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 and their novel compositions 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 exploits information about the semantics and structure of the knowledge base 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 from 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 (Zhou et al., 2018), recommendation (Guo et al., 2022), and information retrieval (Jalota et al., 2021).

Semantic parsing (SP) has been shown to be an effective method for KBQA, where the core idea is to translate the input natural language question into a structured logical form (e.g., SPARQL or S-Expression (Gu et al., 2021)), which is then executed to yield the question answer.

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. Given a large number of entities and relations in ambiguous surface forms, and the flexibility in questions expressed in natural language, this mapping process typically yields a set of candidate entities (relations) for each mentions of entities (relations). *The challenge then becomes to uncover the right composition of entities and relations from the sets of candidates.*

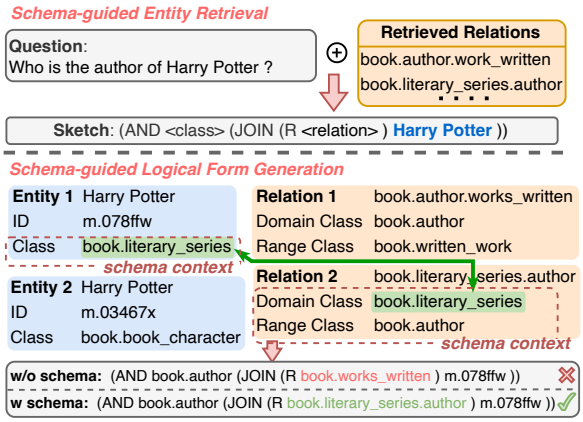


Figure 1: Schema-guided entity retrieval (top) and logical form generation (bottom). Green arrows and boxes highlight schema-level connections (overlapping classes) among KB elements. "w schema" and "w/o schema" logical forms denote whether the composition of KB elements adheres to the KB schema, respectively.

Figure 1 shows an example. Two entities named Harry Potter (a book series and a main character in them) and two authorship relations of similar names were identified as candidates. Combining the top-ranked entity (the book series) with the top-ranked relation book.author.works_written (book authorship) yields an invalid and unexecutable logical form, as the entity is a book series, not a book.

Due to the vast number of KB elements and their compositions, it is difficult (if not impossible) to train a model with all feasible compositions of KB elements that might be queried. 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. When a KB element composition is unseen at training, errors like the example above may occur.

A few studies consider model generalizability to non-I.I.D. settings, where the test set contains schema items (i.e., relations, classes and functions) or their compositions that are unseen during training (i.e., *zero-shot* and *compositional generalization*, respectively). They propose retrieval methods to retrieve KB elements or compositions more relevant to input questions, and use them to construct logical forms (Shu et al., 2022; Gu et al., 2023).

Despite these efforts, achieving compositional and zero-shot generalization remains challenging: (1) **Entity retrieval is a bottleneck.** Current entity retrieval methods often fail to accurately detect entities mentioned in questions containing schema items unseen at training. This is because such items introduce novel contextual patterns in questions, making it difficult to identify the correct boundary of entity mentions. The resulting entity retrieval errors propagate and lead to errors in the logical forms generated. (2) **Schema-level connection between KB elements have been missed.** Existing methods are not explicitly trained to capture compositions of KB elements that are feasible based on their schema. Instead, they tend to reproduce KB element compositions observed at training, making them difficult to generalize to unseen compositions.

To address these challenges, we propose a schema-guided model for KBQA (SG-KBQA), that incorporates KB schema to guide both entity retrieval and logical form generation. Unlike previous approaches that initiate the pipeline with entity retrieval, SG-KBQA adopts a schema-first principle, prioritizing schema understanding as the foundation for downstream logical form generation.

SG-KBQA begins with relation retrieval, employing a pre-trained language model (PLM)-based (Devlin et al., 2019) retriever to retrieve top-ranked relations from the KB that are most relevant to the input question. Benefiting from the generalization capability of pre-trained language models (and that there are much fewer relations and their surface form variants than entities in a KB), relations that are semantically similar to the question—yet unseen at training—can still be included among the top-ranked retrieved relations (as validated in our study). Then, as illustrated in Figure 1, we introduce a schema-guided entity retrieval (SER) module. This module employs a logical form sketch parser that converts the input question and retrieved relations into logical form sketches by a Seq2Seq model. These top-ranked relations provide schema context that is relevant to the question but not observed at training, hence helping the model distinguish actual entity mentions from unseen schema items in the question. More precise entity mentions are then extracted from the generated sketches, thereby improving the zero-shot generalizability of entity retrieval.

To further mitigate error propagation between the retrieval and generation stages, we defer entity disambiguation to the logical form generation stage. For each entity mention, all top-ranked matched candidates are retained as candidate entities for the preceding generation stage.

Further exploiting the schema-guided idea, we propose a schema-guided logical form generation (SLFG) module that fine-tunes a large language model (LLM) to reason over feasible compositions of KB elements based on their underlying schema contexts. As Figure 1 shows, we feed the input question, the retrieved candidate relations and entities, plus their corresponding schema contexts, i.e., the (domain and range) classes of the relations and entities, into the LLM for logical form generation. The domain and range classes of a relation refer to the classes to which its subject and object entities belong. *Together with the class of the candidate entities, they provide explicit training signals to guide the LLM to look for KB elements that can be connected together (and hence are more likely to form executable logical forms).* As a result, our SLFG module generalizes to compositions of KB elements unseen at training.

To summarise: (1) We introduce SG-KBQA to solve the KBQA problem under non-I.I.D. settings, where test input contains unseen schema items or their compositions during training. (2) We introduce schema-guided modules for entity retrieval and logical form generation with deferring entity disambiguation to enhance both compositional and zero-shot generalization. These modules can also be incorporated into existing SP-based KBQA systems to improve their generalization performance. (3) 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 Graph Question Answering Early KBQA solutions can be widely categorized as information retrieval-based (IR-based) (He et al., 2021; Zhang et al., 2022) or semantic parsing-based (SP-based) solutions (Cao et al., 2022; Ye et al., 2022).

Benefiting from the strong natural language understanding and reasoning abilities of LLMs, recent LLM-based KBQA methods have achieved promising results. A branch of work employs LLMs to produce reasoning trajectories or step-by-step tool invocation over a question-specific subgraph to obtain question answers, assuming that topic entities are given (Sun et al., 2024; Luo et al., 2024b; Sui et al., 2025).

Others follow the SP-based paradigm, using LLMs to generate approximate logical form sketches through few-shot in-context learning or fine-tuning (Cao et al., 2022; Li et al., 2023, 2024; Luo et al., 2024a; Wang and Qin, 2024). The inaccurate or ambiguous KB elements in the generated sketches are further refined through a retrieval stage, aligning them with actual KB elements to construct complete logical forms.

However, these methods often fail to generalize over test questions that refer to KB elements or their compositions unseen during training, or when the topic entities are not known. Our SG-KBQA improves KBQA generalizability through a schema-guided approach. While we also use LLMs to generate logical form sketches, we incorporate retrieved relations to guide sketch generation for entity mention extraction, thereby improving the generalizability of entity retrieval, while *we do not refine these sketches to produce the final output logical forms*.

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 (Gu et al., 2021, 2023) start from retrieved entities, traverse the KB, and construct the target logical form by ranking the traversed paths.

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 the top-5 candidate logical forms enumerated from the retrieved entities. Shu et al. (2022) further use top-ranked relations, *disambiguated entities*, and classes (retrieved *separately*). Zhang et al. (2023) use connected pairs of retrieved KB elements.

Our SG-KBQA adopts a generation-based approach, training the LLM to reason over candidate entities and relations, using their schema contexts (i.e., classes) to infer connectivity. This enables the model to compose novel logical forms without seeing them at training, hence generalizing better in larger, noisier search spaces. Additionally, we defer entity disambiguation to the generation stage, mitigating error propagation caused by early disambiguation without context.

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 used for entity mention detection from input questions. To retrieve KB entities for the entity mentions, the FACC1 dataset (Gabrilovich et al., 2013) is often used, with over 10 billion surface forms and popularity scores of Freebase entities. Gu et al. (2021) use 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, $\mathcal{R}$ a set of relations, and $\mathcal{L}$ a set of literals, e.g., textual labels or numerical values. 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 relation between the two.

The ontology defines the rules governing the composition of relational facts within $\mathcal{G}$: $\mathcal{C}$ 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 the *domain class*, i.e., the class of subject entities that satisfy relation r; c_r is the *range class*, i.e., the class of object entities or literals satisfying r. Each ontology triple can be instantiated a set of relational facts, with an example provided in Appendix A.

Problem Statement Given a KB $\mathcal{G}$ and a question q expressed in natural language, i.e., a sequence of word tokens, KBQA aims to find a subset (the answer set) $\mathcal{A} \subseteq \mathcal{E} \cup \mathcal{L}$ of elements from $\mathcal{G}$ that

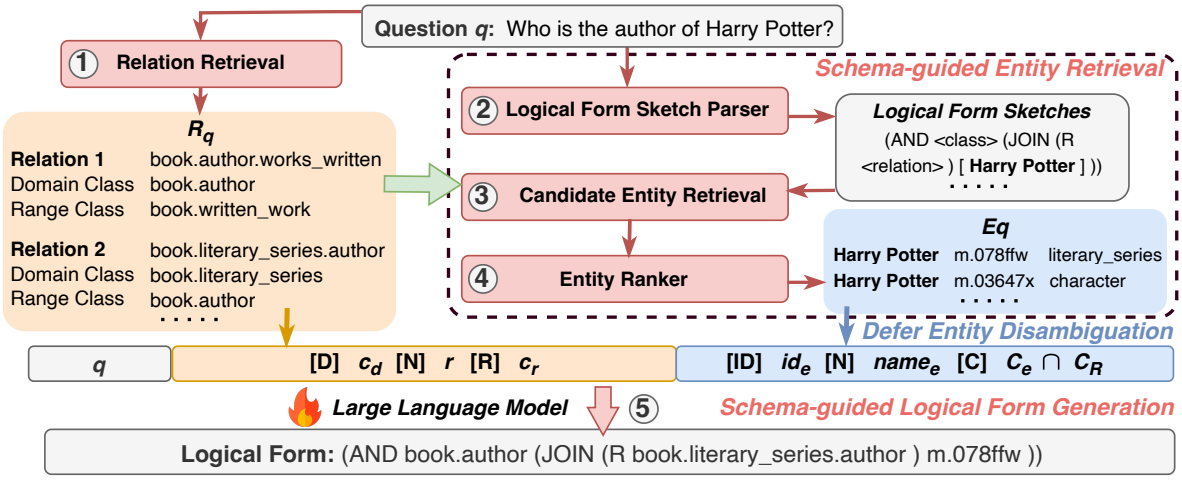


Figure 2: Overview of SG-KBQA. The model consists of two novel modules: *schema-guided entity retrieval* (SER) and *schema-guided logical form generation* (SLFG). Given a question q, the model first retrieves and ranks candidate relations (①). In SER, q and the top-ranked relations R_q are used to generate logical form sketches and extract entity mentions (②). Based on these mentions and R_q, the model retrieves and ranks candidate entities (③), producing the top entities E_q (④). Entity disambiguation is deferred by directly passing E_q to SLFG. In SLFG, q, R_q, E_q, and their class contexts are fed into a fine-tuned language model for logical form generation (⑤).

— with optional application of some aggregation functions (e.g., COUNT) — answers q.

Logical Form We approach the KBQA problem by translating question q into a structured query that can be executed on $\mathcal{G}$ to fetch the answer set $\mathcal{A}$. Following previous works (Shu et al., 2022; Gu et al., 2023; Zhang et al., 2023), we use logical form as the structured query language, expressed in *S-expression* (Gu et al., 2021). 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), with more details in Appendix A.

4 The SG-KBQA Model

SG-KBQA takes a generation-based approach overall. It introduces two novel modules: *Schema-guided Entity Retrieval* (SER) and *Schema-guided Logical Form Generation* (SLFG), designed to enhance generalizability and shown in Figure 2.

SG-KBQA starts with relation retrieval, where a BERT-based relation ranking model retrieves candidate relations and entities from the KB $\mathcal{G}$ that are potentially relevant to the question q.

In SER, q and the top-ranked candidate relations are passed into a logical form parser (i.e. a Seq2Seq model) to generate logical form sketches that contain entity mentions while masking out relations and classes. The retrieved relations provide the most relevant—and potentially unseen—relations as additional schema context, enabling the model to identify boundaries of entity mentions more accurately as explained in Section 1. We then harvest these entity mentions and use them to retrieve candidate entities from $\mathcal{G}$, thereby improving the non-I.I.D. generalizability of entity retrieval.

To further improve the accuracy of entity retrieval, we propose a combined schema-based pruning strategy to filter out unlikely candidates, as a single mention may correspond to multiple entities. The remaining entities are then ranked by a BERT-based model, which estimates the likelihood of each entity being the correct match for a mention. Leveraging relations—a type of schema item—to guide both entity mention extraction and candidate entity pruning enhances model generalizability over entities unseen at training. This in turn helps logical form generation to filter false positive matches for unseen relations or their compositions.

In SLFG, SG-KBQA feeds q, the top-ranked relations and entities (corresponding to each mention), and the schema contexts, i.e., their class information, into an adapted LLM to generate the logical form and produce answer set $\mathcal{A}$. SLFG is novel in that it takes (1) multiple candidate entities (instead of one in existing models) for each mention and (2) the schema contexts as the input.

By deferring *entity disambiguation* to the generation stage, our approach helps mitigate error propagation that often arises from early-stage disam-

biguation. This strategy also brings challenges, as the extra candidate entities (which often share the same or similar names) introduce noise in SLFG.

To address this and enhance generalizability to unseen compositions of KB elements, we incorporate schema context in SLFG. The LLM is fine-tuned to generate logical forms that consist of valid KB element compositions, as connected based on the (domain and range) classes of the relations and entities. As a result, the model is able to select correct compositions from a noisy KB element space, even under non-i.i.d settings.

4.1 Relation Retrieval

For relation retrieval, we follow TIARA (Shu et al., 2022) for its 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 to learn the semantic similarity between q and a relation $r \in \mathcal{R}$:

$$\text{sim}(q, r) = \textsc{Linear}(\textsc{BertCls}([q; r])), \quad (1)$$

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.

4.2 Schema-guided Entity Retrieval

Entity Mention Detection Given R_q, we propose a schema-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.

Candidate Entity Retrieval We follow previous studies (Faldu et al., 2024; Luo et al., 2024a; Shu et al., 2022) and use an entity name dictionary FACC1 (Gabrilovich et al., 2013) to map extracted entity mentions to entities (i.e., their IDs in KB),

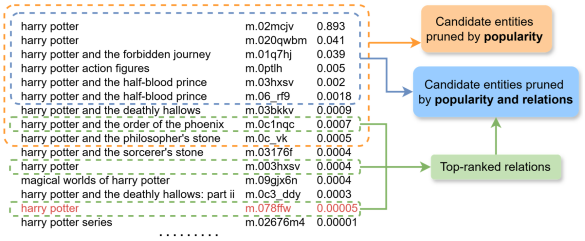


Figure 3: Candidate entity retrieval for 'Harry Potter'. The candidate entity in red is the ground-truth.

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 of the entities (Shu et al., 2022; Ye et al., 2022).

To improve the recall, we propose a combined pruning strategy based on both popularity and relations. As Figure 3 shows, we first select the top-k_{E1} (system parameter) entities for each 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.

4.3 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 class information to help the model learn their connections and generalize to unseen entities, relations, or their compositions. The context of a relation r is described by concatenating its 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 between 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 2 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:

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

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 class 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 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 traverse the KB starting from the retrieved entities. Due to the exponential growth in the candidate paths with each hop, we start from the top-1 entity for each mention and examine its neighborhood for up to two hops. The paths are converted into logical forms. During ranking, a BERT-based ranker scores q and each logical form l (like Eq. 1). We train the ranker using a contrastive objective:

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

where l^* is the ground-truth logical form and C_l is the set of enumerated logical forms. The top-ranked, executable logical form is returned.

5 Experiments

We run experiments to answer: **Q1**: How does SG-KBQA improve generalizability compared with SOTA models? **Q2**: How do model components contribute to generalizability? **Q3**: How can our techniques enhance existing 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 generalizability of KBQA models. It has 64,331 questions with target S-expressions, including complex questions requiring up to 4-hop reasoning over the KG. 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** (SOTA before SG-KBQA, as of 19th May, 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-based (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 LLM-based supervised fine-tuning (SFT) models including **ChatKBQA** (SOTA) (Luo et al., 2024a), **TFS-KBQA** (SOTA) (Wang and Qin, 2024) and **RoG** (Luo et al., 2024b). We also compare with few-shot LLM-based methods: KB-Binder (6)-R, Pangu (Codex), FlexKBQA, **ToG** (Sun et al., 2024) and **FiDeLiS** (Sui et al., 2025). Appendix D details these models. The baseline results are collected from their papers or the GrailQA leaderboard (when available).

Implementation Details All our experiments are run on a machine with an NVDIA A100 GPU and 120 GB of RAM. For each dataset, a T5-base model is fine-tuned for 5 epochs as our logical form sketch parser. 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 are selected empirically. There are only a small number of parameters to consider. As shown in

[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	81.8	58.6	72.3
	Pangu (T5-3B) (ACL 2023)	75.4	81.7	84.4	88.8	74.6	81.5	71.6	78.5
	FC-KBQA (ACL 2023)	73.2	78.7	88.5	91.2	70.0	76.7	67.6	74.0
	TIARA+GAIN (EACL 2024)	76.3	81.5	88.5	91.2	73.7	80.0	71.8	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-shot in-context learning).

	Model	F1
IR-based	SR+NSM (ACL 2022)	69.5
	UniKGQA (ICLR 2023)	75.1
	EPR+NSM (WWW 2024)	71.2
LLM-based (SFT)	Pangu (T5-3B) (ACL 2023)	79.6
	RoG (ICLR 2024)	69.8
	ChatKBQA (ACL 2024)	79.8
	TFS-KBQA (LREC-COLING 2024)	79.9
LLM-based (Few-shot)	KB-Binder (6)-R (ACL 2023)	53.2
	Pangu (Codex) (ACL 2023)	54.5
	FlexKBQA (AAAI 2024)	60.6
	ToG (ICLR 2024)	69.5
	FiDeLiS (ACL 2025)	78.3
Ours (SFT)	**SG-KBQA**	**80.3**
	- Improvement	+0.5%

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

the parameter study in Appendix H, our model performance shows stable patterns against the choice of parameter values. The parameter values do not take excessive fine-tuning. 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 EM and F1 scores. While the performance improvement in the I.I.D. setting is smaller, SG-KBQA achieves substantial gains in non-i.i.d scenarios. For example, under the compositional generalization setting, it increases EM by 4.4% and F1 by 4.0% over the best baseline models. Notably, under non-I.I.D. settings, the improvement in EM is consistently larger than that in F1, indicating that SG-KBQA is more capable of generating logical forms that precisely match both the questions and the KB schema, thanks to the class information that indicate the connections between the KB elements.

Most few-shot LLM-based competitors are generally not very competitive, especially under the non-I.I.D. settings. FiDeLiS, a recent work, achieves results that are close to the SOTA supervised fine-tuning model under the I.I.D. setting (within 2 F1 points). This suggests that LLMs, when guided with appropriate intermediate reasoning steps, hold promising potential for KBQA.

Results on WebQSP On WebQSP, which has an I.I.D. test set, the performance gap of different models are closer. Even in this case, SG-KBQA still performs the best, showing its applicability. Comparing with TFS-KBQA (SOTA) and ChatKBQA, SG-KBQA improves the F1 score by 0.5%. Among IR-based methods, UniKGQA (SOTA) still performs much worse than SG-KBQA. The lower performance of IR-based methods is consistent with existing results (Gu et al., 2022).

Model	GrailQA				WebQSP
	Overall	I.I.D.	Comp.	Zero.	Overall
SG-KBQA	**88.5**	**94.6**	**83.6**	**87.9**	**80.3**
w/o SER	84.9	91.9	79.6	84.0	78.1
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 SER	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

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

5.3 Ablation Study (Q2)

Next, we run an ablation study with the following variants of SG-KBQA: **w/o SER** replaces our schema-guided entity retrieval with the entity linking results from TIARA (Shu et al., 2022) which is commonly used in the baselines (Gu et al., 2023; Faldu et al., 2024); **w/o DED** uses the top-1 candidate entity for each entity mention without deferring entity disambiguation; **w/o SC** omits schema contexts (classes) 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.

Schema-guided Entity Retrieval SG-KBQA w/o SER results drops in F1 by 4.0 and 3.9 points under the compositional and zero-shot generalization settings, respectively. This indicates that incorporating retrieved relations helps the model more accurately identify them and entity mention boundaries in questions involving unseen relations. Since the entity retrieval results directly serve as input to the generation stage, this improvement further enhances the model's overall generalization performance under non-I.I.D. settings. More discussion on entity retrieval results and a case study are provided in Appendix L and Appendix J, respectively.

Schema-guided Logical Form Generation SG-KBQA w/o DED (with schema contexts) reduces the F1 scores on both datasets, demonstrating the effectiveness of our DED strategy in reducing error propagation between the retrieval and generation stages. Meanwhile, SG-KBQA w/o SC (with deferred entity disambiguation but no class information) has the most significant drops in F1 under the compositional (7.2) and zero-shot (14.0) tests. This highlights the contribution of class information in enabling the model to understand the connections among retrieved KB elements, thereby facilitating the generation of correct logical forms. A case study illustrating the schema-guided logical form generation module is provided in Appendix J.

5.4 Module Applicability (Q3)

Our entity retrieval module **SER** and logical form generation module **DED & SC** can be applied to existing models to improve their generalizability under non-I.I.D. settings. We showcase such applicability with TIARA. As shown in Table 4, replacing TIARA's retrieval and generation modules with our schema-guided counterparts leads to F1 score improvements of up to 5.0 and 4.3 points in the compositional and zero-shot settings, respectively.

Table 4 further reports F1 scores of SG-KBQA when replacing LLaMA3.1-8B with **T5-base** (which is used by TIARA) 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 that the performance gains come from the incorporation of the class contexts instead of a more advanced backbone model.*

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 (class) contexts indicate KB element connections. 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

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.

Also, 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. 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.

We further noted several recent works in this highly competitive area, e.g., READS (Xu et al., 2025a) and MemQ (Xu et al., 2025b), which finetune LLMs for reasoning trajectory generation or step-wise tool invocation based on given topic entities. These studies represent parallel efforts to ours with a focus on I.I.D. settings. We plan to evaluate their performance in future work, particularly under settings where the topic entity is unavailable and in our non-I.I.D. scenarios.

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.

Shulin Cao, Jiaxin Shi, Zijun Yao, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jinghui Xiao. 2022. Program Transfer for Answering Complex Questions over Knowledge Bases. *arXiv preprint*. ArXiv:2110.05743 [cs].

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*.

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*.

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. 2024a. 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.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024b. Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning. In *ICLR*.

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, and 7 others. 2018. Never-ending learning. *Communications of the ACM*, 61(5):103–115.

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.

Yuan Sui, Yufei He, Nian Liu, Xiaoxin He, Kun Wang, and Bryan Hooi. 2025. FiDeLiS: Faithful Reasoning in Large Language Model for Knowledge Graph Question Answering. In *ACL*.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph. In *ICLR*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 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.

Mufan Xu, Kehai Chen, Xuefeng Bai, Muyun Yang, Tiejun Zhao, and Min Zhang. 2025a. LLM-based Discriminative Reasoning for Knowledge Graph Question Answering. *arXiv preprint arXiv:2412.12643*.

Mufan Xu, Gewen Liang, Kehai Chen, Wei Wang, Xun Zhou, Muyun Yang, Tiejun Zhao, and Min Zhang. 2025b. Memory-augmented Query Reconstruction for LLM-based Knowledge Graph Reasoning. *arXiv preprint arXiv:2503.05193*.

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 Basic Concepts

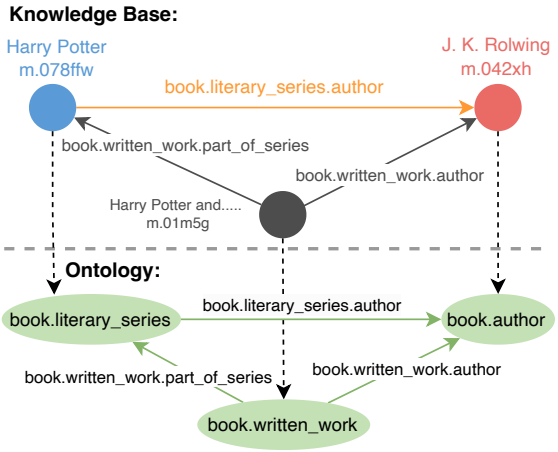


Figure 4: A subgraph of Freebase (top) and its corresponding ontology (bottom).

Ontology As shown in Figure 4, an ontology triple example is:

book.literary_series,
book.literary_series.author,
book.author

An instance of it is:

Harry Potter,
book.literary_series.author,
J.K. Rowling

Here, Harry Potter is an entity that belongs to class book.literary_series; J.K. Rowling is an entity that belongs to class book.author.

S-expressions 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 Table 6.

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

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).

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) is a two-branch model where one follows a ranking-based approach while the other follows a sketch-filling-based logical form construction method. It then uses a discriminator to determine the final output logical form from the two branches.We note that while we also generate logical form sketches. Such sketches are used for entity mention detection only and is not used to form the final output logical forms directly, i.e., our method is *not* sketch-filling-based.

- 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 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

Example Prompt

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

Question: Strong lyrics is the description of which video game rating?

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

[D] cvg.game_version
[N] cvg.game_version.rating
[R] cvg.computer_game_evaluation;

[D] cvg.computer_game_rating
[N] cvg.computer_game_rating.rating_system
[R] cvg.computer_game_rating_system;

[D] cvg.computer_game_rating_system
[N] cvg.computer_game_rating_system.content_descriptors
[R] cvg.computer_game_content_descriptors

...(Continue in the same manner for additional relations)

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

[ID] m.042zlv3
[N] Strong lyrics
[C] cvg.computer_game_content_descriptors

...(Continue in the same manner for additional entities)

Table 6: Example prompt to our fine-tuned LLM-based logical form generator for an input question: Strong lyrics is the description of which video game rating?

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., 2024a) 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.

- RoG (Luo et al., 2024b) fine-tunes an LLM to generate relation paths grounded in the KB, given an input question. These generated relation paths are then used to retrieve valid reasoning paths from the KB, enabling faithful reasoning to derive the final answers.

- ToG (Sun et al., 2024) performs step-by-step relation path selection using an LLM via few-shot in-context learning. At each step, based on newly extended paths, the model prompts the LLM to either answer the question based on the reasoning paths extended so far or continue with another step of path extension.

- FiDeLiS (Sui et al., 2025) combines semantic similarity metrics with graph-based connectivity to incrementally retrieve and extend reasoning paths. It further proposes Deductive-Verification Beam Search (DVBS), which converts the question into a declarative statement and uses an LLM to verify — by combining this statement with the current tail node of the reasoning path — whether the current node can be directly output as the answer or the path should be further extended.

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 one is returned as the answer set.

Following our baselines (Shu et al., 2022; Gu et al., 2023; Faldu et al., 2024), all retrieval and generation models used in our approach are trained separately on the training sets of GrailQA and WebQSP, and then evaluated on their respective validation and test sets. Specifically, for GrailQA, we follow previous works (Shu et al., 2022; Gu et al., 2023; Faldu et al., 2024) and adopt the official data splits described in Section 5.1 to evaluate generalization under non-I.I.D. settings. For WebQSP, we use the same data splits (described in Appendix C) as in previous studies (Shu et al., 2022; Gu et al., 2023; Luo et al., 2024a; Wang and Qin, 2024) to ensure fair comparison.

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.

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 SER	81.0	84.9	90.1	91.9	73.9	79.6	80.0	84.0
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 7: Ablation study results on the validation set of GrailQA.

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 SER	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

Table 8: Full module applicability results on the validation set of GrailQA.

F Full Ablation Study Results (GrailQA)

Table 7 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 schema-guieded entity retrieval modules, SER improves the F1 score by 3.7 points and the EM score by 4.1 points on GrailQA (i.e., SG-KBQA vs. SG-KBQA w/o SER for overall results), while achieving a 1.9-point increase in the F1 score on WebQSP (see Table 7 earlier). It yields an improvement of at least 3.9 F1 points and 4.5 EM points on the compositional and zero-shot test sets, which exceeds the gains observed on the I.I.D. test set (2.5 F1 points and 3 EM points). This shows that SER effectively enhances the generalization capability of KBQA entity retrieval.

For the schema-guided logical form generation module, 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 context (classes) is essential for guiding the LLM to reason over and identify the correct combinations of KB elements that were unseen during training.

In Table 7, 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

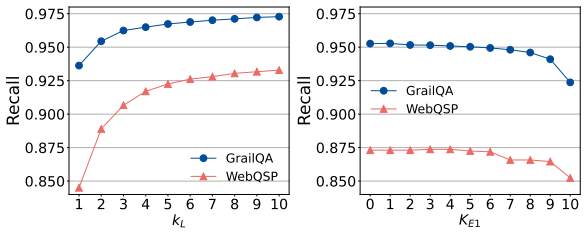
To evaluate the applicability of our proposed modules, we conduct a module applicability study with TIARA (an open-source baseline) and a different generation model T5-base.

Table 8 reports the results. Replacing TIARA's entity retrieval module with ours (TIARA w SER) 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 F1, 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-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.

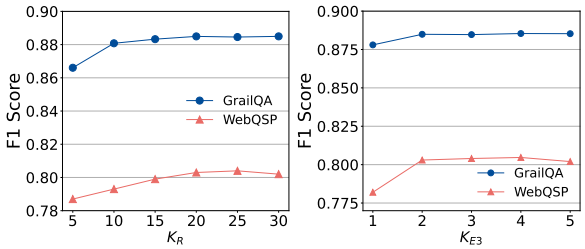
H Parameter Study

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



(a) k_R (b) k_{E3}

Figure 6: Impact of k_R and k_{E3} on the overall F1 score.

Figure 5 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}.

Figure 6 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 generation 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

Question: What is the name for the atomic units of length?	
SpanMD (span classification): length	(✗)
Ours (extracting from generated logical form sketches):	
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]))	(✓)
w/o SER: (AND <class> (JOIN <relation> [length]))	(✗)

Table 9: 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.

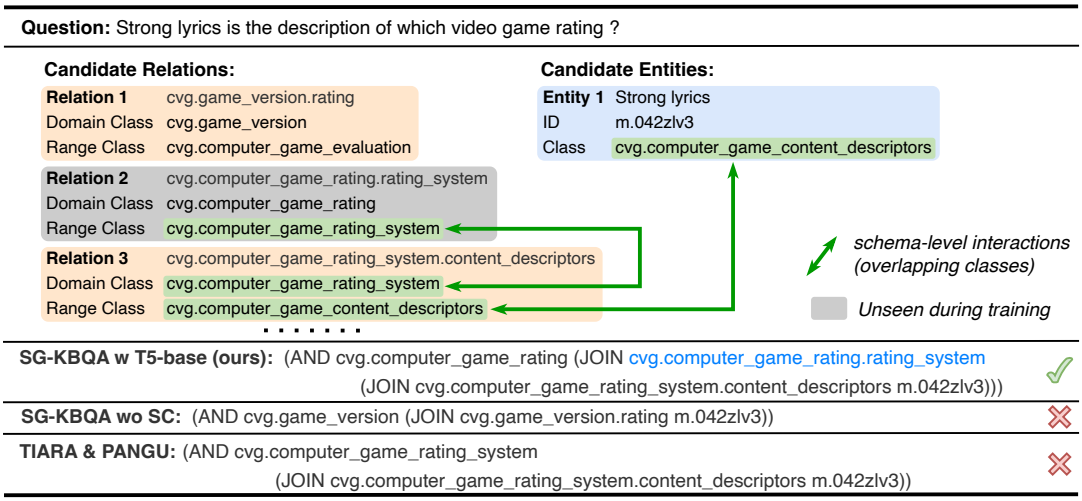


Figure 7: Case study of logical form generation by SG-KBQA and two representative competitors TIARA and PANGU on the GrailQA validation set. Green arrows and boxes highlight schema-level interactions (overlapping classes) among KB elements.

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 Table 9 and Figure 7.

Entity Mention Detection Figure 9 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 schema-guided entity retrieval module, on the other hand, leverages the retrieved relations as additional KB schema contexts to generate a logical form sketch.

By retrieving the unseen relation measurement_unit.measurement_system.length_units from the question through our relation retrieval module, the model is provided with schema-level information not encountered during training. This enables it to accurately identify the correct entity

mention, atomic units, from surrounding relation tokens—even when both the mention and relation are novel. Moreover, removing schema guidance (i.e., using a Seq2Seq model without SER) results in the same incorrect entity detection as SpanMD, highlighting the importance of schema-guidance in improving entity retrieval in non-I.I.D. scenarios.

Logical Form Generation Figure 7 shows a logical form generation example. Here, SG-KBQA and TIARA (a representative generation-based model) and PANGU (a representative ranking-based SOTA model) have both retrieved the same sets of relations and entities in the retrieval stage which include false positives and unseen ground-truth relation. Meanwhile, the three models also share the same retrieved entity m.042zlv3. Despite accessing the correct entity and relevant relations, both TIARA and PANGU fail to generate the correct logical form, instead producing a logical form that consists of a relation composition seen at training. In contrast, SG-KBQA successfully generates the correct logical form by leveraging schema context (i.e. the entity's class as well as the domain and range classes of the retrieved relations which overlap). This enables SG-KBQA to reason about the connectivity between the unseen ground-truth relation and the seen elements, demonstrating its superiority in generalizing to novel compositions of KB elements.

K Error Analysis

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

L Retrieval Performance

Entity Retrieval We report entity retrieval results under both I.I.D. and non-I.I.D. settings in Table 11, comparing our schema-guided entity retrieval module against baseline methods. Our

Steps\Models	SG-KBQA	TIARA	RnG-KBQA
Relation Retrieval	BERT-base-uncased	BERT-base-uncased	BERT-base-uncased
Logical Form Sketch Generation	T5-base	—	—
Entity Mention Detection	—	BERT-base-uncased	BERT-base-uncased
Entity Ranking	BERT-base-uncased	BERT-base-uncased	BERT-base-uncased
Logical Form Generation	LLaMA3.1-8B	T5-base	T5-base

Table 10: Backbone models used by SG-KBQA and baselines at each pipeline step

Model	Overall	I.I.D.	Comp.	Zero.
RnG-KBQA	80.4	86.6	83.3	76.5
TIARA	85.4	91.3	86.9	82.2
SG-KBQA	**90.5**	**94.0**	**90.8**	**88.8**
w/o SER	86.9	92.5	88.1	83.9

Table 11: F1 scores of KBQA entity retrieval methods under different generalization scenarios.

method consistently outperforms the strongest baseline (TIARA), with larger performance gains observed in non-I.I.D. scenarios. Specifically, compared to a 2.7-point improvement in F1 score under the I.I.D. setting, our method achieves gains of 3.9 and 6.6 points under the compositional and zero-shot generalization settings, respectively. Furthermore, removing schema guidance from our logical form parser leads to F1 drops of 2.7 points in the compositional setting and 4.9 points in the zero-shot setting. These results further highlight the effectiveness of our schema-guided entity retrieval in enhancing overall model generalizability.

For ease of comparison, we summarize the backbone models used by SG-KBQA and the baselines at each step of the pipeline in Table 10. In Section G, we report results showing that even when using the same backbone model (T5-base) as the baselines (TIARA), our SG-KBQA still outperforms them.

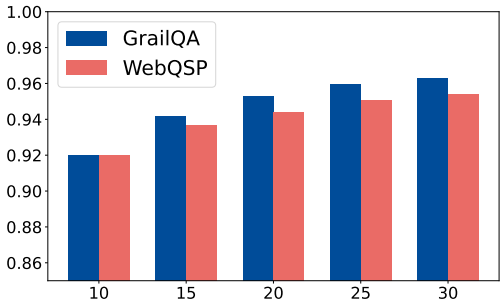


Figure 8: Recall of top-k relation retrieval on validation set of GrailQA and test set of WebQSP.

Relation Retrieval Figure 8 presents the recall of relation retrieval in SG-KBQA under different top-k settings. Overall, as k increases, the recall improves but with diminishing gains. Notably, even when $k = 20$, the relation recall on the GrailQA validation set remains above 94%, despite some question containing relations that were unseen during training. These results indicate that the cross-encoder-based retrieval model achieves high coverage of relevant relations, effectively retrieving both seen and unseen relations.

Although a larger k introduces more noise (negative relations), our SG-KBQA does not rely on highly precise relation retrieval. It leverages the top-20 retrieved relations to provide auxiliary context for the logical form sketch parser, and uses schema context to guide the LLM in selecting valid compositions of KB elements. Our experimental results show that SG-KBQA is capable of identifying the correct relations and their compositions from a noisy candidate set, thereby improving robustness and generalization.