ANSWERABILITY DETECTION IN KNOWLEDGE BASE QUESTION ANSWERING

A THESIS
submitted by

PRAYUSHI FALDU

for the award of the degree of

MASTER OF SCIENCE (by Research)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY DELHI
JULY 2024

This thesis is dedicated to my parents.

For their endless love and support.

THESIS CERTIFICATE

This is to certify that the thesis titled **TOWARDS ROBUST KNOWLEDGE BASE QUESTION ANSWERING**, submitted by **PRAYUSHI FALDU (2021CSY7548)**, to the Indian Institute of Technology DELHI, for the award of the degree of **Master of Science (By Research)**, is a bona fide record of the research work done by her under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Mausam
Research Guide
Department of Computer Science and Engineering
IIT Delhi, 110 016

Dr. Indrajit Bhattacharya
Research Co-Guide
TCS Research, IIT Kharagpur Research Park
Kolkata, 700135

Place: Delhi

Date: July 2024

ACKNOWLEDGEMENTS

A heartfelt shout-out to my parents and family for their unwavering support and encouragement. They have been the strongest pillars throughout my journey, and their belief in me has been a constant source of motivation.

I want to express my deepest gratitude to my advisors, Prof. Mausam and Dr. Indrajit Bhattacharya, for their invaluable guidance, support, and encouragement throughout the course of my research. Their insights and expertise have been instrumental in shaping the direction and success of this thesis. Every interaction with them has not only enhanced my technical skills but also imparted numerous life lessons.

I am also grateful to the members of my thesis committee, Prof. Rohan Paul and Prof. Sumeet Agarwal, for their constructive feedback and suggestions. A special mention goes to Prof. Rohan Paul, who has been a constant supporter and motivator throughout the course of my thesis. He has been my go-to person for any advice I needed, and I am fortunate to have a lifelong mentor like him.

I would like to extend my heartfelt thanks to my project collaborators, Mayur and Avinash from TCS Research, for their invaluable contributions to a part of my thesis. I learned a great deal from both of them. The memories we created hold a special place in my heart, as we navigated the ups and downs of the project together, always cheering each other on through every phase. I would also like to acknowledge Vishal, Aniket, and Tuhin for their collaboration on a different line of work. The work we did together was filled with valuable learning experiences that I will never forget.

Special thanks to my colleagues and friends at DAIR lab - Aayush, Ashish, Dhruvil, Himanshu, Namas, Saurabh, Vishal, and Vipul, who provided a stimulating and collaborative environment that inspired me daily. Their camaraderie and moral support have made this journey enjoyable and memorable. All the lab memories are very special to me and will hold a special place in my heart forever.

Special thanks go to my hostel roommates, Prachi and Nikki, who are also my batchmates, for their unwavering support and for making my stay at the hostel such an enjoyable experience. Prachi, my longest-standing roommate, deserves special recognition for supporting me through times when I was unwell, patiently listening to my never-ending talks, and always being a great source of comfort and motivation. I am also deeply grateful to my mess companions - Shreya, Shreyushi, Soumya, Sanhita, and many more - whose conversations

and shared experiences during breakfast, lunch, and dinner were significant stress relievers.

I would also like to acknowledge my close friends Vishwa and Nevil, who have been like family, always believing in me and being just a call away for me whenever I needed them.

I would like to extend my sincere thanks to the computing facility at IIT Delhi, whose resources and technical support were crucial for conducting my research. I would also like to acknowledge the funding and support provided by Reliance Foundations, which made this research possible.

Lastly, my time here has been filled with countless cherished moments and the formation of lifelong friendships. I want to give special mention to all my batchmates and seniors who have supported me in various ways - Daman, Suraj, Ajay, Harsh, Devesh, Shivangi, and Abhishrut. The experiences and relationships I've cultivated during this journey will hold a lasting place in my heart.

Finally, I want to extend my heartfelt appreciation to all those who have contributed, directly or indirectly, to the completion of this thesis. There are many others whom I haven't mentioned here. Their support and encouragement have been invaluable to me—a sincere thanks to all of them.

© 2025, *Indian Institute of Technology Delhi*

ABSTRACT

The problem of natural language question answering over knowledge bases (KBQA) has garnered substantial attention in recent years. An integral aspect of practical KBQA is robustness against unanswerable questions. This challenge can naturally emerge because any natural language question may be unanswerable within a particular knowledge base due to an incomplete KB schema, or due to the absence of relevant facts or the limited scope of the KB. While the concept of answerability has been explored in other Question Answering domains, it has not been thoroughly examined within the context of KBQA. Creating a dataset of unanswerable questions manually is a labor-intensive and time-consuming task because one needs to meticulously design questions so that they are semantically correct and, simultaneously, fall outside the realm of available knowledge, ensuring that no relevant information can be extracted from the KG to provide a meaningful answer. We propose a generic approach for automatically curating a dataset with unanswerable questions, using a KB and a dataset of answerable questions, without human intervention. With this approach, we create GrailQAbility, the largest and most diverse benchmark dataset for KBQA, containing both answerable and unanswerable questions. We conduct a comprehensive experimental analysis of various state-of-the-art KBQA models on GrailQAbility by systematically adapting them for unanswerable questions. Our study elucidates important research questions that underscore the trade-offs associated with different modeling choices and various categories of answerability & unanswerability. Furthermore, we observe that existing models struggle with answerability detection even after suitable training, and the boost in the performance of unanswerable questions comes at the cost of the performance of answerable questions. In response to these challenges, we present RetinaQA - a multi-staged retrieve, generate, and rank framework for KBQA robust against unanswerability. RetinaQA's robustness stems mainly from its discriminative nature and complimentary retriever methods. Our experiments demonstrate that RetinaQA effectively manages the trade-offs between answerable and unanswerable questions, and it significantly outperforms adaptations of state-of-the-art KBQA models on the GrailQAbility dataset. Remarkably, RetinaQA also establishes a new state-of-the-art answerable KBQA, surpassing the performance of all existing models.

KEYWORDS: *KBQA; Unanswerability; Discrimination; Sketch Generation;*

PUBLICATIONS

1. 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.

2. *Prayushi Faldu*, Indrajit Bhattacharya, and Mausam .. 2024. RetinaQA: A Robust Knowledge Base Question Answering Model for both Answerable and Unanswerable Questions. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 6643–6656, Bangkok, Thailand. Association for Computational Linguistics.

Contents

ACKNOWLEDGEMENTS **1**

ABSTRACT **3**

PUBLICATIONS **4**

1 INTRODUCTION **2**

1.1 Contributions 5

1.2 Thesis Structure 6

2 Related Work **7**

2.1 KBQA Datasets 7

2.2 KBQA Models 9

2.3 Answerability Detection in QA 13

3 KBQA with Answerability Detection: Problem Formulation **15**

3.1 KBQA Preliminaries 15

3.2 Answerability Detection - Task Definition 16

4 GrailQAbility: Dataset Creation **19**

4.1 Dataset Creation Algorithm 19

4.2 Dataset Details 22

4.2.1 Categories of Generalization: Definition 23

4.2.2 Dataset Analysis 25

5 GrailQAbility: Evaluation and Discussions **27**

5.1 Baselines 27
5.2 Adapting for Answerability Detection 28
5.2.1 Thresholding 28
5.2.2 Training with unanswerable questions 28
5.3 Evaluation Metrics 30
5.4 Results and Discussions 31

6 RetinaQA: Robust KBQA model for answerability detection 37
6.1 RetinaQA Model 38
6.2 Experimental Evaluation 41
6.2.1 Results for KBQA with Unanswerability 41
6.2.2 Results for KBQA with Answerable questions 44
6.2.3 Ablation Study 45
6.2.4 Error Analysis 47

7 CONCLUSION AND FUTURE WORK 50
7.1 Limitations and Future Direction 51

Chapter 1

INTRODUCTION

In today's digital age, the rapid increase in information volume and complexity requires efficient and intuitive ways to access and understand data. This growing demand has, in turn, sparked significant interest in the field of Knowledge Base Question Answering (KBQA) in recent years. KBQA systems aim to fetch answers to given natural language questions from large-scale Knowledge Bases (KBs), thereby bridging the gap between structured knowledge repositories—such as databases, knowledge graphs, and ontologies—and the intuitive queries posed by users in natural language. These systems play a pivotal role in enabling users to interact and reason over vast stores of structured facts in a manner that is accessible, efficient, and cognitively aligned with human communication patterns. Users can do this without being exposed to the intricate complexities of the KBs. This has encouraged the development of a wide range of KBQA benchmarking datasets and neural architectures that perform competitively on these datasets. Most of these studies aim to build high-quality KBQA datasets and models, emphasizing different aspects like complexity, diversity, scale, and generalizability. The next chapter discusses them in detail.

It is a well-known fact that knowledge in the real world is constantly evolving, leading to continuous updates in KBs. Consequently, a KB can naturally be incomplete at any given moment, and it's very difficult to ensure its completeness at any time. Incompleteness in KBs can arise due to multiple reasons, including incomplete schema, missing facts, and limited KB scope. Additionally, the possibility exists that a posed question is based on a false premise. However, existing studies operate under the assumption that the question is invariably answerable within the given KB, i.e., a question always has some correct answer. However, this assumption poses potential challenges for practical KBQA systems, especially in critical applications like healthcare or legal systems. In these fields, ensuring the reliability and accuracy of information is paramount. Consider the scenario of a medical QA system with an incomplete KB. Suppose it provides an incorrect answer instead of predicting "I don't know" to a medical query due to the assumption that every question has a correct answer within the KB. This misinformation can have severe consequences, potentially leading to misdiagnosis, inappropriate treatment decisions, or compromised patient care. Therefore, practical KBQA systems must be equipped to handle situations where questions may not have definitive answers within the given KB. This necessitates the development of robust systems capable of acknowledging uncertainty, providing reasoned responses, and, when necessary, admitting the inability to provide a conclusive answer.

However, there is no existing work that studies unanswerability in the context of KBQA. In this thesis, we first systematically define unanswerability in the context of KBQA. We then study different categories of KB incompleteness and how they can lead to different types of unanswerabilities. Based on these studies, we create a benchmarking KBQA dataset with unanswerable questions.

Creating a challenging dataset of unanswerable questions is not a trivial task, even with the involvement of human annotators. Firstly, KBs are large-scale, and modern KBs have large and complex schemas. For example, FREEBASE has around 8,000 schema items(relations and entity types) in total. Furthermore, these schema items have complex relationships with each other such as type hierarchy and domain/range information of relations. One needs to have a deeper understanding of the entire KB to craft questions that fall outside the scope of the current KB, ensuring that no relevant information can be extracted from the KB to provide a meaningful answer. Each question requires thoughtful consideration to ensure its unanswerability while still being grammatically correct and coherent. Moreover, the process of validating unanswerable questions adds another layer of complexity. Thus, generating a dataset of unanswerable questions is a non-trivial task that demands expertise, time, and resources. In this thesis, we propose a method to simulate incompleteness in an existing KB and automatically curate a challenging QA dataset. This dataset includes both unanswerable and answerable questions, derived from a given dataset that originally contains only answerable questions. Notably, this method does not require any human annotation. The algorithmic creation of the dataset involves addressing several challenges. These challenges arise from the various ways different KB elements affect the answerability of questions. Additionally, there is a dependence between different types of KB elements, the shared nature of KB elements across questions, and more (Chapter 4 explains them in detail). Using this method, we create a new benchmark dataset for KBQA, **GrailQAbility**. To make the dataset more challenging, we introduce multiple types of unanswerability, each arising from different types of KB incompleteness (facts, entities, relations, types). The percentage of each type of unanswerability in the dataset is controllable and can be varied accordingly.

Another key aspect of a challenging dataset is its ability to test how well the system can generalize. The training data might not always match the real user data, and test questions may come from a different distribution. This is particularly true for large KBs with many entities and schema items (types and relations). User queries may include previously unseen KB elements (Zero-Shot generalization) or compositions of previously seen KB elements (Compositional generalization) (Gu et al., 2021). Therefore, in addition to three categories of generalization for answerable questions: IID, Compositional, and Zero-Shot, we introduce different generalization scenarios for unanswerable questions as well, namely IID and Zero-Shot (we describe each generalization category in detail in subsequent chapters). With a total of around 50,507 questions, involving both answerable and unanswerable questions with

© 2025, *Indian Institute of Technology Delhi*

different categories and generalizations, GrailQAbility is the first challenging benchmark dataset for the task of KBQA with answerability detection.

Next, we evaluate the robustness of state-of-the-art KBQA models on GrailQAbility. Existing KBQA models can be broadly divided into two categories: retrieval-based and semantic parsing-based. Retrieval-based methods directly rank nodes in the KB to fetch answer nodes, while semantic parsing-based methods first convert a natural language question into an executable logical form, which executes over the KB to fetch the answer. We discuss these methods in detail in the next chapter.

State-of-the-art approaches to KBQA are predominantly based on semantic parsing. Additionally, these methods are more interpretable; that is, semantic parsing-based methods necessitate that the correct answer comes for the correct reasons. Therefore, our study focuses on robust semantic parsing-based KBQA models. Existing semantic parsing-based KBQA models employ a retrieval stage to retrieve relevant KB elements which can be either schema elements (relations and types) or paths (facts). Since the existing models were designed with the assumption of answerability, we adapted them to handle unanswerable questions with appropriate modifications. We show that even after adaptation and training for unanswerability, these models struggle significantly with unanswerable questions. Additionally, we observe that all models witness a significant drop in the performance of answerable questions when trained for unanswerable questions. 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. A robust KBQA model should be able to correctly differentiate between answerable questions and unanswerable questions. This requires good model calibration. Most KBQA models use *generation* of logical form as their final step, which has calibration challenges. Instead, since discriminative models are generally better calibrated, RetinaQA's final stage *discriminates* among candidate logical forms, thus separating questions without any valid logical form better. Secondly, existing models assume answerability, where all logical forms have accompanying paths in the KB. These models start from the mentioned entity (the entity in the question) and retrieve paths from the KB to generate a logical form and predict an answer. However, when a path is broken due to missing facts or entities, these path-based traversal methods cannot generate the correct logical form to predict the correct answer. To handle such unanswerable questions, RetinaQA employs a sketch-filling-based construction. It first generates high-level sketches that are independent of the KB and then grounds these sketches for the specific KB using retrieved schema elements. This approach bypasses the dependence on path traversals.

© 2025, *Indian Institute of Technology Delhi*

Through experiments conducted on GrailQAbility, we demonstrate that RetinaQA significantly outperforms adaptations of multiple state-of-the-art KBQA models that assume answerability, not only across different categories of unanswerable questions but also for answerable questions. We also test the models' performance after training with only answerable questions (which is a more realistic scenario). We show that RetinaQA performs well on unanswerable questions in this scenario as well. Furthermore, we show that, unlike other KBQA models, RetinaQA does not observe a large performance drop on answerable questions after training with unanswerable questions. We present a detailed ablation study to highlight the importance and intuition behind the architectural design choices of RetinaQA. Additionally, we show that RetinaQA performs equivalently well on KBQA datasets with only answerable questions. Finally, we present extensive error analysis and highlight potential areas for improvement in the future.

1.1 Contributions

Our work makes the following key contributions:

1. We motivate and introduce the task of detecting answerability for KBQA.
2. We present a generic approach for creating KBQA datasets with unanswerable questions. We create GrailQAbility - the first benchmark dataset for KBQA with unanswerable questions.
3. Through experiments and analysis on GrailQAbility with state-of-the-art KBQA models, we demonstrate the dataset's effectiveness and identify aspects of unanswerability that existing models struggle with.
4. We propose a novel KBQA model - RetinaQA which becomes a new state-of-art-model for detecting unanswerability by surpassing all existing KBQA models. We perform a thorough examination and show the effectiveness of different architectural components for various categories of answerability and unanswerability. We also present a detailed error analysis which points out interesting future directions
5. We release data[1] and code[2] to the community for further research.

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

© 2025, *Indian Institute of Technology Delhi*

1.2 Thesis Structure

The rest of the thesis is divided into the following chapters:

- **Chapter 2** presents a literature survey on the problem, encompassing existing research on datasets, models, and the concept of unanswerability in general.
- **Chapter 3** formalizes the problem of KBQA with answerability detection and describes the related terminologies and definitions.
- **Chapter 4** provides an in-depth explanation of GrailQAbility, including the dataset creation algorithm, different test scenarios, and generalizations.
- **Chapter 5** explains the metrics and compares the performance of state-of-the-art models on GrailQAbility. It also includes a detailed error analysis for different models, highlighting important research questions about the robustness of KBQA models.
- **Chapter 6** presents RetinaQA. It discusses the model architecture in detail, provides experimental results on GrailQAbility, and compares them with baselines. Additionally, it includes an extensive ablation study and error analysis of RetinaQA.
- **Chapter 7** concludes the thesis, presenting the limitations of the work and potential future directions.

© 2025, *Indian Institute of Technology Delhi*

Chapter 2

Related Work

In this chapter, we delve into a comprehensive review of the existing literature pertaining to KBQA. We explore three main areas: KBQA datasets, KBQA models, and the Answerability Detection task for QA. Firstly, we examine the landscape of KBQA datasets, discussing the diverse range of datasets available for training and evaluation purposes. Secondly, we explore various existing KBQA techniques. Lastly, we investigate the Answerability Detection task for general QA tasks and analyze its relation with KBQA.

2.1 KBQA Datasets

An array of KBQA datasets has been released in recent years. These datasets differ in terms of their size, generalization capabilities, complexities, representational power, etc.

KBQA datasets can be broadly classified into two subsets. One subset consists of datasets (Bordes et al., 2014; Zhang et al., 2017; Bao et al., 2016) that only include question-answer pair annotations. These datasets lack explicit answer paths, making them a weaker choice for complex questions (Yih et al., 2016). This is particularly true for questions that require compositional reasoning over multi-hop answer paths, especially when dealing with large KBs. Hence, we believe that along with question-answer pair annotations, intermediate supervision in terms of answer paths or logical forms is crucial for enabling better reasoning capabilities.

The other subset of datasets includes reasoning paths along with question-answer pairs in annotations. As our work builds on existing datasets, we discuss them in detail here. The **WebQSP** dataset (Yih et al., 2016) comprises 4,737 questions on FREEBASE[1] sourced from real web-query logs. Each question is associated with an executable semantic parse in SPARQL format using standard FREEBASE entity identifiers. The authors demonstrate that learning from labeled semantic parses notably enhances overall performance, leading to significant improvements compared to learning from answers.

However, the dataset is small in size, so **ComplexWebQSP** (Talmor and Berant, 2018) was developed as an extension of WebQSP. This dataset utilizes SPARQL queries from

[1]FREEBASE is on open-source KB which contains over 100 domains, 40 million entities, and 2 billion facts. https://developers.google.com/freebase

WebQSP to automatically generate more complex queries incorporating phenomena like function composition, conjunctions, superlatives, and comparatives. Through Amazon Mechanical Turk (AMT), ComplexWebQSP collects 34,689 question-answer pairs along with corresponding SPARQL queries. However, this method of query extension does not ensure the naturalness of questions, potentially resulting in redundancy or irrelevant information within the questions. Additionally, the split between the training and test sets is uniformly random. Hence, the test set exhibits the same unrealistic distribution as the training set, which is not ideal.

SimpleQ (Bordes et al., 2015) is a factoid question-answering dataset comprising 108,442 natural language questions. These questions were generated by sampling facts (subject-relation-object triples) from the FREEBASE. Notably, the answer to each question is always the object, resulting in a dataset characterized by single-relation queries, making it straightforward and simple.

Several large-scale datasets (Su et al., 2016; Dubey et al., 2019; Cao et al., 2022a) have been released to enhance the complexity of questions in KBQA. **GraphQ** (Su et al., 2016) is a dataset created using a semi-automated approach. It employs an algorithm to sample graph-structured logical forms (graph queries) from FREEBASE, and then human annotators convert these graph queries into questions. Graph Query generation includes 3 steps : (1) Generation of Query Templates: Query templates are generated by traversing the KB ontology. These templates consist only of classes, relations, and functions. (2) Grounding Nodes: The nodes of the query templates are grounded to compatible entities, resulting in graph queries. (3) Query Refinement: The graph queries are refined by reducing redundancy and commonness. **LC-QuAD** (Dubey et al., 2019) is another large-scale KBQA dataset containing 30,000 pairs of questions and corresponding SPARQL queries. The target knowledge base for LC-QuAD is DBPEDIA[2]. **KQAPro** (Cao et al., 2022a) is also a large-scale dataset, comprising 117,970 questions with corresponding SPARQL annotations. It additionally provides annotations of multi-step programs composed of basic functions for each question.

Despite the large size and diversity of these datasets in terms of the complexities and structures of intermediate logical forms, they all focus on the *IID* setting, where the distribution of test and training data is the same, which is unlikely in practical scenarios. Given the large-scale nature of KBs, it is challenging to create a training dataset covering questions from all possible combinations of KB elements. **GrailQA** (Gu et al., 2023) is a large-scale KBQA dataset specifically designed to tackle various challenges in the field. It leverages the GraphQ algorithm (described above) to generate logical forms and introduces a crowdsourcing framework for annotations. Similar to GraphQ, GrailQA generates canonical

[2]DBPEDIA is also an open-source KB developed by crowd-sourcing. It contains around 6 million entities and 9.5 billion facts.https://downloads.dbpedia.org/wiki-archive/dbpedia-version-2016-04.html

© 2025, *Indian Institute of Technology Delhi*

logical forms based on queries per template. Each canonical logical form is validated with the criterion: "Could we reasonably expect a real human user to ask this question?" Valid canonical logical forms are then annotated with canonical questions. The authors developed a crowdsourcing framework using Amazon Mechanical Turk to gather paraphrases of the canonical questions. Additionally, more logical forms are generated by grounding canonical logical forms. From the pool of logical forms and their associated paraphrases, controlled sampling is performed. This involves selecting one from each pool at a time, starting with uniform weights, and reducing the weight of each selected logical form or paraphrase by a factor. Using this framework, the authors created a large-scale dataset with a total of 64,331 questions. These questions include three categories of generalization: *IID* (where logical forms in the test follow the same distribution of KB elements as in training), *Compositional* (where logical forms in the test have novel compositions of KB elements seen in different questions in training), and *Zero-Shot* (where logical forms in the test have unseen KB elements). The questions are annotated with graph queries, SPARQL, and s-expressions (a more interpretable and compressed representation of graph queries). However, like other datasets, GrailQA solely comprises *answerable* questions.

QALD (Perevalov et al., 2022), a multilingual dataset, contains 'out-of-scope' questions that could be considered unanswerable based on our definition. Nonetheless, the number of such questions remains minimal (a few tens across different dataset versions), impeding meaningful benchmarking efforts. Furthermore, QALD lacks fine categorization of such questions. To the best of our knowledge, there exists no high-quality KBQA dataset specifically tailored for unanswerable questions with detailed categorization into various generalizations which we curate in Chapter 4.

2.2 KBQA Models

With the emergence of various challenging datasets, extensive research has been conducted on KBQA models in recent years. Broadly, the solution approaches can be categorized as retrieval methods and semantic parsing-based methods.

Retrieval-based approaches directly rank entity nodes in the KB to find the answer node. These methods first perform entity linking to identify the mentioned entities in question. They then enumerate and score paths from these mentioned entities to directly retrieve the answer. Some methods (Thai et al., 2022a; Das et al., 2022) propose a two-staged approach: one non-parametric retrieval module that finds similar questions from a training set based on KNN, and a parametric component that is trained to identify the reasoning patterns from the sub-graphs of KNN queries and then select the subgraph of the target query. Some methods propose to learn KG embeddings first, which are then used to

© 2025, *Indian Institute of Technology Delhi*

generate query graphs (Mitra et al., 2022; Wang et al., 2022b) or rank the possible answer entities to select final answer (Saxena et al., 2020).

Query generation-based approaches (Cao et al., 2022b; Ye et al., 2022; Chen et al., 2021; Das et al., 2021) learn to generate a logical form or a query (e.g., in SPARQL) based on the question, which is then executed over the KB to obtain an answer. Since the main focus of this thesis is on query generation-based approaches, we provide a detailed explanation of these methods here.

- **ReTraCk** (Chen et al., 2021) presents a neural semantic parsing framework with three stages: Retriever, Transducer, and Checker.

  The Retriever consists of an entity linker and a schema retriever. The entity linker connects explicit entity mentions to corresponding entities, using the pipeline proposed in Gu et al. (2021). The schema retriever identifies relevant schema items (types and relations) mentioned explicitly or implicitly in the question. For this, they train a bi-encoder architecture based on BERT, using a cosine similarity objective to match related schema items with the question embedding. During inference, they retrieve the top-k schema items with the highest scores.

  The Transducer uses a grammar-based decoder to generate the logical form with syntax correctness guarantees. The decoder is based on LSTM, it treats each logical form as a sequence of actions and outputs the corresponding grammar rule or action at each step. The transducer interacts with the Checker to avoid generating programs that are semantically inconsistent with the KB.

  The Checker employs two types of checks: ontology-based checking and instance-based checking. Instance-level checking relies on KB linkage information at the data level (entities and their connected relations), while ontology-level checking uses linkage information at the ontology level (types and bridging relations).

  Once decoding reaches the end, an action sequence is converted into a logical form and then into a SPARQL query. If the query cannot be executed successfully or the result is empty, the next action in the beam is utilized. Decoding continues until a query successfully executes over the KB with a non-empty answer.

- **RnG-KBQA** (Ye et al., 2022) is a rank-and-generate KBQA framework with two main stages: a ranker followed by a generator. The Ranker selects a set of related logical forms from a pool of candidate logical forms obtained by searching over the graph. The Generator then uses both the question and the top-ranked candidates to compose the final logical form.

  For entity linking, authors adopt the mention detection method of Gu et al. (2021) and propose a ranking-based method similar to the ranker for entity disambiguation.

  The Ranker comprises two stages: Logical Form Enumeration and Logical Form Ranking. During Enumeration, for every entity detected in the question, the knowledge base is queried for paths reachable within two hops. They then write down an s-expression corresponding to each of these paths, creating a set of candidates. After enumeration, the candidates are ranked.

  The Ranker model learns to score each candidate logical form by maximizing the similarity between the question and the ground truth logical form while minimizing

© 2025, *Indian Institute of Technology Delhi*

the similarities between the question and the negative logical forms. It uses a BERT-based encoder that takes as input the concatenation of the question and the logical form, outputting a logit representing the similarity between them. The ranker is optimized to minimize contrastive loss based on these similarity scores. However, it is impractical to cover every possible logical form when enumerating, so the authors address this issue via the generation model.

The top-k logical forms from the ranker and the question are fed into the Generator to produce the final logical form. The Generator is a transformer-based seq-to-seq T5 model trained to generate the ground truth logical form autoregressively with a cross-entropy objective using teacher forcing. During inference, beam search is used to decode the top-k target logical forms.

Since the T5 generation model does not guarantee the syntactic correctness or executability of the produced logical forms, an execution-augmented inference is employed. Here, the logical form that executes successfully and generates a non-empty answer is accepted.

- **TIARA** (Shu et al., 2022) is a multi-grained Retrieval Augmented KBQA framework that provides both semantic and syntactic references to a pre-trained language model (PLM) and uses constrained decoding for logical form generation.

  The Multi-grained Retriever consists of three components: Entity Retriever, Exemplary Logical Form Retriever, and Schema Retriever. For entity retrieval, the authors follow a standard pipeline involving mention detection and entity disambiguation. To improve zero-shot mention detection, they propose SpanMD, a mention detector trained with an objective similar to span classification. For entity disambiguation, they use a cross-encoder framework similar to that in RnG-KBQA.

  The Exemplary Logical Form Retriever follows enumeration and ranking methods similar to RnG-KBQA. Additionally, the authors employ dense schema retrieval to find semantically relevant items (classes and relations), independent of entity and exemplary logical form retrieval. Schema retrieval is implemented using a cross-encoder based architecture with an objective similar to sentence-pair classification.

  All the retrieved contexts are fed to the generator for target logical form generation. The generation is performed by a transformer-based seq2seq model, T5, with an objective similar to that of RnG-KBQA.

  During inference, constrained decoding is employed, ensuring the correctness of the generated operators and schema elements in the logical form.

- **Pangu** (Gu et al., 2023) is a generic framework for grounded language understanding that leverages the discriminative abilities of language models (LMs) rather than their generative abilities. The authors implement this framework for complex question answering over knowledge bases (KBQA). Unlike other methods, Pangu does not have a separate retrieval stage. Instead, it performs enumeration and ranking step-by-step, rather than in a one-time process like RnG-KBQA.

  Pangu is based on the hypothesis that a logical form can be incrementally constructed by interacting with the KB. The framework starts with an empty logical form set and a set of entities (mentioned in the question) P_0. At each step, the agent interacts with the KB, enumerates possible 1-hop paths, and extends the existing set of logical forms.

© 2025, *Indian Institute of Technology Delhi*

This process repeats until a termination check is passed and the best logical form is returned.

The authors use an entity linking pipeline similar to TIARA. To score the logical forms at every step, they employ an LM-based scorer and discuss various architectures for the LM (encoder-only like BERT, encoder-decoder like T5, or decoder-only LMs like Codex).

The LM is optimized based on the following criteria: (1) The score of incorrect logical forms is less than the score of correct logical forms. (2) The score of partially correct logical forms is less than the score of correct logical forms. (3) The score of incorrect logical forms is less than the score of partially correct logical forms. After each enumeration and scoring step, a termination check is applied. The process terminates if the highest score of logical forms at step t is lower than the highest score of logical forms at step $t1$, indicating that the logical form generated at step $t1$ is better than the one generated at step t.

Based on these criteria, Pangu learns to assign the highest score to the correct logical form, which is returned after the termination check. During inference, Pangu uses beam search and eliminates logical forms with empty answers.

Complementary to these methods involving intermediate logical forms as supervision is **program induction**-based methods that learn intermediate representations from question-answer pairs (Liang et al., 2017; Saha et al., 2019; Ansari et al., 2019). Typically, they take the answers as weak supervision and search for gold programs using reinforcement learning (RL). Zhang et al. (2022) combines this training technique with retrieval. It trains a subgraph retriever via weak-supervised pre-training, which is then combined with a reasoner trained with answer paths.

In addition to these, KBQA is also studied in **transfer** (Cao et al., 2022b; Ravishankar et al., 2022) and **few-shot** (Li et al., 2023) settings. Here, test questions involve questions from the same KB but different datasets or questions that involve relations and entity types unseen during training. These approaches use the notion of generalizable sketches (also called drafts or skeletons) that capture the syntax of the target language. The skeleton of any logical form can be derived by removing all the KB elements, such as relations, classes, and entities. This leaves only the function operators, parentheses, and placeholders/arguments for KB elements.

However, all of these models assume the answerability of questions and are designed to always return a non-empty answer. No specialized models have been proposed so far for detecting unanswerable questions in KBQA.

© 2025, *Indian Institute of Technology Delhi*

2.3 Answerability Detection in QA

Answerability detection has been explored for extractive QA (Rajpurkar et al., 2018), conversational QA (Choi et al., 2018; Reddy et al., 2019), boolean (Y/N) QA (Sulem et al., 2022) and MCQ (Raina and Gales, 2022). For Conversational QA, the goal is to answer a series of questions (as conversations) given a text passage. The answer can be either span-based or free-form. In this context, a question is unanswerable if there is no relevant answer based on text. Some datasets have been proposed with unanswerable questions (Reddy et al., 2019; Choi et al., 2018). Most of these datasets are crowd-sourced and contain a small percentage of unanswerable questions. While in practical scenarios the percentage of unanswerable questions might be higher.

Various studies have been performed on answerability detection for extractive QA where the task is to extract an answer to a question from a passage. Jia and Liang (2017) proposes a rule-based method to create unanswerable questions by making changes to SQUAD questions. Because of simple edits like replacing entities/numbers with similar words or antonyms - the dataset is not diverse enough. SQUADRUN (Rajpurkar et al., 2018) combines the existing Stanford Question Answering Dataset (SQuAD) with over 50,000 unanswerable questions written adversarially by crowd workers to look similar to answerable ones so that the unanswerable question is relevant to given the passage. For BooleanQA, Sulem et al. (2022) builds on top of the BoolQ dataset to include unanswerable questions. It uses data augmentation methods to generate negative samples by pairing questions with non-relevant context selected on the basis of TF-IDF. Raina and Gales (2022) studies unanswerability in multi-choice machine reading comprehension based QA. It creates data by replacing the correct answer options with (None Of the Above) NOA.

While our work is motivated by these approaches, the nature of unanswerable questions is very different for KBs compared to unstructured contexts. In unstructured contexts, a question can be directly linked to the answer without involving reasoning. In contrast, KBQA requires reasoning over multiple paths to reach an answer node. Additionally, unlike unstructured contexts, the context for KBQA is the KB itself, which involves multiple schema items, entities, and facts tied by a strict ontology. To answer a question with respect to a given KB, one must retrieve paths and KB elements to prepare relevant context. This context should be then logically pieced together to create a meaningful path (or intermediate representation) that leads to the answer, unlike in unstructured contexts where answers can be generated directly. So we cannot directly generate logical positive and negative pairs of (question, context) to create unanswerable questions, as context is shared across all questions. Also, dataset creation using crowd-sourcing is very difficult as explained in Chapter 1.

Among all methods, the most relevant work to KBQA is semantic parsing tasks like text-

© 2025, *Indian Institute of Technology Delhi*

to-SQL. Recently, unanswerability has been explored for text-to-SQL (Wang et al., 2022a). It proposes a counterfactual example generation approach that automatically produces unanswerable text-to-SQL examples using existing datasets by modifying the structured table. While there is a natural connection between text-to-SQL and KBQA, KBQA is more challenging due to two main factors: schema-level and fact-level complexity in large-scale KBs. On one hand, the schema of a modern KB is extremely rich, whereas a relational database typically contains only a few dozen schema items (e.g., table names and column headers) (Yu et al., 2018). This makes schema linking (learning an alignment between natural language and the schema) much more difficult in KBQA. Additionally, as explained in Chapter 1, the KB ontology models complex relationships among schema items, demanding a better understanding of the schema for KBQA compared to other semantic parsing tasks (Gu et al., 2022).

There is work on improving the accuracy of QA over incomplete KBs (Thai et al., 2022b; Saxena et al., 2020). In these approaches, the model assumes answerability and tries to predict the missing link or path to reach the answer node. However, this work focuses only on one aspect of incomplete KBs, namely missing paths, while unanswerability can also arise due to an incomplete schema or missing entities.

In conclusion, our literature survey reveals that there is no existing KBQA dataset or model specifically addressing the issue of unanswerability in KBQA. While answerability detection has been studied in other QA tasks, these methods cannot be directly applied to KBQA. Therefore, we draw inspiration from existing research on answerability detection and conduct a systematic study of unanswerability within the context of KBQA.

© 2025, *Indian Institute of Technology Delhi*

Chapter 3

KBQA with Answerability Detection: Problem Formulation

3.1 KBQA Preliminaries

A *Knowledge Base* (KB) (also called Knowledge Graph) G is a set of facts defined by a *schema* S and an ontology O. The schema consists of *entity-types* (or types) T and *relations* R defined over pairs of entity-types. We refer to entity types and relations together as **schema elements** of the KB. KB also contains *entities* E as instances of entity types, and *facts* F. A fact is a (s, r, o) triple where s is an entity, r is a relation, o is an entity or literal L, therefore $F \subseteq E \times R \times (E \cup L)$. We refer to entities and facts as **data elements** of the KB. The top layer of Figure 3.1(A) shows example schema elements, while the bottom layer shows entities and facts.

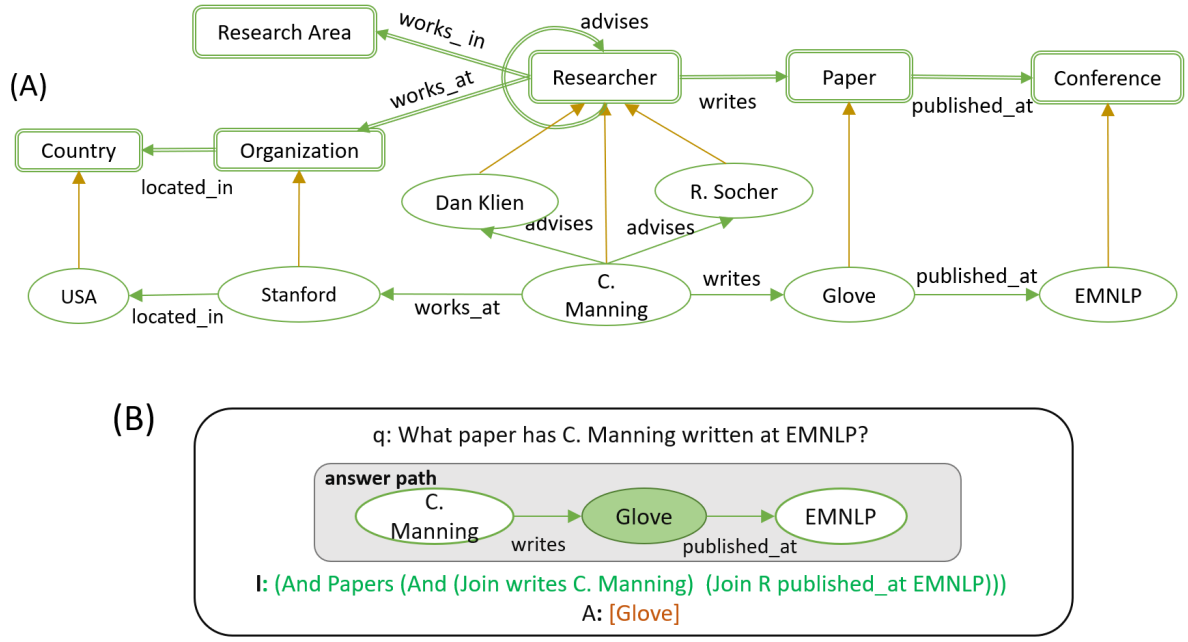

Figure 3.1: Snapshot of KB. (A) The top layer consists of schema which defines KB. The bottom layer is composed of facts i.e. entities linked by relations. (B) Example of question q with answer path, ground truth logical form l and answer A.

For instance, within a schema, the entity type "Researcher" is connected to the entity type "Paper" via the relation "writes" indicating that any instance of the "Researcher" entity type

can be associated with any instance of the "Paper" entity-type through the "writes" relation (for instance, C. Manning writes Glove). Here, we denote entity types such as "Researcher" and "Paper," along with relations like "writes," as schema elements. In contrast, entities like "C. Manning," "Glove," and the fact "C. Manning writes Glove" are considered data elements.

The **task of KBQA** can be defined as - *Translating a natural language* question q *to a* logical form *(or query)* l *that executes over a KB* G *to yield a set of* answers A.
Figure 3.1(B) illustrates an example question alongside its answer path, logical form, and corresponding answer. The answer path (or graph query) represents a reasoning chain over facts in the KB, starting from the entity mentioned in the question (e.g., C. Manning) and leading to the answer. The logical form (s-expression in this case) is a linearized representation of the graph query, consisting of operators, functions, and their arguments (entity types, relations, operators, and literals). All the functions and arguments are defined by a grammar. Additionally, every graph query can be easily converted to an equivalent s-expression. Furthermore, both graph queries and s-expressions can be readily transformed into SPARQL queries to retrieve answers. For this thesis, we will utilize s-expressions as the intermediate logical form.

3.2 Answerability Detection - Task Definition

Before outlining the task of answerability detection, we establish the definitions for answerable and unanswerable questions. These definitions encompass both logical forms and answers, as our emphasis lies in ensuring accurate reasoning leading to the correct answer, rather than merely obtaining correct answers.

A question q is defined to be **answerable** for a KB G, if

- q admits a valid logical form l for G, AND
- l returns a *non-empty* answer set A when executed over G.

And a question q is defined **unanswerable** for a KB G if,

- q does not admit a valid logical form l for G, OR
- the valid l when executed over the G returns an empty answer.

A logical form of a question is considered *valid* for a given G if it executes over G without an error and all KB elements required to construct a correct logical form are present in the KB. For instance, in Figure 3.2(A), l represents a valid logical form. However, if any of the

© 2025, *Indian Institute of Technology Delhi*

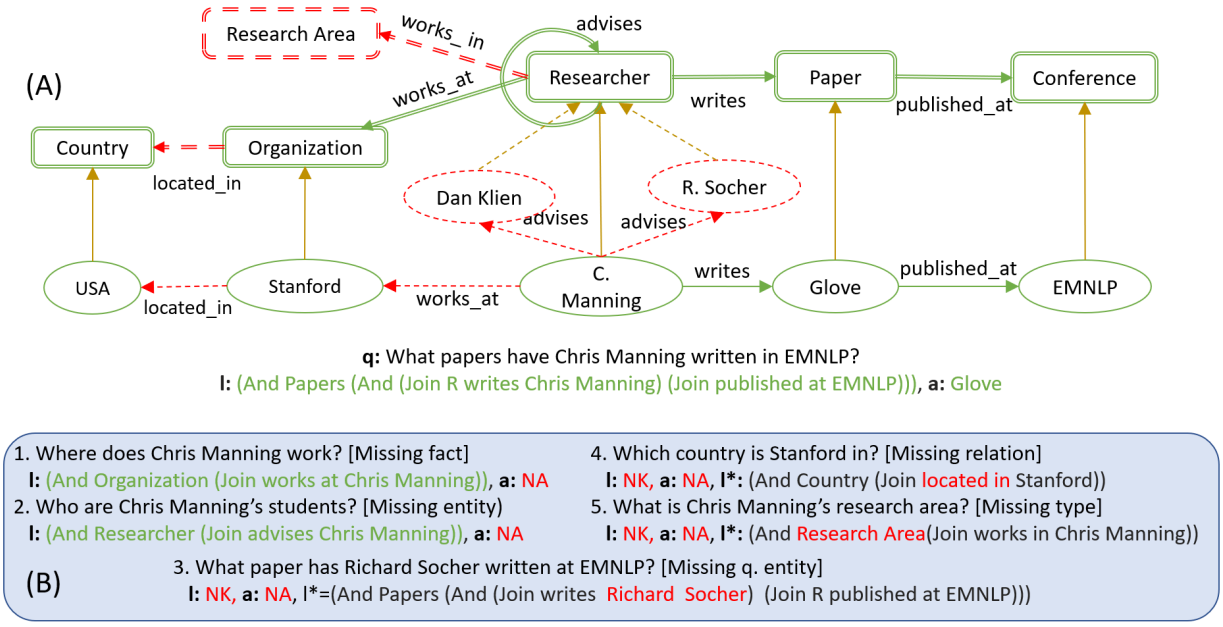

Figure 3.2: (A) KB schema and facts. Elements in red are part of the 'ideal' KB but missing in the given KB for QA. An answerable question is shown for this KB with logical form l in s-expression, answer a, and path (shaded blue). (B) There are 5 types of unanswerable questions for the provided KB. These include actual logical forms l (with respect to the updated KB), answers a, and ideal logical forms l^* (with respect to the original KB). In the ideal logical forms, the missing KB elements are highlighted in red (3-5).

KB elements referenced in the logical form were absent from the KB, the logical form would become invalid.

Let's consider an incomplete KB (non-ideal KB where the KB elements highlighted in red are absent), as shown in Figure 3.2(A). Consequently, logical forms that rely on these missing elements become invalid. For instance, in example 4 of Figure 3.2(B), the absence of the "located in" relation renders the logical form invalid. Here, l^* (ideal logical form) denotes the logical form with respect to the ideal or complete KB. However, a KB may be incomplete due to the absence of schema elements or data elements. In Figure 3.2(B), various types of KB incompleteness are illustrated. *Example* 4 pertains to the absence of the "located in" relation from the schema, while *Example* 5 involves the absence of the entity type "Research Area" from the schema. *Example* 3 highlights the missing question entity "Richard Socher". In all three examples, certain components of the ideal logical form are absent, resulting in the absence of a valid logical form, denoted as "NK." *Example* 1 depicts a missing fact where the statement "C. Manning works at Stanford" is absent from the KB. While a valid logical form can be formulated, the answer to the question is empty, referred to as NA here. In *Example* 2, the absence of the answer entity R. Socher from the KB results in an empty answer, despite a valid logical form being possible.

Hence, a question will be unanswerable when the KB is incomplete, and the reason for

the KB incompleteness can arise from the absence of different KB elements (entity types, relations, entities, facts). Thus, the type of unanswerability is linked to the category of missing KB elements.

For a question q and a KB G which can be incomplete, the task of **KBQA with answerability detection** is defined as follows -

- appropriately label the answer as NA (No Answer) when q is unanswerable for G,
- label the logical form l as NK (No Knowledge) when it is not possible to generate a valid logical form for G,
- generate the correct non-empty answer A and valid logical form l when q is answerable for G.

However, it is interesting to note that it is also possible for a question to be meaningless and not have any valid answer, e.g., "In which country is the USA located?" or "Who is Glove?" For such questions, there is no possible valid answer or logical form. Or it is also possible that a question has an empty answer for example in the above KB (Figure 3.2 if C. Manning has never advised any students then the natural answer to Example 2 will be NA. But it is difficult to distinguish these types of questions with no answer from the second type of unanswerability i.e. a meaningful question with no answer due to missing data elements (facts or answer entities). *The study of meaningless/out-of-scope questions for a KB is not included in this thesis.*

© 2025, *Indian Institute of Technology Delhi*

Chapter 4

GrailQAbility: Dataset Creation

We described the problem formulation of KBQA with Answerability Detection in the previous section. To assess existing KBQA systems for the task of answerability detection, we need a dataset with unanswerable questions. As discussed in Chapter 2, there is no such existing dataset. In this chapter, we first present an algorithm to create a dataset with unanswerable questions. Using this algorithm, we create the first benchmarking dataset that includes both answerable and unanswerable questions, named GrailQAbility. Later in the chapter, we discuss GrailQAbility in detail.

4.1 Dataset Creation Algorithm

In the previous chapter, we discussed that unanswerability can arise from different types of KB incompleteness. Instead of creating a dataset with human intervention, can we develop an automated mechanism to simulate incompleteness in the KB? This would allow us to use existing datasets that contain only answerable questions.

Problem: Extend a dataset of only answerable questions to a dataset with both answerable and unanswerable questions by systematically modifying the KB.

Solution in nutshell: We begin with a standard KBQA dataset comprising solely answerable questions for a given KB (G^*), where G^* denotes an ideal/complete KB for a dataset with all answerable questions. Subsequently, we simulate KB incompleteness in stages by removing schema elements (entity types and relations) and data elements (entities and facts) from the original KB. Questions that become unanswerable due to each drop are annotated with appropriate unanswerability labels. Consequently, we obtain an updated KB (G) and a dataset comprising answerable and unanswerable questions for (G).

Given that we are algorithmically simulating incompleteness, it is crucial to *control* the percentage of questions that become unanswerable following each type of drop. Moreover, to create a challenging dataset, it is essential to incorporate different categories of unanswerability with varying generalizations as discussed in Chapter 1. Therefore, crafting a

This work is jointly done with Mayur Patidar and Avinash Singh from TCS Research. They contributed the initial POCs for dataset creation. The rest of the design and development of GrailQAbility is a collaborative effort.

challenging dataset for KBQA in a controlled manner necessitates a systematic simulation, which poses several challenges.

1. Dropping various KB elements affects answerability in different ways. It impacts both logical forms and answers distinctly. For instance, the schema elements drop influences logical forms, whereas the removal of facts impacts answers exclusively. As illustrated in Figure 3.2 (A), dropping "Research Area" affects the logical form, whereas dropping the fact "C. Manning works at Stanford" affects only the answers.

2. In a KB, elements cannot be dropped independently due to their interdependencies. Entity types are linked with relations and entities, while relations and entities are associated with facts. For instance, as illustrated in Figure 3.2, dropping the type "Research Area" removes both its incoming and outgoing edges (relations). Additionally, all entities labeled with the type "Research Area" will be dropped from the data, dropping all facts associated with these entities.

3. The impact of a dropped KB element on answerability varies depending on its role within the query graph. For instance, if a dropped entity serves as the mentioned entity (i.e., the starting node of the query graph.), it affects the logical form. Conversely, if the dropped entity acts as an intermediate entity (i.e., an intermediate node of the query graph.), it impacts only the answer. In Figure 3.2, the drop of the entity "Richard Socher" influences the logical form in example B.3, whereas it affects only the answer in example B.2.

4. Questions with multiple answers remain answerable as long as the fact paths leading to any of these answers remain intact. In Figure 3.2, the question in example B.2 becomes unanswerable only when both answer entities are dropped from the KB; otherwise, the answer will not be NA.

5. Selecting KB elements for dropping uniformly at random does not accurately reflect the notion of incompleteness in the real world.

Algorithm 1 outlines the proposed approach for data generation at a high level. Dropping different types of KB elements is done in a specific sequence. First, entity types are dropped, followed by relations. Next, entities are removed, and finally, facts are dropped (line 1). This sequence is chosen considering the hierarchy of associations and dependencies between various types of KB elements. Subsequently, we iterate over the four categories of KB elements one by one (line 6). Then, based on the category of KB incompleteness sampling is performed to select the KB element to be dropped (line 9). To emulate real-world KB incompleteness, we employ *inverse popularity-based sampling*, as "better known" KB elements are less likely to be missing. Once a KB element to be dropped is determined, we appropriately relabel its logical forms (line 12) or answers (line 13) and tag the question with the dropped type (line 16). Throughout the process, we track the remaining paths leading to answers for each question. We designate a question as unanswerable only when all paths are disrupted by KB drops. We halt the process when specific percentages of questions become unanswerable for each category (line 8). Consequently, we obtain a new version of

© 2025, *Indian Institute of Technology Delhi*

the KB - KB with dropped elements (line 15) and a dataset containing both answerable and unanswerable questions for the updated KB (line 18).

Algorithm 1 Dataset creation algorithm at high level

1: $kb_incompletness \leftarrow$ {Type, Relation, Entity, Fact}
2: $p_{uc} \leftarrow$ percentage of unanswerable questions for each category
3: $\hat{G} \leftarrow G^*$
4: $Q \leftarrow$ all answerable questions
5: $Q_u = \{\}$
6: **for all** $category \in kb_incompletness$ **do**
7: $\quad Q_{uc} = \{\}$
8: $\quad$ **while** $|Q_{uc}|/|Q| \leq p_{uc}[cat]$ **do**
9: $\quad\quad g \leftarrow Sample(\hat{G})$
10: $\quad\quad$ **for all** $q \in Q$ **do**
11: $\quad\quad\quad$ **if** $q \notin Q_u$ **then**
12: $\quad\quad\quad\quad Update_LogicalForm(lf_q, g)$
13: $\quad\quad\quad\quad Update_Answers(a_q, g)$
14: $\quad\quad\quad\quad$ **if** $Is_Unanswerable(lf_q, a_q)$ **then**
15: $\quad\quad\quad\quad\quad Q_{uc}.add(q)$
16: $\quad\quad \hat{G} \leftarrow \hat{G} \backslash \{g\}$
17: $\quad Q_u.add_all(q_{uc})$
18: $Q_a \leftarrow Q - Q_u$

- G^* refers to complete KB for questions Q.
- $\hat{G}$ refers to updated KB.
- Q_u refers to set of curated unanswerable questions.
- Q_{uc} refers to the set of curated unanswerable questions for each incompleteness category.
- g refers to sample KB elements to be dropped.
- lf_q refers to the logical form for question q.
- a_q refers to the set of answers for question q.
- $Update_LogicalForm$ is a function that updates logical form-in-place, given g.
- $Update_Answers$ is a function that updates the answer set in place, given g.
- $Is_Unanswerable$ is a boolean function that outputs if a question is unanswerable given logical form and answer set.

The specific details about functions and drops for every category of KB incompleteness are described below.

Entity Type Drop: Entities are often tagged with multiple types in a hierarchy (e.g. *C. Manning* may be *Researcher* and *Person*). After dropping an entity type from the KB schema, drop all entities e that are associated *only* with that type. Further, drop all relations associated with the type. For a question affected by entity-type drop, we set its answer as NA and its logical form as NK. For sampling entity types to drop, the popularity of an entity type is defined as the number of facts in the original KB associated with that type or any of its descendant types.

© 2025, *Indian Institute of Technology Delhi*

Relation Drop: To drop a relation, we first drop all facts associated with the dropped relation and then drop the relation from the schema. For a question affected by relation drop, we set its answer as NA and its logical form as NK. Since it is possible to create a valid logical form using the reverse of the dropped relation, we ensure that if a reverse relation exists, it is also removed. To sample relation drop, the popularity of the relation is taken as the number of associated facts in the original KB.

Entity Drop: To drop an entity from the KB, we first drop all its associated facts and then drop the entity itself (Figure3.2(B-3)). Dropping an entity affects answerability in two ways. If the logical form contains the dropped entity as one of the mentioned entities, then it invalidates the logical form. For such questions, we set the logical form as NK and the answer as NA. If entity drop affects some answer path then it invalidates only the answer, and we set it as NA. When sampling entities to be dropped, we consider all entities to be equally popular. This approach is taken because the number of entities in a KB is very large, and the association of facts to entities is sparser than that of schema elements. In other words, the distribution of facts across entities is almost uniform.

Fact Drop: Dropping a KB fact can break the path of one or more answers for a question but cannot affect the logical form. Answers whose paths are broken are removed from the answer list of the question. If the answer list becomes empty as a result of a fact drop, we set its answer to NA but leave its logical form unchanged. For sampling facts to drop, we consider all facts equally popular similar to entity drop.

Using this algorithm, given a KB and a set of answerable questions for the given KB, we simulate KB incompleteness by dropping various KB elements systematically, and finally, we get an updated KB, with a set of answerable and unanswerable questions. With the given set of questions, we create train, dev, and test split with different categories of generalizations in the test set. We describe it more in the next section.

4.2 Dataset Details

We make use of GrailQA (Gu et al., 2021), which is one of the largest and most diverse benchmarks for only-answerable KBQA, and create a new benchmark for KBQA with answerability detection which we name as GrailQAbility.

© 2025, *Indian Institute of Technology Delhi*

4.2.1 Categories of Generalization: Definition

GrailQAbility's test data has two main categories of questions - answerable and unanswerable. Further, we define different generalization types for each category.

1. For answerable questions, there are three generalization categories - **IID, Compositional, Zero-Shot**. We adopt definitions of GrailQA for these categories. For a question q, let S_q denote the schema elements in the logical form for q, let Q_{tr} be training set, then

- a test question q is labeled **IID answerable** (Figure 4.1 Test: Example-1) if it is answerable and it follows the distribution for questions in Q_{tr} and contains only schema elements seen in train $S_q \subseteq S_{Q_{tr}}$ (we have overloaded notation to define $S_{Q_{tr}}$).
- Alternatively, a test question q is labeled **Zero-Shot answerable** (Figure 4.1 Test: Example-2) if it is answerable and involves at least one unseen schema element ($S_q \nsubseteq S_{Q_{tr}}$).
- Finally, a test question q is classified as **Compositional answerable** if it is answerable and $S_q \subseteq S_{Q_{tr}}$, but the specific logical form for q does not match any $q' \in Q_{tr}$. This means that the combination of KB elements in the logical form of the test question is different from those seen during training. For example, in Figure 4.1 Test: Example-3, all KB elements are individually present in the training set, but their combination is not.

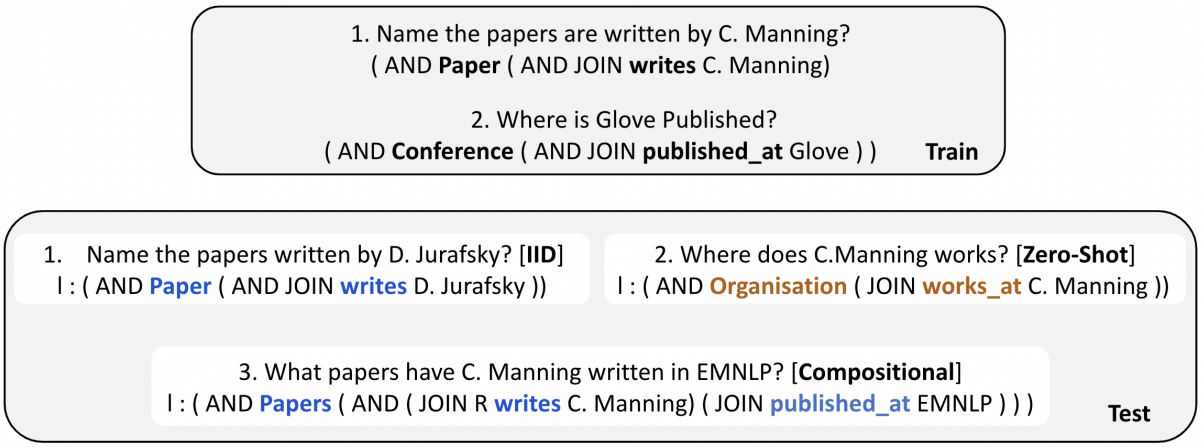

Figure 4.1: Illustration of different generalization categories for answerable questions. Here schema items in logical forms are marked in bold. In Test examples schema items seen in training are marked blue, and unseen schema items are marked red.

2. Mirroring definitions for answerable questions, we define two generalizations for unanswerable questions - **IID and Zero-Shot**.

- An **IID unanswerable** question in the test follows the same distribution as unanswerable questions in train i.e. all missing KB elements in the test set are encountered

in the unanswerable subset of the train set associated with the same category of incompleteness. For example, question 2 of Figure 4.2 (Train) is unanswerable because of missing entity type *Research Area*, so we mark question 1. of Test as *IID* because it is also unanswerable due to missing entity type *Research Area*.

- In contrast, a **Zero-Shot unanswerable** question involves at least one missing KB element (schema element in its ideal logical form or data element in its paths) that is not part of any unanswerable question in train associated with the same category of incompleteness. E.g., the missing schema elements (*located in* and *works at*) for the second and third test questions in Figure4.2 are not covered by any unanswerable question in the train set. We further categorize Zero-Shot unanswerable questions as partial-Zero-Shot and complete-Zero-Shot.
  - A Zero-Shot instance is classified as *partial Zero-Shot* if its ideal logical form includes at least one schema element observed in answerable questions during training. For instance, in Figure 4.2, the logical form for question 3 in the test dataset contains the relation *writes*, which is found in the training data, along with the missing element *R. Socher*, which is absent from the training set.
  - In contrast a Zero-Shot instance is defined as *full Zero-Shot* when it does not involve any schema element seen in logical forms of answerable questions in train. The second test question in Figure 4.2, involving the missing relation *located in* is an example.

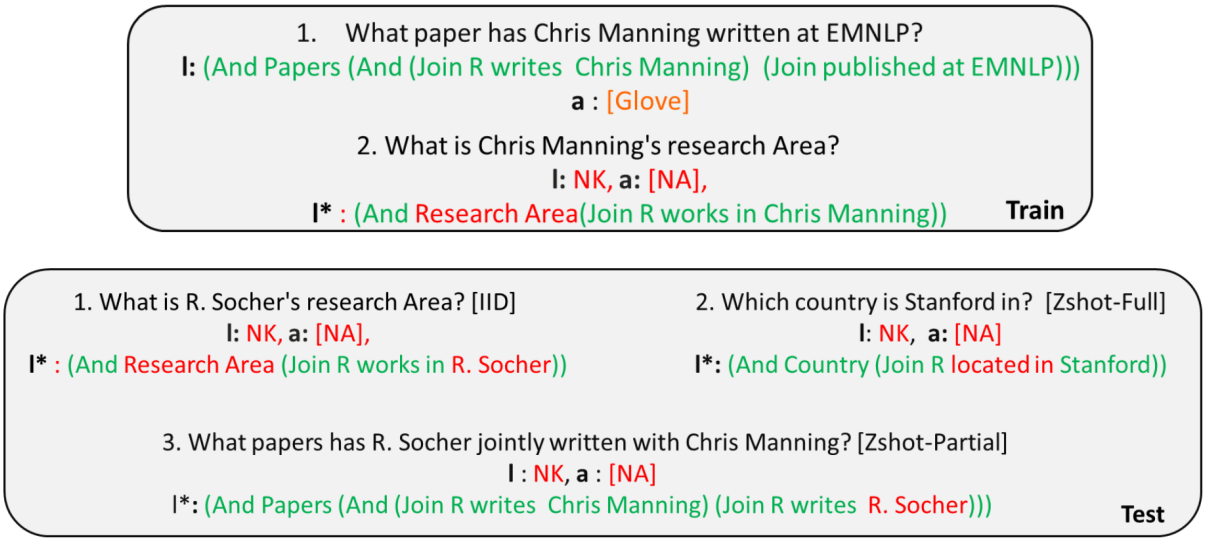

Figure 4.2: Illustration of different generalization categories for unanswerable questions.

Please note that we have not considered Compositional generalization for unanswerable questions. We may define a Compositional unanswerable question as one that contains more than one missing KB element in its ideal logical form or in its ideal paths, all of which have appeared in unanswerable training instances, but not all in the same instance. We hypothesize that detecting unanswerability in this scenario should only be as hard as for IID unanswerability. We plan to validate this experimentally in the future.

© 2025, *Indian Institute of Technology Delhi*

4.2.2 Dataset Analysis

Now we perform a quantitative analysis of GrailQAbility. GrailQA (Gu et al., 2021) (initial dataset for us) contains a total of 64,331 question-logical form pairs, divided into train, validation, and test sets. The test set of GrailQA is hidden, so we make use of the train and validation set to create GrailQAbility. Table 4.1 shows a comparison between GrailQA and GrailQAbility in terms of the number of questions, KB elements coverage, type of questions, and test scenarios.

Dataset	#Q	#LF	#D	#R	#T	#E	Q. Type		Test Scenarios	
							A	U	A	U
GrailQA	64,331	4969	86	3720	1534	32,585	✓	✗	I, C, Z	✗
GrailQAbility	50,507	4165	81	2289	1081	22,193	✓	✓	I, C, Z	I, Z

Table 4.1: Statistics for GrailQA and GrailQAbility. #Q is no. of questions, #LF no. of unique canonical logical forms, #D no. of domains, #R, #T, #E no. of relations, types and entities, A and U denote answerable and unanswerable questions. I, C, and Z denote IID, Compositional, and Zero-Shot.

The Compositional and Zero-Shot questions from the validation set of GrailQA are directly split into the validation and test set of GrailQAbility (answerable questions). Then using the algorithm described in section 4.1, we utilize train questions along with IID validation questions of GrailQA to create unanswerable (IID, partial-Zero-Shot and full-Zero-Shot) and answerable (IID) questions of GrailQAbility.

Split	A	U	
		NK	NA
Train	23,933	7110	4240
Dev	3399	1064	595
Test	6808	2162	1196

Table 4.2: Distribution of GrailQAbility questions across different train, dev, and test splits.

Split	IID	Compositional	Zero-Shot
Train	23,933	0	0
Dev	1691	488	1220
Test	3386	981	2441

Table 4.3: Distribution of GrailQAbility answerable questions across different generalization categories.

Table 4.2 shows train, dev, and test splits for GrailQAbility for both question categories. The test and validation set comprises around 20% and 10% of the data respectively. Aligning with earlier QA datasets with unanswerability (Rajpurkar et al., 2018; Sulem et al., 2022; Choi et al., 2018; Raina and Gales, 2022), we keep the total percentage of unanswerable questions as 33%. Table 4.3 presents details of answerable questions across multiple generalization categories. Zero-Shot and Compositional questions comprise of around 35% and 15% respectively, of the answerable dev and test questions.

© 2025, *Indian Institute of Technology Delhi*

Split	Type Drop			Relation Drop			Entity Drop				Fact Drop	
	IID	Zero-Shot		IID	Zero-Shot		IID		Zero-Shot		IID	Zero-Shot
	NK	P (NK)	F (NK)	NK	P (NK)	F (NK)	NA	NK	NA	NK	NA	NA
Train	2667	0	0	2780	0	0	1288	1663	0	0	2952	0
Dev	211	154	47	211	146	55	91	89	87	151	176	241
Test	422	330	96	425	298	107	178	193	179	291	352	487

Table 4.4: Statistics for unanswerable questions (P: Partial and F: Full) in GrailQA-bility corresponding to different types of KB incompleteness.

Table 4.4 shows data splits for unanswerable questions across various types of KB incompleteness and generalization categories. We equally split unanswerable questions across different KB incompleteness i.e. around 8.25% for each of the dropped entity types, relations, entities, and facts.

The diverse array of answerable and unanswerable question categories, along with their various generalizations, collectively present a challenging benchmarking dataset. In the following chapter, we assess the performance of existing KBQA methods on our dataset.

© 2025, *Indian Institute of Technology Delhi*

Chapter 5

GrailQAbility: Evaluation and Discussions

In this chapter, we evaluate the performance of existing state-of-the-art KBQA models on GrailQAbility. Our experimental analysis shows that while model adaptations help improve performance for unanswerable questions across all models, they still struggle with various categories of unanswerabilities, and no single model excels in handling both answerable and unanswerable questions effectively.

5.1 Baselines

Since GrailQAbility is based on GrailQA, so we select the top four semantic parsing-based published models (with available code) from the GrailQA leaderboard i.e. RnG-KBQA (Ye et al., 2022), ReTraCk (Chen et al., 2021), TIARA (Shu et al., 2022) and Pangu (Gu et al., 2023). These models report state-of-the-art results on GrailQA. We have discussed these methods in detail in Chapter 2.

All of the baselines span different variations in retrieval and logical form prediction approaches. At the same time, all models have two steps in common.

- **Entity Linking**: Here, the task is to detect the mentioned entity in the question, i.e., link the entity in the question to an entity in the KB. For our experiments, we use the same entity linker pipeline as RnG-KBQA (Ye et al., 2022) for all baselines.
- **Execution Guided Check**: Once the top-k logical forms have been generated, each logical form is translated into SPARQL and executed against the KB to obtain an answer. If the answer is empty, the subsequent logical form is selected. This iterative process continues until a logical form yielding a non-empty answer is obtained.

The model architectures were designed assuming that the input questions are always answerable. Therefore, these model architectures cannot handle unanswerable questions. For a fair analysis, we make appropriate modifications (as described in Sec 5.2) to the baselines so that the models have the ability to detect unanswerable questions.

This work is jointly done with Mayur Patidar and Avinash Singh from TCS Research. They contributed to the implementation and analysis of one baseline: RnG-KBQA.

5.2 Adapting for Answerability Detection

As discussed in previous chapters, we categorize unanswerability into two main groups: first, when a valid logical form does not exist, denoted as $l = NK$; and second, when a logical form exists but an answer does not, denoted as $a = NA$. For the latter category, we turn off the execution-guided check for all baselines, thus allowing models to accept logical forms that result in empty answers upon execution. For the former, we make two types of adaptations: training with unanswerability and thresholding. Both adaptations enable the models to predict l as NK.

5.2.1 Thresholding

We introduce thresholding (+**T**) for entity disambiguation and logical form generation. If either of the prediction scores is below the respective thresholds, we classify the prediction as NK. These thresholds are tuned using the validation set. Additionally, any empty predictions—retrieved schema elements, linked entities, or generated logical forms—imply that the logical form is NK.

5.2.2 Training with unanswerable questions

We train the models by including both the answerable and unanswerable questions in the training data (**A+U training**). For this training, we use existing code bases[1] [2] [3] [4] of these models and make appropriate code changes. The modifications and training details are mentioned below.

1. **ReTraCk** (Chen et al., 2021) includes three main components - retriever, transducer, and checker. The retriever consists of an entity linker that links entity mentions to corresponding entities in KB and a schema retriever that retrieves relevant schema items given a question.
   - For entity linking we use the ReTraCk's entity linker pre-trained on GrailQA. To adapt this for GrailQability we remove the dropped entities from the predictions of the entity linker.
   - The schema retriever leverages the dense retriever framework (Mazaré et al., 2018; Humeau et al., 2019; Wolf et al., 2020) for obtaining classes(types) and relations. Same as ReTraCk, we use a pre-trained BERT-base-uncased model as a schema retriever and fine-tune it on GrailQAbility for 10 epochs with a learning rate of

[1]https://github.com/salesforce/rng-kbqa
[2]https://github.com/microsoft/KC/tree/main/papers/ReTraCk
[3]https://github.com/microsoft/KC/tree/main/papers/TIARA
[4]https://github.com/dki-lab/Pangu

© 2025, *Indian Institute of Technology Delhi*

1e-5. The best model is selected on the basis of recall@top_k where top_k is 100 and 150 for types and relations respectively. We train two schema retriever models, one for A and one for A+U. For A, all answerable questions are used for training, while for A+U we use non-NK questions i.e. questions having only valid logical forms.

- Transducer modules consist of a question encoder and a grammar-based decoder. ReTraCk uses a set of grammar rules for logical form. For NA training we have added a new grammar rule i.e. num $\rightarrow$ NK where NK is a terminal symbol representing No Knowledge. So for a question with no logical form, the sequence of grammar rules will be @start@ $\rightarrow$ num and num $\rightarrow$ NK. We have trained the transducer model with updated grammar rules for GrailQAbility.
- Training settings and hyperparameters are the same as ReTraCk i.e. the BERT-base-uncased model with Adam optimizer and learning rate 1e-3, while the learning rate for BERT is set to 2e-5. The best model is selected on the basis of the average exact match calculated between the predicted logical form and the golden logical form.

2. **RnG-KBQA** (Ye et al., 2022) consists of four modules: Entity Linker, Entity Disambiguation, Ranker, and Generator. We use the same training objective and base models for re-training these components on GrailQAbility.
   - Similar to GrailQA (Gu et al., 2021), for mention detection, we fine-tune a BERT-base-uncased model for 3 epochs with a learning rate of 5e-5 and a batch size of 32. For training the Entity Disambiguator, similar to RnG-KBQA, we fine-tuned a BERT-base-uncased (Devlin et al., 2019) model for 3 epochs with a learning rate of 1e-3 and a batch size of 16.
   - During training with A+U, we train mention detection and entity disambiguation model with questions having valid logic form i.e., $l \neq$ NK, and perform only inference for questions where $l =$ NK.
   - We random sample negative logical forms during the training of the ranker and fine-tune a BERT-based-uncased model for 3 epochs with a learning rate of 1e-3 and a batch size of 2. During training with A+U, we train ranker for questions only with $l \neq$ NK similar to entity linker.
   - RnG-KBQA generator takes the (question, logical form) pair as input during training. For A+U trainingGenerator is trained to predict "no logical form" for unanswerable questions with $l =$ NK and valid logical form for remaining training questions.
   - Hyper-parameters for generator training are the same as proposed by the authors. We fine-tune T5-base (Raffel et al., 2020a) for 10 epochs with a learning rate of 3e-5 and a batch size of 8. During inference with the generator, similar to RnG-KBQA, we use a beam size of 10 but due to the presence of NA questions in the test we do not perform execution augmented inference.

3. **TIARA** (Shu et al., 2022) consists of four modules - Entity Linking, Schema Retriever, Exemplary Logical Form Retrieval, and Generator.
   - For fair comparison we have used the same Entity Linker as RnG-KBQA.

© 2025, *Indian Institute of Technology Delhi*

- Schema retrieval is implemented by a cross-encoder using a pre-trained BERT-base-uncased model. The model is trained for 10 epochs and the best model is selected on the basis of recall@top_k where k is 10 for both relations and classes. To train schema-retriever for A+U model we use questions with valid logical forms.
- The Logical Form Retrieval includes enumeration and ranking. It follows the same methods as proposed by Gu et al. (2021) and Ye et al. (2022). So training process for A+U model and hyper-parameters for this module are same as described in the RnG-KBQA section above.
- Generation is performed by a transformer-based seq2seq model - T5(base) (Raffel et al., 2020a). The Generator is fine-tuned for 10 epochs with a learning rate of 3e-5 and batch size of 8. For A+U model we use all answerable and unanswerable questions for training. For unanswerable questions, the model is trained to generate output as "no logical form". For inference, we perform constraint decoding with a beam search of 10 same as proposed by the authors.

4. **Pangu** (Gu et al., 2023) consist of four modules - Entity Linking, Candidate Plan Enumerator, and LM-based scorer.
   - For fair comparison we have used the same Entity Linker as RnG-KBQA.
   - Next we modify Candidate Plan Enumerator to add "none" to candidate logical forms at the first stage. Further, we also modify the LM scorer to accept "none" as logical form and assign an appropriate score to it.
   - We fine tune T5-base model for 10 epochs with learning rate 1e-4 and batch-size of 8. During inference, a beam size of 5 is used (same as proposed by authors).

5.3 Evaluation Metrics

We evaluate the model's performance on the following metrics:

- As the task fundamentally revolves around question answering, we conduct direct answer evaluation. Since a question may inherently have multiple correct answers, we assess predicted answers using the F1 score.

  For questions with non-empty answers, we use the standard F1 metric, where the gold answer is the modified answer based on updated KB. In cases of unanswerable questions where the gold answer is an empty set, computing the **F1** score resembles accuracy, where the score is 1 if the model correctly predicts an empty answer set; otherwise, it is 0. We denote this standard F1 score as **F1(R)**.

  Given our focus on unanswerability, it is also crucial to consider the scores for the original answer (based on the non-updated KG). This is because the models might still find the original answer through a different path, and this behavior should be rewarded. Hence, we also incorporate a lenient answer evaluation, **F1(L)**, which acknowledges model's capability of recovering this answer, possibly through inference. For example, for the fourth question in Figure 3.2(B), F1(R)-evaluation only rewards NA as the answer, whereas F1(L)-evaluation rewards both NA and *{USA}* as perfect answers.

© 2025, *Indian Institute of Technology Delhi*

- However, the correctness of the answer does not necessarily imply the correctness of the logical form, particularly for unanswerable questions where many different incorrect logical forms can result in an empty answer set. Therefore, we also assess the predicted logical form to determine if the models arrive at the correct answer through correct reasoning. We compare the predicted logical form with the gold-standard one using **exact match (EM)** (Gu et al., 2021), which computes semantic equivalence between the predicted and gold logical forms. To determine semantic equivalence, first logical forms are converted to query graphs, and then the similarity is computed by graph isomorphism.

5.4 Results and Discussions

We train the models according to their original settings using two different paradigms. The first paradigm involves training with only the answerable subset of questions, excluding the unanswerable ones (**A training**). The second paradigm includes both answerable and unanswerable questions in the training data (**A+U training**). For both settings, we have two versions of the models: one without thresholding and one with thresholding (+**T**).

Train	Model	Overall			Answerable			Unanswerable		
		F1(L)	F1(R)	EM	F1(L)	F1(R)	EM	F1(L)	F1(R)	EM
A	ReTraCk	69.20	67.00	50.70	67.00	66.90	62.40	73.80	67.20	27.10
	ReTraCk+T	69.90	67.90	52.00	65.30	65.30	61.20	79.30	73.20	33.40
	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**
A+U	ReTraCk	69.70	68.40	56.50	61.40	61.30	57.30	86.50	82.80	54.70
	ReTraCk+T	70.30	69.10	56.60	61.20	61.10	57.10	88.70	85.10	55.50
	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	92.40	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

Table 5.1: Performance of different models on GrailQAbility overall, answerable and unanswerable questions. EM is an exact match on logical forms and F1(L) and F1(R) are lenient and regular evaluations of answers. A and A+U indicate training with only answerable questions and with both answerable and unanswerable questions. Models with suffix +T have additional thresholds for entity disambiguation and logical form fine-tuned on the dev set.

Table 5.1 presents the **overall performance** of the baselines on the GrailQAbility test

set for two training configurations. The values in bold represent the highest scores within individual training settings. Here are the key observations:

- Both A+U training and thresholding improve performance on unanswerable questions. A+U training helps models predict "NK" when it is impossible to generate a valid logical form based on the input context and question. Thresholding helps predict "NK" when the model isn't confident enough in its predictions. Further, it can be seen that discriminative style model like Pangu benefits more from thresholding, while generation-based models benefit more from A+U training.

- When training with only answerable questions (A training), all models perform poorly on unanswerable questions in terms of EM, with Pangu+T being the best among all. We discuss the reasons for this in more detail later in the chapter.

- Not surprisingly, improvement for unanswerable questions comes at the expense of answerable question performance for both settings. For example, the performance of the best model on answerable questions (TIARA) drops by around 7 percentage points after A+U training. Furthermore, it can be observed that the answerable performance is also affected by thresholding. This is because, for both forms of adaptation, the models incorrectly predict l = NK for answerable questions.

- The difference in performance between answerable and unanswerable questions varies across models - it is approximately 1% for RnG-KBQA+T and TIARA+T, and approximately 15% for Pangu+T. Also, there is a very large gap between EM and F1(R) for unanswerable questions. This is because correct NA (no answer) predictions are often linked to spurious logical form predictions in all three models.

- Performance is better (by about 2-4 percentage points) with lenient answer evaluation than with the regular counterpart. We found that this is often because the models generate logical forms with schema elements similar to the dropped ones and return subsets or supersets of the old answer instead of NA. For example, consider the question: *What are papers written by C. Manning at EMNLP?* The original answer is "Glove." Now, if the entity EMNLP is missing in the updated KB, the model should ideally predict $l = NK$ and $a = NA$. However, the model incorrectly predicts the logical form for a different question: *What are papers written by C. Manning?* In such cases, F1(L) will be greater than F1(R).

In Table 5.2, we break down **performance for unanswerable questions according to different forms of KB incompleteness**. Note that we have further decomposed entity drop into two categories: the drop of mentioned entities, which affects the logical form, and the drop of other entities in the path, which affects only the answer paths. Unanswerable questions with l = NK and l = NA fall under missing schema elements, while questions with only a = NA fall under missing data elements. The following are the main takeaways.

- A+U training significantly enhances performance for missing schema elements. This is primarily because schema-level patterns are easy to learn because number of schema

© 2025, *Indian Institute of Technology Delhi*

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	ReTraCk	71.10	34.80	59.30	18.90	80.70	63.70	72.60	11.20	64.40	11.90
	ReTraCk+T	75.70	47.90	64.90	28.80	83.50	70.30	81.00	10.90	72.30	12.00
	RnG-KBQA	40.10	0.00	44.20	0.00	27.40	0.0	45.10	13.50‘	46.00	16.80
	RnG-KBQA+T	55.50	49.50	57.10	46.60	44.70	40.30	56.00	11.50	58.60	13.90
	TIARA	59.55	0.71	58.07	0.00	50.63	0.00	64.15	**23.53**	62.71	**26.01**
	TIARA+T	66.27	21.70	70.21	28.06	61.01	23.43	68.91	22.97	68.29	23.63
	Pangu	24.06	19.46	21.13	16.14	41.67	38.84	12.89	0.00	13.06	0.95
	Pangu+T	**87.97**	**87.50**	**80.07**	**79.63**	**90.57**	**90.41**	**83.19**	0.00	**76.48**	1.07
A+U	ReTraCk	89.60	82.20	86.40	74.40	90.30	85.90	79.00	9.80	71.70	10.80
	ReTraCk+T	90.60	83.10	87.80	76.00	91.20	86.80	83.20	9.80	76.40	10.80
	RnG-KBQA	91.60	75.81	86.40	66.60	87.6	72.00	84.00	37.51	82.40	39.10
	RnG-KBQA+T	**93.40**	86.80	89.70	**85.50**	92.10	89.60	87.10	30.80	86.00	32.50
	TIARA	90.09	76.53	88.95	62.62	92.14	63.68	91.32	**55.18**	90.14	**54.99**
	TIARA+T	91.63	83.84	**90.90**	72.37	**94.50**	71.38	**91.60**	50.42	**90.38**	52.85
	Pangu	25.24	17.57	19.72	13.65	42.45	37.42	11.76	0.00	12.35	0.71
	Pangu+T	90.80	**90.68**	78.66	78.44	90.41	**90.25**	74.51	0.00	69.71	0.95

Table 5.2: Performance of different models for the unanswerable questions in GrailQAbility, grouped by categories of KB incompleteness. Note that missing mentioned entities result in an invalid logical form, while other missing entities lead to a valid logical form with no answer.

elements is relatively low in count. Additionally, thresholding also improves performance for the missing schema elements category. It is noteworthy that Pangu experiences the most significant benefits from thresholding. We speculate that this might be attributed to the prediction scores of logical forms being more calibrated for discriminative models compared to generation-based models such as ReTraCk, RnG-KBQA, and TIARA.

- For all models, the performance (EM) is notably worse for all forms of missing data elements compared to missing schema elements. This is true even after applying thresholding and retraining. Among the models, TIARA demonstrates the highest EM performance for missing data elements with A+U training. The category of data elements comprises facts or intermediate entities, where one or more paths are disrupted to arrive at an answer. Consequently, methods reliant solely on retrieved data paths for generating logical forms (path traversal-based retrieval), such as RnG-KBQA or Pangu, encounter challenges when relevant data elements are absent. In such scenarios, these models fail to retrieve any recognizable input patterns and predict $l = \text{NK}$. Notably, TIARA serves as an exception by also retrieving schema elements alongside data paths. Thus, it learns to generate correct logical forms using the retrieved schema elements even when data elements are missing.

- The disparity between EM and F1(R) is minimal for missing schema elements ($l = \text{NK}$) but significantly pronounced for missing data elements ($l \neq \text{NK}$). This discrepancy arises from models predicting either spurious logical forms with empty answers or incorrectly marking the predicted logical form as $l = \text{NK}$ through thresholding, resulting

© 2025, *Indian Institute of Technology Delhi*

in empty answers.

- Our observation reveals that there is not a single model that excels in both categories of unanswerabilities. Integrating various retrieval mechanisms enhances performance for the missing data element category, but introduces confusion for questions related to missing schema elements. On the other hand, thresholding aids in enhancing performance for the missing schema element category, it adversely affects performance for missing data elements.

Model	IID		Zero-Shot	
	F1(R)	EM	F1(R)	EM
ReTraCk	88.7	66.5	77.7	44.4
ReTraCk+T	90.1	66.6	80.7	45.7
RnG-KBQA	91.9	73.3	81.7	47.1
RnG-KBQA+T	94.3	75.9	85.9	**59.5**
TIARA	93.8	75.2	86.4	50.8
TIARA+T	**95.1**	**77.8**	**87.9**	56.9
Pangu	21.4	12.2	22.3	15.3
Pangu+T	80.5	57.5	80.2	52.9

Table 5.3: Performance of different models for unanswerable IID and zero-shot test scenarios in GrailQAbility (trained with A+U).

Model	Full Z-Shot		Partial Z-Shot	
	F1(R)	EM	F1(R)	EM
ReTraCk	86.2	65.0	73.6	54.5
ReTraCk+T	88.2	67.0	75.6	56.7
RnG-KBQA	87.2	75.9	78.0	40.0
RnG-KBQA+T	89.7	86.7	**83.1**	71.0
TIARA	90.2	68.9	80.3	40.4
TIARA+T	**90.6**	78.8	82.6	54.1
Pangu	24.63	20.69	21.2	15.76
Pangu+T	89.7	**89.7**	79.9	**79.5**

Table 5.4: Performance of different models for partial zero-shot and full-zero test scenarios in GrailQAbility (trained with A+U).

Next, we study the **performance across various categories of generalization**. We analyze this separately for answerable and unanswerable subsets.

- **Unanswerable Questions**. Recall that we introduced two generalization levels for unanswerable questions: IID and Zero-Shot. Note that the definitions of IID and Zero-Shot utilize unanswerable training instances, so only A+U training makes sense for this comparison. The decomposition of unanswerable performance in terms of IID and Zero-Shot subsets is shown in Table 5.3. The following are the main takeaways.
  - As expected, IID performance surpasses Zero-Shot for all models, with TIARA+T demonstrating the best performance in the IID scenario and RnG-KBQA+T excelling in Zero-Shot scenarios. This disparity arises because the schema elements retrieved by TIARA often lead to confusion in the model, resulting in incorrect logical forms. This effect is particularly pronounced for zero-shot questions where the schema elements were not encountered during training.
  - However, a deeper analysis of Zero-Shot instances(Table 5.4) reveals more interesting insights. We observe that all models find answering full Zero-Shot questions considerably easier compared to partial Zero-Shot questions. Specifically, for Pangu+T, which is the top-performing model based on EM, there is approximately a 10 percentage point difference in EM scores between the two types. The reason behind this difference is that partial Zero-Shot questions include some KB elements that were encountered during training in answerable contexts, alongside

© 2025, *Indian Institute of Technology Delhi*

Zero-Shot KB elements that were not encountered during training. This mixture confuses the models, leading them to incorrectly label such questions as answerable. In contrast, full Zero-Shot instances bear no resemblance to any training answerable questions, making them less confusing for the models to classify as unanswerable correctly.

Train	Model	IID			Compositional			Zero-Shot		
		F1(L)	F1(R)	EM	F1(L)	F1(R)	EM	F1(L)	F1(R)	EM
A	ReTraCk	79.60	79.50	75.60	63.10	63.10	55.40	51.00	51.00	46.80
	ReTraCk+T	79.00	78.90	75.20	61.60	61.60	53.90	47.80	47.80	44.50
	RnG-KBQA	85.50	85.40	83.20	65.90	65.90	60.20	72.70	72.70	67.30
	RnG-KBQA+T	79.00	79.00	77.30	58.80	58.80	54.50	65.80	65.80	61.90
	TIARA	**86.53**	**86.47**	**84.52**	**72.02**	**72.02**	64.93	74.24	74.24	67.60
	TIARA+T	83.76	83.71	82.01	65.68	65.68	60.86	64.02	64.02	58.58
	Pangu	82.00	81.97	79.09	71.63	71.63	**65.95**	**77.02**	**77.02**	**70.18**
	Pangu+T	81.15	81.14	78.88	68.69	68.69	64.73	70.46	70.46	66.41
A+U	ReTraCk	77.80	77.60	73.90	60.60	60.60	53.50	39.00	39.00	35.80
	ReTraCk+T	77.70	77.50	73.80	59.90	59.90	52.8	38.90	38.90	35.70
	RnG-KBQA	**85.40**	**85.30**	**83.30**	65.80	65.80	60.80	66.90	66.90	62.60
	RnG-KBQA+T	80.90	80.90	79.20	60.50	60.50	56.10	61.10	61.10	57.60
	TIARA	82.38	82.36	80.57	65.16	65.16	59.84	58.50	58.50	54.65
	TIARA+T	81.86	81.84	80.09	64.23	64.23	59.02	55.56	55.56	51.99
	Pangu	81.08	81.01	76.85	**77.43**	**77.43**	**69.52**	**78.01**	**78.01**	**70.42**
	Pangu+T	81.07	81.00	75.72	77.43	77.43	69.22	**78.00**	**78.00**	66.90

Table 5.5: 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 5.1.

- **Answerable Questions**. The breakdown of performance on answerable questions for IID, Compositional, and Zero-Shot subsets is shown in Table 5.5. The following are the main takeaways.
  - All models find IID easier than the other two generalizations, which is intuitive.
  - Pangu performs best for Compositional questions because it is trained to discriminate between incorrect, partially correct, and correct logical forms.
  - Interestingly, A+U training affects different models in different ways. For generation-based models like ReTraCk, RnG-KBQA, and TIARA, the performance on Zero-Shot questions drops significantly. This happens because of lack of generalization in generation-based models which overfits A+U training and often confuses Zero-Shot questions with $l = NK$ type of unanswerable questions. While Pangu has a better generalization ability and can effectively differentiate between unanswerable and Zero-Shot questions.
  - Thresholding also affects Zero-Shot the most compared to other generalizations. This is because the models are better calibrated (or confident) for the data seen during training and generally confuse Zero-Shot answerable questions as unanswerable questions.

© 2025, *Indian Institute of Technology Delhi*

In summary, we observe that no single model excels equally in handling both answerable and unanswerable questions simultaneously. Even with appropriate retraining and thresholding, the performance gap between these two categories remains significant across models, although both adaptation methods contribute substantially to performance enhancement. Additionally, we explored several significant trade-offs: robustness to missing schema elements vs. missing data elements, performance of answerable questions for A vs. A+U training, and trade-offs across different generalizations following A+U training and thresholding.

© 2025, *Indian Institute of Technology Delhi*

Chapter 6

RetinaQA: Robust KBQA model for answerability detection

In the previous chapter, we analyzed the performance of existing state-of-the-art KBQA models on GrailQAbility. We delved into how various architectural choices impact distinct categories of unanswerabilities. Here we outline some key insights:

- First, effective model calibration plays a pivotal role in distinguishing between answerable questions (possessing a valid logical form) and those deemed unanswerable due to missing schema elements (lacking a valid logical form). Also, the quality of calibration directly influences the efficacy of thresholding.
- Furthermore, while path traversal-based retrieval works well for finding candidate logical forms for answerable questions, it fails when relevant data elements are missing from the KB, even if a valid logical form exists. Detecting this type of unanswerability requires a different approach to logical form construction that does not rely on path traversal.
- When generating logical forms, it is crucial to include constraints from the KB. This helps to avoid errors by eliminating invalid logical forms that are syntactically correct but semantically incorrect. However, identifying data-level constraints can affect the performance of unanswerable questions. Therefore, using a set of robust constraints is crucial.

Based on these insights, we propose a new multi-staged **RETrIeve**, ge**N**erate, and r**A**nk model for KBQA that is robust against unanswerability, named **RetinaQA**. For better calibration, instead of *generating* logical forms as the final stage, RetinaQA *discriminates* between candidate logical forms. To construct candidate logical forms, RetinaQA complements KB-traversal-based retrieval with sketch-filling-based construction, which generates KB-independent sketches and then grounds these by directly retrieving schema elements relevant to the question. This enables the identification of logical forms for questions with missing data elements and therefore no connected KB path. We then perform schema-level constraint checks to ensure that the logical form candidates are syntactically and semantically valid for easier discrimination.

In the upcoming sections, we will discuss the model architecture in detail, evaluate the performance of RetinaQA on GrailQAbility, and present a detailed ablation study and error analysis of the proposed model.

6.1 RetinaQA Model

Figure 6.1 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 **KB-traversal based retrieval**. Starting from linked entities in the question, RetinaQA traverses *data-level KB paths* and transforms these into 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 any data-level information. Once candidate logical forms are identified, RetinaQA uses *discriminative* scoring to rank these logical forms with respect to the question.

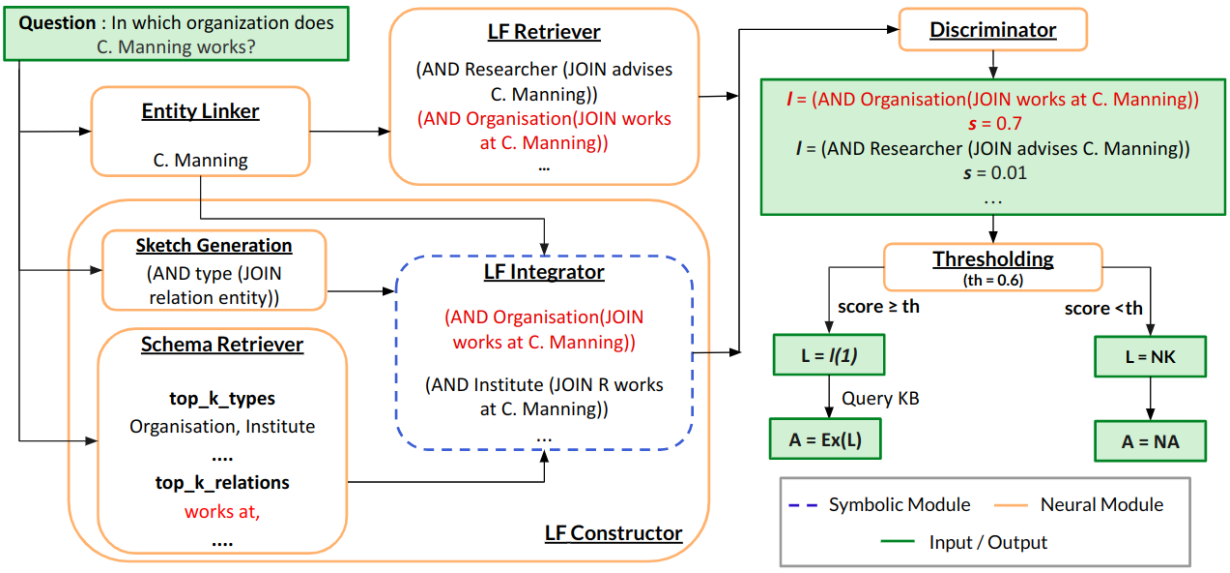

Figure 6.1: RetinaQA Architecture showing different components illustrated with an example question. Symbols l and s represent the candidate logical form and its score as Discriminator output, L the output logical form, A the final answer, $l(1)$ the top-ranked logical form, Ex(l) the answer obtained 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.

We next explain each of these components in more detail.

© 2025, *Indian Institute of Technology Delhi*

- **Entity Linker**.

  The pipeline starts with linking the 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 similar to all baselines (as described in Sec 5) If the mentioned entities are missing in the KB, the entity linker returns an empty set.

- **Logical Form Reteriver**.

  As the first approach for enumerating logical forms, RetinaQA uses KB data path traversal similar to Ye et al. (2022). It traverses 2-hop paths starting from the linked entities and transforms these to logical forms in s-expression. Let L be the set of all candidate logical forms. 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 Figure 6.1. Following Ye et al. (2022), we score a logical form l and question q as:

  $$s(l, q) = \textsc{Linear}(\textsc{BertCls}([l; q])) \tag{6.1}$$

  and optimize a contrastive objective for ranking:

  $$\mathcal{L}_{ret} = -\frac{\exp(s(l^*, q))}{\exp(s(l^*, q)) + \sum_{l \in L \wedge l \neq l^*} \exp(s(l, q))} \tag{6.2}$$

  where l^* is the gold-standard logical form for q.

  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 this enumeration approach is not guaranteed to cover all logical forms. Also, this approach cannot retrieve the logical form when the relevant data path in the KB is broken. For example in Figure 6.1, LF Retriever cannot enumerate the logical form in red when the fact *(C. Manning, works at, Stanford)* is missing.

- **Logical Form Constructor**.

  The second approach used by RetinaQA for logical form enumeration is sketch-filling. Drawing inspiration from transfer approaches for KBQA (Cao et al., 2022b; Ravishankar et al., 2022; Li et al., 2023), RetinaQA utilizes logical form sketches. These sketches capture the KB-independent syntax of s-expressions, including functions, operators, and literals, and replace KB-specific elements such as entities, entity types, and relations with arguments. RetinaQA first generates sketches using a **sketch generator**. In parallel, it retrieves relevant schema elements as candidates for arguments using a **schema retriever**. Finally, it fills in arguments for each candidate sketch with the retrieved argument candidates in all possible valid ways using a **logical form integrator**. Since this style bypasses path-based KB retrieval, it can construct valid logical forms even if some relevant data elements for the question are missing in the KB. This allows for the creation of accurate logical forms whenever they exist. For example even though the data element *(C. Manning, works at, Stanford)* is missing in the KB, the relation *works at* is present in the KB schema and LF Constructor adds the correct logical form to the pool of candidate logical forms.

  – **Sketch Generator:** The sketch generator takes the question q as input and

© 2025, *Indian Institute of Technology Delhi*

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-base (Raffel et al., 2020b) with a batch size of 8 and learning rate $3e-5$ for 10 epochs. Since T5 is an encoder-decoder framework, it often hallucinates when generating sketches. Therefore, we implement constraint decoding, where it checks the syntactic correctness of the generated sketches. The constraints include checks for correctly balanced brackets, proper placement of arguments, and the use of permitted operators and functions. During inference, we perform constraint decoding with beam search, using a beam size of 10. This step is unaffected by any KB incompleteness.

- **Schema Retriever:** To retrieve candidate arguments for generated sketches, we follow the schema retriever pipeline of TIARA (Shu et al., 2022). 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 Figure 6.1, this step is affected 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 linked entities to construct complete logical form candidates.

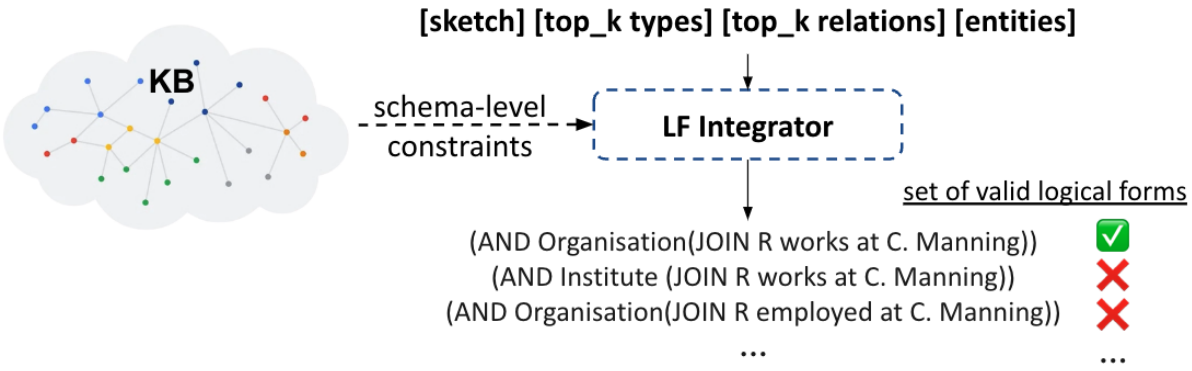

Figure 6.2: LF Integrator.

Each candidate sketch is grounded using every possible combination of arguments - retrieved types, relations, and entities. A symbolic checker ensures schema-level validity of the grounded logical forms for the KB. Here we first convert the logical form into a graph query and check if the graph follows the schema of KB. For example, the second candidate logical form in Figure 6.2 is invalid because according to KB schema, an entity *C. Manning* of type *Researcher* cannot be linked to type *Institute* by relation *works at*. Symbolic check 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**.

Finally, this component takes the union of logical form candidates from both the retriever and constructor components. It then scores and ranks these candidates'

logical forms. Following Zhuang et al. (2023), we fine-tune a T5 encoder-decoder model to score a (question, logical form) pair. Here decoding is done only for one step and the decoding probability of a special token is used as the ranking score.[1]

This component uses a contrastive learning-based optimization objective similar to Eqn.6.2. We perform random negative sampling, a our discriminator does not need bootstrapping with hard-negatives separately. Because the set of negative candidates is very small, so for most of the questions negative samples cover the entire negative candidate set. We fine-tune t5-base with a batch size of 4, 64 negative samples, and a learning rate of 1e-4 for 10 epochs.

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 =NK. Otherwise, the top-ranked candidate is predicted as the logical form and is executed over KB to fetch the answer.

We observe that such a discriminator is able to separate valid and invalid logical forms for a question. It helps identify questions for which valid logical forms do not exist due to missing schema elements. For example, in Figure 6.1, if the relation *works at* is missing from the KB, the logical form in red under the discriminator box will be absent from the candidate list. Consequently, the discriminator assigns a low score to the other candidate's logical forms. So the final output after thresholding is l =NK.

6.2 Experimental Evaluation

In this section, we try to address the following research questions:

1. How does RetinaQA compare against baselines on GrailQAbility in two different train settings: A and A+U?
2. How does RetinaQA perform for different categories of unanswerable questions?
3. How does RetinaQA perform across different generalization categories?
4. How does RetinaQA compare against baselines on other datasets with only answerable questions?
5. What are the individual contributions of various model components towards the performance of RetinaQA in (1),(2), (3) and (4) above?

6.2.1 Results for KBQA with Unanswerability

We first address the research question (1). In Table 6.1, we report the high-level performance of different models on GrailQAbility. With A+U training, RetinaQA+T outperforms all

[1]We use $< extra_id_6 >$ token of T5 for tuning ranking score.

© 2025, *Indian Institute of Technology Delhi*

models overall and is about 9 pct points ahead of the closest competitor (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.

The findings from the A training are also interesting. Ideally, we aim for models to perform well on unanswerable questions solely with A training, as this is a more practical scenario. In the A training setting, RetinaQA achieves comparable overall performance to the A+U setting, with only a 4 percentage point gap in EM. Other trends closely mirror those observed in the A+U setting.

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	78.70	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	**83.3**	**82.18**	**73.76**	77.74	77.74	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	84.46	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	**83.3**	**82.69**	**77.45**	77.91	77.91	75.16	**94.21**	**92.38**	**82.10**

Table 6.1: 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, and +T indicates thresholding.

Next, we address the research question (2). Table 6.2 drills down on performance for different categories of unanswerability. First, we observe that for the baselines, performance varies significantly across different categories. None of the baselines is equally good for both missing data elements and missing schema elements. However RetinaQA performs the best by a large margin for questions with missing data elements, and comparably with Pangu for missing schema elements, making it the overall model of choice across different categories of unanswerability.

Also, we observe a significant disparity between F1(R) and EM scores for the baselines.

© 2025, *Indian Institute of Technology Delhi*

In contrast, RetinaQA shows a much smaller gap, suggesting that it excels in reasoning by consistently providing correct answers alongside the correct intermediate logical form.

We also note that RetinaQA with thresholding results in minimal or no loss for questions with missing data elements (which have valid logical forms), and in huge gains for questions with missing schema elements.

Train	Model	Missing schema elements						Missing data elements			
		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	21.7	70.21	28.06	61.01	23.43	68.91	22.97	68.29	23.63
	Pangu+T	**87.97**	**87.5**	**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	**90.9**	72.37	**94.5**	71.38	91.6	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**	**94.3**	**76.13**

Table 6.2: Performance of different models for the unanswerable questions in GrailQAbility, grouped by categories of KB incompleteness. Note that missing mentioned entities result in an invalid logical form, while other missing entities lead to a valid logical form with no answer.

Next, we address the research question (3). We see that RetinaQA largely outperforms existing models across different generalization categories for answerable (Table 6.3) and unanswerable questions (Table 6.4 and Table 6.5).

For answerable questions in A setting, RetinaQA beats the previous best results for IID and compositional generalization, but for zero-shot generalization, RetinaQA has a slightly worse performance than Pangu. This happens because Pangu utilizes path-based traversal, which helps it eliminate invalid logical forms. This reduces the chances of model confusion, particularly in zero-shot scenarios where the model encounters unseen KB elements. On the other hand, RetinaQA relies on schema-level constraints, making it more likely to generate logical forms that resemble each other visually. This tendency can confuse the discriminator, especially when handling zero-shot questions.

For unanswerable questions (Table 6.4), 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.

Additionally, In Table 6.5, 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

© 2025, *Indian Institute of Technology Delhi*

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 6.3: 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 6.1.

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 6.2. As seen earlier, here too all the baseline models have huge variations in performance across full Zero-Shot and partial Zero-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	IID		Zero-Shot	
	F1(R)	EM	F1(R)	EM
RnG-KBQA+T	94.3	75.9	85.9	**59.5**
Tiara+T	**95.1**	**77.8**	**87.9**	56.8
Pangu+T	80.5	57.5	80.1	52.8
RetinaQA+T	**97.0**	**89.9**	**88.3**	**75.2**

Table 6.4: Performance of different models for unanswerable IID and zero-shot test scenarios in GrailQAbility (trained with A+U).

Model	Full Z-Shot		Partial Z-Shot	
	F1(R)	EM	F1(R)	EM
RnG-KBQA+T	89.7	86.7	**83.1**	71.0
Tiara+T	**90.6**	78.8	82.6	54.1
Pangu+T	89.6	**89.6**	79.9	**79.4**
RetinaQA+T	88.6	77.8	80.8	70.5

Table 6.5: Performance of different models for partial zero-shot and full-zero test scenarios in GrailQAbility (trained with A+U).

6.2.2 Results for KBQA with Answerable questions

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 (4), by evaluating it for traditional KBQA benchmarks with only answerable questions. Since all questions are answerable in this setting,

© 2025, *Indian Institute of Technology Delhi*

we re-introduce Execution Guided Check (EGC) as the final step for all models including RetinaQA. We find (Table 6.6) 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 similar to 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.

Model	Overall		IID		Compositional		Zero-Shot	
	F1	EM	F1	EM	F1	EM	F1	EM
RnG-KBQA	85.50	83.20	85.50	83.20	65.90	60.20	72.70	67.30
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 6.6: Performance of different models on GrailQA (validation set) for IID, compositional, and zero-shot test scenarios. RetinaQA' denotes RetinaQA with EGC

In Table 6.7, 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.

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

Table 6.7: 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

6.2.3 Ablation Study

Finally, we address the research question (5). Here we assess the 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). Note that if we remove the schema retriever (SR), then the sketch generator (SG) cannot contribute complete logical forms, hence those components have to be ablated together.

The last three rows of Table 6.8 show that at the aggregate level, all components contribute towards RetinaQA's performance on GrailQAbility to different extents for answerable and unanswerable questions.

© 2025, *Indian Institute of Technology Delhi*

Train	Model	Overall		Answerable		Unanswerable	
		F1(R)	EM	F1(R)	EM	F1(R)	EM
A+U	RetinaQA+T	**82.69**	**77.45**	**77.91**	**75.16**	**92.38**	**82.10**
	RetinaQA+T - LFR	76.37	65.37	73.39	70.90	82.43	54.17
	RetinaQA+T - LFI	73.53	53.89	70.85	68.07	78.95	25.13
	RetinaQA+T - (SG ∪ SR)	62.58	52.46	72.95	68.13	41.54	20.70

Table 6.8: 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).

Next, we drill down into specific question categories. Since training the discriminator and computing thresholds for every ablation is expensive, so for stages before the discriminator instead of studying performance on EM we study the *recall* of the correct logical form within the candidate set. Recall can be considered as an upper bound of the performance of the discriminator.

First, we study the recall of the correct logical form within the candidate set for unanswerable questions with missing data elements (Table 6.9). We see that if we remove SR and SG, there is a significant 65 pct points decrease in recall. In contrast, removing LFR has little impact. This is because LFR relies on path traversal to retrieve candidate logical forms. This agrees with our intuition that when essential data elements are missing, we need retrieval techniques that are independent of path traversal.

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	10.00	14.11

Table 6.9: 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.

Next, we do a similar study of recall for answerable questions (Table 6.10). 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. We use EGC only when datasets assume answerability so we perform this ablation study on GrailQA (Table 6.11). We can see that

© 2025, *Indian Institute of Technology Delhi*

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 6.10: Ablation experiment of Logical Form Coverage(%) on GrailQAbility test set for Answerable questions. LFR refers to Logical Form Retriever, SP refers to Sketch Parser and SR refers to Schema Retriever.

both LFI and EGC reduce the space of logical form candidates for the discriminator, by pruning out invalid logical forms. 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 drop. This shows that these components can compensate for each other, but RetinaQA needs at least one of them for good performance.

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	68.10	79.03	73.58
RetinaQA' - LFI	78.65	73.10	88.10	84.81	75.00	67.31	76.04	70.40
RetinaQA' - LFR	71.8	68.33	87.33	85.56	69.00	63.47	66.19	62.83
RetinaQA' - (SG ∪ SR)	73.20	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 6.11: Ablation experiment on GrailQA dev set. EGC refers to Execution Guided Check and LFI refers to Logical Form Integrator, RetinaQA' = RetinaQA + EGC

6.2.4 Error Analysis

We have done a detailed error analysis for RetinaQA on GrailQAbility. For this, we used the best version RetinaQA, which is RetinaQA+T with A+U training. We found three main error categories:

- *Thresholding error*, where, due to thresholding, RetinaQA incorrectly predicts NK for a question with a valid logical form;
- *Reranking error*, where the discriminator makes a mistake in scoring, though the candidates contain the correct logical form, and
- *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.

Table 6.12 presents a component-wise error analysis on the answerable subset of GrailQAbility. Unsurprisingly, the majority of errors across all categories occur for the zero-shot

© 2025, *Indian Institute of Technology Delhi*

generalization questions. Thresholding and reranking errors occur for around 38%, and 30% of the total errors respectively. The most frequent error is recall error (70%). Among these, entity linking errors occur 80% of the time. This clearly suggests that improving the entity linker can significantly enhance the overall performance of KBQA models. Unsurprisingly, the majority of the errors across categories occur for the zero-shot generalization questions.

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 6.12: 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 6.13: Component wise errors of RetinaQA+T (A+U) for data element missing unanswerable questions . Names have the same meanings as in Table 6.12.

For unanswerable questions, we first look at those with missing data elements (Table 6.13). 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 account for 5% of total error. This suggests that the discriminator is calibrated well for the relative ranking of logical forms but still makes mistakes in assigning correct absolute scores to logical forms.

Finally, we look at the subset of unanswerable questions with missing schema elements. Since for these, the gold logical form is NK, thresholding error can be the only source of error. This occurs only 14% of the time, out of which 90% of errors occur for zero-shot generalization. This indicates that the model is largely good in this setting but makes

© 2025, *Indian Institute of Technology Delhi*

some mistakes in the absolute scoring of logical forms with schema elements not seen during training.

We have demonstrated that RetinaQA is a robust KBQA model that performs well for both answerable and unanswerable questions, across its various sub-categories and generalization categories. Our ablation study highlighted the importance of different components of RetinaQA for handling different categories of unanswerabilities. The error analysis revealed errors in multiple components of the architecture. Therefore, RetinaQA can be seen as a general framework for robust KBQA, where individual components can be improved for better overall performance.

© 2025, *Indian Institute of Technology Delhi*

Chapter 7

CONCLUSION AND FUTURE WORK

We have formally introduced the task of answerability detection for Knowledge Graph Question Answering (KBQA). Initially, we devised an algorithm to simulate incompleteness in a KB for curating a dataset containing both answerable and unanswerable questions based on a dataset comprising solely answerable questions. This algorithm enabled us to create a benchmark dataset for KBQA with unanswerable questions, named GrailQAbility. Subsequently, we adapted state-of-the-art KBQA models to handle unanswerable questions and evaluated their performance on GrailQAbility. Our results demonstrate that both proposed model adaptations—training with unanswerable questions and thresholding—substantially improve performance. However, even with these adaptations, we observed that no model could replicate its performance for answerable questions when presented with unanswerable questions, underscoring the necessity for a robust KBQA model. Additionally, we highlighted important trends and trade-offs between the robustness and performance of answerable questions.

Furthermore, we introduced RetinaQA, the first specialized model for KBQA that exhibits robustness for both answerable and unanswerable questions. RetinaQA 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. RetinaQA achieves a new state-of-the-art performance on GrailQAbility, enhancing unanswerability performance with thresholding and unanswerability training while experiencing only a minimal drop in performance for answerable questions. RetinaQA also performs well with only answerable training, rendering the model suitable for realistic scenarios. We conducted a detailed ablation study and demonstrated the importance of different model components for various categories of unanswerability. The RetinaQA model architecture can be viewed as a framework wherein each component (entity linking, retrieval, discriminator) can be individually improved. We presented a comprehensive error analysis, delineating the percentage of errors associated with every component.

We believe that both GrailQAbility as a benchmarking dataset and RetinaQA, as a framework, serve as a good starting point for further research toward robust KBQA models.

7.1 Limitations and Future Direction

1. GrailQAbility dataset:

GrailQAbility does not differentiate between the missing data element category (where a valid logical form exists but there is no answer) and questions where the answer is actually an empty set. As discussed in the thesis, distinguishing between these categories poses a significant challenge, but it would be a valuable enhancement and will add significantly to the difficulty of the dataset.

Since we constructed the dataset based on an existing Knowledge Base (KB), GrailQAbility does not include any out-of-domain questions (i.e., questions of domains not covered in the KB). Introducing this as a new sub-category under unanswerable questions could be a beneficial addition.

2. Logical form-based answer evaluation and unaswerability: For answerable questions, we consider only a single answer path to create the ground truth logical form, as is standard in current evaluation metrics. However, it's often possible to reach an answer through an alternative reasoning path, which may be correct or incorrect. For example, the relation "grandfather" is equivalent to the path "father_of-father_of." Current studies do not account for alternative correct reasoning paths in their evaluations.

This limitation also affects unanswerability, where a question is defined as unanswerable based on one ground truth logical form, even though it might be answerable through an alternative path. While we do not address this issue directly, we mitigate it by rewarding answers consistent with the original KB, which captures correct answers from alternate reasoning paths.

Exploring logical form equivalence for answerable questions would be a promising direction for future research.

3. RetinaQA model

From a deployment standpoint, RetinaQA exhibits significantly high inference times. We identify the retrieval stage as the primary bottleneck, particularly in scenarios involving large-scale Knowledge Bases (KBs). Future directions may involve optimizing the current architecture and exploring trade-offs between inference time and performance to address this challenge effectively.

In our analysis, we noted that entity linking errors constitute a substantial proportion of the overall error. We firmly believe that enhancing the performance of the entity linker alone

© 2025, *Indian Institute of Technology Delhi*

could significantly narrow the performance gap.

Additionally, for unanswerable questions, thresholding errors are the second most common type of error. This indicates that while discriminators are good at relatively ranking logical forms, they often fail to correctly score individual logical forms. This might be because the discriminator is trained contrastively to differentiate between logical forms rather than to score them. Implementing regularized contrastive training to correctly scale the absolute scores may help reduce this error.

4. Interpretable models

In this thesis, our initial objective is to solely identify whether a question is answerable or unanswerable, with the output of a logical form provided only for answerable questions. However, investigating the underlying cause of unanswerability presents a promising avenue for future research. Ideally, a model should generate an incomplete logical form, thereby indicating the type of missing KB element. For missing schema elements we can determine whether a relation or entity type is absent from the schema and for missing data elements, we can pinpoint the exact path or link absent in the KB from the equivalent query-graph. This capability could greatly benefit applications such as KB completion.

5. Extensions to Related Tasks

Expanding the insights and observations gleaned from the RetinaQA framework to additional tasks aimed at detecting unanswerability (such as NL-to-SQL or comprehension-based QA) represents a promising avenue for future research.

Given the prevalence of Large Language Models (LLMs) today, considerable recent work has focused on leveraging LLMs for KBQA tasks. FuSIC-KBQA (Patidar et al., 2023) studies retrieval-augmented LLMs for KBQA, focusing on low-resource and transfer scenarios. It uses different supervised models as retrievers and employs LLMs for final logical form generation. Sawhney et al. (2024) extends FuSIC-KBQA to address unanswerability. It iteratively prompts the LLM, with multiple feedbacks involving syntactic, semantic, and execution-guided checks. The authors test unanswerability in a transfer learning scenario. Interestingly, they propose using RetinaQA as a retriever in their framework and demonstrate that the combination of RetinaQA with their framework can outperform fully supervised models on the task of answerability detection in transfer learning settings. However, they do not discuss in-domain settings. It would be interesting to investigate the utility of LLMs for answerability detection in different settings, including in-domain, transfer, and low-resource scenarios.

© 2025, *Indian Institute of Technology Delhi*

Bibliography

G. A. Ansari, A. Saha, V. Kumar, M. Bhambhani, K. Sankaranarayanan, and S. Chakrabarti. Neural program induction for kbqa without gold programs or query annotations. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4890–4896. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/679. URL https://doi.org/10.24963/ijcai.2019/679.

J. Bao, N. Duan, Z. Yan, M. Zhou, and T. Zhao. Constraint-based question answering with knowledge graph. In Y. Matsumoto and R. Prasad, editors, *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2503–2514, Osaka, Japan, Dec. 2016. The COLING 2016 Organizing Committee. URL https://aclanthology.org/C16-1236.

A. Bordes, S. Chopra, and J. Weston. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*, 2014.

A. Bordes, N. Usunier, S. Chopra, and J. Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

S. Cao, J. Shi, L. Pan, L. Nie, Y. Xiang, L. Hou, J. Li, B. He, and H. Zhang. KQA Pro: A large diagnostic dataset for complex question answering over knowledge base. In *ACL'22*, 2022a.

S. Cao, J. Shi, Z. Yao, X. Lv, J. Yu, L. Hou, J. Li, Z. Liu, and J. Xiao. 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)*, 2022b.

S. Chen, Q. Liu, Z. Yu, C.-Y. Lin, J.-G. Lou, and F. Jiang. 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*, 2021.

E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang, and L. Zettlemoyer. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Oct.-Nov. 2018.

R. Das, M. Zaheer, D. Thai, A. Godbole, E. Perez, J. Y. Lee, L. Tan, L. Polymenakos, and A. McCallum. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

R. Das, A. Godbole, A. Naik, E. Tower, M. Zaheer, H. Hajishirzi, R. Jia, and A. Mccallum. Knowledge base question answering by case-based reasoning over subgraphs. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

M. Dubey, D. Banerjee, A. Abdelkawi, and J. Lehmann. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *The Semantic Web – ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II*, page 69–78, Berlin, Heidelberg, 2019. Springer-Verlag. ISBN 978-3-030-30795-0. doi: 10.1007/978-3-030-30796-7_5. URL https://doi.org/10.1007/978-3-030-30796-7_5.

Y. Gu, S. Kase, M. Vanni, B. Sadler, P. Liang, X. Yan, and Y. Su. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, WWW '21, 2021.

Y. Gu, V. Pahuja, G. Cheng, and Y. Su. Knowledge base question answering: A semantic parsing perspective, 2022.

Y. Gu, X. Deng, and Y. Su. 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, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.270. URL https://aclanthology.org/2023.acl-long.270.

S. Humeau, K. Shuster, M.-A. Lachaux, and J. Weston. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*, 2019.

R. Jia and P. Liang. Adversarial examples for evaluating reading comprehension systems. In M. Palmer, R. Hwa, and S. Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1215. URL https://aclanthology.org/D17-1215.

T. Li, X. Ma, A. Zhuang, Y. Gu, Y. Su, and W. Chen. Few-shot in-context learning for knowledge base question answering. *arXiv preprint arXiv:2305.01750*, 2023.

C. Liang, J. Berant, Q. Le, K. D. Forbus, and N. Lao. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In R. Barzilay and M.-Y. Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1003. URL https://aclanthology.org/P17-1003.

P.-E. Mazaré, S. Humeau, M. Raison, and A. Bordes. Training millions of personalized dialogue agents. *arXiv preprint arXiv:1809.01984*, 2018.

S. Mitra, R. Ramnani, and S. Sengupta. 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*, 2022.

© 2025, *Indian Institute of Technology Delhi*

M. Patidar, A. Singh, R. Sawhney, I. Bhattacharya, et al. Combining transfer learning with in-context learning using blackbox llms for zero-shot knowledge base question answering. *arXiv preprint arXiv:2311.08894*, 2023.

A. Perevalov, D. Diefenbach, R. Usbeck, and A. Both. Qald-9-plus: A multilingual dataset for question answering over dbpedia and wikidata translated by native speakers. In *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*, 2022.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020a. URL http://jmlr.org/papers/v21/20-074.html.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020b.

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

P. Rajpurkar, R. Jia, and P. Liang. 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)*, 2018.

S. Ravishankar, D. Thai, I. Abdelaziz, N. Mihindukulasooriya, T. Naseem, P. Kapanipathi, G. Rossiello, and A. Fokoue. A two-stage approach towards generalization in knowledge base question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022.

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

A. Saha, G. A. Ansari, A. Laddha, K. Sankaranarayanan, and S. Chakrabarti. Complex program induction for querying knowledge bases in the absence of gold programs. *Transactions of the Association for Computational Linguistics*, 7:185–200, 2019. doi: 10.1162/tacl_a_00262. URL https://aclanthology.org/Q19-1012.

R. Sawhney, I. Bhattacharya, et al. Robust few-shot transfer learning for knowledge base question answering with unanswerable questions. *arXiv preprint arXiv:2406.14313*, 2024.

A. Saxena, A. Tripathi, and P. Talukdar. 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*, 2020.

Y. Shu, Z. Yu, Y. Li, B. Karlsson, T. Ma, Y. Qu, and C.-Y. Lin. 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*, 2022.

© 2025, *Indian Institute of Technology Delhi*

Y. Su, H. Sun, B. Sadler, M. Srivatsa, I. Gür, Z. Yan, and X. Yan. On generating characteristic-rich question sets for QA evaluation. In J. Su, K. Duh, and X. Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572, Austin, Texas, Nov. 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1054. URL https://aclanthology.org/D16-1054.

E. Sulem, J. Hay, and D. Roth. 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*, 2022.

A. Talmor and J. Berant. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018.

D. Thai, S. Ravishankar, I. Abdelaziz, M. Chaudhary, N. Mihindukulasooriya, T. Naseem, R. Das, P. Kapanipathi, A. Fokoue, and A. McCallum. Cbr-ikb: A case-based reasoning approach for question answering over incomplete knowledge bases, 2022a.

D. Thai, S. Ravishankar, I. Abdelaziz, M. Chaudhary, N. Mihindukulasooriya, T. Naseem, R. Das, P. Kapanipathi, A. Fokoue, and A. McCallum. Cbr-ikb: A case-based reasoning approach for question answering over incomplete knowledge bases, 2022b. URL https://arxiv.org/abs/2204.08554.

B. Wang, Y. Gao, Z. Li, and J.-G. Lou. Know what i don't know: Handling ambiguous and unanswerable questions for text-to-sql. *arXiv preprint arXiv:2212.08902*, 2022a. URL https://arxiv.org/abs/2212.08902.

Y. Wang, V. Srinivasan, and H. Jin. 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*, 2022b.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

X. Ye, S. Yavuz, K. Hashimoto, Y. Zhou, and C. Xiong. 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)*, 2022.

W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh. 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)*, 2016.

T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In E. Riloff, D. Chiang,

© 2025, *Indian Institute of Technology Delhi*

J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1425. URL https://aclanthology.org/D18-1425.

J. Zhang, X. Zhang, J. Yu, J. Tang, J. Tang, C. Li, and H. Chen. 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)*, 2022.

Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song. Variational reasoning for question answering with knowledge graph. *CoRR*, abs/1709.04071, 2017. URL http://arxiv.org/abs/1709.04071.

H. Zhuang, Z. Qin, R. Jagerman, K. Hui, J. Ma, J. Lu, J. Ni, X. Wang, and M. Bendersky. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2308–2313, 2023.

© 2025, *Indian Institute of Technology Delhi*