Execution as Verification: Fine-Grained Self-Correcting Reasoning for Complex KBQA

Minghan Zhang, Zhen Yang, Haodong Zou, Jie Chen, Zhen Duan, Shu Zhao*
School of Computer Science and Technology, Anhui University, China
z17333121752@163.com, uscyz094@gmail.com
zou_hd@163.com, chenjie200398@163.com
dz@ahu.edu.cn, zhaoshuzs2002@hotmail.com

Abstract

Knowledge Base Question Answering (KBQA) leverages structured knowledge bases to offer superior interpretability and hallucination resistance, making it a critical technology for precise knowledge reasoning. However, the prevailing LLM-based generate-then-execute formulation of semantic parsing is limited by strict syntactic constraints, making it primarily prone to structural deviations that render queries unexecutable, while suffering from semantic deviations that yield incorrect execution results. To address these challenges, we propose the Execution as Verification (EVER) framework, reframing semantic parsing as an iterative, self-correcting reasoning process driven by execution feedback. First, motivated by the insight that query executability serves as a strong proxy for answer correctness, we introduce Fine-Grained Execution-Aware Planning. This mechanism decomposes complex semantic parsing into a sequence of stepwise reasoning processes oriented by executability verification, ensuring high query executability. We further design a Self-Guided Semantic Correction mechanism based on execution result verification, utilizing execution feedback to verify and calibrate semantic deviations, thereby ensuring the semantic correctness of executable queries. Experimental results on the WebQSP and CWQ datasets demonstrate that our method achieves significant improvements in both query executability and answer accuracy, achieving state-of-the-art performance, particularly in complex multi-hop scenarios. Our code is available at https://github.com/ahu-zmh/EVER.

1 Introduction

Knowledge Base Question Answering (KBQA) serves as a critical technology for bridging the gap between unstructured natural language and structured knowledge bases, playing an indispensable role in precise knowledge reasoning (Lan et al., 2021). Compared to pure language modeling approaches, KBQA leverages the factual precision of knowledge bases to generate answers with high interpretability and resistance to hallucinations (Ji et al., 2023; Chen, 2024). Recently, the integration of Large Language Models (LLMs) has infused KBQA with enhanced semantic understanding and generalization capabilities, enabling systems to handle increasingly complex natural language queries (Brown et al., 2020).

Current LLM-based KBQA methods primarily divide into Information Retrieval (IR) (Sun et al., 2024; Pan et al., 2024; Liang and Gu, 2025) and Semantic Parsing (SP) (Li et al., 2023; Jiang et al., 2023a; Luo et al., 2024a; Tian et al., 2025). The LM-based IR paradigm typically retrieves relevant knowledge subgraphs to augment the LLM for direct answer generation. While it excels in precise retrieval (high Hit@1), its lack of explicit logical modeling leads to suboptimal F1 scores in complex tasks such as counting and comparison, limiting its applicability in complex real-world scenarios, as illustrated in Figure 1(a). In contrast, LLM-based SP has emerged as a more promising paradigm due to its capability to generate explicit logical structures. However, as shown in Figure 1(b), existing LLM-based SP methods are constrained by the strict syntactic constraints of logical forms, where even minor structural deviations can render queries unexecutable—a fragility that is particularly pronounced when generating long logical chains. Simultaneously, these methods are also susceptible to semantic deviations, leading to incorrect execution results even if the query itself is executable.

Our research is first driven by a critical insight: the executability of a query serves as a strong proxy for its correctness. This is supported by our empirical analysis of representative methods (e.g., ChatKBQA (Luo et al., 2024a)), which reveals a high correlation (approximately 90%) between successful execution and answer accuracy. This high cor-

*Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 37646–37663
July 2-7, 2026

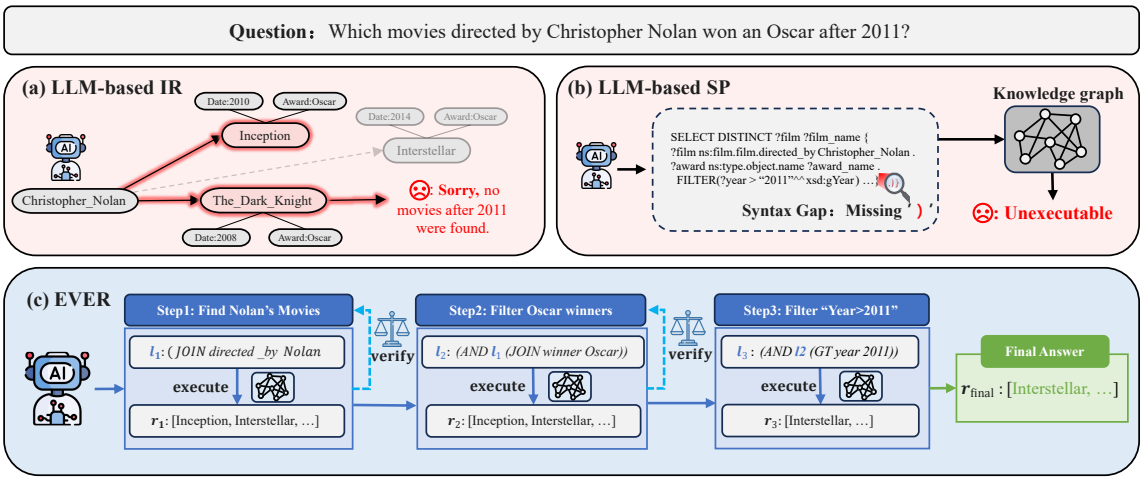


Figure 1: Different paradigms of KG-enhanced LLM reasoning. (a) LM-based IR: Misses the correct answer due to limited exploration width; (b) LLM-based SP: Fails as a minor syntax error renders the entire logical query unexecutable; (c) EVER: Achieves effective reasoning for complex questions via fine-grained execution-aware planning and self-guided semantic correction.

relation stems from the KB's Structural Sparsity, where incorrect logic typically fails to match valid graph paths, making executability a robust indicator of alignment with valid graph paths. This suggests that the primary bottleneck limiting current LLM-based SP performance lies in structural deviations—where syntactically invalid or structurally flawed logic prevents execution entirely. Therefore, our primary objective is to maximize executability by establishing execution as a foundational verification signal. However, solely maximizing executability is insufficient due to the risk of semantic deviations—cases where logic is executable but stems from semantic misalignment (e.g., querying a physicist instead of a director). Thus, a robust KBQA system must ensure both structural validity and semantic consistency.

To address these challenges, we propose the Execution as Verification (EVER) framework, reframing semantic parsing as an execution-driven, self-correcting reasoning process, as illustrated in Figure 1(c). First, to overcome structural deviations and maximize executability, we introduce Fine-Grained Execution-Aware Planning, which decomposes complex semantic parsing into a sequence of stepwise reasoning processes oriented by executability verification. We implement this via a hybrid decoding strategy: using greedy search to deterministically anchor the current step's intent, followed by beam search to explore intent-aligned executable sub-query skeletons within a constrained space. This fine-grained planning approach significantly reduces the structural complexity of single-step generation, effectively mitigating query failures caused by structural deviations.

Second, to rectify semantic deviations and ensure answer accuracy, we design a Self-Guided Semantic Correction mechanism based on execution result verification. Specifically, this module leverages feedback from each execution step to verify the consistency of input entity semantics, relation path intents, and result types. Upon detecting potential semantic shifts, the system triggers a dynamic rollback and candidate replacement process. This enables the reasoning process to iteratively converge toward the correct semantics under the guidance of execution feedback, thereby simultaneously achieving both syntactic executability and semantic correctness.

Finally, to equip the model with these capabilities, we construct a hybrid instruction tuning dataset featuring both positive reasoning and negative self-repair trajectories. Derived from standard logical forms via bottom-up fine-grained decomposition and augmented by explicitly injecting distractor samples such as entity ambiguity, this dataset simulates realistic reasoning failure scenarios. This design enables the model to learn fine-grained semantic verification and dynamic correction strategies. Extensive experiments on WebQSP (Yih et al., 2016) and CWQ (Talmor and Berant, 2018) demonstrate that our method achieves highly competitive results against existing baselines, delivering significant gains in query executability and answer

accuracy, especially for complex reasoning tasks. To conclude, our contributions are:

- To tackle the bottleneck of structural deviations in complex queries, we design the Fine-Grained Execution-Aware Planning mechanism. By reframing semantic parsing as a stepwise reasoning process oriented by executability verification, we maximize the executability rate of logical forms.
- To mitigate the risk of semantic deviations, we design the Self-Guided Semantic Correction mechanism. Leveraging execution result verification and dynamic rollback strategies, this module ensures strict semantic alignment between the logical form and the natural language question.
- We construct a hybrid instruction-tuning dataset incorporating both positive reasoning chains and negative self-repair trajectories to equip the model with robust execution and error-correction capabilities. Extensive experiments on WebQSP and CWQ demonstrate that our method outperforms existing baselines, achieving significant gains in both query executability and answer accuracy.

2 Related Work

2.1 Traditional KBQA Approaches

Early KBQA generally relied on supervised learning or template-based methods. Traditional Semantic Parsing approaches utilized pattern matching or ranking models to map natural language questions into logical forms (Cao et al., 2022; Ye et al., 2022; Zhang et al., 2023). Conversely, traditional Information Retrieval methods focused on extracting question-specific subgraphs and ranking candidate entities based on distributed representations (Saxena et al., 2020; He et al., 2021; Zhang et al., 2022). However, these approaches heavily depend on human-annotated logical templates or massive labeled data, limiting their generalization capability to unseen complex queries.

2.2 LLM-based Information Retrieval

With the emergence of Large Language Models (LLMs), the IR paradigm has increasingly adopted an iterative reasoning paradigm. In this framework, LLMs function as agents to dynamically explore multiple reasoning paths within the KBs and make decisions accordingly (Sun et al., 2024; Luo et al., 2024c; Chen et al., 2024; Ma et al., 2025; Xiao et al., 2025; Zhang et al., 2026). While these methods excel at pinpointing relevant entities (achieving high Hit@1), they face inherent limitations in explicit logical modeling. The lack of structured symbolic representation makes it difficult for LM-based IR methods to handle questions involving precise arithmetic operations (e.g., counting), comparisons, or complex constraints, often leading to suboptimal F1 scores in multi-answer scenarios.

2.3 LLM-based Semantic Parsing

The LLM-based SP paradigm aims to directly translate natural language into executable logical forms, adopting a "Generate-then-Execute" strategy (Jiang et al., 2023a; Fang et al., 2024; Luo et al., 2025; Feng and He, 2025; Tian et al., 2025; Zhang et al., 2025). Recent works like KB-Binder (Li et al., 2023) and ChatKBQA (Luo et al., 2024a) utilize In-Context Learning (ICL) or fine-tuning to enhance logical form generation. However, these methods are constrained by the intrinsic rigidity of logical languages, where minor structural deviations or entity misalignments render queries unexecutable. Due to the absence of error correction mechanisms, a single local deviation often irreversibly precipitates total system failure. Some approaches (e.g., PER-KBQA (Guo et al., 2023)) attempt iterative refinement via execution feedback. However, such post-hoc compensation fails to recover if fatal structural errors render the initial logical form completely unexecutable.

In contrast to such brittle generation or post-hoc refinement processes, EVER reframes semantic parsing as an self-correcting reasoning process. By treating execution feedback as a continuous verification signal, our method ensures both syntactic executability and semantic consistency. This approach effectively mitigates the dual risks of structural fragility and semantic misalignment.

3 Preliminaries

Knowledge Base. We formalize the knowledge base as a large-scale multi-relational graph $\mathcal{K}$, where facts are stored as triples in the form of (s, r, o). Here, s denotes the subject entity, r the relation, and o the object entity or literal. Each entity within the KB is indexed by a Unique Identifier (UID) to ensure uniqueness.

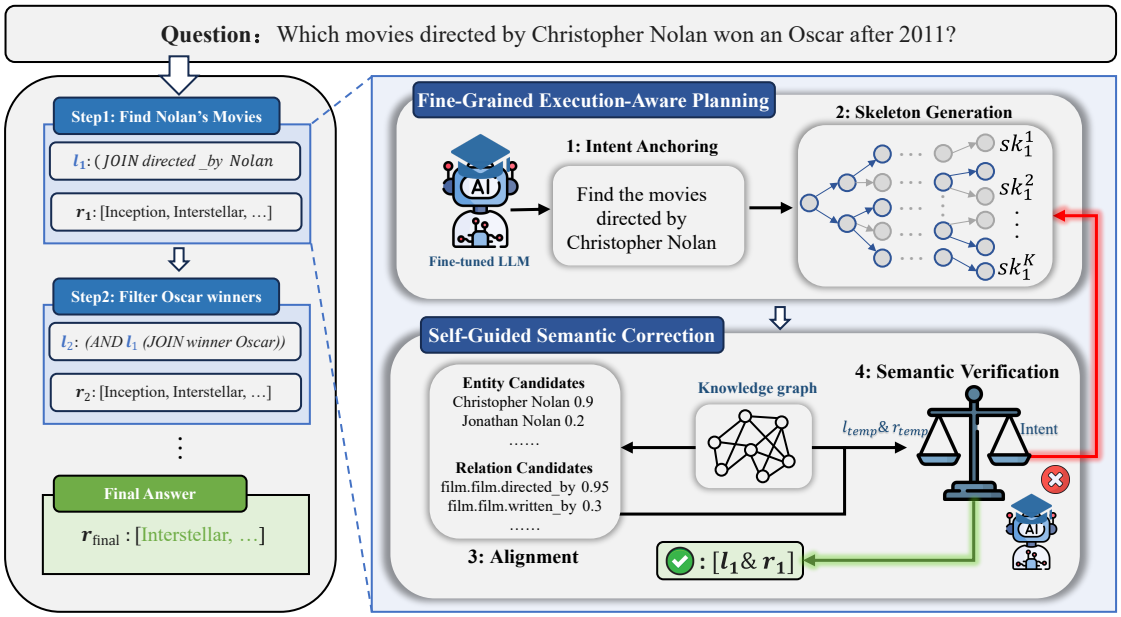


Figure 2: The overview of our Execution as Verification framework. The system reframes semantic parsing as an iterative reasoning process, consisting of two core components: (1) Fine-Grained Execution-Aware Planning, which decomposes complex queries into stepwise atomic operations via hybrid decoding; and (2) Self-Guided Semantic Correction, which leverages execution feedback to verify semantic consistency and trigger dynamic rollbacks for error recovery.

Logical Form. We adopt S-expressions as the logical form due to their superior compositionality. The core comprises projection operations and high-order operators. Projections perform single-hop traversals using the `JOIN` operator for both forward $(?, r, o)$ and backward $(s, r, ?)$ queries. Furthermore, high-order operators facilitate set operations (e.g., `AND`), aggregations (e.g., `COUNT`, `ARGMAX`), and comparisons (e.g., `GT`). This enables complex multi-hop reasoning via recursive nesting. Our logical form maps deterministically to SPARQL, allowing direct conversion. Detailed definitions are provided in Appendix B.

Problem Definition. The KBQA task is formalized as a semantic parsing process: given a natural language question Q and a knowledge base $\mathcal{K}$, the objective is to first parse Q into a structured logical form L, which is subsequently converted into an executable query q to retrieve the answer set A via the execution function $\text{Execute}(q|\mathcal{K})$.

4 Methodology

In this section, we present the Execution as Verification (EVER) framework, which reframes complex multi-hop KBQA as an iterative, self-correcting reasoning process. As illustrated in Figure 2, the verification role of execution is explicitly manifested in two dimensions: (1) Fine-Grained Execution-Aware Planning, which employs executability status (i.e., success or failure) as a verification signal to ensure structural validity; and (2) Self-Guided Semantic Correction, which leverages execution results to verify semantic consistency.

4.1 Fine-Grained Execution-Aware Planning

To address the bottleneck of structural deviations—where generated complex logic often fails to execute due to minor structural errors—we propose Fine-Grained Execution-Aware Planning. The core insight is that decomposing complex reasoning into a sequence of atomic, verifiable decision processes can significantly reduce the structural complexity of single-step generation, thereby inherently enhancing the executability rate of each step.

Specifically, we decompose the generation process into T discrete time steps. At each step t, we maintain two key state variables based on the historical reasoning trajectory $H_{t-1} = \{(l_0, r_0), \ldots, (l_{t-1}, r_{t-1})\}$:

- **Logical Form (l_t):** The cumulative query statement constructed up to the current step. It is a nested structure formed by combining

the currently generated atomic operator with the logic l_{t-1} from the previous step.

- **Execution Result (r_t):** The feedback obtained by executing l_t on the knowledge base $\mathcal{K}$, i.e., $r_t = \text{Execute}(l_t|\mathcal{K})$.

Under this planning framework, to ensure that the generated single-step logic is not only structurally correct but also intentionally precise, we employ a Hybrid Decoding Strategy. This strategy leverages the historical context H_{t-1} to further decouple single-step reasoning into two phases:

Phase 1: Intent Anchoring. First, the model utilizes greedy search to generate the reasoning intent (I_t) in natural language. This step aims to anchor the reasoning direction and prevent semantic drift before the logic is grounded:

$$I_t = \text{LLM}_{\text{greedy}}(Q, H_{t-1}). \quad (1)$$

Phase 2: Skeleton Generation. Next, conditioned on the generated intent I_t, the model utilizes beam search to explore K candidate logical forms $\mathcal{S}_t = \{sk_t^1, \ldots, sk_t^K\}$. The core motivation for employing beam search over greedy decoding is Structural Fault Tolerance. Complex logic generation is prone to local optima, where a single skeleton with a structural error leads to irreversible reasoning failure. By retaining K high-confidence candidate skeletons, we maximize the coverage of correct logical structures with minimal computational cost, providing a search space for subsequent structure-level rollback:

$$\mathcal{S}_t = \text{LLM}_{\text{beam}}(Q, H_{t-1}, I_t). \quad (2)$$

4.2 Self-Guided Semantic Correction

To mitigate the risk of semantic deviations—where a query executes successfully but returns factually incorrect results due to semantic misalignment—we introduce the Self-Guided Semantic Correction mechanism. This mechanism leverages execution feedback as a signal to dynamically identify the first logical form from the skeleton set $\mathcal{S}_t$ that simultaneously satisfies both syntactic executability and semantic consistency.

Phase 3: Alignment. The candidate skeletons generated in the previous step contain ungrounded surface names for entities (N_{ent}) and relations (N_{rel}). To transform these into executable queries, we must precisely map them to their Unique Identifiers (UIDs) in the KB.

For Entity Alignment, we first attempt to match N_{ent} exactly against KB labels. Matches are ranked by popularity using FACC1 (Gabrilovich et al., 2013) scores. If exact matching fails, we employ the BM25 (Robertson and Zaragoza, 2009) algorithm to retrieve semantically relevant entities. In both scenarios, the top-k_{ent} entities are retained as the candidate set $\mathcal{E}_{cand}$. For Relation Alignment, we employ SimCSE (Gao et al., 2021) to compute semantic similarity with KB relations, retaining the top-k_{rel} results as $\mathcal{R}_{cand}$. Finally, we generate candidate combinations from $\mathcal{E}_{cand}$ and $\mathcal{R}_{cand}$, sorting them in descending order based on the joint score of FACC1 and SimCSE to form a prioritized verification list.

Phase 4: Semantic Verification. We perform an "Execute-Verify" test on prioritized candidates. For each combination $c_i \in \mathcal{E}_{cand} \times \mathcal{R}_{cand}$, the system instantiates the current skeleton sk_t and integrates the historical state l_{t-1} to construct a temporary logical form l_{temp}, yielding the execution result:

$$r_{temp} = \text{Execute}(l_{temp}|\mathcal{K}), \quad (3)$$

Subsequently, the framework retrieves metadata (e.g., descriptions, types). The fine-tuned LLM evaluates the semantic consistency of both the candidate and the result with the query intent (I_t). Verification failures (e.g., the intent is "movies" but results are "books") trigger a candidate-level rollback, discarding the current candidate to attempt the next without reverting earlier reasoning steps.

The search loop terminates immediately upon identifying the first executable and semantically aligned instance, formalizing the current state (i.e., $l_t \leftarrow l_{temp}, r_t \leftarrow r_{temp}$). To maintain controllable inference costs and guarantee system termination, we impose a *Max Retry Limit* that bounds the number of candidate combinations evaluated within each individual reasoning step. At this stage, the model makes a final decision based on the result: if the task goal is deemed achieved (i.e., the current result is the desired answer), it outputs a termination token (`Final Answer: <`r_t`>`) to conclude reasoning; otherwise, the system stores the current state in the historical context and advances to the next time step to generate a new plan.

4.3 Instruction Tuning Strategy

To equip the model with fine-grained planning capabilities and dynamic error-correction skills, we

construct a hybrid instruction tuning dataset comprising both Positive Reasoning Chains and Negative Self-Repair Trajectories.

The construction process begins by converting the SPARQL queries from the training set into equivalent S-expression logical forms. To simulate the ungrounded state prior to entity linking, we replace the Machine Identifiers (MIDs) with their natural language surface names (e.g., `m.03bxz` → `[Christopher Nolan]`). Subsequently, we employ a Bottom-Up recursive parsing algorithm to perform a post-order traversal of the syntax tree, decomposing the complex query into a linear sequence of atomic operations $l_1, l_2, \ldots, l_T$. This sequence serves as the fundamental skeleton for stepwise execution.

Building on these skeletons, we utilize GPT-4 to synthesize explicit reasoning thoughts for each atomic step. The model generates a `[Plan]` to articulate the intent and a `[Reflection]` to validate the execution feedback. These samples constitute the positive training data, teaching the model standard intent planning and result verification.

To address semantic deviations—where queries execute successfully but yield incorrect answers—we introduce Negative Self-Repair Augmentation. We employ GPT-4 to synthesize scenarios involving executable but semantically flawed logic, such as linking to an entity with an identical name but divergent type (e.g., a Physicist instead of a Director). This trains the system to detect semantic inconsistencies within the execution feedback and explicitly trigger a `<RETRY>` action. The detailed prompts used for data synthesis and illustrative examples of the reasoning trajectories are provided in Appendix F.

5 Experiments

5.1 Experimental Setup

Datasets. We conduct experiments on two standard benchmarks grounded in the Freebase knowledge base (Bollacker et al., 2008). WebQSP (Yih et al., 2016) contains 4,737 natural language questions paired with SPARQL queries, serving as a fundamental testbed for semantic parsing. To evaluate performance on more sophisticated reasoning, we also employ CWQ (Talmor and Berant, 2018). This dataset comprises 34,689 samples involving intricate multi-hop logic and constraints, which demand significantly stronger reasoning capabilities from the model. Detailed statistics and construction procedures for the instruction tuning data are provided in Appendix C.

Baselines. We compare our proposed method with a comprehensive set of KBQA baselines, primarily categorized into the following four groups: 1) Traditional IR methods, including NSM+h (He et al., 2021), Rigel (Sen et al., 2021), and UniKGQA (Jiang et al., 2023b); 2) Traditional SP methods, including QGG (Lan and Jiang, 2020), UnifiedSKG (Xie et al., 2022), TIARA (Shu et al., 2022), and FC-KBQA (Zhang et al., 2023); 3) LLM-based IR methods, including ToG (Sun et al., 2024), RoG (Luo et al., 2024b), PoG (Chen et al., 2024), and EPERM (Long et al., 2025); and 4) LLM-based SP methods, including KB-Binder (Li et al., 2023), ChatKBQA (Luo et al., 2024a), RGR-KBQA (Feng and He, 2025), CompKBQA (Tian et al., 2025), and Rule-KBQA (Zhang et al., 2025). Detailed descriptions of these baselines are provided in Appendix A.

Evaluation Metrics. Following previous works (Luo et al., 2024a; Feng and He, 2025; Tian et al., 2025), we adopt two standard metrics to evaluate the performance: F1 score and Hits@1. Specifically, Hits@1 measures whether the top-ranked prediction contains the correct answer, F1 score evaluates the overlap between the predicted and ground-truth answer sets, and Acc denotes the strict exact-match accuracy.

Hyperparameters and Environment . We employ Llama-2-7B and Llama-2-13B (Touvron et al., 2023) as backbone models for WebQSP and CWQ, respectively. The models are fine-tuned for 50 epochs on WebQSP and 5 epochs on CWQ using LoRA with a rank of $r = 8$. All experiments are conducted on a single NVIDIA A100 GPU. Detailed implementation settings are provided in Appendix E.

5.2 Main Results

Table 1 presents the performance comparison of EVER against state-of-the-art baselines on the WebQSP and CWQ datasets. Overall, our framework consistently outperforms other baseline methods. First, we observe that LLM-based methods significantly outperform traditional baselines, validating the necessity of leveraging the powerful generalization capabilities of LLMs for complex semantic parsing tasks.

Compared with LLM-based Semantic Parsing

Method	WebQSP		CWQ	
	F1	Hits@1	F1	Hits@1
Traditional IR Methods				
NSM+h (He et al., 2021)	67.4	74.3	44.0	48.8
Rigel (Sen et al., 2021)	-	73.3	-	48.7
UniKGQA (Jiang et al., 2023b)	77.2	72.2	49.4	51.2
Traditional SP Methods				
QGG (Lan and Jiang, 2020)	73.0	74.0	40.4	44.1
UnifiedSKG (Xie et al., 2022)	-	73.9	-	68.8
TIARA (Shu et al., 2022)	78.9	75.2	-	-
FC-KBQA (Zhang et al., 2023)	78.8	82.1	56.4	70.4
LLM-based IR Methods				
ToG (Sun et al., 2024)	-	82.6	-	69.5
RoG (Luo et al., 2024b)	70.8	85.7	56.2	62.6
PoG (Chen et al., 2024)	-	87.3	-	75.0
EPERM (Long et al., 2025)	72.4	**88.8**	58.9	66.2
LLM-based SP Methods				
KB-Binder (Li et al., 2023)	74.4	-	-	-
ChatKBQA (Luo et al., 2024a)	79.8	83.2	77.8	82.7
RGR-KBQA (Feng and He, 2025)	80.7	84.5	76.6	82.0
CompKBQA (Tian et al., 2025)	81.3	84.2	79.4	83.6
Rule-KBQA (Zhang et al., 2025)	**81.9**	84.1	-	73.5
EVER (Ours)	81.8	86.9	**81.2**	**84.9**

Table 1: Performance comparison on WebQSP and CWQ datasets. The best results are bolded.

methods, EVER achieves consistent improvements. Specifically, against ChatKBQA, which employs a standard "generate-then-execute" paradigm, our method demonstrates a substantial advantage, improving F1 scores by 2.0% on WebQSP and expanding the lead to 3.4% on the complex CWQ dataset. Furthermore, EVER consistently maintains superior performance over other state-of-the-art baselines. This trend suggests that while recent SP methods (e.g., Rule-KBQA) achieve highly comparable performance on simple queries, our proposed "Execution as Verification" paradigm is far more effective in handling complex multi-hop reasoning, as fine-grained planning and verification are critical for generating structurally correct logical forms.

Compared with LLM-based IR methods, EVER demonstrates superior robustness. On the complex CWQ dataset, IR methods suffer from error propagation inherent in iterative retrieval. In contrast, EVER mitigates this via Self-Guided Semantic Correction, outperforming the best IR baseline by 10.8% in F1 and 15.4% in Hits@1. Furthermore, while IR methods achieve decent Hits@1 on WebQSP, their limited retrieval width leads to low F1 scores in multi-answer scenarios. Conversely, EVER employs fine-grained planning to generate symbolic queries that comprehensively cover semantic constraints, ensuring both high precision and recall. Zero-shot experiments in Appendix G further confirm our model's robust transferability to unseen datasets.

Model Variants	F1	Hits@1
EVER (Full Method)	**81.2**	**84.9**
Generation		
w/o Fine-Grained Planning	78.4 (-2.8)	82.9
w/o Semantic Correction	79.3 (-1.9)	83.0
w/o Negative Training	80.1 (-1.1)	84.1
Retrieval		
w/o Relation Retrieval	79.8 (-1.4)	83.2
w/o Entity Retrieval	77.1 (-4.1)	80.5

Table 2: Ablation study of different components on the CWQ dataset. Values in parentheses denote the performance drop compared to the full model.

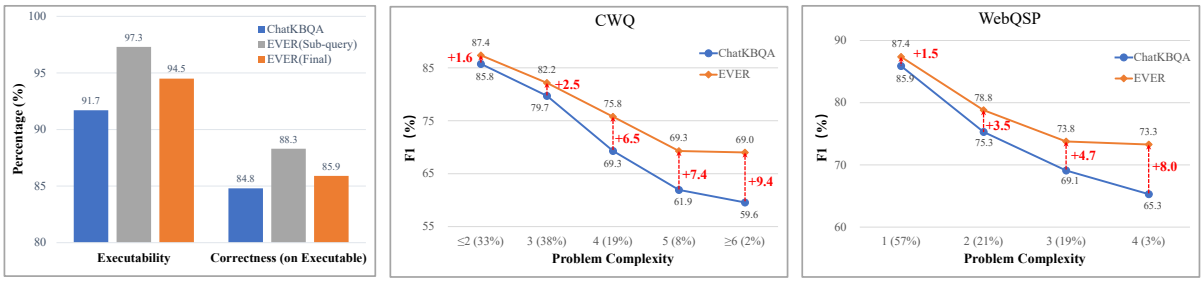
5.3 Ablation Study

To investigate the contributions of each component in our Execution as Verification framework, we conduct ablation studies on the CWQ dataset. The results are presented in Table 2.

First, removing Fine-Grained Planning causes the most significant drop (ΔF1 = -2.8%), confirming that stepwise decomposition is essential for managing structural complexity in multi-hop queries, whereas one-shot generation often yields irreversible structural errors due to the lack of intermediate checkpoints to arrest error propagation. Second, ablating Semantic Correction leads to a 1.9% decline. This underscores that execution feedback is vital for filtering out semantic mismatches that linguistic checks miss. Additionally, excluding Negative Training results in a slight drop (ΔF1 = -1.1%), validating the benefit of explicit learning from failure-recovery scenarios. Finally, the disparity between removing Entity Retrieval (ΔF1 = -4.1%) and Relation Retrieval (ΔF1 = -1.4%) highlights that bridging the vocabulary gap for massive entities necessitates robust retrieval, whereas closed relation sets favor exact matching.

5.4 In-depth Analysis

In this section, we provide a comprehensive analysis of the two core mechanisms in EVER: Fine-Grained Planning and Self-Guided Semantic Correction. We aim to empirically validate their effectiveness and demonstrate how they contribute to structural stability and semantic correctness.



(a) (b$_1$) (b$_2$)

Figure 3: Detailed performance analysis. (a) Execution Quality Analysis: We report the Final Executability and Correctness on Executable queries for ChatKBQA and EVER on the complex CWQ dataset. The gray bars specifically denote the execution success rate of EVER's intermediate atomic sub-queries. (b) Performance by Complexity: We plot the F1 scores against problem complexity, defined by the number of relations in the logical form, across both CWQ and WebQSP.

5.4.1 Robustness of Fine-Grained Planning

We evaluate the Fine-Grained Execution-Aware planning mechanism by examining its foundational execution stability and its capability to mitigate error propagation in complex scenarios.

Execution Quality. As illustrated in Figure 3(a), our Fine-Grained Planning strategy significantly enhances structural stability on the CWQ dataset compared to the one-shot generation of ChatKBQA. By decomposing complex logic into atomic operations, EVER achieves a remarkable sub-query execution rate of 97.3%. This foundational stability effectively mitigates structural collapse, resulting in a final query executability of 94.5%, which surpasses the baseline by 2.8%. Furthermore, EVER maintains a higher answer accuracy on the executable subset, confirming that our planning yields logic that is not only structurally executable but also semantically aligned.

Resilience to Complexity. Leveraging high execution quality, we further verify robustness against increasing reasoning depth. As illustrated in Figure 3(b), EVER demonstrates exceptional resilience. While performing comparably on simple queries, the advantage of EVER expands significantly as the reasoning chain lengthens. Notably, on the most challenging subset of CWQ (relations ≥ 6), EVER achieves a 9.4% improvement over ChatKBQA. This confirms that fine-grained planning effectively mitigates error propagation, ensuring reasoning robustness in long logical chains by continuously anchoring the reasoning process to valid KB facts.

Metric Category	Dataset	
	WebQSP	**CWQ**
Trigger Rate	14.5%	20.2%
Rescue Rate	69.5%	58.8%
Retry Distribution:		
1 Iteration	86.7%	79.9%
2 Iterations	8.1%	12.1%
$\geq$ 3 Iterations	5.2%	8.0%

Table 3: Statistical breakdown of the Self-Guided Correction mechanism. We report the Trigger Rate (percentage of atomic sub-queries triggering rollbacks), the Rescue Rate (percentage of triggered queries successfully corrected), and the Retry Distribution to evaluate the computational cost.

5.4.2 Effectiveness of Self-Guided Correction

To validate the effectiveness of the Self-Guided Correction mechanism, we report the triggering and resolution patterns in Table 3. The substantial Trigger Rates underscore the ubiquity of semantic deviations (e.g., entity linking errors). In response, our mechanism demonstrates a robust Rescue Rate, successfully salvaging 69.5% and 58.8% of these erroneous queries on WebQSP and CWQ, respectively. This indicates that EVER effectively converts potential failures (i.e., executable but incorrect results) into semantically aligned queries via semantic calibration. Furthermore, the Retry Distribution confirms the system's manageability regarding efficiency: the vast majority of corrections are resolved within a single iteration. This demonstrates fast convergence, ensuring precise logic grounding without incurring excessive com-

putational overhead. For a comprehensive analysis of the system's efficiency, including comparisons of average LLM calls and inference time against baselines, please refer to Appendix D.

6 Conclusion

In this paper, we introduce the EVER framework, reframing semantic parsing as an iterative, self-correcting reasoning process. We propose Fine-Grained Execution-Aware Planning to maximize executability via stepwise decomposition, and Self-Guided Semantic Correction to rectify semantic misalignment using execution feedback. Extensive experiments on WebQSP and CWQ demonstrate that EVER significantly outperforms state-of-the-art baselines, particularly in complex multi-hop scenarios. Our findings underscore the critical role of execution feedback, offering a robust paradigm for neuro-symbolic reasoning.

Limitations

While the computational overhead of the EVER framework remains within a manageable range, it does incur an increase compared to one-shot generation methods due to its fine-grained planning and interactions with the knowledge base. Additionally, although our system verifies the consistency between the reasoning intent and the logical form, LLMs face a challenge in distinguishing whether an empty execution result stems from a semantic misalignment or knowledge base incompleteness. This absence of result-based grounding mechanisms may occasionally complicate the decision-making process for error recovery under sparse data conditions.

Acknowledgments

Our work is supported by the National Natural Science Foundation of China (62476003), Anhui Province Excellent Scientific Research and Innovation Team (2024AH010004), Anhui Provincial Natural Science Foundation- Water Science Joint Fund (2408055US006), the University Synergy Innovation Program of Anhui Province (GXXT2023-050), and SMP-Zhipu AI Large Model CrossDisciplinary Fund (SMP-Zhipu20240210). We also acknowledge the support from Zhipu AIAnhui University Joint Research Center, and the High-Performance Computing Platform of Anhui University.

References

Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250. ACM.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Shulin Cao, Jiaxin Shi, Zijun Yao, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jinghui Xiao. 2022. Program transfer for answering complex questions over knowledge bases. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8128–8140. Association for Computational Linguistics.

Huajun Chen. 2024. Large knowledge model: Perspectives and challenges. *DATA INTELLIGENCE*, 6(3):587–620.

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. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Haishuo Fang, Xiaodan Zhu, and Iryna Gurevych. 2024. DARA: decomposition-alignment-reasoning autonomous language agent for question answering over knowledge graphs. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 3406–3432. Association for Computational Linguistics.

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, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pages 3057–3070. Association for Computational Linguistics.

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

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics.

Yu Gu, Sue Kase, Michelle Vanni, Brian M. 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 '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 3477–3488. ACM / IW3C2.

Wangzhen Guo, Linyin Luo, Hanjiang Lai, and Jian Yin. 2023. From parse-execute to parse-execute-refine: Improving semantic parser for complex question answering over knowledge base. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 11771–11780. Association for 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 *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, pages 553–561. ACM.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12):248:1–248:38.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023a. Structgpt: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9237–9251. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2023b. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

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 and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 969–974. Association for Computational Linguistics.

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.

Xujian Liang and Zhaoquan Gu. 2025. Fast think-on-graph: Wider, deeper and faster reasoning of large language model on knowledge graph. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 24558–24566. AAAI Press.

Xiao Long, Liansheng Zhuang, Aodi Li, Minghong Yao, and Shafei Wang. 2025. EPERM: an evidence path enhanced reasoning model for knowledge graph question and answering. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 12282–12290. AAAI Press.

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. 2024a. 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, Jiaxin Ju, Bo Xiong, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2025. Chatrule: Mining logical rules with large language models for knowledge graph reasoning. In *Advances in Knowledge Discovery and Data Mining - 29th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2025, Sydney, NSW, Australia, June 10-13, 2025, Proceedings, Part II*, volume 15871 of *Lecture Notes in Computer Science*, pages 314–325. Springer.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024b. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Man Luo, Xin Xu, Zhuyun Dai, Panupong Pasupat, Mehran Kazemi, Chitta Baral, Vaiva Imbrasaite, and Vincent Zhao. 2024c. Dr.icl: Demonstration-retrieved in-context learning. *DATA INTELLIGENCE*, 6(4):909–922.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. 2025. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Trans. Knowl. Data Eng.*, 36(7):3580–3599.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Apoorv Saxena, Aditay Tripathi, and Partha P. Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4498–4507. Association for Computational Linguistics.

Priyanka Sen, Armin Oliya, and Amir Saffari. 2021. Expanding end-to-end question answering on differentiable knowledge graphs with intersection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 8805–8812. Association for Computational Linguistics.

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 bases. *CoRR*, abs/2210.12925.

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.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 641–651. Association for Computational Linguistics.

Llama Team. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

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.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Peng Xiao, Chao Liu, Wei Jia, and Lijun Dong. 2025. Aligned-entities-based fusion embedding on hetero-field knowledge graphs. *DATA INTELLIGENCE*, 7(3):618–635.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, and 4 others. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 602–631. Association for Computational Linguistics.

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, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. The Association for Computer Linguistics.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 5773–5784. Association for Computational Linguistics.

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), ACL 2023, Toronto, Canada, July 9-14, 2023, pages 1002–1017. Association for Computational Linguistics.

Minghan Zhang, Shu Zhao, Zhen Yang, Hongsheng Wu, Yongxing Lin, Haodong Zou, Jie Chen, and Zhen Duan. 2026. Thinking beyond the local: Multi-view instructed adaptive reasoning in kg-enhanced llms. In *Findings of the Association for Computational Linguistics: EACL 2026, Rabat, Morocco, March 24-29, 2026*, Findings of ACL, pages 6173–6188. Association for Computational Linguistics.

Zhiqiang Zhang, Liqiang Wen, and Wen Zhao. 2025. Rule-kbqa: Rule-guided reasoning for complex knowledge base question answering with large language models. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pages 8399–8417. Association for Computational Linguistics.

A Baseline Descriptions

We compare our proposed method with a comprehensive set of KBQA baselines. These methods are categorized into four groups: Traditional Information Retrieval (IR) methods, Traditional Semantic Parsing (SP) methods, LLM-based IR methods, and LLM-based SP methods.

1) Traditional IR Methods These methods typically retrieve answer candidates directly from the knowledge bases by scoring paths or subgraphs.

- **NSM+h** (He et al., 2021) introduces a teacher-student framework that utilizes the distributional consistency between forward and backward reasoning. It derives supervision signals at intermediate steps to mitigate spurious reasoning in the Neural State Machine.
- **Rigel** (Sen et al., 2021) extends the differentiable knowledge bases framework by incorporating a specialized intersection operation, enabling the explicit handling of multi-entity constraints in a fully differentiable manner.
- **UniKGQA** (Jiang et al., 2023b) unifies retrieval and reasoning within a shared parameter space, employing a semantic matching module combined with information propagation to jointly optimize both tasks.

2) Traditional SP Methods These methods focus on translating natural language questions into executable logical forms (e.g., SPARQL, S-expressions).

- **QGG** (Lan and Jiang, 2020) optimizes staged query graph generation. It incorporates constraints early in the generation process to effectively prune the search space for complex multi-hop questions.
- **UnifiedSKG** (Xie et al., 2022) frames various structured knowledge grounding tasks into a unified text-to-text format, leveraging the T5 model to generate outputs from linearized structured inputs.
- **TIARA** (Shu et al., 2022) utilizes constrained decoding to strictly regulate the output space of PLMs. It ensures the generation of syntactically executable logical forms while integrating multi-grained retrieval context.
- **FC-KBQA** (Zhang et al., 2023) proposes a fine-to-coarse composition framework. It reformulates fine-grained KB components into middle-grained knowledge pairs to balance the generalization and executability of logical expressions.

3) LLM-based IR Methods These methods leverage the parametric knowledge and reasoning capabilities of Large Language Models to guide the retrieval process.

- **ToG** (Sun et al., 2024) treats the LLM as a reasoning agent within an interactive paradigm, actively exploring external KGs via beam search to answer complex queries.
- **RoG** (Luo et al., 2024b) presents a planning-retrieval-reasoning framework. The LLM first generates relation paths as faithful plans, which subsequently guide the retrieval of valid reasoning paths from the KG.
- **PoG** (Chen et al., 2024) introduces a self-correcting adaptive planning paradigm. It decomposes questions into sub-objectives and employs a reflection mechanism to dynamically adjust reasoning paths during KG exploration.
- **EPERM** (Long et al., 2025) adopts a three-stage framework that retrieves subgraph information and explicitly weights evidence paths, prioritizing knowledge that faithfully supports the reasoning process.

4) LLM-based SP Methods These methods utilize LLMs to generate logical forms, often combined with grounding mechanisms to ensure validity.

- **KB-Binder** (Li et al., 2023) establishes a training-free few-shot framework. It generates Skeleton logical forms via LLMs and grounds them to the KB schema using BM25-based matching to ensure executability.
- **ChatKBQA** (Luo et al., 2024a) follows a generate-then-execute paradigm. A fine-tuned LLM first predicts a logical form Skeleton, which is then grounded to the KB via unsupervised entity and relation retrieval.
- **RGR-KBQA** (Feng and He, 2025) employs a multi-stage strategy combining factual knowl-

Category	Operator	Semantics
Projection	(JOIN e r)	Returns objects o where $(e, r, o) \in \mathcal{K}$.
	(JOIN r e)	Returns subjects s where $(s, r, e) \in \mathcal{K}$.
Intersection	(AND E_1 E_2)	Intersection of sets: $E_1 \cap E_2$.
Aggregation	(COUNT E)	Returns cardinality $\|E\|$.
	(ARGMAX E r)	Entities in E maximizing value of r.
	(ARGMIN E r)	Entities in E minimizing value of r.
Comparative	(GT/LT E l)	Filters E where value $> l$ or $< l$.
	(GE/LE E l)	Filters E where value $\geq l$ or $\leq l$.

Table 4: Definitions of atomic operators. E denotes a set of entities, r a relation, e an entity, and l a literal value.

edge injection before generation with a subsequent retrieval step for entity and relation refinement.

- **CompKBQA** (Tian et al., 2025) optimizes logical form generation by decomposing the problem into progressive subtasks—specifically skeleton, topic entity, and relation generation—to reduce component-wise errors.

- **Rule-KBQA** (Zhang et al., 2025) combines rule induction and rule-guided reasoning to integrate symbolic rules with large language models, enabling more controllable and reliable logical form generation for complex KBQA while reducing hallucinations.

B Logical Form Definitions

In our framework, we adopt S-expressions (Lisp-like symbolic expressions) as the intermediate logical form. This choice is driven by their hierarchical structure, which naturally aligns with the compositional nature of complex multi-hop questions and allows for deterministic compilation into standard SPARQL queries for execution against the knowledge base.

In this context, the construction of a logical form is decomposed into a sequence of atomic operations. We categorize these operators into four types: *Projection*, *Intersection*, *Aggregation*, and *Comparative*. The detailed semantics and syntax of each operator are presented in Table 4.

C Datasets and Data Construction

To comprehensively evaluate the performance of the EVER framework, we conduct experiments on two widely used KBQA benchmarks based on Freebase. WebQSP (Yih et al., 2016) consists of 4,737 questions, predominantly involving 1-hop or 2-hop relations, serving as a baseline for fundamental semantic parsing capabilities. In contrast, CWQ (Talmor and Berant, 2018) exhibits significantly higher complexity. With 34,689 questions, it necessitates multi-hop reasoning (up to 4 hops) and involves compositionality, aggregation, and comparatives, thereby imposing stringent demands on the model's long-range reasoning abilities.

Dataset	Original Split			Constructed SFT Data (Train Only)		
	Train	Dev	Test	Positive	Negative	Total
WebQSP	3,098	-	1,639	3,098	500	3,598
CWQ	27,639	3,519	3,531	27,639	4,000	31,639

Table 5: Statistics of the benchmark datasets and the constructed instruction tuning data.

Instruction Tuning Data Construction. We construct the instruction tuning data based on the official training splits of the aforementioned benchmarks. To balance training efficiency with error-correction robustness, we adopt an asymmetric construction strategy. For Positive Trajectories, we generate standard step-by-step reasoning chains using all original training samples to ensure mastery of correct foundational logic. For Negative Trajectories, considering the sparsity of errors in real-world scenarios, we restrict the proportion of samples containing self-correction processes to approximately 10%–20% of the total data. Table 5 details the statistics of the original datasets and the constructed instruction tuning data.

Data Quality and Consistency. We ensure the reliability of the GPT-4 generated annotations through deterministic parsing and manual auditing. By extracting S-expressions directly from gold-standard SPARQL queries via a bottom-up recursive algorithm, we guarantee 100% execution-level accuracy and simplify complex multi-hop logic into single-step atomic operations, which significantly reduces the risk of LLM hallucinations. Furthermore, a manual quality audit of 200 random instances revealed that only 5.5% exhibited minor noise, such as verbosity or slight semantic deviations. These instances still accurately conveyed the core reasoning goals without compromising the executability of the logical forms, demonstrating the high reliability of our synthesized data.

Method	Avg. LLM Calls	Avg. Inference Time
ToG	23.1	>60.0s
ChatKBQA*	1	∼3.4s
EVER (Ours)*	4.1	∼7.5s

Table 6: Efficiency comparison regarding LLM calls and inference time, averaged over WebQSP and CWQ datasets. Models marked with * require fine-tuning.

Hyperparameter	WebQSP	CWQ
Model & Training		
LLM Backbone	**Llama-2-7B**	**Llama-2-13B**
Fine-tuning Method	LoRA	LoRA
Train Batch Size	{1, 2, 3, **4**}	{1, **2**, 3, 4}
Learning Rate	{**5e-5**, 5e-4, 5e-3}	{**5e-5**, 5e-4, 5e-3}
Train Epochs	{10, 30, **50**, 100}	{5, **10**, 30, 50}
Inference		
Skeleton Beam Size	{1, 3, 5, **8**}	{1, 3, **5**, 8}
Max Retry Limit	{1, 2, **3**, 5}	{1, 2, **3**, 5}
Retrieval		
Retrieval Model	{**SimCSE**, Contriever, BM25}	
Entity Candidate (k_e)	50	50
Relation Candidate (k_r)	15	15

Table 7: Detailed hyperparameter settings. Values in braces denote the search space, and bold values indicate the optimal settings.

D Efficiency Analysis

We report the statistics of average LLM calls and inference latency per query, as shown in Table 6.

Although LLM-based IR methods like ToG possess the advantage of being training-free, their excessive average inference latency—caused by exhaustive exploration over entity graphs—renders them impractical for real-world deployment. In contrast, LLM-based SP methods like ChatKBQA leverage fine-tuning to generate the logical form of the question in a single pass. While this approach achieves minimal latency, the strict nature of query languages poses challenges when handling complex questions. EVER effectively balances this trade-off: by utilizing instruction tuning to constrain the search space within logical schemas and employing a dynamic correction loop, it achieves efficient and robust reasoning for complex questions with reduced LLM calls and acceptable inference latency.

E Hyperparameter Settings

In this section, we present the implementation details of the EVER framework. The detailed hyperparameter settings for both WebQSP and CWQ datasets are summarized in Table 7, including the search space and the optimal values selected.

F Prompts and Data Examples

To facilitate reproducibility and transparency, we provide the full set of prompts and a detailed training example used in our framework.

Inference Setup. Table 10 details the system prompt enforcing the Execution as Verification protocol, which defines the iterative reasoning process and the <RETRY> mechanism. The model input at step t is constructed by concatenating this prompt, the user question, and the history sequence H_{t-1}.

Data Construction. We utilize GPT-4 for dataset synthesis. Tables 11 and 12 present the prompts used to generate positive reasoning chains and negative self-repair trajectories, respectively.

Trajectory Example. Figure 4 illustrates a training instance featuring negative self-repair. The model identifies an entity linking error via grounding feedback and utilizes the <RETRY> action to correct the mismatch before proceeding.

G Benchmark Expansion and Transferability Analysis

To rigorously verify EVER's generalizability to unseen relations and diverse logical compositions, we conducted zero-shot cross-dataset evaluations. Specifically, we applied the model trained solely on the complex CWQ dataset to two benchmarks without target-domain fine-tuning: (1) WebQSP, for basic cross-dataset transferability, and (2) GrailQA (Gu et al., 2021), for zero-shot generalization to unseen schemas.

Table 8: Zero-shot transferability evaluation. The model trained exclusively on CWQ is evaluated on WebQSP and GrailQA without any target-domain fine-tuning.

Training Source	Evaluation Target	F1	Hits@1
WebQSP (In-domain)	WebQSP	81.8	86.9
CWQ (Transfer)	WebQSP	80.2	84.2
CWQ (Transfer)	GrailQA	62.2	68.3

As shown in Table 8, EVER achieves strong performance retention when transferring from CWQ to WebQSP. Furthermore, under the strict zero-shot setting on GrailQA, EVER achieves competitive results. It effectively mitigates the vocabulary gap of unseen KB schemas through our Execution as Verification paradigm. Decoupling logical skeleton

generation from KB alignment inherently prevents over-fitting, while the Self-Guided Semantic Correction mechanism leverages execution feedback to dynamically calibrate noisy retrieval candidates, ensuring reliable logic grounding.

H Performance Analysis Across Different Backbone Models

To further investigate whether stronger Large Language Models (LLMs) can inherently resolve the query executability issue in KBQA systems, we evaluated the performance of the EVER framework using Llama-3.1-8B (Team, 2024) as the backbone on the complex CWQ dataset.

Table 9: Performance comparison of EVER across different backbone models on the CWQ dataset.

Backbone Model	F1	Hits@1
EVER (Llama-2-13B)	81.2	84.9
EVER (Llama-3.1-8B)	**81.9**	**86.2**

As shown in Table 9, although Llama-3.1-8B possesses a more advanced architecture and stronger instruction-following capabilities, the improvement in F1 score is relatively limited (0.7%) compared to Llama-2-13B. This result demonstrates that simply scaling the model size or enhancing reasoning capacity does not fully eliminate the structural deviations encountered during semantic parsing into logical forms (e.g., S-expressions).

In contrast, the Fine-Grained Execution-Aware Planning in EVER provides structural fault tolerance by decomposing complex queries, which acts as a critical compensation mechanism even for moderate-sized models. This suggests that the current performance bottleneck is no longer primarily determined by the model's raw reasoning capacity, but is instead more constrained by knowledge base incompleteness and the vocabulary gap in entity alignment.

Section	Content
Role Definition	You are an expert assistant proficient in Knowledge Base Question Answering (KBQA). Your goal is to decompose complex user questions into a sequence of executable logical steps.
Guidelines	1. **Iterative Reasoning**: Do **NOT** generate the full plan at once. Only generate the Thought and Action for the **current step**. 2. **Logical Form**: Use standard logical operators: JOIN, AND, COUNT, ARGMAX, ARGMIN, GT, LT, GE, LE. 3. **Variable Usage**: If previous steps produced results, use variables (e.g., `<r1>`) directly. 4. **Structured Thinking**: Must use `[Reflection]` to verify and `[Plan]` to state intent. 5. **Grounding**: Use natural language surface names; KB IDs are prohibited.
Output Format	Select one of the following output modes based on the current state: **Mode 1 (Normal)**: Thought: `[Reflection] <Confirm expectation> [Plan] <Describe intent>` **Action**: `<S-expression>` **Mode 2 (Correction)**: Thought: `[Reflection] <Analyze why the result is wrong>` **Action**: `<RETRY>` **Mode 3 (Completion)**: Thought: `[Reflection] <Confirm final result obtained>` **Action**: `Final Answer: <Rn>`

Table 10: System prompt used during the inference phase.

Section	Content
Role Definition	You are an expert KBQA Data Annotator. Your task is to generate a structured natural language "Thought" for **each step** in the provided decomposed logic sequence, ensuring alignment with the user's question.
Input Data	**Question**: {{question}} **Logic Steps**: {{steps_str}}
Guidelines	1. **Structure Tags**: Strictly use `[Reflection]` and `[Plan]`. 2. **Step 1 Strategy**: Format: `[Plan] <Explain intent starting from question>`. (No reflection). 3. **Step 2 to N**: Format: `[Reflection] <Verify previous result> [Plan] <Explain intent>`. 4. **Variable Handling**: Use natural language (e.g., "these movies") to refer to variables like `<r1>`.
Output	Return ONLY a JSON list of objects: `["step_index": 1, "thought_content": "...", ...]`

Table 11: Prompt for generating Positive Reasoning Thoughts.

Section	Content
Role Definition	You are an expert in Entity Disambiguation and Logic Reasoning. Your task is to create a Failure & Retry scenario where the model initially selects a wrong entity/type and then corrects itself.
Input Data	**Entity Name**: {{entity_name}} **Question**: {{question}}
Guidelines	1. **Invent Distractor**: Create a distractor with the *same name* but *different type/profession* (e.g., Director vs. Physicist). 2. **Rejection Thought**: Start with `[Reflection]`. Explain the semantic mismatch based on the observation and decide to retry. 3. **Correction Thought**: Start with `[Reflection]`. Confirm the correct entity matches user intent, then append `[Plan]`.
Output	Return ONLY a JSON object: `{distractor_description, rejection_thought, correction_thought}`.

Table 12: Prompt for generating Negative Self-Repair Trajectories.

```
{
   "question": "Which movies directed by Christopher Nolan won an Oscar?",
   "conversations": [
     {
       "role": "system",
       "content": "You are an expert assistant proficient in Knowledge Base Question Answering...
(See Table 10 for full prompt)"
     },
     {
       "role": "user",
       "content": "Which movies directed by Christopher Nolan won an Oscar?"
     },
     {
       "role": "assistant",
       "content": "[Plan] First, based on the question intent, I need to find all movies directed
by Christopher Nolan.\nAction: (JOIN (R [ film.director.film ]) [ Christopher Nolan ])"
     },
     {
       "role": "observation",
       "content": "Execution Success\nExecuted Query <l1>: (JOIN (R [ film.director.film ])
[ m.03bxz ])\n[Grounding Info]\n- Entity Christopher Nolan -> m.03bxz (American theoretical
physicist)\n- Relation film.director.film -> (Film Director)\n[Execution Result <r1>]\n- []"
     },
     {
       "role": "assistant",
       "content": "[Reflection] The entity Christopher Nolan was linked to a theoretical physicist
(m.03bxz), which does not match the intent of searching for a director.\nAction: <RETRY>"
     },
     {
       "role": "observation",
       "content": "Execution Success\nExecuted Query <l1>: (JOIN (R [ film.director.film ])
[ m.01234 ])\n[Grounding Info]\n- Entity Christopher Nolan -> m.01234 (Famous film director,
screenwriter, producer)\n[Execution Result <r1>]\n- <r1> = [Inception (2010 Sci-Fi action film),
Interstellar (2014 epic Sci-Fi film), ...]"
     },
     {
       "role": "assistant",
       "content": "[Reflection] The entity was correctly linked to film director, and returned a
result consistent with the intent <r1>. [Plan] Next, I will filter <r1> for movies that won an
Oscar.\nAction: (AND <r1> (JOIN (R [ film.film.award ]) [ Oscar ]))"
     },
     {
       "role": "observation",
       "content": "Execution Status: Success\n[Execution Result]\n- <r2> = [Inception (Won 4 Oscars
including Best Cinematography), The Dark Knight (Won 2 Oscars including Best Sound Editing)]"
     },
     {
       "role": "assistant",
       "content": "[Reflection] Filtering complete. <r2> contains the final answer.\nAction: Final
Answer: [Inception, The Dark Knight]"
     }
   ]
}
```


Figure 4: An example of a training trajectory containing a Negative Self-Repair turn.