Beyond Seen Data: Improving KBQA Generalization Through Schema-Guided Logical Form Generation

Anonymous ACL submission

Abstract

Knowledge base question answering (KBQA) aims to answer user questions in natural language using rich human knowledge stored in large KBs. As current KBQA methods struggle with unseen knowledge base elements at test time, we introduce **SG-KBQA**: a novel model that injects schema contexts into entity retrieval and logical form generation to tackle this issue. It uses the richer semantics and awareness of the knowledge base structure provided by schema contexts to enhance generalizability. We show that SG-KBQA achieves strong generalizability, outperforming state-of-the-art models on two commonly used benchmark datasets across a variety of test settings. Our source code is available at `https://anonymous.4open.science/r/SG-KBQA-7895`.

1 Introduction

Knowledge base question answering (KBQA) aims to answer user questions expressed in natural language with information from a knowledge base (KB). This offers user-friendly access to rich human knowledge stored in large KBs such as Freebase (Bollacker et al., 2008), DBPedia (Auer et al., 2007) and Wikidata (Vrandečić and Krötzsch, 2014), and it has broad applications in QA systems (Zhou et al., 2018), recommender systems (Guo et al., 2022), and information retrieval systems (Jalota et al., 2021).

State-of-the-art (SOTA) solutions often take a semantic parsing (SP)-based approach. They translate an input natural language question into a structured, executable form (AKA logical form (Lan et al., 2021)), which is then executed to retrieve the question answer. Figure 1 shows an example. The input question, Who is the author of Harry Potter, is expressed using the *S-expression* (Gu et al., 2021) (a type of logical form), which is formed by a set of functions (e.g., JOIN) operated over

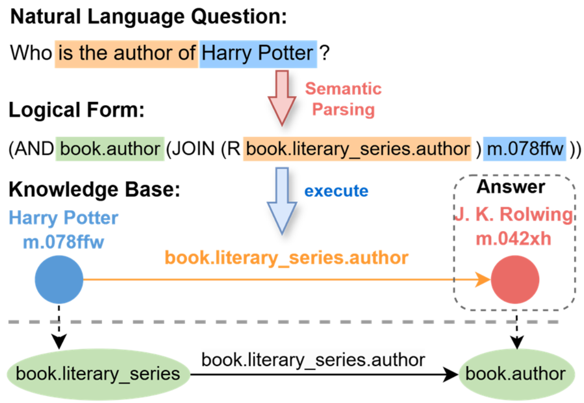


Figure 1: Example of KBQA and SP-based solutions.

elements of the target KB (e.g., entity m.078ffw refers to book series Harry Potter, book.author a class of entities, and book.literary_series.author a relation in Freebase).

A key challenge here is to learn a mapping between mentions of entities and relations in the input question to corresponding KB elements to form the logical form. Meanwhile, the mapping of KB element compositions has to adhere to the structural constraints (schema) of the KB. The schema defines entities' classes and the relationships between these classes within the KB. Take the KB subgraph in Figure 1 as an example, the relationship between the entity Harry Potter and the entity J.K. Rolwing is defined by the relation book.literary_series.author between their respective classes (i.e., class book.literary_series and class book.author).

However, due to the vast number of entities, relations, classes, and their compositions, it is difficult (if not impossible) to train a model with all feasible compositions of the KB elements. For example, Freebase (Bollacker et al., 2008) has over 39 million entities, 8,000 relations, and 4,000 classes. Furthermore, some KBs (e.g., NED (Mitchell et al., 2018)) are not static as they continue to grow.

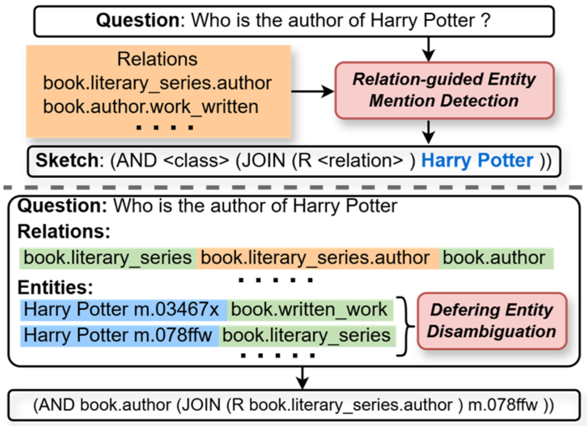


Figure 2: Relation-guided entity mention detection and schema-guided logical form generation.

A few studies consider model generalizability to non-I.I.D. settings, where the test set contains schema items (i.e., relations and classes) or compositions that are unseen during training (i.e., *zero-shot* and *compositional generalization*, respectively). In terms of methodology, these studies typically use ranking-based or generation-based models. Ranking-based models (Gu et al., 2021, 2023) retrieve entities relevant to the input question and then, starting from them, perform path traversal in the KB to obtain the target logical form by ranking. Generation-based models (Shu et al., 2022; Zhang et al., 2023) retrieve relevant KB contexts (e.g., entities and relations) for the input question, and then feed these contexts into a Seq2Seq model together with the input question to generate the logical form.

We observe that both types of models terminate their entity retrieval prematurely, such that each entity mention in the input question is mapped to only a single entity before the logical form generation stage. As a result, the logical form generation stage loses the freedom to explore the full combination space of relations and entities. This leads to inaccurate logical forms (as validated in our study).

To address this issue, our strategy is to defer entity disambiguation — i.e., to determine the most relevant entity for an entity mention (Section 2) — to the logical form generation stage. This allows our model to explore a larger combination space of the relations and entities, and ultimately leads to stronger model generalizability because low-ranked (but correct) relations or entities would still be considered during generation. We call our approach SG-KBQA (schema-guided logical form generator for KBQA). Concretely, SG-KBQA follows the generation-based approach but with deferred entity disambiguation. As shown in Figure 2, it feeds the input question, the retrieved candidate relations and entities, plus their corresponding schema information (the domain and range of classes of relations and entities; Section 4) into a large language model (LLM) for logical form generation. The schema information reveals the connectivity between the candidate relations and entities, hence guiding the LLM to uncover their correct combination in the large search space.

Further exploiting the schema-guided idea, we propose a relation-guided module for SG-KBQA to enhance its entity mention detection from the input question. As shown in Figure 2, this module adapts a Seq2Seq model to generate logical form sketches based on the input question and candidate relations, where relations, classes, and literals are masked by special tokens, such that the entity mentions can be identified more easily without confusions caused by these elements.

To summarise:

- We introduce SG-KBQA to solve the KBQA problem under non-I.I.D. settings, where test input contains unseen schema items or compositions during training.

- We propose to defer entity disambiguation to logical form generation, and additionally guide this generation step with corresponding schema information, allowing us to explore a larger combination space of relations and entities to consider unseen relations, entities, and compositions. We further propose a relation-guided module to strengthen entity retrieval by generating logical form sketches.

- We conduct experiments on two popular benchmark datasets and find SG-KBQA outperforming SOTA models on both datasets. In particular, on non-I.I.D GrailQA our model tops all three leaderboards for the overall, zero-shot, and compositional generalization settings, outperforming SOTA models by 3.3%, 2.9%, and 4.0% (F1) respectively.

2 Related Work

Knowledge Base Question Answering Most KBQA solutions use information retrieval-based (IR-based) or semantic parsing-based (SP-based) methods (Wu et al., 2019; Lan et al., 2021). IR-based methods construct a question-specific subgraph starting from the retrieved entities (i.e., the *topic entities*). They then reason over the subgraph to derive the answer. SP-based methods focus on transforming input questions into logical forms, which are then executed to retrieve answers. SOTA solutions are mostly SP-based, as detailed next.

KBQA under I.I.D. Settings Recent KBQA studies under I.I.D. settings fine-tune LLMs to map input questions to rough KB elements and generate approximate logical form drafts (Luo et al., 2024; Wang and Qin, 2024). The approximate (i.e., inaccurate or ambiguous) KB elements are then aligned to exact KB elements through a subsequent retrieval stage. These solutions often fail over test questions that refer to KB elements unseen during training. While we also use LLMs for logical form generation, we ground the generation with retrieved relations, entities, and schema contexts, thus addressing the non-I.I.D. issue.

KBQA under Non-I.I.D. Settings Studies considering non-I.I.D. settings can be largely classified into *ranking-based* and *generation-based* methods.

Ranking-based methods start from retrieved entities, traverse the KB, and construct the target logical form by ranking the traversed paths. Gu et al. (2021) enumerate and rank all possible logical forms within two hops of retrieved entities, while Gu et al. (2023) incrementally expand and rank paths from retrieved entities.

Generation-based methods transform an input question into a logical form using a Seq2Seq model (e.g., T5 (Raffel et al., 2020)). They often use additional contexts beyond the question to augment the input of the Seq2Seq model and enhance its generalizability. For example, Ye et al. (2022) use top-5 candidate logical forms enumerated from retrieved entities as the additional context. Shu et al. (2022) further use top-ranked relations, *disambiguated entities*, and classes (retrieved *separately*) as the additional context. Zhang et al. (2023) use connected pairs of retrieved KB elements.

Our SG-KBQA is generation-based. We use schema contexts (relations and classes) from retrieved relations and entities, rather than separate class retrieval (as in Shu et al. (2022)) which could introduce noise. We also defer entity disambiguation to the logical form generation stage, thus avoiding error propagation induced by premature entity disambiguation without considering the generation context, as done in existing works outlined below.

KBQA Entity Retrieval KBQA entity retrieval typically has three steps: entity mention detection, candidate entity retrieval, and entity disambiguation. BERT (Devlin et al., 2019)-based named entity recognition is widely used for entity mention detection from input questions. To retrieve KB entities corresponding to entity mentions, the FACC1 dataset (Gabrilovich et al., 2013) is commonly used, which contains over 10 billion surface forms (with popularity scores) of Freebase entities. Gu et al. (2021) use the popularity scores for entity disambiguation, while Ye et al. (2022) and Shu et al. (2022) adopt a BERT reranker.

3 Preliminaries

A graph structured-KB $\mathcal{G}$ is composed of a set of relational facts $\{\langle s, r, o\rangle | s \in \mathcal{E}, r \in \mathcal{R}, o \in \mathcal{E} \cup \mathcal{L}\}$ and an ontology $\{\langle c_d, r, c_r\rangle | c_d, c_r \in \mathcal{C}, r \in \mathcal{R}\}$. Here, $\mathcal{E}$ denotes a set of entities, $\mathcal{R}$ denotes a set of relations, and $\mathcal{L}$ denotes a set of literals, e.g., textual labels, numerical values, or date-time stamps. In a relational fact $\langle s, r, o\rangle$, $s \in \mathcal{E}$ is the *subject*, $o \in \mathcal{E} \cup \mathcal{L}$ is the *object*, and $r \in \mathcal{R}$ represents the relationship between the *subject* and the *object*.

The ontology defines the rules governing the composition of relational facts within $\mathcal{G}$. In its formulation, $\mathcal{C}$ denotes a set of classes, each of which defines a set of entities (or literals) sharing common properties (relations). Note that an entity can belong to multiple classes. In an ontology triple $\langle c_d, r, c_r\rangle$, c_d is called a *domain class*, and it refers to the class of subject entities that satisfy relation r; c_r is called the *range class*, and it refers to the class of object entities or literals satisfying r. Each ontology triple can be instantiated as a set of relational facts. In Figure 1, <book.literary_series, book.literary_series.author, book.author> is an ontology triple. An instance of it is <Harry Potter, book.literary_series.author, J.K. Rowling>, where Harry Potter is an entity that belongs to class book.literary_series.

Problem Statement Given a KB $\mathcal{G}$ and a question q expressed in natural language, i.e., a sequence of word tokens, knowledge base question answering (KBQA) aims to find a subset (the answer set) $\mathcal{A} \subseteq \mathcal{E} \cup \mathcal{L}$ of elements from $\mathcal{G}$ that — with optional application of some aggregation functions (e.g., COUNT) — answers q.

Logical Form We solve the KBQA problem by translating the input question q into a structured

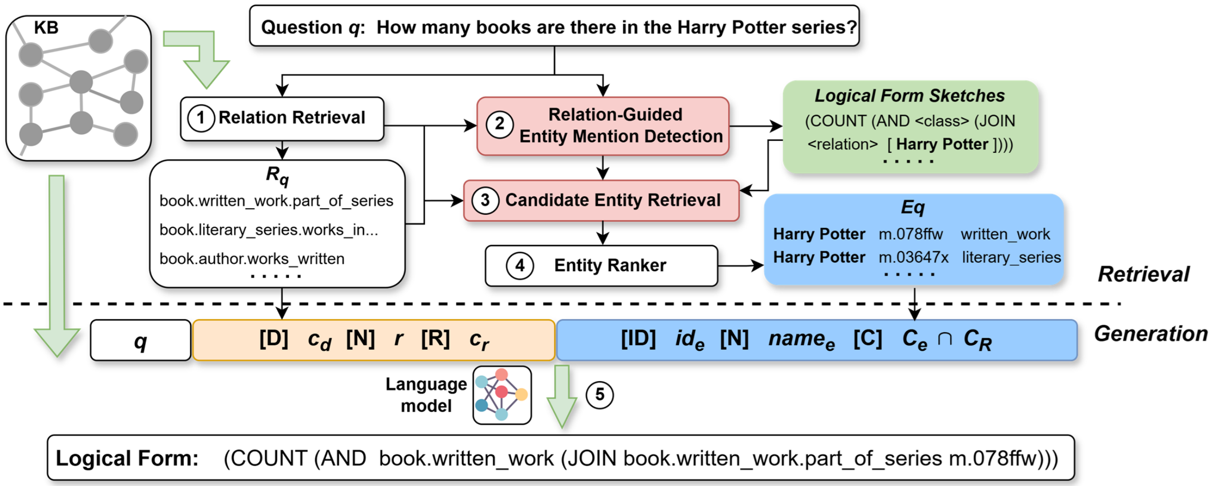


Figure 3: Overview of SG-KBQA. The model has two stages: *retrieval* and *generation*. In the retrieval stage, we first retrieve and rank candidate relations based on the input question q (①). Using q and the top-ranked candidate relations R_q, we generate logical form sketches and extract entity mentions from them (②). Based on the entity mentions and retrieved relations, we retrieve candidate entities from the KB (③) and rank them (the top-ones being E_q, ④). In the generation stage, q, R_q, E_q, and their class contexts, are fed into a fine-tuned language model for logical form generation (⑤). Here, the colored modules come with our new design.

query that can be executed on $\mathcal{G}$ to fetch the answer set $\mathcal{A}$. Following previous works (Shu et al., 2022; Ye et al., 2022; Gu et al., 2023; Zhang et al., 2023), we use logical form as the structured query language, expressed with the *S-expression* (Gu et al., 2021). The S-expression offers a readable representation well-suited for KBQA. It uses set semantics where functions operate on entities or entity tuples without requiring variables (Ye et al., 2022). Figure 1 shows an example: the S-expression of the given question Who is the author of Harry Potter? is (AND book.author (JOIN (R book.literary_series.author) m.078ffw)). This S-expression queries a set of entities that belong to the class book.author from the objects of triples whose subject entity is m.078ffw while the relation is book.literary_series.author. More details about the S-expression is in Appendix A.

4 The SG-KBQA Model

As shown in Figure 3, SG-KBQA follows the common structure of generation-based models. It has two overall stages: *relation and entity retrieval* and *logical form generation*. We propose novel designs in both stages to strengthen model generalizability.

In the relation and entity retrieval stage (Section 4.1), SG-KBQA retrieves candidate relations and entities from KB $\mathcal{G}$ which may be relevant to the input question q. It starts with a BERT-based relation ranking model to retrieve candidate relations relevant to q. Together with q, the set of top-ranked candidate relations are fed into a novel, relation-guided Seq2Seq model to generate logical form sketches that contain entity mentions while masking the relations and classes. We harvest the entity mentions and use them to retrieve candidate entities from $\mathcal{G}$. We propose a combined relation-based strategy to prune the entities (as there may be many). The remaining entities are ranked by a BERT-based model, indicating their likelihood of being the entity that matches each entity mention.

Leveraging relations to guide both entity mention extraction and candidate entity pruning enhances the model generalizability over entities unseen during training. This in turn helps the logical form generation stage to filter false positive matches for unseen relations or their combinations.

In the logical form generation stage (Section 4.2), SG-KBQA feeds q, the top-ranked relations and entities (corresponding to each entity mention), and the schema contexts (i.e., domain and range classes of the relations and classes of the entities), into an adapted LLM to generate the logical form and produce answer set $\mathcal{A}$.

Our schema-guided logical form generation procedure is novel in that it takes (1) multiple candidate entities (instead of one in existing models) for each entity mention and (2) the schema contexts as the input. Using multiple candidate entities essentially defers *entity disambiguation*, which is usually done in the retrieval stage by existing models (Shu et al., 2022; Gu et al., 2023), to the generation stage, thus mitigating error propagation. This strategy also brings challenges, as the extra candidate entities (which are ambiguous as they often share the same name) may confuse the logical form generation model. We address the challenges with the schema contexts, which instruct the model the connectivity structures between the candidate entities and relations. The connectivity structures further help SG-KBQA generalize to unseen entities, relations, or their combinations.

4.1 Relation and Entity Retrieval

Relation Retrieval For relation retrieval, we follow the schema retrieval model of TIARA (Shu et al., 2022), as it has high accuracy. We extract a set R_q of top-k_R (system parameter) relations with the highest semantic similarity to q. This is done by a BERT-based cross-encoder that learns the semantic similarity $\text{sim}(q, r)$ between q and a relation $r \in \mathcal{R}$:

$$\text{sim}(q, r) = \text{LINEAR}(\text{BERTCLS}([q; r])), \qquad (1)$$

where ';' denotes concatenation. This model is trained with the sentence-pair classification objective (Devlin et al., 2019), where a relevant question-relation pair has a similarity of 1, and 0 otherwise.

Relation-Guided Entity Mention Detection Given R_q, we propose a relation-guided logical form sketch parser to parse q into a logical form sketch s. Entity mentions in q are extracted from s.

The parser is an adapted Seq2Seq model. The model input of each training sample takes the form of "q <relation> $r_1; r_2; \ldots; r_{k_R}$" ($r_i \in R_q$, hence "relation-guided"). In the ground-truth logical form corresponding to q, we mask the relations, classes, and literals with special tokens '<relation>', '<class>', and '<literal>', to form the ground-truth logical form sketch s. Entity IDs are also replaced by the corresponding entity names (entity mentions), to enhance the Seq2Seq model's understanding of the semantics of entities.

At model inference, from the output top-k_L (system parameter) logical form sketches (using beam search), we extract the entity mentions.

Relation-Guided Candidate Entity Retrieval We follow previous studies (Gu et al., 2021; Shu et al., 2022; Faldu et al., 2024; Luo et al., 2024) and use an entity name dictionary FACC1 (Gabrilovich et al., 2013) to map extracted entity mentions to

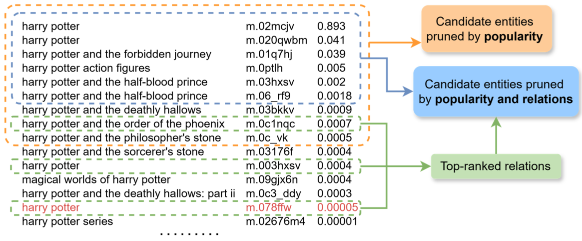


Figure 4: Candidate entity retrieval for the mention 'aloha'. The candidate entity in red is the ground-truth.

entities (i.e., their IDs in KB), although other retrieval models can be used. Since different entities may share the same name, the entity mentions may be mapped to many entities. For pruning, existing studies use popularity scores associated to entities (Shu et al., 2022; Ye et al., 2022).

To improve the recall of candidate entity retrieval, we propose a combined pruning strategy based on both popularity and relation connectivity. As Figure 4 shows, we first select the top-k_{E1} (system parameter) entities for each entity mention based on popularity and then extract k_{E2} (system parameter) entities from the remaining candidates that are connected to the retrieved relations R_q. Together, these form the candidate entity set E_c.

Entity Ranking We follow existing works (Shu et al., 2022; Ye et al., 2022) to score and rank each candidate entity in E_c by jointly encoding q and the context (entity name and its linked relations) of the entity using a cross-encoder (like Eq. 1). We select the top-k_{E3} (system parameter) ranked entities for each mention as the entity set E_q for each question.

4.2 Schema-Guided Logical Form Generation

Given relations R_q and entities E_q, we fine-tune an open-souce LLM (LLaMA3.1-8B (Touvron et al., 2023) by default) to generate the final logical form.

Before being fed into the model, each relation and entity is augmented with its schema context (i.e., class information) to help the model to learn their connections and generalize to unseen entities, relations, or their compositions. The context of a relation r is described by concatenating the relation's domain class c_d and range class c_r, formatted as "[D] c_d [N] r [R] c_r". For an entity e, its context is described by its ID ("id_e"), name ("$name_e$"), and the intersection of its set of classes C_e and the set of all domain and range classes C_R of all relations in R_q, formatted as "[ID] id_e [N] $name_e$ [C] class($C_e \cap C_R$)".

As Figure 3 shows, we construct the input to the logical form generation model by concatenating q with the context of each relation in R_q and the context of each entity in E_q. The model is fine-tuned with a cross-entropy-based objective:

$$\mathcal{L}_{generator} = -\sum_{t=1}^{n} \log p\left(l_t \mid l_{<t}, q, K_q\right), \tag{2}$$

where l denotes a logical form of n tokens and l_t is its t-th token, and K_q is the retrieved knowledge (i.e., relations and entities with contexts) for q. At inference, the model runs beam search to generate top-k_O logical forms – the executable one with the highest score is selected as the output. See Appendix B for a prompt example used for inference.

It is possible that no generated logical forms are executable. In this case, we fall back to following Shu et al. (2022) and Ye et al. (2022) and retrieve candidate logical forms in two stages: enumeration and ranking. During enumeration, we search the KB by traversing paths starting from the retrieved entities. Due to the exponential growth in the number of candidate paths with each hop, we start from the top-1 entity for each mention and searches its neighborhood for up to two hops. The paths retrieved are converted into logical forms. During ranking, a BERT-based ranker scores q and each enumerated logical form l (like Eq. 1). We train the ranker using a contrastive objective:

$$\mathcal{L} = -\frac{\exp(\text{sim}(q, l^*))}{\exp(\text{sim}(q, l^*)) + \sum_{l \in C_l \wedge l \neq l^*} \exp(\text{sim}(q, l))}, \tag{3}$$

where l^* is the ground-truth logical form and C_l is the set of enumerated logical forms. We run the ranked logical forms from the top and return the first executable one.

5 Experiments

We run experiments to answer: **Q1**: How does SG-KBQA compare with SOTA models in their accuracy for the KBQA task? **Q2**: How do model components impact the accuracy of SG-KBQA? **Q3**: How do our techniques generalize to other KBQA models?

5.1 Experimental Setup

Datasets Following SOTA competitors (Shu et al., 2022; Gu et al., 2023; Zhang et al., 2023), we use two benchmark datasets built upon Freebase.

GrailQA (Gu et al., 2021) is a dataset for evaluating the generalization capability of KBQA models. It contains 64,331 questions with annotated target S-expressions, including complex questions requiring up to 4-hop reasoning over Freebase, with aggregation functions including comparatives, superlatives, and counting. The dataset comes with training (70%), validation (10%), and test (20%, hidden and only known by the leaderboard organizers) sets. In the validation and the test sets, 50% of the questions include KB elements that are unseen in the training set (**zero-shot** generalization tests), 25% consist of unseen compositions of KB elements seen in the training set (**compositional** generalization tests), and the remaining 25% are randomly sampled from the training set (**I.I.D.** tests).

WebQuestionsSP (**WebQSP**) (Yih et al., 2016) is a dataset for the I.I.D. setting. While our focus is on non-I.I.D. settings, we include results on this dataset to show the general applicability of SG-KBQA. WebQSP contains 4,937 questions. More details of WebQSP are included in Appendix C.

Competitors We compare with both IR-based and SP-based methods including the SOTA models.

On GrailQA, we compare with models that top the leaderboard[1], including **RnG-KBQA** (Ye et al., 2022), **TIARA** (Shu et al., 2022), **DecAF** (Yu et al., 2023), **Pangu** (previous SOTA as of 15th February, 2025) (Gu et al., 2023), **FC-KBQA** (Zhang et al., 2023), **TIARA+GAIN** (Shu and Yu, 2024), and **RetinaQA** (Faldu et al., 2024). We also compare with few-shot LLM (training-free) methods: KB-BINDER (6)-R (Li et al., 2023), Pangu (Gu et al., 2023), and FlexKBQA (Li et al., 2024). These models are SP-based. On the non-I.I.D. GrailQA, IR-based methods are uncompetitive and excluded.

On WebQSP, we compare with IR-based models **SR+NSM** (Zhang et al., 2022), **UNIKGQA** (Jiang et al., 2023), and **EPR+NSM** (Ding et al., 2024), plus SP-based models **ChatKBQA** (SOTA) (Luo et al., 2024) and **TFS-KBQA** (SOTA) (Wang and Qin, 2024), both of which use a fine-tuned LLM to generate logical forms. We also compare with TIARA, Pangu, and FC-KBQA as above, which represent SOTA models using pre-trained language models (PLMs). Appendix D details these models. The baseline results are collected from their papers or the GrailQA leaderboard (if available).

Implementation Details All our experiments are run on a machine with an NVDIA A100 GPU and 120 GB of RAM. We fine-tuned three `bert-base-uncased` models for a maximum of

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

	Model	Overall		I.I.D.		Compositional		Zero-shot	
		EM	F1	EM	F1	EM	F1	EM	F1
SP-based (SFT)	RnG-KBQA (ACL 2021)	68.8	74.4	86.2	89.0	63.8	71.2	63.0	69.2
	TIARA (EMNLP 2022)	73.0	78.5	87.8	90.6	69.2	76.5	68.0	73.9
	Decaf (ICLR 2023)	68.4	78.7	84.8	89.9	73.4	<u>81.8</u>	58.6	72.3
	Pangu (T5-3B) (ACL 2023)	75.4	<u>81.7</u>	84.4	88.8	<u>74.6</u>	81.5	71.6	<u>78.5</u>
	FC-KBQA (ACL 2023)	73.2	78.7	<u>88.5</u>	<u>91.2</u>	70.0	76.7	67.6	74.0
	TIARA+GAIN (EACL 2024)	<u>76.3</u>	81.5	<u>88.5</u>	<u>91.2</u>	73.7	80.0	<u>71.8</u>	77.8
	RetinaQA (ACL 2024)	74.1	79.5	-	-	71.9	78.9	68.8	74.7
SP-based (Few-shot)	KB-Binder (6)-R (ACL 2023)	53.2	58.5	72.5	77.4	51.8	58.3	45.0	49.9
	Pangu (Codex) (ACL 2023)	56.4	65.0	67.5	73.7	58.2	64.9	50.7	61.1
	FlexKBQA (AAAI 2024)	62.8	69.4	71.3	75.8	59.1	65.4	60.6	68.3
Ours (SFT)	**SG-KBQA**	**79.1**	**84.4**	**88.6**	**91.6**	**77.9**	**85.1**	**75.4**	**80.8**
	- Improvement	+3.6%	+3.3%	+0.1%	+0.4%	+4.4%	+4.0%	+5.0%	+2.9%

Table 1: *Hidden* test results (%) on GrailQA (best results are in boldface; best baseline results are underlined; "SFT" means supervised fine-tuning; "few-shot" means few-show in-context learning).

	Model	F1
IR-based	SR+NSM (ACL 2022)	69.5
	UniKGQA (ICLR 2023)	75.1
	EPR+NSM (WWW 2024)	71.2
SP-based (SFT)	TIARA (EMNLP 2022)	76.7
	Pangu (T5-3B, ACL 2023)	79.6
	FC-KBQA (ACL 2023)	76.9
	ChatKBQA (ACL 2024)	79.8
	TFS-KBQA (LREC-COLING 2024)	<u>79.9</u>
SP-based (Few-shot)	KB-Binder (6)-R (ACL 2023)	53.2
	Pangu (Codex) (ACL 2023)	54.5
	FlexKBQA (AAAI 2024)	60.6
Ours (SFT)	**SG-KBQA**	**80.3**
	- Improvement	+0.5%

Table 2: Test results (%) on WebQSP (I.I.D.).

three epochs each, for relation retrieval, entity ranking, and fallback logical form ranking. For each dataset, a T5-base model is fine-tuned for 5 epochs as our logical form sketch parser. Finally, we fine-tune a LLaMA3.1-8B with LoRA (Hu et al., 2022a) for 5 epochs on GrailQA and 20 epochs on WebQSP to serve as the logical form generator. Our system parameters have been chosen empirically, and a parameter study is provided in Appendix H. More implementation details are in Appendix E.

Evaluation Metrics On GrailQA, we report the exact match (**EM**) and **F1** scores, following the leaderboard. EM counts the percentage of test samples where the model generated logical form (an S-expression) that is semantically equivalent to the ground truth. F1 measures the answer set correctness, i.e., the F1 score of each answer set, average over all test samples. On WebQSP, we report the F1 score as there are no ground-truth S-expressions.

5.2 Overall Results (Q1)

Tables 1 and 2 show the overall comparison of SG-KBQA with the baseline models for GrailQA and WebQSP, respectively. SG-KBQA shows the best results across both datasets.

Results on GrailQA On the overall hidden test set of GrailQA, SG-KBQA outperforms the best baseline Pangu by 4.9% and 3.3% in the EM and F1 scores, respectively. Under the compositional and zero-shot generalization settings (both are non-I.I.D.), similar performance gaps are observed, i.e., 4.0% and 2.9% in F1 compared to the best baseline models, respectively. This validates that SG-KBQA can extract relations and entities more accurately from the input question, even when these are unseen in the training set, and it creates more accurate logical forms to answer the questions.

The fine-tuned baseline models do not use relation semantics to enhance entity retrieval, and they either omit the class contexts in logical form generation or use these classes separately for retrieval. As such, they do not generalize as well in the non-I.I.D. settings. The few-shot LLM-based competitors are generally not very competitive, especially under the non-I.I.D. settings. This suggests that the current generation of LLMs are unable to infer from a few input demonstrations the process of logical form generation from user questions. Fine-tuning is still required.

Results on WebQSP On WebQSP, which has an I.I.D. test set, the performance gap of the different models are closer. Even in this case, SG-KBQA still performs the best, showing its general applicability. Comparing with TFS-KBQA (SOTA) and

	GrailQA				WebQSP
Model	**Overall**	**I.I.D.**	**Comp.**	**Zero.**	**Overall**
SG-KBQA	**88.5**	**94.6**	**84.6**	**87.9**	**80.3**
w/o RG-EMD	85.3	92.4	80.2	84.3	78.4
w/o RG-CER	86.5	92.1	81.1	86.3	79.5
w/o DED	87.8	94.0	82.4	87.2	78.2
w/o SC	79.2	92.9	77.4	73.9	77.1

Table 3: Ablation study results (F1 score) on the validation set of GrailQA and the test set of WebQSP.

Model	Overall	I.I.D.	Comp.	Zero.
TIARA (T5-base)	81.9	91.2	74.8	80.7
w RG-EMD & RG-CER	84.3	92.3	78.1	83.3
w DED & SC	85.6	92.3	79.8	85.0
SG-KBQA	**88.5**	**94.6**	**83.6**	**87.9**
w T5-base	84.9	92.6	81.0	83.3
w DS-R1-8B	87.5	94.0	82.4	86.7

Table 4: Module applicability results (F1 score) on the validation set of GrailQA. EM scores are in Appendix G.

ChatKBQA, SG-KBQA improves the F1 score by 0.5%. Among IR-based methods, UniKGQA (SOTA) still performs substantially worse compared to SG-KBQA. The lower performance of IR-based methods is consistent with existing results (Gu et al., 2022).

5.3 Ablation Study (Q2)

Next, we run an ablation study with the following variants of SG-KBQA: **w/o RG-EMD** replaces our relation-guided entity mention detection with SpanMD (Shu et al., 2022) which is commonly used in existing models (Pang et al., 2022; Faldu et al., 2024); **w/o RG-CER** omits candidate entities retrieved from the top relations; **w/o DED** uses the top-1 candidate entity for each entity mention without deferring entity disambiguation; **w/o SC** omits schema contexts from logical form generation.

Table 3 shows the results on the validation set of GrailQA and the test set of WebQSP. Only F1 scores are reported for conciseness, as the EM scores on GrailQA exhibit similar comparative trends and are provided in Appendix F.

All model variants have lower F1 scores than those of the full model, confirming the effectiveness of the model components. SG-KBQA w/o DED (with schema contexts) reduces the F1 scores across various generalization settings on both datasets, demonstrating the effectiveness of our DED strategy in reducing error propagation during the retrieval and generation stages. Furthermore, SG-KBQA w/o SC (with deferred entity disambiguation) has the most significant drops in the F1 score under the compositional (7.2) and zero-shot (14.0) generalization tests. It highlights the importance of schema contexts in constraining the larger search space introduced by DED and in generalizing to unseen KB elements and their combinations. Meanwhile, the lower F1 of SG-KBQA w/o RG-EMD emphasizes the capability of our relation-guided entity mention detection module in strengthening KBQA entity retrieval.

5.4 Module Applicability (Q3)

Our relation-guided entity retrieval (**RG-EMD & RG-CER**) module and schema-guided logical form generation (**DED & SC**) module can be applied to existing KBQA models. We showcase such applicability with the TIARA model. As shown in Table 4, by replacing the retrieval and generation modules of TIARA with ours, the F1 scores increase consistently for the non-I.I.D. tests.

Table 4 further reports F1 scores of SG-KBQA when we replace LLaMA3.1-8B with **T5-base** (which is used by TIARA), and DeepSeek-R1-Distill-Llama-8B (**DS-R1-8B**) (Guo et al., 2025) for logical form generation. We see that, even with the same T5-base model for the logical form generator, SG-KBQA outperforms TIARA consistently. This further confirms the effectiveness of our model design. As for DS-R1-8B, it offers accuracy slightly lower than that of the default LLaMA3.1-8B model. We conjecture that this is because DS-R1-8B is distilled from DeepSeek-R1-Zero, which focuses on reasoning capabilities and is not specifically optimized for the generation task.

We also have results on parameter impact, model running time, a case study, and error analyses. They are documented in Appendices H to K.

6 Conclusion

We proposed SG-KBQA for the KBQA task. Our core innovations include: (1) using relation to guide the retrieval of entities; (2) deferring entity disambiguation to the logical form generation stage; and (3) enriching logical form generation with schema contexts to constrain search space. Together, we achieve a model that tops the leaderboard of a popular non-I.I.D. dataset GrailQA, outperforming SOTA models by 4.0%, 2.9%, and 3.3% in F1 under compositional generalization, zero-shot generalization, and overall test settings, respectively. Our model also performs well in the I.I.D. setting, outperforming SOTA models on WebQSP.

Limitations

First, like any other supervised models, SG-KBQA requires annotated samples for training which may be difficult to obtain for many domains. Exploiting LLMs to generate synthetic training data is a promising direction to address this issue. Second, as discussed in the error analysis in Appendix K, errors can still arise from the relation retrieval, entity retrieval, and logical form generation modules. There are rich opportunities in further strengthening these modules. Particularly, as we start from relation extraction, the overall model accuracy relies on highly accurate relation extraction. It would be interesting to explore how well SG-KBQA performs on even larger KBs with more relations.

Ethics Statement

This work adheres to the ACL Code of Ethics and is based on publicly available datasets, used in compliance with their respective licenses. As our data contains no sensitive or personal information, we foresee no immediate risks. To promote reproducibility and further research, we also open-source our code.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *The Semantic Web*, pages 722–735.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Wentao Ding, Jinmao Li, Liangchuan Luo, and Yuzhong Qu. 2024. Enhancing complex question answering over knowledge graphs through evidence pattern retrieval. In *WWW*, pages 2106–2115.

Prayushi Faldu, Indrajit Bhattacharya, and Mausam. 2024. RetinaQA : A knowledge base question answering model robust to both answerable and unanswerable questions. In *ACL*, pages 6643–6656.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of clueweb corpora, version 1(release date 2013-06-26, format version 1, correction level 0).

Yu Gu, Xiang Deng, and Yu Su. 2023. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In *ACL*, pages 4928–4949.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond I.I.D.: Three levels of generalization for question answering on knowledge bases. In *WWW*, pages 3477–3488.

Yu Gu, Vardaan Pahuja, Gong Cheng, and Yu Su. 2022. Knowledge base question answering: A semantic parsing perspective. In *AKBC*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2022. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *WSDM*, pages 553–561.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022a. LoRA: Low-rank adaptation of large language models. In *ICLR*.

Xixin Hu, Xuan Wu, Yiheng Shu, and Yuzhong Qu. 2022b. Logical form generation via multi-task learning for complex question answering over knowledge bases. In *COLING*, pages 1687–1696.

Rricha Jalota, Daniel Vollmers, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. 2021. LAUREN - Knowledge graph summarization for question answering. In *ICSC*, pages 221–226.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2023. UniKGQA: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *ICLR*.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. In *IJCAI*, pages 4483–4491.

Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In *ACL*, page 6433–6441.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot in-context learning for knowledge base question answering. In *ACL*, pages 6966–6980.

Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2024. FlexKBQA: A flexible LLM-powered framework for few-shot knowledge base question answering. In *AAAI*, pages 18608–18616.

Haoran Luo, Haihong E, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, and Wei Lin. 2024. ChatKBQA: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. In *Findings of the ACL*, pages 2039–2056.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2018. Never-ending learning. *Communications of the ACM*, 61(5):103–115.

Junbiao Pang, Yongheng Zhang, Jiaxin Deng, and Xiaoqing Zhu. 2022. A survey on information retrieval method for knowledge graph complex question answering. In *CAC*, pages 1059–1064.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Yiheng Shu and Zhiwei Yu. 2024. Distribution shifts are bottlenecks: Extensive evaluation for grounding language models to knowledge bases. In *EACL*, pages 71–88.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. TIARA: Multi-grained retrieval for robust question answering over large knowledge base. In *EMNLP*, pages 8108–8121.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Shouhui Wang and Biao Qin. 2024. No need for large-scale search: Exploring large language models in complex knowledge base question answering. In *LREC-COLING*, pages 12288–12299.

Peiyun Wu, Xiaowang Zhang, and Zhiyong Feng. 2019. A survey of question answering over knowledge base. In *CCKS*, pages 86–97.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RnG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In *ACL*, pages 6032–6043.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *ACL*, pages 201–206.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2023. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases. In *ICLR*.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *ACL*, pages 5773–5784.

Lingxi Zhang, Jing Zhang, Yanling Wang, Shulin Cao, Xinmei Huang, Cuiping Li, Hong Chen, and Juanzi Li. 2023. FC-KBQA: A fine-to-coarse composition framework for knowledge base question answering. In *ACL*, pages 1002–1017.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.

A S-Expression

S-expressions (Gu et al., 2021) use set-based semantics defined over a set of operators and operands. The operators are represented as functions. Each function takes a number of arguments (i.e., the operands). Both the arguments and the return values of the functions are either a set of entities or entity tuples (or tuples of an entity and a literal). The functions available in S-expressions are listed in Table 5, where a set of entities typically refers to a class (recall that a class is defined as a set of entities sharing common properties) or individual entities, and a binary tuple typically refers to a relation.

B Prompt Example

We show an example prompt to our fine-tuned LLM-based logical form generator containing top-20 relations and top-2 entities per mention retrieved by our model in Figure 5.

C Additional Details on the WebQSP Dataset

WebQuestionsSP (WebQSP) (Yih et al., 2016) is an I.I.D. dataset. It contains 4,937 questions collected from Google query logs, including 3,098 questions for training and 1,639 for testing, each annotated with a target SPARQL query. We follow GMT-KBQA (Hu et al., 2022b), TIARA (Shu et al., 2022) to separate 200 questions from the training questions to form the validation set.

D Baseline Models

The following models are tested against SG-KBQA on the GrailQA dataset:

- RnG-KBQA (Ye et al., 2022) enumerates and ranks all possible logical forms within two hops from the entities retrieved by an entity retrieval step. It uses a Seq2Seq model to generate the target logical form based on the input question and the top-ranked candidate logical forms.

- TIARA (Shu et al., 2022) shares the same overall procedure with RnG-KBQA. It further retrieves entities, relations, and classes based on the input question and feeds these KB elements into the Seq2Seq model together with the question and the top-ranked candidate logical forms to generate the target logical form.

- TIARA+GAIN (Shu and Yu, 2024) enhances TIARA using a training data augmentation strategy. It synthesizes additional question-logical form pairs for model training to enhance the model's capability to handle more entities and relations. This is done by a graph traversal to randomly sample logical forms from the KB and a PLM to generate questions corresponding to the logical forms (i.e., the "GAIN" module). TIARA+GAIN is first tuned using the synthesized data and then tuned on the target dataset, for its retriever and generator modules which both use PLMs.

- Decaf (Yu et al., 2023) uses a Seq2Seq model that takes as input a question and a linearized question-specific subgraph of the KG and jointly decodes into both a logical form and an answer candidate. The logical form is then executed, which produces a second answer candidate if successful. The final answer is determined from these two answer candidates with a scorer model.

- Pangu (Gu et al., 2023) formulates logical form generation as an iterative enumeration process starting from the entities retrieved by an entity retrieval step. At each iteration, partial logical forms generated so far are extended following paths in the KB to generate more and longer partial logical forms. A language model is used to select the top partial logical forms to be explored in the next iteration, under either fined-tuned models (T5-3B) or few-shot in-context learning (Codex).

- FC-KBQA (Zhang et al., 2023) employs an intermediate module to test the connectivity between the retrieved KB elements, and it generates the target logical form using the connected pairs of the retrieved KB elements through a Seq2Seq model.

- RetinaQA (Faldu et al., 2024) uses both a ranking-based method and a generation-based method (TIARA) to generate logical forms, which are then scored by a discriminative model to determine the output logical form.

- KB-BINDER (Li et al., 2023) uses a training-free few-shot in-context learning model based on LLMs. It generates a draft logical form by showcasing the LLM examples of questions and logical forms (from the training set) that

Function	Return value	Description
(`AND` u_1 u_2)	a set of entities	The `AND` function returns the intersection of two sets u_1 and u_2
(`COUNT` u)	a singleton set of integers	The `COUNT` function returns the cardinality of set u
(`R` b)	a set of (entity, entity) tuples	The `R` function reverses each binary tuple (x, y) in set b to (y, x)
(`JOIN` b u)	a set of entities	Inner `JOIN` based on entities in set u and the second element of tuples in set b
(`JOIN` b_1 b_2)	a set of (entity, entity) tuples	Inner `JOIN` based on the first element of tuples in set b_2 and the second element of tuples in set b_1
(`ARGMAX` u b) (`ARGMIN` u b)	a set of entities	These functions return x in u such that $(x, y) \in b$ and y is the largest / smallest
(`LT` b n) (`LE` b n) (`GT` b n) (`GE` b n)	a set of entities	These functions return all x such that $(x, v) \in b$ and $v < / \leq / > / \geq n$

Table 5: Functions (operators) defined in S-expressions (u: a set of entities, b: a set of (entity, entity or literal) tuples, n: a numerical value).

Please translate the following question into logical form using the provided relations and entities.

Question: Captain pugwash makes an appearance in which comic strip?

Candidate relations with their corresponding Domain [D], Name [N], Range [R]:

[D] comic_strips.comic_strip_character [N] comic_strips.comic_strip_character.comic_strips_appeared_in [R] comic_strips.comic_strip ;
[D] comic_strips.comic_strip [N] comic_strips.comic_strip.characters [R] comic_strips.comic_strip_character ;
[D] comic_books.comic_book_character [N] comic_books.comic_book_character.regular_featured_appearances [R] comic_books.comic_book_series ;
[D] comic_strips.comic_strip [N] comic_strips.comic_strip.syndicate [R] comic_strips.comic_strip_syndicate_duration ;
[D] comic_books.comic_book_character [N] comic_books.comic_book_character.first_appearance [R] comic_books.comic_book_issue ;
[D] comic_strips.comic_strip_syndicate [N] comic_strips.comic_strip_syndicate.comic_strips_syndicated [R] comic_strips.comic_strip_syndicate_duration ;
[D] comic_books.comic_book_character [N] comic_books.comic_book_character.cover_appearances [R] comic_books.comic_book_issue ;

. . . .

Candidate entities with their corresponding id [ID], Name [N], Class [C]:

[ID] m.04fgkzf [N] captain pugwash [C] comic_strips.comic_strip ;
[ID] m.02hcty [N] captain pugwash [C] comic_strips.comic_strip_character ;

. . . .

Figure 5: Example prompt to our fine-tuned LLM-based logical form generator for an input question: Captain pugwash makes an appearance in which comic strip?

are similar to the given test question. Subsequently, a retrieval module grounds the surface forms of the KB elements in the draft logical form to specific KB elements.

- FlexKBQA (Li et al., 2024) considers limited training data and leverages an LLM to generate additional training data. It samples executable logical forms from the KB and utilizes an LLM with few-shot in-context learning to convert them into natural language questions, forming synthetic training data. These data, together with a few real-world training samples, are used to train a KBQA model. Then, the model is used to generate logical forms with more real world questions (without ground truth), which are filtered through an execution-guided module to prune the erroneous ones. The remaining logical forms and the corresponding real-world questions are used to train a new model. This process is repeated, to align the distributions of synthetic training data and real-world questions.

The following models are tested against SG-KBQA on the WebQSP dataset:

- Subgraph Retrieval (SR) (Zhang et al., 2022) focuses on retrieving a KB subgraph relevant to the input question. It does not concern retrieving the exact question answer by reasoning over the subgraph. Starting from the topic entity, it performs a top-k beam search at each step to progressively expand into a subgraph, using a scorer module to score the candidate relations to be added to the subgraph next.

- Evidence Pattern Retrieval (EPR) (Ding et al., 2024) aims to extract subgraphs with fewer noise entities. It starts from the topic entities and expands by retrieving and ranking atomic (topic entity-relation or relation-relation) patterns relevant to the question. This forms a set of relation path graphs (i.e., the candidate *evidence patterns*). The relation path graphs are then ranked to select the most relevant one. By further retrieving the entities on the selected relation path graph, EPR obtains the final subgraph relevant to the input question.

- Neural State Machine (NSM) (He et al., 2021) is a reasoning model to find answers for the KBQA problem from a subgraph (e.g., retrieved by SR or EPR). It address the issue of lacking intermediate-step supervision signals when reasoning through the subgraph to reach the answer entities. This is done by training a so-called teacher model that follows a bidirectional reasoning mechanism starting from both the topic entities and the answer entities. During this process, the "distributions" of entities, which represent their probabilities to lead to the answer entities (i.e., intermediate-step supervision signal), are propagated. A second model, the so-called student model, learns from the teacher model to generate the entity distributions, with knowledge of the input question and the topic entities but not the answer entities. Once trained, this model can be used for KBQA answer reasoning.

- UniKGQA (Jiang et al., 2023) integrates both retrieval and reasoning stages to enhance the accuracy of multi-hop KBQA tasks. It trains a PLM to learn the semantic relevance between every relation and the input question. The semantic relevance information is propagated and aggregated through the KB to form the semantic relevance between the entities and the input question. The entity with the highest semantic relevance is returned as the answer.

- ChatKBQA (Luo et al., 2024) fine-tunes an open-source LLM to map questions into draft logical forms. The ambiguous KB items in the draft logical forms are replaced with specific KB elements by a separate retrieval module.

- TFS-KBQA (Wang and Qin, 2024) fine-tunes an LLM for more accurate logical form generation with three strategies. The first strategy directly fine-tunes the LLM to map natural language questions into draft logical forms containing entity names instead of entity IDs. The second strategy breaks the mapping process into two steps, first to generate relevant KB elements, and then to generate draft logical forms using the KB elements. The third strategy fine-tunes the LLM to directly generate the answer to an input question. After applying the three fine-tuning strategies, the LLM is used to map natural language questions into draft logical forms at model inference. A separate entity linking module is used to further map the entity names in draft logical forms into entity IDs.

E Implementation Details

All our experiments are run on a machine with an NVDIA A100 GPU and 120 GB of RAM. We fine-tuned three `bert-base-uncased` models for a maximum of three epochs each, for relation retrieval, entity ranking, and fallback logical form ranking. For relation retrieval, we randomly sample 50 negative samples for each question to train the model to distinguish between relevant and irrelevant relations.

For each dataset, a `T5-base` model is fine-tuned for 5 epochs as our logical form sketch parser, with a beam size of 3 (i.e., $k_L = 3$) for GrailQA, and 4 for WebQSP. For candidate entity retrieval, we use the same number (i.e., $k_{E1} + k_{E2} = 10$) of candidate entities per mention as that used by the baseline models (Shu et al., 2022; Ye et al., 2022). The retrieved candidate entities for a mention consist of entities with the top-k_{E1} popularity scores and k_{E2} entities connected to the top-ranked relations in R_q, where $k_{E1} = 1$, $k_{E2} = 9$ for GrailQA, $k_{E1} = 3$, $k_{E2} = 7$ for WebQSP. We select the top-20 (i.e., k_R = 20) relations and the top-2 (i.e., $k_{E3} = 2$) entities (for each entity mention) retrieved by our model. For WebQSP, we also use the candidate entities obtained from the off-the-shelf entity linker ELQ (Li et al., 2020).

Finally, we fine-tune LLaMA3.1-8B with LoRA (Hu et al., 2022a) for logical form generation. On GrailQA, LLaMA3.1-8B is fine-tuned for 5 epochs with a learning rate of 0.0001. On WebQSP, it is fine-tuned for 20 epochs with the same learning rate (as it is an I.I.D. dataset where more epochs are beneficial). During inference, we generate logical forms by beam search with a beam size of 10 (i.e., $K_O = 10$). The generated logical forms are executed on the KB to filter non-executable ones. If none of the logical forms are executable, we check candidate logical forms from the fallback procedures, and the result of the first executable

Model	Overall		I.I.D.		Compositional		Zero-shot	
	EM	F1	EM	F1	EM	F1	EM	F1
SG-KBQA	**85.1**	**88.5**	**93.1**	**94.6**	**78.4**	**83.6**	**84.4**	**87.9**
w/o RG-EMD	81.3	85.3	90.6	92.4	74.4	80.2	80.2	84.3
w/o RG-CER	82.8	86.5	90.2	92.1	75.4	81.1	82.7	86.3
w/o DED	84.3	87.8	92.6	94.0	77.1	82.4	83.7	87.2
w/o SC	76.6	79.2	91.7	92.9	72.3	77.4	71.7	73.9
w/o Fallback LF	81.8	84.6	92.8	94.1	77.3	81.8	78.7	81.5

Table 6: Ablation study results on the validation set of GrailQA.

one is returned as the answer set.

Our system parameters are selected empirically. There are only a small number of parameters to consider. As shown in the parameter study later, our model performance shows stable patterns against the choice of parameter values. The parameter values do not take excessive fine-tuning.

F Full Ablation Study Results (GrailQA)

Table 6 presents the full ablation study results on the validation set of GrailQA. We observe a similar trend to that of the F1 score results reported earlier – all ablated model variants yield lower EM scores compared to the full model.

For the retrieval modules, RG-EMD improves the F1 score by 3.2 points and the EM score by 3.8 points on GrailQA (i.e., SG-KBQA vs. SG-KBQA w/o RG-EMD for overall results), while achieving a 1.9-point increase in the F1 score on WebQSP (see Table 2 earlier). It achieves an increase of 3.4 points or larger in the F1 score on the compositional and zero-shot tests, which is larger than the 2.2-point improvement on the I.I.D. tests. This shows that relation-guided mention detection effectively enhances the generalization capability of KBQA entity retrieval. For the other module RG-CER, removing it (SG-KBQA w/o RG-CER) results in a 2.5-point drop in the F1 score for both the I.I.D. and compositional tests, while the impact is smaller on the zero-shot tests (1.6 points). This is because the lower accuracy in relation retrieval under zero-shot tests leads to error propagation into relation-guided candidate entity retrieval, reducing the benefits of this module.

For the generation modules, SG-KBQA w/o DED negatively impacts the F1 scores on both GrailQA and WebQSP, confirming that deferring entity disambiguation effectively mitigate error propagation between the retrieval and generation stages. For SG-KBQA w/o SC, it reduces the F1 score by 1.7 points and 3.2 points on the GrialQA I.I.D. tests and on WebQSP. The drop is more significant on the compositional and zero-shot tests, i.e., by 6.2 points and 14.0 points, respectively. This indicates that schema contexts can effectively guide the LLM to reason and identify the correct combinations of KB elements unseen at training.

In Table 6, we present an additional model variant, SG-KBQA w/o Fallback LF, which removes the fall back logical form generation strategy from SG-KBQA. We see that SG-KBQA has lower accuracy without the strategy. We note that this fallback strategy is *not* the reason why SG-KBQA outperforms the baseline models. TIARA also uses this fallback strategy, while RetinaQA uses the top executable logical form from the fallback strategy as one of the options to be selected by its discriminator to determine the final logical form output.

G Full Module Applicability Results

To evaluate the applicability of our proposed modules, we conduct a module applicability study with TIARA (an open-source retrieve-then-generate baseline) and different generation models (i.e., T5-base and DeepSeek-R1-Distill-Llama-8B).

Table 7 reports the results. Replacing TIARA's entity retrieval module with ours (TIARA w RG-EMD & RG-CER) helps boost the EM and F1 scores by 4.2 and 2.4 points overall, comparing against the original TIARA model. This improvement is primarily from the tests with KB elements or compositions that are unseen at training, as evidenced by the larger performance gains on the compositional and zero-shot tests, i.e., 3.3 and 2.6 points in the F1 score, respectively. Similar patterns are observed for TIARA w DED & SC that replaces TIARA's logical form generation module with ours. These results demonstrate that our proposed modules can enhance the retrieval and generation steps of other compatible models, especially under non-I.I.D. settings.

Further, using the same language model (i.e., T5-

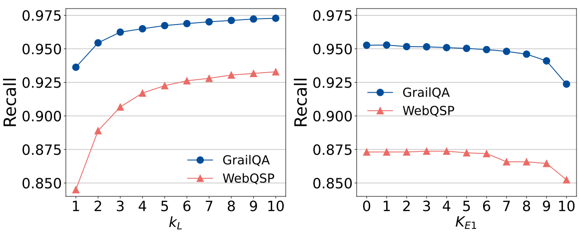
Model	Overall		I.I.D.		Compositional		Zero-shot	
	EM	F1	EM	F1	EM	F1	EM	F1
TIARA (T5-base)	75.3	81.9	88.4	91.2	66.4	74.8	73.3	80.7
w RG-EMD & RG-CER	79.5	84.3	90.3	92.3	71.2	78.1	78.3	83.3
w DED & SC	79.9	85.6	88.6	92.3	72.7	79.8	79.0	85.0
SG-KBQA	**85.1**	**88.5**	**93.1**	**94.6**	**78.4**	**83.6**	**84.4**	**87.9**
w T5-base	80.6	84.9	89.9	92.6	73.8	81.0	79.4	83.3
w DS-R1-8B	83.6	87.5	92.3	94.0	75.4	82.4	83.1	86.7

Table 7: Full module applicability results on the validation set of GrailQA.

base in TIARA) to form logical form generation modules, our model SG-KBQA w T5-base still outperforms TIARA by 5.3 points 3.0 points in the EM and F1 scores for the overall tests. This confirms that the overall effectiveness of our model stems from its design rather than the use of a larger model for logical form generation. As for SG-KBQA w/ DS-R1-8B, it reports close performance to SG-KBQA, indicating that SG-KBQA does not rely on a particular LLM.

H Parameter Study

We conduct a parameter study to investigate the impact of the choice of values for our system parameters. When the value of a parameter is varied, default values as mentioned in Appendix E are used for the other parameters.



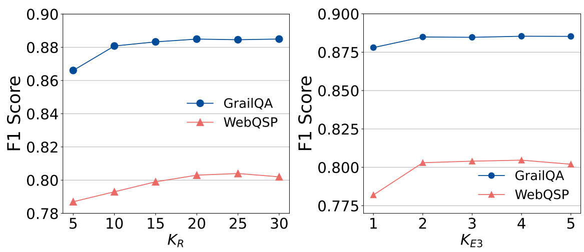
(a) k_L (b) k_{E1}

Figure 6: Impact of k_L and k_{E1} on the recall of candidate entity retrieval.

Figure 6 presents the impact of k_L and k_{E1} on the recall of candidate entity retrieval (i.e., the average percentage of ground-truth entities returned by our candidate entity retrieval module for each test sample). Here, for the GrailQA dataset, we report the results on the overall tests (same below). Recall that k_L means the number of logical form sketches from which entity mentions are extracted, while k_{E1} refers to the number of candidate entities retrieved based on the popularity scores.

As k_L increases, the recall of candidate entity retrieval grows, which is expected. The growth diminishes gradually. This is because a small number of questions contain complex entity mentions that are difficult to handle (see error analysis in Appendix K). As k_L increases, the precision of the retrieval also reduces, which brings noise into the entity retrieval results and additional computational costs. To strike a balance, we set $k_L = 3$ for GrailQA and $k_L = 4$ for WebQSP. We also observe that the recall on WebQSP is lower than that on GrailQA. This is because WebQSP has a smaller training set to learn from.

As for k_{E1}, when its value increases, the candidate entity recall generally drops. This is because an increase in K_{E1} means to select more candidate entities based on popularity while fewer from those connected to the top retrieved relations but with lower popularity scores. Therefore, we default k_{E1} at 1 for GrailQA and 3 for WebQSP, which yield the highest recall for the two datasets, respectively. Recall that we set the total number of candidate entities for each entity mention to 10 ($K_{E1} + K_{E2} = 10$), following our baselines (e.g., TIARA, RetinaQA, and Pangu). Therefore, we omit another study on K_{E2}, as it varies with K_{E1}.



(a) k_R (b) k_{E3}

Figure 7: Impact of k_R and k_{E3} on the overall F1 score.

Figure 7 further shows the impact of k_R and k_{E3} – recall that k_R is the number of top candidate relations considered, and k_{E3} is the number of candidate entities matched for each entity mention. Now we show the F1 scores, as these parameters are used by our schema-guided logical form gener-

Question: What is the name for the atomic units of length?	
SpanMD: What is the name for the atomic units of length?	(✗)
Ours:	
Retrieved Relations: measurement_unit.measurement_system.length_units, measurement_unit.time_unit.measurement_system, measurement_unit.measurement_system.time_units...	
Generated Logical Form Sketch: (AND <class> (JOIN <relation> [atomic units]))	(✓)

Table 8: Case study of entity mention detection by our model and SpanMD (a mention detection method commonly used by SOTA KBQA models) on the GrailQA validation set. The incorrect entity mention detected is colored in red, while the correct entity mention detected is colored in blue.

ation module. They directly affect the accuracy of the generated logical form and the corresponding question answers.

On GrailQA, increasing either k_R or k_{E3} leads to higher F1 scores, although the growth becomes marginal eventually. On WebQSP, the F1 scores peak at $k_R = 25$ and $k_{E3} = 4$. These results suggest that feeding an excessive number of candidate entities and relations to the logical form generator module has limited benefit. To avoid the extra computational costs (due to more input tokens) and to limit the input length for compatibility with smaller Seq2Seq models (e.g., T5-base), we use $k_R = 20$ and $k_{E3} = 2$ on both datasets.

I Model Running Time

SG-KBQA takes 26 hours to train on the GrailQA dataset and 13.6 seconds to run inference for a test sample. It is faster on WebQSP which is a smaller dataset. Note that more than 10 hours of the training time were spent on the fallback logical form generation. If this step is skipped (which does not impact our model accuracy substantially as shown earlier), SG-KBQA can be trained in about half a day. Another five hours were spent on fine-tuning the LLM for logical form generation, which can also be reduced by using a smaller model.

As there is no full released code for the baseline models, it is infeasible to benchmark against them on model training time. For model inference tests, TIARA has a partially released model (with a closed-source mention detection module). The model takes 11.4 seconds per sample (excluding the entity mention detection module) for inference on GrailQA, which is close to that of SG-KBQA. Therefore, we have achieved a model that is more accurate than the baselines while being at least as fast in inference as one of the top performing baselines (i.e., TIARA+GAIN which shares the same inference procedure with TIARA).

J Case Study

To further show SG-KBQA's generalizability to non-I.I.D. KBQA applications, we include a case study from the GrailQA validation set as shown in Tables 8 and 9.

Entity Mention Detection Figure 8 shows an entity mention detection example, comparing our entity detection module with SpanMD which is a mention detection method commonly used by SOTA KBQA models (Shu et al., 2022; Ye et al., 2022; Faldu et al., 2024). In this case, SpanMD incorrectly detects length as an entity mention, which is actually part of the ground-truth relation (measurement_unit.. . ..length_units) that is unseen in the training data. Our entity mention detection module, on the other hand, leverages the retrieved relations to generate a logical form sketch. The correct entity mention, atomic units, is isolated from the relations and can be corrected extracted, even though this entity mention has not been seen at training.

Logical Form Generation Table 9 shows a logical form generation example. Here, SG-KBQA and TIARA (a representative generation-based model) have both retrieved the same sets of relations in the retrieval stage which include false positives. The two models also share the same top-1 retrieved entity m.04fgkzf, while SG-KBQA has retrieved a second entity m.02hcty in addition. TIARA is misled by the erroneous KB relations retrieved and produces an incorrect logical form. SG-KBQA, on the other hand, is able to produce the correct logical form by leveraging the schema information (i.e., the entity's class and the relation's domain and range classes).

K Error Analysis

Following TIARA (Shu et al., 2022) and Pangu (Gu et al., 2023), we analyze 200 incorrect predictions

Question: Captain pugwash makes an appearance in which comic strip?			
	Relation Retrieval	**Entity Retrieval**	
TIARA	...comic_strips_appeared_in ...character	Captain Pugwash m.04fgkzf	
	(AND comic_strips.comic_strip_character (JOIN comic_strips.comic_strip_character.comic_strips_appeared_in m.04fgkzf))		(✗)
Ours	[D] comic_strip_character [N] comic_strips_appeared_in [R] comic_strip	[ID] m.04fgkzf [N] Captain Pugwash [C] comic_strip	
	[ID] comic_strip [N] character [R] comic_strip_character	[ID] m.02hcty [N] Captain Pugwash [C] comic_strip_character	
	(AND comic_strips.comic_strip (JOIN comic_strips.comic_strip.characters m.02hcty))		(✓)

Table 9: Case study of logical form generation by SG-KBQA and a representative competitor TIARA on the GrailQA validation set. Incorrect relations and entities are marked in red, while the correct relations and entities are colored in blue.

randomly sampled from each of the GrailQA validation set and the WebQSP test set where our model predictions are different from the ground truth. The errors of SG-KBQA largely fall into the following three types:

- **Relation retrieval errors** (35%). Failures in the relation retrieval step (e.g., failing to retrieve any ground-truth relations) can impinge the capability of our entity mention detection module to generate correct logical form sketches, which in turn leads to incorrect entity mention detection and entity retrieval.

- **Entity retrieval errors** (32%). Errors in the entity mentions generated by the logical form sketch parser can still occur even when the correct relations are retrieved, because some complex and unseen entity mentions require domain-specific knowledge. An example of such entity mentions is 'Non-SI units mentioned in the SI', which refers to units that are not part of the International System (SI) of Units but are officially recognized for use alongside SI units. This entity mention involves two concepts that are very similar in their surface forms (Non-SI and SI). Without a thorough understanding of the domain knowledge (SI standing for International System of Units), it is difficult for the entity mention detection module to identify the correct entity boundaries.

- **Logical form generation errors** (31%). Generation of inaccurate or inexecutable logical forms can still occur when the correct entities and relations are retrieved. The main source of such errors is questions involving operators rarely seen in the training data (e.g., ARGMIN and ARGMAX). Additionally, there are highly ambiguous candidate entities that may confuse the model, leading to incorrect selections of entity-relation combinations. For example, for the question Who writes twilight zone, two candidate entities m.04x4gj and m.0d_rw share the same entity name twilight zone. The former refers to a reboot of the TV series The Twilight Zone produced by Rod Serling and Michael Cassutt, while the latter is the original version of The Twilight Zone independently produced by Rod Serling. They share the same entity name and class (tv.tv_program). There is insufficient contextual information for our logical form generator to differentiate between the two. The generator eventually selected the higher-ranked entity which was incorrect, leading to producing an incorrect answer to the question Rod Serling and Michael Cassutt.

- The remaining errors (2%) stem from incorrect annotations of comparative questions in the dataset. For example, larger than in a question is annotated as LE (less equal) in the ground-truth logical form.