Subgraph-Guided Executable Logical Form Generation for Knowledge Base Question Answering

Yuhang Tian[1], Dandan Song[1*], Zhijing Wu[1], Changzhi Zhou[1], Jun Yang[1], Huipeng Ma[1], Chenhao Li[1], Luan Zhang[1], Yading Li[1], Xudong Li[1], Shenxi Liu[1], Jing Jiang[2]

[1]School of Computer Science and Technology, Beijing Institute of Technology, China

[2]School of Computing, Australian National University

{tianyuhang,sdd}@bit.edu.cn

Abstract

Large Language Models (LLMs) have shown great potential in Knowledge Base Question Answering (KBQA) via semantic parsing. However, existing retrieval-augmented approaches typically retrieve entities and relations in isolation based solely on semantic similarity, ignoring the structural information of the Knowledge Base (KB) and the question. To address this limitation, we propose SELF-KBQA (**S**ubgraph-Guided **E**xecutable **L**ogical **F**orm Generation), a novel framework that empowers LLMs to generate logical forms conditioned on structurally aligned and semantically relevant subgraphs. Specifically, we introduce a structure-aware subgraph retrieval stage that ranks candidate subgraphs by aligning them with the question's structure, along with semantic relevance. Subsequently, we employ a token-budgeted evidence condensation strategy to distill the top-ranked subgraphs into compact contexts for the generation stage. Extensive experiments on GrailQA, WebQSP, and GraphQuestions demonstrate that SELF-KBQA achieves state-of-the-art performance.

1 Introduction

Large-scale Knowledge Bases (KBs) such as DBpedia (Auer et al., 2007), FreeBase (Bollacker et al., 2008), and Wikidata (Vrandečić and Krötzsch, 2014) contain millions of factual triples. Knowledge Base Question Answering (KBQA) is a fundamental task that aims to answer natural language questions based on the knowledge from the KBs (Lan et al., 2022).

With the rise of Large Language Models (LLMs) (Zhao et al., 2023), two main paradigms have emerged: (i) direct answer generation, where the LLM retrieves KB-relevant information and generates an answer (Sun et al., 2024; Chen et al., 2024; He et al., 2024; Mavromatis and Karypis, 2025; Sui et al., 2025; Li et al., 2024a); and (ii) semantic parsing, where the model generates an executable logical form (e.g., SPARQL (Pérez et al., 2009) or S-expressions (Gu et al., 2021)) that is executed against the KB to obtain the answer (Li et al., 2023; Nie et al., 2024a; Luo et al., 2024; Tian et al., 2024, 2025; Gao et al., 2025). The semantic parsing approach is adopted to achieve high precision through executable logical forms and to provide inherent interpretability for transparent error localization (Lan et al., 2021).

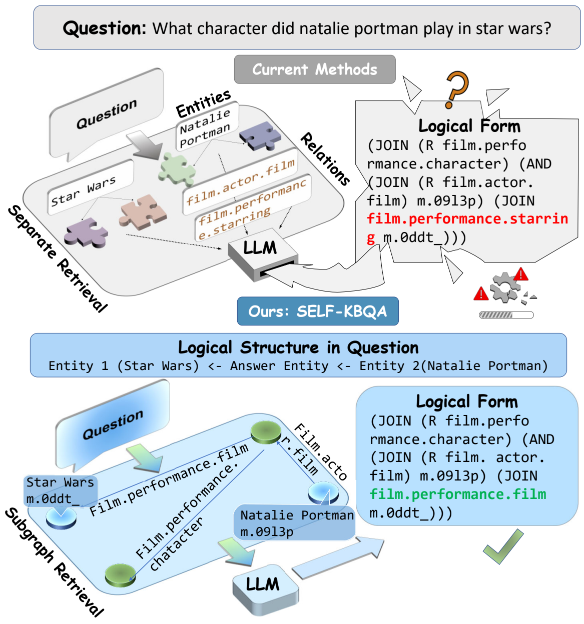

Figure 1: Comparison between current methods and our method. Identifiers such as m.0ddt_ denote entity IDs in the KB.

Early LLM-based semantic parsing approaches typically adopt a generate-then-retrieve paradigm (Luo et al., 2024). Due to the lack of direct access to the knowledge base, LLMs often generate incorrect entities or non-existent relations in the logical form generation stage. To mitigate this issue, recent methods (Nie et al., 2024a; Tian

[*]Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 3614–3635
July 2-7, 2026 ©2026 Association for Computational Linguistics

et al., 2024; Gao et al., 2025; Tian et al., 2025; Li et al., 2026b) retrieve question-relevant entities and relations from the knowledge base prior to generation, thereby providing question-related context and reducing factual errors in the generated logical forms.

Although current retrieval-augmented methods alleviate errors in LLM-generated logical forms, they typically retrieve entities and relations in isolation and ignore the structural information of the KB; consequently, they fail to capture the correspondence between a question's implicit logical structure and the KB structure, which is a major limitation. For example, as shown in the upper part of Figure 1, consider the question *"What character did natalie portman play in star wars?"*. Existing methods rank candidate entities and relations primarily by semantic similarity, thereby overlooking the crucial relation *film.performance.film*. This prevents the LLM from generating the correct logical form. In fact, a KBQA question can be mapped to a subgraph within the KB. For this question, the corresponding graph structure consists of the answer entity connected to the topic entity "Star Wars" and "Natalie Portman", which can be simplified as Entity_1 $\leftarrow$ answer_entity $\leftarrow$ Entity_2, as shown in the lower part of Figure 1. If we incorporate the KB's structural information during retrieval by retrieving from structured subgraphs and considering both semantic relevance and structural alignment between questions and subgraphs, the retrieved results will better support generating the correct logical form.

To address this problem, we propose **SELF-KBQA** (**S**ubgraph-Guided **E**xecutable **L**ogical **F**orm Generation for **K**nowledge **B**ase **Q**uestion **A**nswering), a framework that enables LLMs to generate logical forms under the guidance of question-relevant subgraphs that are both structurally aligned and semantically relevant. SELF-KBQA addresses this problem through a two-stage design. In the **Structure-aware Subgraph Retrieval** stage, we enumerate subgraph candidates starting from entity mentions and rank them along both structural and semantic dimensions. Specifically, for the structural dimension, we first predict a logical structure from the natural language question (e.g., topic_entity $\rightarrow$ mid_entity $\rightarrow$ answer_entity). We then score and rank candidate subgraphs based on two criteria: structural similarity to the predicted logical structure and semantic similarity to the question. In the **Subgraph-conditioned Logical Form Generation** stage, we employ a token-budgeted evidence condensation strategy to distill the top-ranked subgraphs into compact subgraph context, which is then used to guide the LLM in generating the correct logical form.

The main contributions of this work are three-fold:

- We propose SELF-KBQA, a novel framework that formulates KBQA as subgraph-guided logical form generation.
- We design a structure-aware retrieval strategy that ranks subgraphs based on both their structural similarity to the question's logical structure and semantic consistency.
- Extensive experiments on GrailQA, WebQSP, and GraphQuestions demonstrate that SELF-KBQA achieves state-of-the-art performance, validating the effectiveness of our method.

2 Related Work

Semantic Parsing for KBQA. Semantic parsing-based KBQA translates natural language questions into executable logical forms (e.g., SPARQL) (Pérez et al., 2009; Gu et al., 2021). Recently, Large Language Models (LLMs) have advanced the paradigm through few-shot prompting (Li et al., 2023) and code-style generation (Nie et al., 2024a; Luo et al., 2024). However, without explicit grounding in the KB structure, direct generation often suffers from *schema hallucination*, producing plausible but non-existent relations (Lan et al., 2021; Tian et al., 2025). This underscores the necessity of retrieval-augmented strategies to ground generation in valid evidence.

Retrieval-Augmented Generation for Semantic-based KBQA. To mitigate errors, recent KBQA methods retrieve knowledge from the KB to constrain the search space (Liu et al., 2023; Li et al., 2026a). Early approaches treated KB knowledge as unstructured text, such as linearizing triples (Yu et al., 2023) or retrieving isolated schema items (Shu et al., 2022), which often lose explicit structural information. To better capture connectivity, later work adopted path-based retrieval, including ranking candidate paths (Ye et al., 2022) and iterative hop-by-hop searches (Hu et al., 2022; Feng and He, 2025). SG-KBQA (Gao et al., 2025) further introduces schema-first retrieval with deferred entity disambiguation. However, these methods still struggle to model the *global* structural connectivity required by complex queries, as they rely on restricted path templates or greedy local decisions. In contrast, SELF-KBQA retrieves and prioritizes *subgraphs* by jointly considering their **structural compatibility** with the question logic and their **semantic relevance**, providing more comprehensive context for the LLM.

3 Preliminaries

In this section, we will introduce the two key components of a KBQA system: a Knowledge Base (KB) and Logical Forms.

Knowledge Base (KB) . A Knowledge Base stores knowledge as triples (s, r, o), where s is an entity, r is a relation (predicate), and o is either an entity or a literal. Literals represent attribute values (e.g., strings, numbers, dates) and typically appear only in the object position, since the subject is usually required to be an identifiable entity that can participate in further relations. The

Logical Forms. A logical form is an executable query over a KB that retrieves answers to a natural language question, such as SPARQL or S-Expressions. We focus on S-Expressions, which are compact Lisp-style queries written in prefix notation, where each expression compositionally denotes a set of entities (or a value). For example, `(AND common.image (JOIN common.image.appears_in_topic_gallery m.04qs1r))` is an S-Expression in which `JOIN` performs a one-hop traversal, i.e., `(JOIN r e)` returns entities x such that (x, r, e) holds in the KB, and `AND` computes the intersection of entity sets. `m.04qs1r` denotes entity in the KB. Details of all operators are provided in Appendix A.

4 Methodology

SELF-KBQA consists of two stages: **Structure-aware Subgraph Retrieval** and **Subgraph-conditioned Logical Form Generation**, as shown in Figure 2. In the first stage, we retrieve and rank candidate subgraphs from the knowledge base and jointly consider structural compatibility and semantic relevance according to the question. In the second stage, conditioned on the question and the ranked subgraph candidates, an LLM is employed to generate executable logical forms.

4.1 Structure-aware Subgraph Retrieval

In this stage, for a given question, we predict a logical structure and retrieve subgraphs from the KB. Subsequently, we rank these subgraphs using a joint objective that explicitly evaluates structural compatibility by quantifying the alignment between each candidate and the predicted structure, together with multi-granular semantic relevance.

4.1.1 Question-Guided Core Reasoning Pattern Prediction

In knowledge base question answering, the logical form underlying a question typically follows a relational structure that connects the topic entity to the answer entity. We refer to this structure as the **core reasoning pattern (CRP)**, which captures the reasoning pattern of the question. For example, given the question *"What kind of money to take to Bahamas?"*, the logical form is `(JOIN (R location.country.currency_used) m.0160w)`. The corresponding CRP is *topic_entity* $\rightarrow$ *answer_entity*, reflecting that a single relation connects the topic entity (*Bahamas*) to the answer entity.

In this work, we restrict CRP patterns to at most two hops (i.e., single-hop and two-hop patterns, as well as answer-centered variants). This choice is mainly motivated by the hop distribution of the benchmarks we consider (see Appendix B), while keeping the framework extensible to longer-hop patterns when needed. We define patterns as follows:

- **Single-hop:** Direct connection via $e_t \rightarrow e_a$ or $e_t \leftarrow e_a$.
- **Two-hop:** Connection via e_m (4 directions): $e_t \rightarrow e_m \rightarrow e_a$, $e_t \rightarrow e_m \leftarrow e_a$, $e_t \leftarrow e_m \rightarrow e_a$, and $e_t \leftarrow e_m \leftarrow e_a$.
- **Answer-centered:** e_a acts as a bridge: $e_1 \rightarrow e_a \rightarrow e_2$, $e_1 \leftarrow e_a \leftarrow e_2$, or $e_1 \rightarrow e_a \leftarrow e_2$.

Given a question, our goal is to predict its corresponding CRP based solely on the question text. To this end, we extract the gold CRP from annotated logical forms and formulate CRP prediction as a sequence generation task. Specifically, we fine-tune a lightweight sequence-to-sequence model, Flan-T5-base, to generate the CRP pattern conditioned on the input question.

Extensibility of CRP Patterns. The current CRP patterns are restricted to at most two hops, which empirically covers the vast majority of benchmark questions (100% of WebQSP, 94.90% of GrailQA, and 92.43% of GraphQuestions; see Appendix B). Importantly, the CRP design is *declarative and modular*: each pattern is defined by a hop count

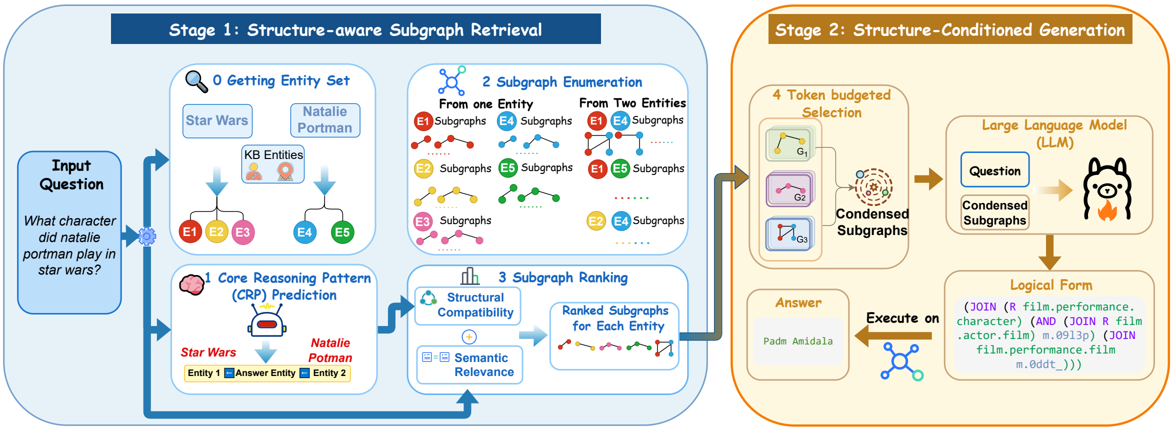

Figure 2: Overview of SELF-KBQA. The framework consists of two stages: **(Left)** *Structure-aware Subgraph Retrieval*, which predicts a Core Reasoning Pattern (CRP) from the question, enumerates candidate subgraphs from the KB, and ranks them by jointly considering structural compatibility and semantic relevance; **(Right)** *Subgraph-conditioned Generation*, which condenses the top-ranked subgraphs under a token budget and feeds them to a fine-tuned LLM to produce executable logical forms.

and a sequence of edge directions, independent of KB-specific relation names. Extending to longer reasoning chains (e.g., 3-hop patterns such as $e_t \rightarrow e_{m_1} \rightarrow e_{m_2} \rightarrow e_a$) or additional anchor entities only requires augmenting the pattern set; the subgraph enumeration, ranking, and generation modules remain unchanged. Furthermore, since CRP patterns are defined at the level of graph topology (nodes and directed edges), they are *transferable across different KBs*. For example, Wikidata's statement-qualifier structure is topologically analogous to Freebase's CVT nodes, both realizing a two-hop topology from topic to answer via an intermediate node. We provide empirical cross-KB evidence and further discussion in Appendix O.

4.1.2 Subgraph Enumeration

For each question q, we can obtain a set of candidate entities E from the entity mention in the question. Then, in this stage, we enumerate candidate subgraphs by expanding from each entity in E under a fixed set of reasoning patterns mentioned above. In our setting, a subgraph consists of a topic entity, relations, and adjacent entities (following the hop and direction constraints of the template). Starting from the topic entity, a single relation may connect to multiple adjacent entities. Since the concrete adjacent entities are often unnecessary for LLM-based logical-form generation, we replace them with an *abstract placeholder node* representing the adjacent type. By doing so, we increase the information density of the retrieved subgraphs, which helps logical-form generation. For example, we use an abstract adjacent node to represent the set of member countries: `United Kingdom – base.locations.continents.countries_within->base.locations.countries`.

Specifically, for each entity $e \in E$, we instantiate all CRP patterns and traverse the knowledge base accordingly, as shown in Figure 2. If the question contains a single entity mention, we enumerate all one-hop and two-hop subgraphs for each entity in the corresponding entity set. When the question contains two entity mentions, in addition to enumerating subgraphs centered at each entity individually, we also consider subgraphs that jointly include both entity mentions. For example, in Figure 2, for the question *"What character did Natalie Portman play in Star Wars?"*, two entity mentions, *Star Wars* and *Natalie Portman*, are identified. Accordingly, we enumerate not only the subgraphs centered at *Star Wars* and *Natalie Portman* separately, but also subgraphs that contain both entities.

4.1.3 Subgraph Ranking

Given the enumerated candidate subgraphs, this stage aims to identify the most relevant ones for logical form generation. We rank subgraphs by jointly considering their *structural compatibility* and *semantic relevance* according to the question.

Structural Compatibility. We define a *soft structural compatibility* score to measure how well a candidate subgraph g aligns with the predicted Core Reasoning Pattern (CRP). Let $S_{\text{struct}}(g)$ denote the structural compatibility score, which is decomposed into *hop consistency* and *directional* *consistency*.

Hop & Direction Template. We represent the predicted CRP as a directed path $\mathcal{P} = (v_0 \xrightarrow{d_1} v_1 \xrightarrow{d_2} \cdots \xrightarrow{d_H} v_H)$, where H is the hop count and $d_i \in \{\rightarrow, \leftarrow\}$. For each subgraph g, we extract its reasoning pattern $\mathcal{P}_g$ with hop count H_g and direction sequence $\mathbf{d}_g = (d_{g,1}, \ldots, d_{g,H_g})$.

Hop Consistency.

$$S_{\text{hop}}(g) \;=\; \frac{1}{1 + |H_g - H|}. \tag{1}$$

Directional Consistency. Let $m = \min(H_g, H)$ and $M = \max(H_g, H)$. We define

$$S_{\text{dir}}(g) \;=\; \begin{cases} 1, & M = 0, \\ \frac{1}{M} \sum_{i=1}^{m} \mathbb{I}\left[d_{g,i} = d_i\right], & M > 0. \end{cases} \tag{2}$$

Finally, the structural compatibility score is computed as

$$S_{\text{struct}}(g) \;=\; \lambda_{\text{hop}} S_{\text{hop}}(g) + \lambda_{\text{dir}} S_{\text{dir}}(g), \tag{3}$$

where λ_{hop} and λ_{dir} are hyperparameters.

Semantic Relevance. Beyond structural compatibility, we measure the semantic relevance between the question q and a candidate subgraph g at three granularities: *node level*, *relation-text level*, and *subgraph level* (Liu et al., 2025). Let $\phi(\cdot)$ be a text encoder (`BAAI/bge-m3` in our implementation) and $\cos(\cdot,\cdot)$ denote cosine similarity. We first obtain the question embedding $\mathbf{h}_q = \phi(q)$.

Node-level. For each node $v \in \mathcal{V}(g)$, we textualize it as $t(v)$ (e.g., entity surface form, types of entity) and compute $s(q, v) = \cos(\mathbf{h}_q, \phi(t(v)))$. We aggregate over nodes as

$$S_{\text{node}}(q, g) = \frac{1}{|\mathcal{V}(g)|} \sum_{v \in \mathcal{V}(g)} \cos(\phi(q), \phi(t(v))). \tag{4}$$

Relation-text level. For each relation $r \in \mathcal{R}(g)$, we convert its id into natural language tokens by splitting on delimiters (e.g., `location.country.currency_used` $\rightarrow$ "location country currency used") and denote the resulting text as $t(r)$. Then

$$S_{\text{rel}}(q, g) = \frac{1}{|\mathcal{R}(g)|} \sum_{r \in \mathcal{R}(g)} \cos(\phi(q), \phi(t(r))). \tag{5}$$

Subgraph-level. We linearize the whole subgraph g into a single text sequence $t(g)$ (nodes and triples) and compute

$$S_{\text{sub}}(q, g) = \cos(\mathbf{h}_q, \phi(t(g))). \tag{6}$$

Finally, we combine the three similarities with a weighted sum:

$$\begin{aligned} S_{\text{sem}}(q, g) = \alpha\, S_{\text{node}}(q, g) + \beta\, S_{\text{rel}}(q, g) \\ + \gamma\, S_{\text{sub}}(q, g). \end{aligned} \tag{7}$$

where α, β, and γ are weighting coefficients.

Overall Scoring. Finally, we combine structural compatibility and semantic relevance into an overall subgraph score:

$$S(g \mid q) = \lambda \cdot S_{\text{struct}}(g) + (1 - \lambda) \cdot S_{\text{sem}}(q, g), \tag{8}$$

where $\lambda \in [0, 1]$ controls the trade-off between structural and semantic information.

Candidate subgraphs are ranked by $S(g \mid q)$, and the top-ranked subgraphs are selected for subsequent logical form generation.

4.2 Subgraph-Conditioned Logical Form Generation

Given a question q, we take the top-k ranked subgraphs $\mathcal{G}_k = \{g_1, \ldots, g_k\}$ from the previous stage as evidence to generate an executable logical form.

Directly concatenating $\mathcal{G}_k$ yields redundant tokens due to overlapping nodes/edges, slowing training and inference. We therefore perform token-budgeted evidence condensation to produce a compact, structure-preserving evidence representation.

Token-budgeted Subgraph Selection. Top-k subgraphs often overlap, leading to redundant tokens when concatenated. We select a subset $\mathcal{H} \subseteq \mathcal{G}_k$ under a token budget B, where $\ell(g)$ denotes the token length of linearized subgraph g:

$$\max_{\mathcal{H} \subseteq \mathcal{G}_k} F(\mathcal{H}) \quad \text{s.t.} \quad \sum_{g \in \mathcal{H}} \ell(g) \leq B.$$

We define

$$F(\mathcal{H}) = \sum_{g \in \mathcal{H}} s(g) + |U(\mathcal{H})|,$$

where $s(g)$ is the subgraph ranking score and $U(\mathcal{H}) = \bigcup_{g \in \mathcal{H}} U(g)$ is the set of unique units covered by $\mathcal{H}$ (entities and relation types). This formulation discourages overlap by counting repeated units only once. We approximately optimize F using a greedy budgeted selection based on marginal gain per token.

Finally, we serialize the input context $\mathcal{C}(\mathcal{S})$, a full example prompt is provided in Appendix C.

LoRA Fine-Tuning for Executable Logical Forms. Given a question q and condensed evidence context $\mathcal{C}(\mathcal{S})$, we fine-tune an open-source LLM to generate the gold logical form y^* by minimizing the negative log-likelihood:

$$\mathcal{L}(\theta) = -\log P_\theta(y^* \mid q, \mathcal{C}(\mathcal{S})). \tag{9}$$

We adopt LoRA for parameter-efficient tuning, adding a low-rank update $\Delta W = BA$ to each weight matrix.

Execution and fallback. The generated logical forms are directly executed against the knowledge base. While the fine-tuned model produces executable logical forms in most cases (without additional entity/relation normalization), a small portion may still be non-executable due to entity, relation, or structural errors. In that case, we adopt a fallback mechanism: when execution fails, we return the logical form transformed from the top-1 ranked subgraph from previous stage.

5 Experiments

Our analysis is organized around the following research questions (**RQs**): **RQ1**: How does SELF-KBQA perform in comparison to other approaches? **RQ2:** How does the retrieval of semantically and structurally aligned subgraphs impact performance? **RQ3:** How much does the Core Reasoning Pattern (CRP) structural prior contribute?

5.1 Experimental Setup

Datasets. We evaluate our approach on three widely used KBQA benchmarks: GrailQA (Gu et al., 2021), WebQSP (Yih et al., 2016), and GraphQuestions (Su et al., 2016). Detailed dataset statistics are provided in Appendix B.

Baselines. We compare our method against representative baselines from three categories: IR-based methods, SP-based methods, and recent LLM-era approaches, including LLM prompting and LLM fine-tuning. Detailed descriptions of all baselines are provided in Appendix D.

Evaluation Metric. For GrailQA, following previous work (Tian et al., 2024; Luo et al., 2025a; Gao et al., 2025), we adopt Exact Match (EM) and F1 score as evaluation metrics, computed using the official evaluation script. For WebQSP and GraphQuestions, following prior work (Li et al., 2023; Luo et al., 2025a), we report F1 score.

Implementation Details. For CRP prediction, we fine-tune **Flan-T5-base**, and for logical form generation, we fine-tune **Llama-3.1-8B-Instruct**. We perform a sensitivity analysis on the subgraph ranking hyperparameters k and λ, and observe stable performance for $k \in [10, 50]$, with the best results around $k = 40$, which is used by default. An intermediate λ (approximately 0.4) performs best, highlighting the importance of balancing structural compatibility and semantic relevance; detailed analyses are provided in Appendix H. Training and hardware details are reported in Appendix F.

5.2 Main Results (RQ1)

Tables 1–3 report the main results on GrailQA, WebQSP, and GraphQuestions. We reproduce SG-KBQA with Llama-3.1-8B for a fair comparison, while other numbers are taken from the original papers [1]. In addition, for comparison with some other methods (e.g.,ToG (Sun et al., 2024)), since they randomly sample 1,000 instances from the validation set of the GrailQA dataset, we also conduct our evaluation on the same subset for a fair comparison. Detailed settings and results can be found in the Appendix E.

GrailQA. As shown in Table 1, our method achieves the best overall performance on GrailQA. The gains are consistent across all three splits, with the largest improvement observed on the i.i.d. split and steady gains on the compositional and zero-shot splits. These results indicate that SELF-KBQA not only improves in-distribution accuracy but also generalizes better to compositional and unseen-schema questions. We further study the performance of SELF-KBQA across different LLM backbones, with detailed results reported in Appendix G. We also analyze the execution success rate of the logical forms generated by SELF-KBQA, with detailed analyses provided in Appendix I, and examine the effect of removing the fallback mechanism, with results summarized in Table 4.

[1]SG-KBQA was originally reported with LLaMA-3-8B; different LLaMA versions may lead to different performance.

Method	LLM	I.I.D		Compositional		Zero-shot		Overall	
		EM	F1	EM	F1	EM	F1	EM	F1
Prompting Methods									
KB-BINDER	GPT-3.5-turbo	75.8	80.9	48.3	53.6	45.4	50.7	53.2	58.5
KB-Coder	GPT-3.5-turbo	76.9	81.0	52.7	57.8	48.9	54.1	56.3	61.3
ARG-KBQA	GPT-3.5-turbo	79.0	81.5	48.4	55.8	55.9	61.4	59.6	64.9
StructGPT	GPT-4-turbo	-	70.4	-	44.3	-	50.5	-	54.6
QueryAgent	GPT-4-turbo	-	-	-	-	-	-	-	66.8
TARGA	GPT-4-turbo	-	-	-	-	-	-	-	69.8
Fine-tune-based Methods									
RnG-KBQA	–	86.7	89.0	61.7	68.9	68.8	74.7	69.5	76.9
DecAF	–	88.7	92.4	71.5	79.8	65.9	77.3	72.5	81.4
TIARA	–	88.4	91.2	66.4	74.8	73.3	80.7	75.3	81.9
FC-KBQA	–	89.0	91.5	70.4	77.3	78.1	83.1	79.0	83.8
KGAgent	Llama-2-7B	-	92.0	-	80.0	-	**86.3**	-	<u>86.1</u>
KBQA-o1	Llama-3.1-8B	77.8	85.5	**76.3**	77.6	68.1	76.1	71.9	78.5
SG-KBQA	Llama-3.1-8B	<u>90.7</u>	<u>92.2</u>	74.4	<u>80.9</u>	<u>79.2</u>	84.2	<u>80.8</u>	85.4
SELF-KBQA (Ours)	Llama-3.1-8B	**94.0**	**95.2**	<u>75.4</u>	**81.6**	**80.1**	<u>84.8</u>	**82.3**	**86.5**

Table 1: Performance on the dev set of GrailQA. The **bold** and <u>underlined</u> numbers indicate the best and second-best performance, respectively.

WebQSP and GraphQuestions. On WebQSP and the more compositional GraphQuestions dataset, SELF-KBQA further improves the best Llama-3.1-8B baseline, suggesting its effectiveness on complex graph-structured reasoning. Overall, the consistent improvements across datasets demonstrate the robustness of our subgraph-guided executable logical form generation framework.

5.3 Analysis of Structure-aware Subgraph Retrieval (RQ2)

To answer **RQ2**, we investigate whether retrieving subgraphs that align with both the question's semantics and structure improves performance, and determine if both constraints are necessary.

Necessity of Aligned Subgraph Evidence. We first investigate whether subgraph evidence is beneficial through an ablation study. We compare the full model with a baseline without retrieved context (**w/o Subgraphs**) and a baseline using independently retrieved evidence (**w/ Independent Retrieval**) which employs the same encoder to retrieve entities and relations from the KB independently. As shown in Table 4, removing subgraph evidence causes a catastrophic drop. Furthermore, **w/ Independent Retrieval** significantly underperforms the subgraph-based approach. These results confirm that treating the KB as a "bag of items" disrupts the reasoning chain. By explicitly enumerating connected subgraphs, SELF-KBQA bridges the "structural gap", providing the necessary topological integrity for generating executable logical forms.

Retrieval Performance Comparison across Retrieval Sizes (k). Figure 3 compares the retrieval match rate of SELF-KBQA and its variants as the number of retrieved subgraphs k increases. Here, we define the **match rate** as the percentage of questions where the retrieved evidence fully covers the entities and relations in the gold logical form. As shown in Figure 3, removing either the semantic or structural scores, or adopting independent retrieval, leads to a distinct decline in retrieval performance. Crucially, this trend remains consistent as k increases from 10 to 50. These results demonstrate that explicitly enforcing both structural compatibility and semantic relevance is essential for identifying subgraphs that faithfully mirror the question's intent.

Impact of Retrieval Strategy on Generation. As shown in Table 4, the variants **w/o Structure Score**, **w/o Semantic Score**, and **w/ Independent Retrieval** all suffer from consistent degradation in EM and F1 scores across different difficulty levels. Crucially, this downstream performance drop strictly mirrors the retrieval recall gaps observed in Figure 3.

Thus, providing subgraphs that align with the

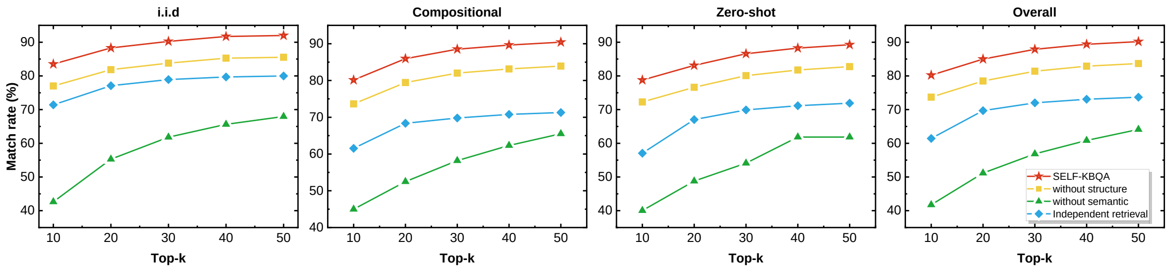

Figure 3: Retrieval performance on the validation set of GrailQA of SELF-KBQA and its variants across varying k. The full model (red line) consistently outperforms variants that lack structural or semantic constraints, highlighting the robustness of the joint ranking strategy.

Method	LLM	F1
IR-based KBQA Methods		
NSM	–	68.7
KGT5	–	56.1
SP-based KBQA Methods		
RnG-KBQA	–	75.6
DecAF	–	77.1
TIARA	–	76.7
FC-KBQA	–	76.9
LLMs-based Methods (Prompting)		
KB-Binder	GPT-3.5-turbo	52.6
ARG-KBQA	GPT-3.5-turbo	58.8
KELDaR	GPT-3.5-turbo	76.7
ToG	GPT-4-turbo	76.0
FIDELIS	GPT-4-turbo	78.3
Interactive-KBQA	GPT-4-turbo	71.2
LLMs-based Methods (Fine-tuning)		
ChatKBQA	Llama-2-7B	<u>79.8</u>
KBQA-o1	Llama-3.1-8B	57.8
MCTS-KBQA	Llama-3.1-8B	76.0
SG-KBQA	Llama-3.1-8B	<u>79.8</u>
GCR	Llama-3.1-8B	74.1
SELF-KBQA (ours)	Llama-3.1-8B	**80.6**

Table 2: Results on the test set of WebQSP. The **bold** and <u>underlined</u> numbers indicate the best and second-best performance.

Method	LLM	F1
Prompting-based Methods		
KB-Binder	GPT-3.5-turbo	32.5
KB-Coder	GPT-3.5-turbo	36.6
Fine-tuning-based Methods		
SPARQA	–	21.5
BERT+Ranking	–	25.0
ArcaneQA	–	31.8
KBQA-o1	Llama-3.1-8B	<u>48.7</u>
SELF-KBQA (ours)	Llama-3.1-8B	**49.9**

Table 3: Results on the test set of GraphQ. The **bold** and <u>underlined</u> numbers indicate the best and second-best performance.

question in *both* structure and semantics is a prerequisite for generating executable logical forms.

5.4 Impact of the CRP Structural Prior (RQ3)

The CRP captures reasoning patterns (specifically hop structure and directionality) that connect the topic entity to the answer entity, abstracting away auxiliary constraints.

CRP prediction quality. We evaluate CRP prediction accuracy by deterministically extracting gold patterns from annotated logical forms and training the model to generate the corresponding discrete patterns. As shown in Table 5, CRP prediction is reasonably accurate on GrailQA, WebQSP, and GraphQuestions, but errors still occur, which may propagate to retrieval and generation.

Impact on downstream QA. To quantify the impact of CRP accuracy and isolate the benefit of structural guidance, we conduct an ablation study on the GrailQA dev set (Table 4). Injecting ground-truth CRPs ("w/ Golden CRP") yields consistent gains across splits, indicating substantial headroom from better structural priors. In contrast, removing CRP-based guidance ("w/o Structure Score") leads to clear degradation, confirming that CRP helps filter irrelevant candidates and steer retrieval toward the correct logical structure.

Performance on ≥3-hop questions. Although CRP patterns cover at most two hops, we evaluate on GrailQA questions requiring ≥3-hop reasoning. SELF-KBQA achieves EM 56.4 / F1 62.6, outperforming SG-KBQA's EM 44.2 / F1 54.3, showing that the retrieval and generation modules compensate effectively beyond CRP coverage.

Model Variant	I.I.D		Compositional		Zero-shot		Overall	
	EM	F1	EM	F1	EM	F1	EM	F1
SELF-KBQA (Full Model)	94.04	95.18	75.36	81.63	80.11	84.76	82.33	86.51
w/ Golden CRP	95.50	96.50	76.80	83.00	81.60	86.20	83.77	87.89
w/o Structure Score	92.50	93.90	73.00	80.00	78.50	83.50	80.54	85.15
w/o Semantic Score	89.89	90.78	69.88	74.51	75.16	74.97	75.43	78.59
w/ Independent Retrieval	90.30	91.50	71.10	77.80	76.10	80.90	78.30	82.68
w/o Fallback Mechanism	91.50	92.60	72.80	78.90	77.50	82.20	79.72	83.89
w/o Subgraphs	67.55	71.95	54.16	59.62	68.57	74.87	65.10	70.77

Table 4: Results of our Ablation Study on the validation set of GrailQA. Cell shading indicates performance changes relative to the full model (SELF-KBQA): blue tones stand for improvements and amber tones for degradations, with darker shades indicating larger changes.

Dataset	CRP Accuracy (%)
GrailQA	85.64
WebQSP	87.56
GraphQ	76.81

Table 5: CRP prediction accuracy on different Datasets.

5.5 Efficiency Analysis

We report inference latency on GrailQA dev (6,763 questions, single A6000 GPU). The end-to-end latency of SELF-KBQA is ∼16.9 s per question, comprising subgraph enumeration (206 ms), BGE-M3 semantic scoring (16.4 s), LLM generation (275 ms, vLLM), and SPARQL execution (27 ms). This represents a ∼25× speedup over ChatKBQA and KB-BINDER, which require ∼7 min per question for post-hoc combinatorial entity/relation grounding. We further validate retrieval quality with a frozen GPT-4o: providing retrieved subgraphs via in-context learning raises F1 from 16.1 (no subgraphs) to 46.9, confirming that the retrieved context provides effective grounding independent of fine-tuning. Detailed results are in Appendix K and L.

6 Conclusion

We presented SELF-KBQA, a subgraph-guided framework for executable logical form generation in KBQA that aligns retrieval with both *structural compatibility* (via Core Reasoning Patterns) and *semantic relevance*. By ranking candidate subgraphs with a joint scoring function and performing token-budgeted evidence condensation, our method supplies compact, high-utility KB context to LLM decoding, enabling reliable execution. Extensive experiments on GrailQA, WebQSP, and GraphQuestions demonstrate consistent gains and state-of-the-art performance.

Limitations

While SELF-KBQA achieves state-of-the-art performance, several limitations remain. First, our CRP patterns are currently restricted to at most two hops. Although this covers the vast majority of benchmark questions (92–100%), questions requiring longer reasoning chains are not explicitly modeled. As discussed in Section 4, extending CRP to longer-hop patterns is modular and straightforward, and we leave this for future work. Second, our subgraph enumeration may face scalability challenges with hub entities connected to many relation types. In practice, schema-level enumeration with pruning keeps candidate sizes manageable (median 51 per entity on GrailQA; only 1.7% of questions exceed 10K candidates), but extreme cases remain a potential bottleneck. Third, we primarily evaluate on Freebase-based benchmarks. While CRP patterns are defined at the topology level and are largely transferable (e.g., 77.7% coverage on Wikidata-based KQA Pro; see Appendix O), full evaluation on other KBs is left for future work. Finally, SELF-KBQA relies on supervised training with gold logical forms, which is standard for SP-KGQA. However, our retrieval module is largely annotation-free (subgraph enumeration and semantic scoring require no gold forms), and the CRP prediction component could be replaced by rule-based heuristics or LLM few-shot prompting, making the framework a promising candidate for extension to IR-KGQA settings where executable annotations are unavailable.

Ethics Statement

In this work, we use publicly available datasets and do not collect any personally identifiable information. All datasets and models are utilized in full compliance with their intended purposes and respective licenses. The primary goal of this work is to strengthen LLM's ability to generate correct logical form; we condemn any potential misuse.

Acknowledgements

This work was supported by the National Key Research and Development Program of China (Grant No. 2024YFE0210800) and the National Natural Science Foundation of China (Grant No. 62476025). This work was also supported by the Fundamental Research Funds for the Central Universities.

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 *international semantic web conference*, pages 722–735. Springer.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. 2024. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. *Advances in Neural Information Processing Systems*, 37:37665–37691.

Tengfei Feng and Liang He. 2025. Rgr-kbqa: Generating logical forms for question answering using knowledge-graph-enhanced large language model. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3057–3070.

Shengxiang Gao, Jey Han Lau, and Jianzhong Qi. 2025. Beyond seen data: Improving KBQA generalization through schema-guided logical form generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8764–8783, Suzhou, China. Association for Computational Linguistics.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488.

Yu Gu and Yu Su. 2022. Arcaneqa: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, pages 1718–1731. International Committee on Computational Linguistics.

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 *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 553–561, New York, NY, USA. Association for Computing Machinery.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907.

Xixin Hu, Xuan Wu, Yiheng Shu, and Yuzhong Qu. 2022. Logical form generation via multi-task learning for complex question answering over knowledge bases. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1687–1696, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Xiang Huang, Sitao Cheng, Shanshan Huang, Jiayu Shen, Yong Xu, Chaoyun Zhang, and Yuzhong Qu. 2024. Queryagent: A reliable and efficient reasoning framework with environmental feedback based self-correction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 5014–5035. Association for Computational Linguistics.

Xiang Huang, Jiayu Shen, Shanshan Huang, Sitao Cheng, Xiaxia Wang, and Yuzhong Qu. 2025. TARGA: targeted synthetic data generation for practical reasoning over structured data. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 2704–2726. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. 2025. KG-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9505–9523, Vienna, Austria. Association for Computational Linguistics.

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 *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4483–4491. ijcai.org.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Complex knowledge base question answering: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(11):11196–11215.

Bo Li, Tian Tian, Zhenghua Xu, Hao Cheng, Shikun Zhang, and Wei Ye. 2026a. Modeling uncertainty trends for timely retrieval in dynamic RAG. In *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pages 31527–31535. AAAI Press.

Bo Li, Mingda Wang, Gexiang Fang, Shikun Zhang, and Wei Ye. 2026b. Retrieval as generation: A unified framework with self-triggered information planning. *Preprint*, arXiv:2604.11407.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 6966–6980. Association for Computational Linguistics.

Yading Li, Dandan Song, Changzhi Zhou, Yuhang Tian, Hao Wang, Ziyi Yang, and Shuhao Zhang. 2024a. A framework of knowledge graph-enhanced large language model based on question decomposition and atomic retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11472–11485, Miami, Florida, USA. Association for Computational Linguistics.

Yading Li, Dandan Song, Changzhi Zhou, Yuhang Tian, Hao Wang, Ziyi Yang, and Shuhao Zhang. 2024b. A framework of knowledge graph-enhanced large language model based on question decomposition and atomic retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11472–11485, Miami, Florida, USA. Association for Computational Linguistics.

Peiyang Liu. 2024. Unsupervised corrupt data detection for text training. *Expert Systems with Applications*, 248:123335.

Peiyang Liu, Xi Wang, Ziqiang Cui, and Wei Ye. 2025. Queries are not alone: Clustering text embeddings for video search. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 874–883.

Peiyang Liu, Jinyu Yang, Lin Wang, Sen Wang, Yunlai Hao, and Huihui Bai. 2023. Retrieval-based unsupervised noisy label detection on text data. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 4099–4104.

Haoran Luo, Haihong E, Yikai Guo, Qika Lin, Xiaobao Wu, Xinyu Mu, Wenhao Liu, Meina Song, Yifan Zhu, and Anh Tuan Luu. 2025a. Kbqa-o1: Agentic knowledge base question answering with monte carlo tree search. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net.

Haoran Luo, Haihong E, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, Yifan Zhu, and Anh Tuan Luu. 2024. Chatkbqa: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 2039–2056. Association for Computational Linguistics.

Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Yuan-Fang Li, Chen Gong, and Shirui Pan. 2025b. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net.

Costas Mavromatis and George Karypis. 2025. GNN-RAG: graph neural retrieval for efficient large language model reasoning on knowledge graphs. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 16682–16699. Association for Computational Linguistics.

Zhijie Nie, Richong Zhang, Zhongyuan Wang, and Xudong Liu. 2024a. Code-style in-context learning for knowledge-based question answering. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 18833–18841. AAAI Press.

Zhijie Nie, Richong Zhang, Zhongyuan Wang, and Xudong Liu. 2024b. Code-style in-context learning for knowledge-based question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18833–18841.

Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. 2009. Semantics and complexity of sparql. *ACM Transactions on Database Systems (TODS)*, 34(3):1–45.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. *arXiv preprint arXiv:2203.10321*.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje F. Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. TIARA: Multi-grained retrieval for robust question answering over large knowledge base. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8108–8121, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gür, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for qa evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572.

Yuan Sui, Yufei He, Nian Liu, Xiaoxin He, Kun Wang, and Bryan Hooi. 2025. Fidelis: Faithful reasoning in large language models for knowledge graph question answering. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 8315–8330. Association for Computational Linguistics.

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 *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yuhang Tian, Dandan Song, Zhijing Wu, Pan Yang, Changzhi Zhou, Jun Yang, Hao Wang, Huipeng Ma, Chenhao Li, and Luan Zhang. 2025. CompKBQA: Component-wise task decomposition for knowledge base question answering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 293–309, Suzhou, China. Association for Computational Linguistics.

Yuhang Tian, Dandan Song, Zhijing Wu, Changzhi Zhou, Hao Wang, Jun Yang, Jing Xu, Ruanmin Cao, and Haoyu Wang. 2024. Augmenting reasoning capabilities of llms with graph structures in knowledge base question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 11967–11977. Association for Computational Linguistics.

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

Guanming Xiong, Junwei Bao, and Wen Zhao. 2024. Interactive-kbqa: Multi-turn interactions for knowledge base question answering with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 10561–10582. Association for Computational Linguistics.

Guanming Xiong, Haochen Li, and Wen Zhao. 2025. Mcts-kbqa: Monte carlo tree search for knowledge base question answering. *arXiv preprint arXiv:2502.13428*.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 6032–6043. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2023. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

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 *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1002–1017, Toronto, Canada. Association for Computational Linguistics.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Appendix

A Operators in S-expression

In this section, we introduce the operators used in S-expressions, as detailed in Table 6.

B Datasets

GrailQA (Gu et al., 2021) is designed to evaluate zero-shot generalization in KBQA. It contains 64,331 questions and is carefully split to assess three levels of generalization: (1) *i.i.d.*, where test questions share the same distribution as training data; (2) *compositional*, which evaluates generalization to unseen compositions of seen schema items; and (3) *zero-shot*, which tests generalization to entirely unseen KB schema items. The test set comprises 25%, 25%, and 50% of these three settings, respectively. Beyond generalization challenges, GrailQA also presents difficulties due to its large-scale schema (over 3,000 relations), complex logical forms (up to 4 hops), and noisy entity mentions in questions.

WebQSP (Yih et al., 2016) is a widely-used KBQA benchmark based on Freebase, comprising 4,737 natural language questions. It evaluates model performance under the *i.i.d.* setting, where training and test sets share the same distribution of entities and relations. The dataset requires reasoning chains of up to 2 hops.

GraphQuestions (Su et al., 2016) is a benchmark dataset designed to evaluate KBQA models on complex graph-structured reasoning. It contains a total of 5,166 questions, with 2,508 questions for training and 2,658 for testing. Compared to simpler KBQA benchmarks, GraphQuestions places a stronger emphasis on multi-hop reasoning over the knowledge graph, requiring models to navigate compositional and structurally complex relational paths to derive correct answers.

Table 7 shows the hop distribution of questions across different datasets. Most questions require 1-2 hops to answer, with WebQSP being the simplest (no 3-hop questions) and GraphQ being the most complex (7.57% questions require 3 or more hops).

C Prompt Used in SELF-KBQA

Figure 4 illustrates the prompt used by SELF-KBQA for logical form generation, including the question and the top-k retrieved subgraphs.

D Baselines

We compare SELF-KBQA with the following baselines divided into five classes: 1) IR-based KBQA Methods, 2) SP-based KBQA Methods, 3) LLM-based KBQA Methods (Prompting), 4) LLM-based KBQA Methods (Fine-tuning).

D.1 IR-based KBQA Methods

NSM (He et al., 2021) proposes a teacher-student approach for multi-hop KBQA, where the teacher network uses bidirectional reasoning to generate reliable intermediate supervision signals, improving the student network's reasoning capacity.

KGT5 (Saxena et al., 2022) demonstrates that an off-the-shelf encoder-decoder Transformer model, when applied as a sequence-to-sequence task for knowledge graph link prediction and question answering.

D.2 SP-based KBQA Methods

Rng-KBQA (Ye et al., 2022) first enumerates all possible queries and then finetune T5 model to get the final output.

DecAF (Yu et al., 2023) proposes a framework that jointly generates logical forms and direct answers for KBQA, combining their strengths to improve accuracy.

TIARA (Shu et al., 2022) enhances PLM-based KBQA by incorporating multi-grained retrieval to provide relevant KB context and constrained decoding to improve logical form generation.

FC-KBQA (Zhang et al., 2023) proposes a fine-to-coarse compositional framework that improves KBQA generalization and executability by reformulating fine-grained KB components into middle-grained knowledge pairs for logical expression generation.

D.3 LLM-based KBQA Methods (Prompting)

KB-Binder (Li et al., 2023) enables few-shot KBQA using large language models and BM25 matching, achieving strong performance across datasets without training.

KB-Coder (Nie et al., 2024b) proposes a code-style in-context learning method for KBQA, converting logical form generation into code generation for LLMs, reducing format errors.

ARG-KBQA (Tian et al., 2024) proposes a framework that enhances LLMs for KBQA by retrieving question-related graph structures using a two-stage ranker.

Syntax	Return Type	Semantics
(AND E_1 E_2)	Set of Entities	Intersection of two entity sets E_1 and E_2.
(COUNT E)	Integer	Returns the cardinality $\|E\|$ of the entity set.
(R P)	Set of Pairs	Inverse of the binary relation P, i.e., $\{(y, x) \mid (x, y) \in P\}$.
(JOIN P E)	Set of Entities	Entity retrieval: returns entities related to any element in E via relation P.
(JOIN P_1 P_2)	Set of Pairs	Relational composition of P_1 and P_2.
(ARGMAX E P) (ARGMIN E P)	Set of Entities	Selects entity $x \in E$ that maximizes (or minimizes) the value associated via relation P.
(CMP P n)	Set of Entities	Filters entities where the related value compares to n. CMP $\in$ {LT, LE, GT, GE}.

Table 6: Definitions of logical operators. E denotes a set of entities, P denotes a binary relation (set of tuples), and n is a literal value.

Dataset	1 hop	2 hop	$\geq$ 3 hop
WebQSP	61.04%	37.22%	0.00%
GrailQA	68.78%	26.11%	5.10%
GraphQ	59.83%	32.59%	7.57%

Table 7: Question hop distribution across different datasets

KELDaR (Li et al., 2024b) enhances LLMs for KGQA by introducing question decomposition and atomic retrieval, using a decomposition tree to guide atomic-level KG subgraph retrieval for answering complex questions.

FIDELIS (Sui et al., 2025) introduces a retrieval-augmented reasoning method that enhances KGQA by anchoring responses to verifiable reasoning paths, using keyword-enhanced retrieval and step-wise beam search to ensure accuracy.

QueryAgent (Huang et al., 2024) proposes a step-by-step semantic parsing framework with an environmental feedback-based self-correction method (ERASER) that selectively fixes hallucinations to improve reliability and efficiency.

TARGA (Huang et al., 2025) proposes a targeted synthetic data generation framework that builds high-relevance query–question pairs via entity/relation-guided expansion to provide synthetic in-context demonstrations without manual annotation.

KG-Agent (Jiang et al., 2025) proposes an autonomous LLM-based agent with tool use and memory, fine-tuned on synthesized code-based instructions to perform multi-hop KG reasoning effectively.

ToG (Sun et al., 2024) proposes an LLM⊗KG paradigm where an LLM agent iteratively performs beam search over a KG to find promising reasoning paths for traceable, correctable, training-free graph reasoning with reduced hallucinations.

PoG (Chen et al., 2024) proposes a self-correcting adaptive planning paradigm for KG-augmented LLMs that decomposes questions into sub-objectives and iteratively explores, memorizes, and reflects to refine reasoning paths on graphs.

D.4 LLM-based KBQA Methods (Fine-tuning)

ArcaneQA (Gu and Su, 2022) dynamically generates the query based on results from previous steps.

ChatKBQA (Luo et al., 2024) presents a generate-then-retrieve KBQA framework that uses fine-tuned LLMs to generate logical forms and unsupervised retrieval to replace entities and relations.

CompKBQA (Tian et al., 2025) decomposes KBQA logical-form generation into sequential subtasks (skeleton → topic entity → relations → full logical form) and augments generation with an R3 retriever that injects KB evidence, reducing errors and achieving SOTA on WebQSP and CWQ.

Interactive-KBQA (Xiong et al., 2024) frames an LLM as an agent that iteratively interacts with a knowledge base via three unified SPARQL-based tools (SearchNodes, SearchGraphPatterns, ExecuteSPARQL) to incrementally construct executable logical forms in few-shot settings, improving interpretability and achieving competitive results across multiple KBQA benchmarks.

KBQA-o1 (Luo et al., 2025a) treats KBQA as an agentic search problem: a ReAct-style agent explores the KB step-by-step to build logical forms, and Monte Carlo Tree Search (guided by policy/reward models) balances exploration vs. efficiency while also auto-generating high-quality annotations for incremental fine-tuning.

System: You are an expert in answering questions based on knowledge graphs. Your task is to provide a logical form which can be executed on the Knowledge Base to get the answers based on the provided subgraph and question. The subgraph contains nodes and paths representing entities and their relationships.

Question: oersted is the magnetic field strength unit in what measurement system?

Subgraphs:
Entities:
- `m.0fksj` megagauss oersted *type:* `freebase.unit_profile` `measurement_unit.magnetic_field_strength_unit`
- `measurement_unit.measurement_system`
- `measurement_unit.magnetic_field_strength_unit`
- `measurement_unit.dimension`
- `type.float`
- `m.038z7b2` oersted *type:* `music.release` `media_common.creative_work`
- `m.01jkj07` oersted (album) *type:* `music.album`
- `m.0265qkf` hans christian Ørsted *type:* `chemistry.element_discoverer` `symbols.name_source` `book.author`

Subgraph 1:
- Edges: `measurement_unit.measurement_system -[measurement_unit.measurement_system.magnetic_field_strength_units]-> m.0fksj`

Subgraph 2:
- Edges: `m.0fksj -[measurement_unit.magnetic_field_strength_unit.measurement_system]-> measurement_unit.measurement_system`

Subgraph 3:
- Edges: `measurement_unit.measurement_system -[measurement_unit.measurement_system.acceleration_units]-> measurement_unit.magnetic_field_strength_unit` `m.0fksj -[measurement_unit.measurement_system.magnetic_field_strength_units]-> measurement_unit.magnetic_field_strength_unit`

[...] *(more subgraphs omitted for brevity)*

Logical Form:

Figure 4: Example prompt for logical form generation with subgraph-enhanced context.

MCTS-KBQA (Xiong et al., 2025) improves LLM-based semantic-parsing KBQA by replacing linear tool-use with Monte Carlo Tree Search guided by step-wise, prompt-based rewards, enabling better exploration and stronger accuracy—especially in low-resource settings—while also distantly annotating intermediate reasoning traces for training.

SG-KBQA (Gao et al., 2025) improves KBQA generalization beyond seen data by injecting KB schema (class/domain/range contexts) to guide both entity retrieval and logical-form generation, enabling valid unseen compositions and stronger zero-shot/compositional performance.

GCR (Luo et al., 2025b) proposes a graph-constrained reasoning framework that integrates KG structure into LLM decoding via a trie-based KG-Trie to generate faithful KG-grounded reasoning paths and reduce hallucinations.

E Performance on the subset extracted by ToG

Following the settings of ToG (Sun et al., 2024), we evaluate SELF-KBQA using their official evaluation script on their extracted GrailQA Subset, and report the Hits@1 results in Table 8. As shown in the table, SELF-KBQA outperforms all baseline methods, demonstrating the effectiveness of structured subgraph guidance.

F Implementation Details

In this section, we introduce the implementation details of each module. For the entity mention detection, following previous methods, the entity mentions are obtained by ELQ. Then for each entity mention, we merge the linking results based on FACC1 and elq linking.

For the Core relational path (CRP) prediction stage, We extract gold CRP patterns from annotated logical forms via a deterministic parser and fine-tune Flan-T5-base to generate one of the nine predefined CRP patterns. We train for 10 epochs with batch size 16, learning rate 3e-5 for GrailQA, 15 epochs with batch size 8, learning rate 3e-5 for WebQSP and GraphQuestions. At inference, we use greedy decoding and take the top-1 predicted pattern.

At the subgraph ranking stage, semantic rel-

Method	LLM	I.I.D	Compositional	Zero-shot	Overall
ToG	GPT-4	79.4	67.3	86.5	81.4
PoG	GPT-4	87.9	69.7	88.6	84.7
SELF-KBQA	Llama-3.1-8B	**94.6**	**81.8**	**89.0**	**88.9**

Table 8: Performance comparison on the GrailQA development set under the ToG evaluation setting.

evance S_{sem} combines node-, relation-, and subgraph-level similarities with $(\alpha, \beta, \gamma) = (0.4, 0.3, 0.3)$. We encode the question and textualized entities/relations using BAAI/bge-m3 and compute cosine similarity. Relation text is obtained by splitting relation IDs (e.g., `location.country.currency_used`) into natural language tokens. For structural compatibility, we set $(\lambda_{\text{hop}}, \lambda_{\text{dir}}) = (0.5, 0.5)$. The final ranking score uses $\lambda = 0.4$, tuned on the dev set via grid search over $[0, 1]$.

At the token-budgeted evidence condensation stage, we take the top-k ranked subgraphs with $k = 40$ and select a subset under the token budget $B = 2048$.

At the logical form generation stage, we fine-tune Llama-3.1-8B-Instruct to generate executable logical forms conditioned on the question and condensed evidence. For the GrailQA dataset, we train for 3 epochs with a learning rate of 5×10^{-5}. For the WebQSP and GraphQuestions datasets, we train for 20 epochs with the same learning rate of 5×10^{-5}. During decoding, we use greedy decoding with a temperature of 0.1.

G Performance on the dev set of GrailQA across different LLMs.

In this section, we report the results of SELF-KBQA on the GrailQA dev set using different LLM backbones, as shown in Table 9. The results indicate that SELF-KBQA performs well across different LLMs.

H Sensitivity to Retrieval Parameters.

We further study the sensitivity of our retrieval module to key hyperparameters, including the number of retrieved subgraphs k and the weighting coefficient λ in Eq. 8, while keeping all other settings fixed.

Effect of k. Figure 5 reports performance under different k values. Across settings (i.i.d., compositional, and zero-shot), the results are generally stable when k ranges from 10 to 50, with a mild

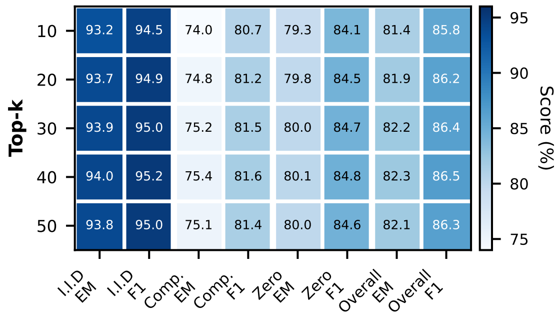

Figure 5: Sensitivity analysis of top_k.

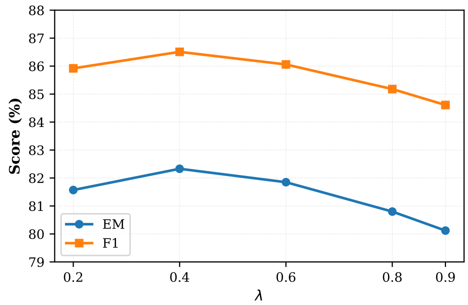

Figure 6: Sensitivity analysis of λ.

improvement as k increases from 10 to 30/40 and a plateau thereafter. This trend is consistent with the intended coverage–precision trade-off: a larger k provides broader evidence coverage, but beyond a certain point additional subgraphs are increasingly redundant or noisy, and the ranking module limits their marginal utility. Overall, we observe the best performance around $k = 40$, and use this value as the default in all experiments.

Effect of λ. Figure 6 shows the end-to-end EM/F1 as we vary λ, which controls the trade-off between structural compatibility and semantic relevance in subgraph ranking. Overall, performance is maximized at an intermediate value (around $\lambda = 0.4$), indicating that *both* structural and semantic signals are necessary for effective retrieval. When λ is too small, retrieval becomes overly semantic-driven and may include topically related but structurally mismatched subgraphs; when λ is too large, the model over-emphasizes structural

Method	LLM	I.I.D		Compositional		Zero-shot		Overall	
		EM	F1	EM	F1	EM	F1	EM	F1
	Llama-3.1-8B	**94.04**	**95.18**	75.36	81.63	**80.11**	84.76	**82.33**	86.51
SELF-KBQA	Qwen-2.5-7B	93.35	94.91	75.30	81.35	79.81	84.46	81.99	86.22
	Qwen-3-8B	93.47	94.97	**75.69**	**81.91**	**80.11**	**84.85**	82.27	**86.57**

Table 9: Performance of SELF-KBQA on the GrailQA development set with different LLM backbones. Best results are in **bold** and second-best results are underlined.

constraints and may discard semantically correct candidates.

I Logical Form Execution Analysis

In KBQA systems, a generated logical form should be executable on the underlying knowledge base to retrieve answers. However, for various reasons—such as errors in SPARQL conversion or schema mismatches—some generated logical forms may fail to execute. Table 10 reports the execution success rates on the GrailQA validation set. Overall, SELF-KBQA achieves a high execution success rate, indicating that it can generate valid logical forms in most cases.

Dataset	Success Rate
GrailQA	90.60%
i.i.d.	96.55%
compositional	90.09%
zero-shot	88.21%

Table 10: Logical form execution success rates on the validation set of GrailQA.

We observe that execution success rates are higher in the i.i.d. setting compared to compositional and zero-shot settings, which aligns with the increased complexity and schema diversity in these challenging scenarios.

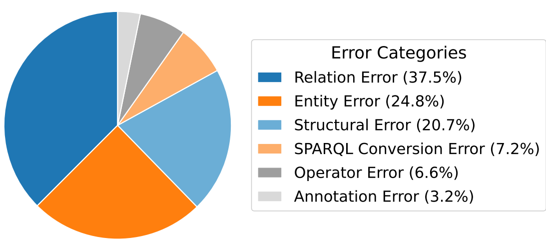

Figure 7: Error analysis on the validation set of GrailQA.

J Error Analysis

To better understand the remaining limitations of SELF-KBQA, we conduct an error analysis of the errors on the GrailQA dev set. We categorize these errors as follows, as shown in Figure 7:

- **Relation Error (37.5%):** The predicted logical form uses incorrect relations, leading to traversing wrong edges in the knowledge base.
- **Entity Error (24.8%):** The predicted logical form is anchored to incorrect entities (entity IDs completely mismatch the gold LF), causing the reasoning chain to start from the wrong node.
- **Structural Error (20.7%):** The predicted logical form differs from the gold logical form structurally (e.g., mixed operators/constraints).
- **SPARQL Conversion Error (7.2%):** The model outputs a logical form consistent with the gold logical form, but the converted SPARQL fails to execute, possibly due to information loss during the conversion from logical form to SPARQL.
- **Operator Error (6.6%):** The logical form uses incorrect operators or reasoning patterns (e.g., missing/extra ARGMAX/ARGMIN, incorrect COUNT, top-level operator mismatch, or large hop-count mismatch), which changes the query semantics.
- **Annotation Error (3.2%):** This mainly occurs in comparative questions, i.e., "larger than" is annotated as $\geq$ in some cases.

K Inference Efficiency

Table 11 reports a per-question latency breakdown of SELF-KBQA on the GrailQA dev set (6,763 questions, single NVIDIA A6000 GPU) and a comparison with representative baselines.

Stage / Method	Latency
SELF-KBQA Pipeline Breakdown	
Subgraph Enumeration	206 ms
Subgraph Ranking (BGE-M3)	16.4 s
LLM Generation (vLLM)	275 ms
SPARQL Execution	27 ms
End-to-End Total	**$\sim$16.9 s**
Baseline Methods	
ChatKBQA / KB-BINDER	$\sim$7 min
SG-KBQA (excl. retrieval)	$>$13.7 s

Table 11: Per-question inference latency breakdown and comparison with baselines on GrailQA dev set.

SELF-KBQA achieves a $\sim$25$\times$ speedup over ChatKBQA and KB-BINDER, which require post-hoc combinatorial entity/relation grounding after LLM generation. Compared to SG-KBQA ($>$13.7 s excluding its retrieval stages), our end-to-end latency is comparable while achieving higher accuracy.

L Frozen LLM Validation

To validate the quality of our retrieved subgraphs independently of fine-tuning, we evaluate a frozen GPT-4o with in-context learning on 1,000 GrailQA test samples (ToG subset). We disable the fallback mechanism to focus on LLM generation quality.

Setting	F1	EM	Invalid %
Fine-tuned LLaMA-3.1-8B (Ours)	**83.9**	**79.7**	**9.4**
GPT-4o 40-shot + subgraphs	46.9	45.6	48.9
GPT-4o 5-shot + subgraphs	43.0	41.5	51.4
GPT-4o 40-shot, no subgraphs	16.1	15.7	82.8

Table 12: Frozen LLM (GPT-4o) performance with and without retrieved subgraphs. Invalid % measures the fraction of outputs that fail to parse or execute.

Retrieved subgraphs substantially improve frozen-LLM performance, raising GPT-4o from 16.1 to 46.9 F1 and reducing invalid outputs from 82.8% to 48.9%. This confirms that the retrieved context is informative and provides effective grounding even without fine-tuning. Fine-tuning remains important for reliably enforcing the syntax and constraints of the target logical form language.

M Semantic Score Weight Sensitivity

We analyze the sensitivity of the semantic relevance weights (α, β, γ) in $S_{\text{sem}} = \alpha \cdot S_{\text{node}} + \beta \cdot S_{\text{rel}} + \gamma \cdot S_{\text{sub}}$ on the GrailQA dev set.

(α, β, γ)	Granularity	Match %	EM	F1
(1, 0, 0)	Node only	85.12	78.6	83.1
(0, 1, 0)	Relation only	82.47	76.8	81.4
(0, 0, 1)	Subgraph only	79.83	74.5	79.7
(0.33, 0.33, 0.33)	Uniform	88.76	81.9	86.1
(0.4, 0.3, 0.3)	**Ours**	**89.39**	**82.3**	**86.5**

Table 13: Sensitivity of semantic score weights (α, β, γ). Match % is the retrieval match rate (fraction where evidence fully covers gold entities and relations).

Any single granularity alone leads to notable degradation (EM drops by 3.7–7.8), validating the necessity of multi-granularity matching. Our default $(0.4, 0.3, 0.3)$ performs comparably to the uniform setting (82.3 vs. 81.9 EM), indicating that performance is not sensitive to exact weights as long as all granularities are included.

N Retrieval vs. Generation Error Decomposition

We conduct a fine-grained error analysis on the GrailQA dev set under the Oracle CRP setting, decomposing errors into **retrieval failures** (gold entities/relations absent from evidence) and **generation failures** (evidence sufficient but LLM generates incorrect logical form).

Generalization Level	n	Retrieval	Generation
I.I.D.	1,593	5.6%	2.1%
Compositional	1,514	14.4%	15.9%
Zero-shot	3,656	21.3%	12.8%
Overall	**1,829 err.**	**59.4%**	**40.6%**

Table 14: Retrieval vs. generation error rates (%) on GrailQA dev under Oracle CRP setting.

On i.i.d. questions, both error rates are minimal. On compositional questions, generation failure slightly exceeds retrieval failure—the LLM struggles to compose known relations into novel structures even with sufficient evidence. On zero-shot questions, retrieval failure dominates (21.3%), as unseen schemas are inherently harder to retrieve. Further analysis shows that 75.8% of generation failures involve the LLM predicting extraneous relations absent from the gold logical form, suggesting that such noise in model outputs could benefit from data-level detection strategies (Liu, 2024).

O Cross-KB CRP Transferability

To evaluate the transferability of CRP patterns beyond Freebase, we extract CRP pattern coverage on KQA Pro, a Wikidata-based KBQA benchmark (94K train, 12K val).

	CRP-Covered				Not Covered	
Split	**1-hop**	**2-hop**	**Ans-c**	**Total**	**Multi**	**Attr**
Train	56.4	14.4	6.9	**77.7**	4.9	17.4
Val	56.5	14.5	6.8	**77.7**	4.9	17.3

Table 15: CRP pattern coverage (%) on KQA Pro (Wikidata). “Ans-c” = answer-centered patterns; “Multi” = >2-hop; “Attr” = zero-hop attribute lookups.

All nine CRP patterns are observed on KQA Pro, covering 77.7% of questions. The Attr-only portion (17.3–17.4%) corresponds to zero-hop attribute lookups that can be handled by adding a simple zero-hop CRP. The Multi-hop portion (4.9%) would benefit from extending CRP to longer-hop variants. Overall, adapting SELF-KBQA to a new KB mainly requires replacing KB-specific components (entity linking, relation textualization) while the retrieval framework and CRP patterns remain applicable.

P Case Study: SELF-KBQA Pipeline Execution

To better understand the advantages of SELF-KBQA over separate retrieval methods, we present a detailed case study in Tables 16, 17, and 18.

As shown in Table 16, the baseline fails to generate the true logical form because it retrieves entities and relations in isolation, missing the critical relation `audio_books_read` which lacks explicit semantic overlap with the question.

In contrast, our method first explicitly models the ambiguity of the entity "Michael Prichard". Table 17 shows that SELF-KBQA successfully enumerates the subgraphs for three distinct interpretations: the Audio Reader (`m.0g8psb0`), the Actor (`m.0jwcpkb`), and the Author (`m.05xcc34`).

Finally, Table 18 demonstrates the effectiveness of our structure-aware ranking. Although the “Book Author” subgraph (E3) presents a strong semantic distractor (Score: 0.66) due to the correlation between "author" and "published", our model correctly identifies the "Audio Reader" subgraph (E1) as the top candidate (Score: **0.70**).

Separate Retrieval Methods (Baseline)	SELF-KBQA (Ours)
Entity Retrieval • `m.0g8psb0(people.person)` • `m.0jwcpkb(people.person)`	**Entity Set and CRP Prediction** • **E1**: `m.0g8psb0` (**`book.audio_book_reader`**) • **E2**: `m.0jwcpkb` (`film.actor`) • **E3**: `m.05xcc34` (`book.author`) • Path: topic_entity → mid_entity → answer_entity
Relation Retrieval • **Retrieved**: – `book.book_edition.place_of_publication` – `book.periodical.publisher` – `book.journal.place_of_publication` – ... – `travel.travel_destination.guidebooks` – ...	**Subgraph Enumeration** **Method**: Expands k-hop subgraphs (Nodes + Edges) for each entity. **Enumerated Subgraphs**: • **E1 (Reader)**: $\texttt{E1} \xrightarrow{\texttt{book.audio\_book\_reader.audio\_books\_read}} \texttt{Book}$ • **E2 (Actor)**: $\texttt{E2} \xrightarrow{\texttt{film.performance.actor}} \texttt{FilmActor}$ • **E3 (Author)**: $\texttt{E3} \xrightarrow{\texttt{book.written\_work.author}} \texttt{Author}$
Logical Form Generation **Wrong Logical Form**: `(AND travel.travel_destination (JOIN (R book.book_edition.place_of_publication) (JOIN` **`book.book_edition.book_authors`** `m.0g8psb0)))`	**Subgraph Ranking** **Method**: Ranking subgraphs based on structural score and semantic score for each entity. • **Subgraph of E1 (Reader)**: Edges: $\xrightarrow{\texttt{...audio\_books\_read}} \ldots$ Score: 0.70 • **Subgraph of E3 (Author)**: Edges: $\xrightarrow{\texttt{...written\_work.author}} \ldots$ Score: 0.66 • **Subgraph of E2 (Actor)**: Edges: $\xrightarrow{\texttt{...performance.actor}} \ldots$ Score: 0.64
Analysis • **Error due to the wrong relation**	**Logical Form Generation** **Correct Logical Form** `(AND travel.travel_destination (JOIN (R book.book_edition.place_of_publication) (JOIN (R book.audio_book_reader.audio_books_read) m.0g8psb0)))`

Table 16: Case Study of SELF-KBQA.

Entity Candidate	Enumerated Subgraphs (Main Path & Edges)
E1: m.0g8psb0 *Type: Audio Reader*	**Subgraph 1:** **Main Path**: `topic_entity -> mid_entity -> answer_entity` **Edges**: $\texttt{E1} \xrightarrow{\texttt{book.audio\_book\_reader.audio\_books\_read}} \texttt{AudioBook}$ $\texttt{Pattern} \xrightarrow{\texttt{book.book\_edition.place\_of\_publication}} \texttt{Location}$
	Subgraph 2: **Pattern**: `topic_entity -> answer_entity` **Edges**: $\texttt{E1} \xrightarrow{\texttt{people.person.nationality}} \texttt{Country}$
	Subgraph 3: **Pattern**: `topic_entity -> mid_entity -> answer_entity` **Edges**: $\texttt{E1} \xrightarrow{\texttt{education.education.student}} \texttt{School}$
E3: m.05xcc34 *Type: Book Author*	**Subgraph 1:** **Main Path**: `topic_entity -> mid_entity -> answer_entity` **Edges**: $\texttt{E3} \xrightarrow{\texttt{book.written\_work.author}} \texttt{WrittenWork}$ $\texttt{WrittenWork} \xrightarrow{\texttt{book.book\_edition.place\_of\_publication}} \texttt{Location}$
	Subgraph 2: **Pattern**: `topic_entity -> answer_entity` **Edges**: $\texttt{E3} \xrightarrow{\texttt{people.person.nationality}} \texttt{Country}$
E2: m.0jwcpkb *Type: Film Actor*	**Subgraph 1:** **Main Path**: `topic_entity -> mid_entity -> answer_entity` **Edges**: $\texttt{E2} \xrightarrow{\texttt{film.performance.actor}} \texttt{FilmCharacter}$ $\texttt{FilmCharacter} \xrightarrow{\texttt{film.film.starring}} \texttt{Movie}$
	Subgraph 2: **Pattern**: `topic_entity -> mid_entity -> answer_entity` **Edges**: $\texttt{E2} \xrightarrow{\texttt{people.profession.people\_with\_this\_profession}} \texttt{Profession}$

Table 17: Case study of Subgraph Enumeration and Structural Expansion.

Entity	Rank	Subgraph Candidate	Sem.	Str.	Final
E1: m.0g8psb0 *(Audio Reader)*	**1**	**Subgraph 1**	**0.50**	**1.00**	**0.70**
	2	Subgraph 2	0.60	0.40	0.50
	3	Subgraph 3	0.20	0.30	0.25
E3: m.05xcc34 *(Book Author)*	**1**	**Subgraph 1**	0.43	1.00	0.66
	2	Subgraph 2	0.61	0.40	0.51
E2: m.0jwcpkb *(Film Actor)*	**1**	**Subgraph 1**	0.38	1.00	0.64
	2	Subgraph 2	0.30	0.50	0.40

Table 18: Case Study of Fine-grained Subgraph Ranking.