

KG-Hopper: Empowering Compact Open LLMs with Knowledge Graph Reasoning via Reinforcement Learning

Shuai Wang, Yinan Yu

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-41296 Gothenburg, Sweden
{shuaiwa, yinan}@chalmers.se

Abstract—Large Language Models (LLMs) demonstrate impressive natural language capabilities but often struggle with knowledge-intensive reasoning tasks. Knowledge Base Question Answering (KBQA), which leverages structured Knowledge Graphs (KGs) exemplifies this challenge due to the need for accurate multi-hop reasoning. Existing approaches typically perform sequential reasoning steps guided by predefined pipelines, restricting flexibility and causing error cascades due to isolated reasoning at each step. To address these limitations, we propose KG-Hopper, a novel Reinforcement Learning (RL) framework that empowers compact open LLMs with the ability to perform integrated multi-hop KG reasoning within a single inference round. Rather than reasoning step-by-step, we train a *Reasoning LLM* that embeds the entire KG traversal and decision process into a unified “thinking” stage, enabling global reasoning over cross-step dependencies and dynamic path exploration with backtracking. Experimental results on eight KG reasoning benchmarks show that KG-Hopper, based on a 7B-parameter LLM, consistently outperforms larger multi-step systems (up to 70B) and achieves competitive performance with proprietary models such as GPT-3.5-Turbo and GPT-4o-mini, while remaining compact, open, and data-efficient. The code is publicly available at: <https://github.com/Wangshuaiia/KG-Hopper>.

Index Terms—LLM, knowledge graph, question answering

I. INTRODUCTION

Large Language Models (LLMs) have achieved remarkable success across diverse domains but still exhibit notable shortcomings, such as hallucinations and factual inaccuracies, particularly in knowledge-intensive tasks. This is largely due to the implicit storage of knowledge within their model weights, making knowledge updates cumbersome and resource-intensive through fine-tuning. To address this limitation, Retrieval-Augmented Generation (RAG) has emerged as an effective strategy, enabling LLMs to dynamically access external knowledge during inference [1]. Among various external resources, Knowledge Graphs (KGs) stand out as structured and reliable knowledge bases, offering explicit, interpretable, and easily updateable knowledge beneficial in critical applications such as medicine and finance.

Knowledge Base Question Answering (KBQA), which aims to leverage structured knowledge from KGs to answer questions, often requires complex multi-hop reasoning—traversing multiple interconnected relationships within a KG [2]. Current

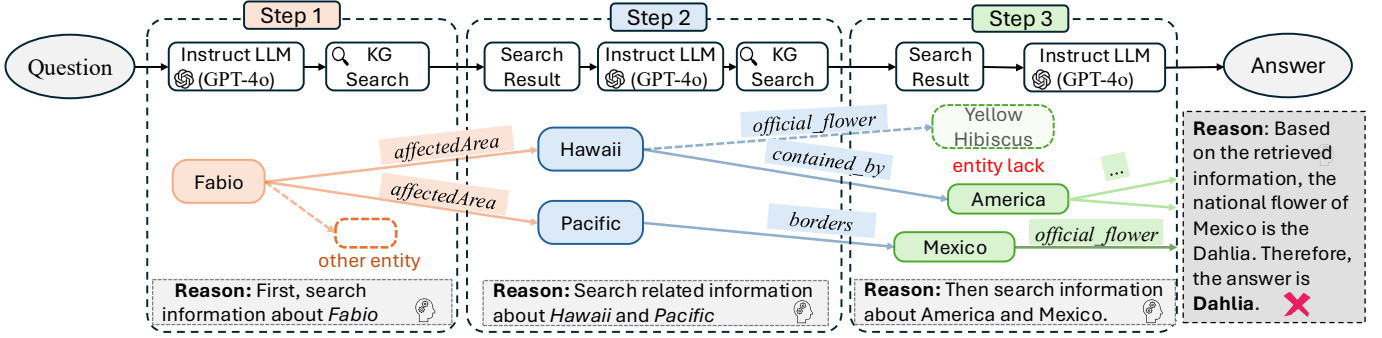
approaches typically follow step-by-step reasoning strategies, sequentially processing entities and relationships from a predefined pipeline. For example, as illustrated in Figure 1(a), answering the question “What is the official flower of the area affected by Tropical Storm Fabio?” involves first identifying the affected area and subsequently retrieving its official flower. However, such rigid frameworks present critical drawbacks:

- **Limited Flexibility and Local Optima Susceptibility.** Traditional methods sequentially aggregate information through individual inference steps guided by predefined reasoning paths. Consequently, these methods struggle to dynamically adjust when facing incomplete or misleading KG data. For instance, as shown in Figure 1(a), missing the critical entity “Yellow Hibiscus” could mislead a multi-step method employing beam search into incorrectly choosing the national flower of Mexico, with limited capability for backtracking.
- **Error Cascading and Reasoning Bias.** Stepwise methods inherently propagate errors from earlier steps. Incorrectly choosing a node, such as “Mexico” in Figure 1(a), directly influences subsequent reasoning steps. Additionally, treating each reasoning step independently neglects inter-step dependencies, causing biases and potential deviations from the original query intent.

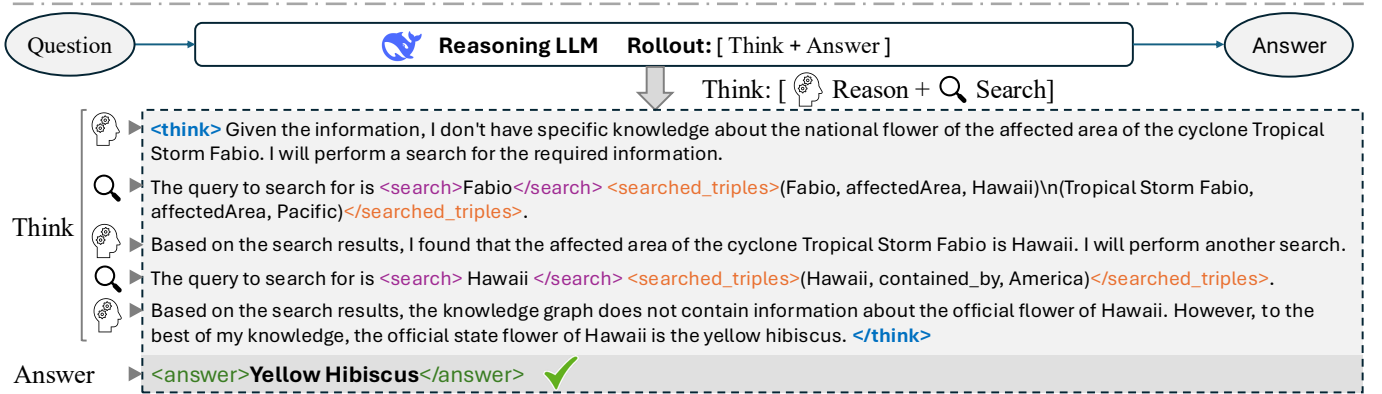
To mitigate these limitations, two lines of research offer promising advances. First, Reinforcement learning (RL) have demonstrated its effectiveness in navigating discrete and combinatorial decision spaces, such as KG traversal by optimizing reasoning policies through exploration and long-term rewards [3]–[5]. Second, recent work on *Reasoning LLMs*, such as ChatGPT-o1 [6] and DeepSeek-R1 [7], has significantly improved the reasoning capabilities of language models. These models perform a dedicated “thinking” phase before generating answers, where each token in the thought sequence can be viewed as a latent variable. Importantly, later tokens can revise earlier ones, enabling self-reflection and correction within the reasoning process [8]. This iterative refinement mechanism aligns naturally with the multi-hop nature of complex KBQA tasks, allowing LLMs to reason more coherently and correct potential errors before answer generation.

Question: What is the official flower of the affected area of the cyclone Tropical Storm Fabio?

Answer: Yellow Hibiscus



(a) Multi-step Reasoning (Multiple LLM Calls)



(b) One-round Reasoning (Single LLM Call)

Fig. 1. Multi-step vs one-round multi-hop reasoning over a knowledge graph: (a) multi-step reasoning and (b) our one-round reasoning. The multi-step pipeline invokes multiple sequential LLM calls and fails due to the missing entity *Yellow Hibiscus*, leading to an incorrect path. In contrast, our one-round approach performs the entire reasoning process within a *single Reasoning LLM* call, maintaining coherence and demonstrating robustness to incomplete knowledge.

Building on these insights, we propose **KG-Hopper**, a novel KBQA framework integrating the entire multi-hop reasoning process within a single LLM call, as depicted in Figure 1(b). Specifically, KG-Hopper leverages RL to enhance the reasoning capabilities of LLMs by training them to utilize KG retrieval tools effectively.

KG-Hopper addresses previous limitations with the following advantages: (1) By embedding the complete reasoning chain within a single inference step, KG-Hopper captures cross-step dependencies, mitigating local biases and ensuring coherent reasoning. (2) Integrating reasoning into the *Reasoning LLM*’s “thinking” phase allows flexible exploration and effective backtracking, significantly reducing cascading errors without predefined reasoning patterns. (3) Our framework leverages RL to explicitly guide and refine reasoning processes, substantially improving the model’s performance in complex multi-hop scenarios. (4) It remains compact and deployable, using a 7B LLM to achieve competitive results with much larger models.

To effectively train KG-Hopper, we initially generate cold-start data and employ Supervised Fine-Tuning (SFT), enabling autonomous KG retrieval. We subsequently strengthen multi-hop reasoning capabilities through targeted RL training, guided by carefully designed reward functions for retrieving and

reasoning process. Additionally, we mask the retrieved triples from KG to minimize external knowledge interference and employ history resampling to enhance training efficiency.

Our main contributions are as follows:

- We introduce a RL framework for LLM–KG integration, combining structured reward signals for retrieval and reasoning with masked supervision over KG content. To the best of our knowledge, this is the first work to apply RL to enable end-to-end KG reasoning within LLMs.
- We propose a one-round KBQA framework that performs multi-hop reasoning entirely within the “thinking” phase of a single LLM inference, enabling autonomous and iterative KG traversal without multi-step orchestration.
- We achieve strong empirical results on eight KBQA benchmarks: our compact 7B-parameter model consistently outperforms multi-step methods using models up to 70B, and matches or exceeds the performance of proprietary models such as GPT-3.5-Turbo and GPT-4o-mini.

II. TASK DEFINITION

We address the task of Knowledge Base Question Answering (KBQA) over a knowledge graph $G = \{(e, r, e') \mid e, e' \in \mathcal{E}, r \in \mathcal{R}\}$, where each triple (e, r, e') denotes a relation r

between two entities e and e' . Given a natural language question q , along with an identified topic entity $e_t \in \mathcal{E}$ mentioned in the question, the objective is to infer the correct answer entity $e_a \in \mathcal{E}$ by reasoning over the knowledge graph G . The answering process typically starts from the topic entity e_t and involves exploring the graph to identify the answer entity e_a . In practice, large-scale knowledge graphs are often sparse and incomplete, making it difficult to locate the answer through simple one-hop queries. As a result, multi-hop reasoning is frequently required, where the answer entity e_a may reside several hops away from the topic entity e_t .

III. METHOD

Our approach leverages reinforcement learning to train LLMs for KG retrieval and reasoning. This section introduces two key components: the construction of a knowledge graph retrieval tool, and the RL framework for training LLMs to interact with the knowledge graph.

A. Knowledge Graph Retrieval Tool

Large-scale knowledge graphs are typically stored in graph databases and queried using SPARQL for retrieval. Our retrieval process starts from a topic entity and expands the search. Due to the vast number of triples connected to a single node, it is crucial to filter out irrelevant information. To address this, we adopt a two-stage strategy: first retrieving the set of directly connected edges (predicates) and then selecting the ones relevant to the input query.

Given a topic entity, our tool first retrieves all predicates and corresponding object entities linked to it. For example, given the question “What books did J.K. Rowling write?”, the topic entity is J.K. Rowling. The following SPARQL query is issued to retrieve all outgoing predicates and their objects:

```
SELECT ?predicate ?object
WHERE {
  ns:m.05b6w ?predicate ?object .
}
```

Here, `ns:m.05b6w` denotes the Freebase ID for J.K. Rowling, `?predicate` retrieves all relations where she appears as the subject, and `?object` returns the corresponding object entities.

For example, the retrieved predicates include:

```
ns:book.author.works_written,
ns:people.person.place_of_birth,
ns:people.person.nationality.
```

Based on the semantic relevance to the question, the most related predicate, such as `ns:book.author.works_written`, is selected. This predicate indicates the books authored by the subject. Next, the tool issues another SPARQL query to retrieve the tail entities (i.e., books) associated with the selected predicate:

```
SELECT ?tailEntity
WHERE {
  ns:m.05b6w ns:book.author.works_written ?
    tailEntity .
}
```

This query fetches all book entities linked to J.K. Rowling through the `works_written` relation, effectively answering the question.

In summary, the retrieval tool takes an entity as input and returns a set of relevant triples from the knowledge graph by identifying and filtering meaningful relations and associated entities.

B. Cold Start

To avoid the unstable cold-start phase typically seen in early RL training. Specifically, we construct and collect a small set of CoT annotated data to fine-tune the base LLM, which is then used as the initial RL actor. The cold-start dataset is created with two primary objectives: (1) to demonstrate how the model should properly invoke the knowledge graph retrieval tool; and (2) to enforce a consistent, structured format for answer generation.

We define a rule such that when the LLM generates the special tokens `<search>` and `</search>`, it triggers the KG retrieval tool to search the enclosed entity. The retrieved triples are then wrapped with `<searched_triples>` tags and appended to the current context, enabling the LLM to continue generation with access to the retrieved knowledge. To collect cold start data, we use few-shot prompting with a long CoT example to elicit responses from a powerful LLM. We then select examples that correctly invoke KG queries and exhibit high readability. For instance, given the question: *what timezone is Utah in?* A preferred response would begin with an explicit motivation for search, such as:

*<think>Given the information, I don't have specific knowledge about Utah's timezone. I will perform a search for the required information. The query to search for is <search>utah</search> <searched_triples> (Utah, timeZone, Mountain Time Zone)</searched_triples>. Based on the search results, I found that Utah is in the Mountain Time Zone.</think>
<answer>Mountain Standard Time</answer>*

We select such examples, which exhibit clarity, appropriate tool invocation, and well-structured reasoning, as the cold-start training data. Such structure is the desired output format of the LLM. The `<think>` section encapsulates the full CoT reasoning process, including autonomous invocation of the KG tool (`<search>`), retrieval results (`<triples>`), and intermediate deductions. The `<answer>` section then summarizes the reasoning into a final answer. This also represents the desired output format of the LLM.

We use the collected data to fine-tune the base LLM. During training, we mask the tokens within the `<triples>` to prevent the model from being distracted by retrieved knowledge. This encourages the model to generalize its reasoning strategy while preserving tool-use behaviors. The fine-tuned model is then further optimized via reinforcement learning.

C. Reasoning-oriented Reinforcement Learning

Given the difficulty of obtaining sufficient high-quality long CoT reasoning data, we leverage Reinforcement Learning with Human Feedback (RLHF) as a principled alternative

to explicitly guide the model toward effective multi-hop reasoning. We design a composite reward function to guide the model’s behavior throughout the reasoning process. The reward comprises four components: retrieval reward, format reward, reasoning reward, and final answer reward.

- **Retrieval Reward** To encourage the model to search for answers through the knowledge graph rather than relying solely on its internal knowledge, we provide a positive reward for each invocation of the query tool. However, to prevent the model from overusing the tool solely for reward accumulation, we apply a cap on the maximum reward. The retrieval reward is defined as:

$$R_{\text{search}} = \min(0.5 \cdot n, 0.8) \quad (1)$$

where n denotes the number of times the query tool is invoked. This design incentivizes query usage while discouraging excessive or redundant retrievals.

- **Format Reward** To enforce structured reasoning and tool usage, we define a format reward that requires the generated text to follow a predefined format. Specifically, the model must use the tags `<think>`, `<search>`, and `<answer>` appropriately. If all tags appear in the correct positions and order, a fixed reward is granted:

$$R_{\text{format}} = \begin{cases} 0.5 & \text{if the format meets all requirements} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

This constraint ensures that the model explicitly separates its reasoning, search actions, and final answer.

- **Reasoning Reward** In multi-hop KG reasoning, an error at any intermediate step can lead to an incorrect final answer. To mitigate this, we introduce a reward signal that directly evaluates the quality of the reasoning process itself, encouraging the model to make sound decisions at each step, rather than focusing solely on the final answer. The model is expected to adapt its behavior based on the informativeness of retrieved triples. When the retrieved information is sufficient, the model should organize and synthesize it into a coherent answer. Otherwise, it should continue reasoning by selecting the next entity or relation to query. In cases where the KG lacks the necessary facts (e.g., due to incompleteness), the model is allowed to fall back on its internal knowledge to infer a plausible answer. To assess the quality of the model’s reasoning behavior, we use an external LLM to evaluate the full reasoning trace enclosed in the `<think>` tag. The reasoning reward is computed as:

$$R_{\text{reason}} = f_r(\text{reasoning process}) \in (0, 1) \quad (3)$$

where f_r denotes an external LLM, and a higher score indicates a more reasonable and logically valid reasoning trace.

- **Answer Reward** Finally, we reward the model if it provides a correct answer. Since the generated answer may differ from the ground truth due to variations such as abbreviations or phrasing, we again use a separate LLM

to perform semantic similarity assessment between the predicted answer and the ground truth (**LLM as Evaluator**). The final answer reward is defined as:

$$R_{\text{answer}} = f_a(\text{predicted_answer}, \text{ground_truth}) \in \{0, 1\} \quad (4)$$

where f_a denotes an external LLM, and the predicted answer is extracted from the `<answer>` tag. A reward of 1 is given if the LLM determines the answer is semantically correct, and 0 otherwise. In the absence of ground truth, we directly use a (preferably larger) LLM with rich knowledge as both the **Judge** and **Evaluator**, i.e., we rely on the LLM’s internal knowledge to assess whether the predicted answer is correct.

By combining the above four reward functions, the total reward provides comprehensive guidance to the model throughout the reasoning process, promoting structured, accurate, and knowledge-grounded answers, which can be represented as:

$$R_{\text{final}} = R_{\text{search}} + R_{\text{format}} + R_{\text{reason}} + R_{\text{answer}} \quad (5)$$

D. Optimization

We optimize the reasoning policy using Group Relative Policy Optimization (GRPO), which trains the LLM to maximize the expected reward of generated reasoning trajectories. Formally, the objective is simplified as:

$$\theta^* = \arg \max_{\theta} \mathbb{E}_{q \sim \pi_{\theta}} [R_{\text{final}}(q)], \quad (6)$$

where π_{θ} denotes the LLM policy and $R_{\text{final}}(q)$ is the corresponding reward function. GRPO estimates relative advantages within sampled output groups to stabilize optimization and encourage higher-quality reasoning paths.

Masking Retrieved Triples. Retrieved triples are provided as auxiliary context but are not expected to be generated by the model. We therefore mask tokens enclosed by `<triples>` and `</triples>` during loss computation to prevent the model from learning to reproduce retrieved content.

History Resampling for Efficient Training. In KBQA, simple one-hop queries often produce uniformly high rewards, leading to near-zero normalized advantages and inefficient learning. Following a history resampling strategy [9], we remove one-hop questions after an initial training phase, encouraging the model to focus on multi-hop reasoning in a curriculum learning manner [10].

IV. EXPERIMENTS

A. Datasets

We evaluate our approach on eight widely-used datasets for KBQA, leveraging two large-scale general-purpose knowledge graphs: Freebase and WikiData. Four of the datasets are based on the Freebase knowledge graph and are standard KBQA benchmarks: ComplexWebQuestions (CWQ) [11], WebQuestionsSP (WebQSP) [12], WebQuestions [13], and GrailQA [14]. The other four datasets are grounded in the WikiData knowledge graph. Among them, QALD10-en [15] is a KBQA dataset, while T-REx [16] and Zero-Shot RE [17]

are designed for slot filling tasks, and Creak [18] focuses on factual verification. Following prior work [19]–[21], we use Hit@1 score as the evaluation metric for both question answering and slot filling tasks, while accuracy is used to evaluate performance on the Creak dataset.

B. Implementation Details

We conduct experiments using two instruction-tuned language models as backbones: LLaMA-3.1-8B-Instruct and Qwen-2.5-7B. To mitigate the unstable cold-start phase of reinforcement learning, we first construct a set of 500 high-quality examples to teach the models how to properly invoke the knowledge retrieval tool. We randomly sample 2,000 examples from 8 datasets for RL training. The scoring model f_r for the reasoning process is Llama-3.3-70B, while the model f_a used to determine whether the predicted answer matches the ground truth is Llama-3.2-3B. Starting from the second epoch, we apply a resampling strategy to dynamically filter out trivial questions and retain more informative ones for continued training. The training is performed on 8 NVIDIA A100 80G GPUs in total. For each input query, we generate 16 outputs (rollouts). We train for 2 epochs with a batch size of 16 and a learning rate of $1e-6$. The rollout temperature is set to 1, the PPO clip ratio is 0.2, and the KL divergence penalty coefficient is $1e-5$.

C. Main Results

We design our experiments to explore the reasoning effectiveness, training efficiency, and design trade-offs of our method KG-Hopper. To contextualize our approach within the broader KBQA landscape, Table I summarizes different LLM+KG integration strategies. It highlights variations across model integration methods, inference modes, post-processing techniques, and model accessibility. KG-Hopper is designed as an open-source, compact, and flexible solution that promotes transparency in the training process and supports targeted modifications. Through our experiments, we investigate how RL contributes to efficient multi-hop KBQA. Results are reported in Table II.

In terms of integration strategy, prompt-only LLMs, including strong proprietary models like GPT-4o, consistently underperform on multi-hop KBQA tasks. This highlights the limitations of relying solely on parametric knowledge and the lack of explicit, structured reasoning. Adding KG retrieval tools improves performance significantly (often by 10–30 Hits@1), but without any form of model-level adaptation, these systems exhibit fixed reasoning patterns and struggle to generalize beyond shallow queries. SFT on KG reasoning tasks further improves results, especially for moderately complex questions. However, its imitation-based learning process tends to be brittle, it teaches the model to reproduce specific reasoning paths rather than adaptively exploring alternatives. Our RL-trained model, KG-Hopper, consistently outperforms SFT-based models of similar (7-8B) or larger size (13B, 70B) and matches or exceeds the performance of GPT-4o-mini + KG, particularly on more complex multi-hop reasoning tasks.

D. Ablation

a) *RL vs SFT*: How does reinforcement learning compare to SFT in the context of multi-hop KBQA performance? (**RQ1**) Table III reports the Hits@1 scores across eight KBQA datasets, comparing models trained with SFT and RL using two LLM backbones: Qwen-2.5-7B and LLaMA-3.1-8B. In both cases, the RL-trained models consistently outperform their SFT counterparts, achieving gains of +4% to +10%. This performance gap can be attributed to the inherent misalignment issue [22] in SFT, where the model learns to produce fixed, pattern-based responses. Such rigidity often leads the model to rely on spurious correlations from training examples, which can interfere with its ability to reason based on its own knowledge. In contrast, RL allows the model to adaptively coordinate with the retrieval module, perform more coherent multi-hop reasoning, and explicitly generate step-by-step solutions.

Notably, the performance gains are especially significant on complex datasets such as CWQ and QALD10-en, with improvements of around 10%. These datasets demand longer reasoning chains and deeper traversal of the KG. The results illustrate the strength of RL in capturing long-horizon dependencies, an area where supervised fine-tuning falls short due to its imitation-based nature and limited exposure to diverse reasoning patterns. In addition to accuracy, RL achieves better performance with fewer annotated examples, since it learns from scalar rewards rather than full reasoning traces. This improves data efficiency and generalization to unseen multi-hop patterns.

b) *RL Reward Design*: We analyze the impact of individual reward components in our RL framework for KBQA. Specifically, we address the following questions:

Which components (retrieval, format, reasoning, or answer) contribute most to multi-hop KBQA performance? In particular, how do token-level action rewards (i.e. retrieval reward and format reward) compare to global sequence-level reward (i.e. reasoning and result reward)? (**RQ2**) As shown in Table IV, removing the reasoning reward leads to the largest performance drop across most datasets, indicating that sequence-level feedback is essential for acquiring robust, long-horizon reasoning capabilities. In contrast, the retrieval and format rewards yield smaller but consistent improvements, helping to regularize tool use and output structure. These two rewards operate at the token level, primarily guiding the generation of literal tags (e.g., <search>, <answer>) to enforce structured output and tool invocation. As such, these behaviors are effectively learned through SFT – RL does not offer a clear efficiency advantage in this context. While token-level rewards contribute to training stability and early convergence, their gains are marginal compared to the global benefits provided by reasoning-level supervision.

For the final answer reward, can LLMs reliably serve as automated judges for final answers in RL? (**RQ3**) Specifically, we compare two modes: (i) **Evaluator-only**, where the LLM assesses the answer against a known ground truth, and (ii) **Judge+Evaluator**, where no ground truth is provided and the LLM both infers a reference answer and evaluates the

TABLE I
LLM+KG INTEGRATION TECHNIQUES. WE HIGHLIGHT OUR FOCUS IN GREEN.

Method	Short Description	Pros	Cons
Model-KG Integration Strategies			
Prompt-only LLM	LLM answers with parametric knowledge only	Easy to use; no dependencies	Poor on multi-hop or unseen queries
LLM + KG (no training)	Uses KG retrieval tools, no adaptation	Quick to integrate KG; flexible	No reasoning adaptation; tool orchestration needed
LLM + KG (SFT)	LLM fine-tuned with KG reasoning traces	More accurate; learns tool use	Requires labeled CoT data; brittle over long chains
LLM + KG (RL)	RL-trained LLM with KG traversal tools	Strong generalization; fewer examples needed	Requires reward design
Inference and Reasoning Modes			
Direct prompting	Direct answer generation in one pass	Fastest inference; minimal cost	Cannot model intermediate reasoning
Multi-step reasoning	Sequential steps with intermediate tool calls	Fine-grained control; interpretable traces	High latency; error-prone; complex logic needed
One-round reasoning	Full reasoning in one pass with embedded retrieval	Enables globally coherent reasoning; low latency	Requires strong reasoning capability; requires model adaptation
Post-hoc Ranking and Aggregation			
Node ranking	Rank multiple answer node candidates	Boosts precision when candidates are relevant	No explicit reasoning path; requires candidate set
Path ranking	Rank and select from reasoning chains	More robust to noise or ambiguity	High compute cost; poor scalability at runtime
LLM Source and Accessibility			
Open LLM	Models such as LLaMA, Qwen, DeepSeek	Transparent; reproducible; modifiable; cheaper to deploy	May underperform on some benchmarks without adaptation
Proprietary LLM	Closed-source APIs like GPT-4o	Often stronger zero-shot reasoning	Expensive; limited control; no model access; data privacy issues

TABLE II
PERFORMANCE COMPARISON (HITS@1). RESULTS MARKED WITH ‘*’ ARE TAKEN DIRECTLY FROM THE CORRESPONDING ORIGINAL PAPERS. BOLD NUMBERS INDICATE THE **BEST PERFORMANCE**, WHILE UNDERLINED NUMBERS DENOTE THE SECOND-BEST.

Method	Size	Freebase				WikiData			
		CWQ	WebQSP	WebQuestion	GrailQA	QALD10-en	T-REx	Zero-Shot RE	Creak
LLM Prompting Only									
Qwen-2.5-7B	7B	31.25	46.97	44.23	29.53	41.88	31.15	7.84	73.26
LLaMA-3.1-8B	8B	32.33	45.07	45.88	28.35	40.25	23.00	12.54	75.80
LLaMA-3.3-70B	70B	37.20	71.12	59.73	33.79	56.00	20.12	18.55	83.72
DeepSeek-R1-Distill-Llama-70B	70B	31.92	77.43	68.84	31.70	43.10	34.21	22.27	79.10
GPT-4o-mini	–	42.32	65.97	57.26	36.22	51.98	26.90	18.85	83.72
GPT-4o	–	41.77	72.55	64.79	35.01	56.20	44.46	48.20	90.70
KG-Augmented LLMs without Fine-Tuning									
Qwen-2.5-7B + KG	7B	44.82	72.60	56.33	41.10	56.54	64.21	70.32	79.04
LLaMA-3.1-8B + KG	8B	45.64	71.32	57.05	40.40	55.73	65.80	68.66	80.50
LLaMA-3.3-70B + KG	70B	44.00	81.90	72.60	57.70	71.78	65.42	80.42	87.04
DeepSeek-R1-Distill-Llama-70B + KG	70B	52.38	81.34	<u>75.80</u>	<u>59.02</u>	66.10	72.04	78.04	<u>91.20</u>
GPT-4o-mini + KG	–	54.35	84.40	81.02	60.00	<u>72.86</u>	69.70	75.12	<u>90.20</u>
KG-CoT w/GPT 3.5-Turbo [20]	–	51.6*	82.1*	66.5*	-	-	-	-	-
KG-Augmented LLMs with Supervised Fine-Tuning									
Interactive-KBQA w/LLaMA-7B [21]	7B	39.9*	43.57*	-	-	-	-	-	-
KG-CoT w/LLaMA-7B [20]	7B	46.7*	72.4*	-	-	-	-	-	-
Qwen-2.5-7B (SFT) + KG	7B	51.84	74.80	61.42	46.18	65.18	68.77	71.46	83.43
LLaMA-3.1-8B (SFT) + KG	8B	47.40	72.98	60.00	47.70	59.63	70.23	64.08	84.47
Interactive-KBQA w/LLaMA-13B [21]	13B	42.5*	54.86*	-	-	-	-	-	-
KG-CoT w/LLaMA-13B [20]	13B	50.0*	74.6*	-	-	-	-	-	-
KG-Augmented LLMs with RL Fine-Tuning									
KG-Hopper w/Qwen-2.5-7B (ours)	7B	61.07	<u>83.20</u>	66.90	50.10	74.28	72.14	<u>78.64</u>	91.82

model’s output accordingly. Using GPT-4o in Judge+Evaluator mode results in a noticeable performance drop, though it still performs better than removing answer reward entirely. This suggests LLMs can act as fallback evaluators when ground truth is unavailable, but are less reliable than direct supervision.

c) *RL Sampling Efficiency.*: How effective is history resampling in mitigating overfitting to trivial or overrepresented examples during RL training? (RQ4) Figure 2 presents training trajectories for three key metrics: average reward, response length, and retrieval count [23], comparing models trained with and without history resampling. All metrics show a general upward trend over time, indicating learning progress. When history resampling is introduced at the start of the second epoch, a temporary dip in average reward is observed, reflecting the removal of simpler, one-hop questions in favor of more

challenging multi-hop queries. As training proceeds, the model adapts to these harder examples, resulting in a steady recovery and continued reward improvement.

Importantly, models trained with history resampling exhibit longer response lengths and higher retrieval counts, both indicative of more complex, multi-step reasoning. These results confirm that resampling effectively shifts training focus toward higher-quality samples, improving the model’s capacity for reasoning without being biased toward shallow patterns.

V. RELATED WORK

KBQA methods are broadly categorized as retrieval-based or semantic parsing-based. The former extract relevant KG subgraphs, while the latter translate questions into executable queries [24], [25]. Reinforcement learning (RL) has been used

TABLE III
COMPARISON OF RL AND SFT. (RQ1)

Method	Freebase				WikiData			
	CWQ	WebQSP	WebQuestion	GrailQA	QALD10-en	T-REx	Zero-Shot RE	Creak
Qwen-2.5-7B (SFT) + KG	51.84	74.80	61.42	46.18	65.18	68.77	71.46	83.43
Qwen-2.5-7B (RL) + KG	61.07 (+9.23)	83.20 (+8.40)	66.90 (+5.48)	50.10 (+3.92)	74.28 (+9.10)	72.14 (+3.37)	78.64 (+7.18)	91.82 (+8.39)
LLaMA-3.1-8B (SFT) + KG	47.40	72.98	60.00	47.70	59.63	70.23	64.08	84.47
LLaMA-3.1-8B (RL) + KG	58.20 (+10.80)	76.90 (+3.92)	67.28 (+7.28)	55.41 (+7.71)	67.66 (+8.03)	74.20 (+3.97)	70.25 (+6.17)	88.31 (+3.84)

TABLE IV
ABLATION STUDY OF THE RL TRAINING PROCESS. THE TABLE REPORTS PERFORMANCE ACROSS DIFFERENT RL VARIANTS COMPARED TO THE BASELINE QWEN-2.5-7B (RL) + KG MODEL. REPORTED (+/-) VALUES INDICATE THE CHANGE IN HITS@1 RELATIVE TO THE BASELINE.

Method	Freebase				WikiData			
	CWQ	WebQSP	WebQuestion	GrailQA	QALD10-en	T-REx	Zero-Shot RE	Creak
Qwen-2.5-7B (RL) + KG	61.07	83.20	66.90	50.10	74.28	72.14	78.64	91.82
reward signal ablation (RQ2)								
w/o Retrieval Reward	60.75 (-0.32)	82.38 (-0.82)	66.01 (-0.89)	49.71 (-0.39)	73.54 (-0.74)	72.81 (+0.67)	77.81 (-0.83)	91.62 (-0.20)
w/o Format Reward	60.45 (-0.62)	83.63 (+0.43)	66.19 (-0.71)	49.97 (-0.13)	72.77 (-1.51)	72.10 (-0.04)	78.10 (-0.54)	91.30 (-0.52)
w/o Reasoning Reward	57.94 (-3.13)	80.50 (-2.70)	64.42 (-2.48)	49.92 (-0.18)	71.69 (-2.59)	70.22 (-1.92)	75.75 (-2.89)	91.72 (-0.10)
w/o Answer Reward	51.96 (-9.11)	75.68 (-7.52)	57.15 (-9.75)	42.19 (-7.91)	64.81 (-9.47)	68.87 (-3.27)	72.31 (-6.33)	82.06 (-9.76)
without ground truth – LLM being Judge+Evaluator (RQ3)								
Evaluate Reward by GPT-4o	53.49 (-7.58)	77.07 (-6.13)	57.46 (-9.44)	47.46 (-2.64)	70.53 (-3.75)	69.31 (-2.83)	73.82 (-4.82)	89.23 (-2.59)
without history resampling (RQ4)								
w/o History Resampling	57.83 (-3.24)	80.64 (-2.56)	67.19 (+0.29)	47.50 (-2.60)	72.16 (-2.12)	70.15 (-1.99)	77.39 (-1.25)	91.13 (-0.69)

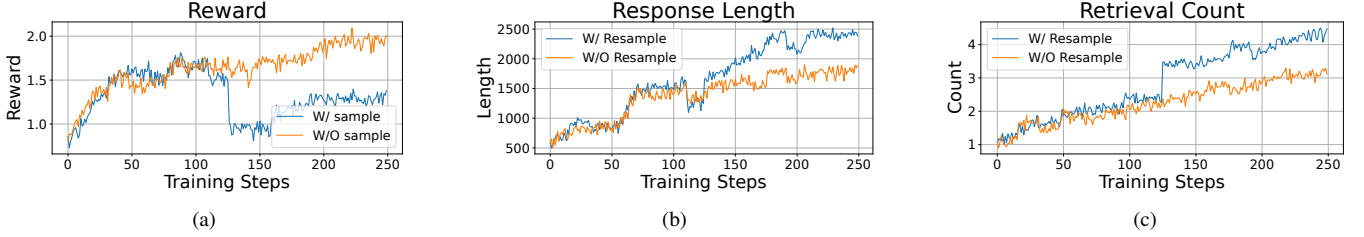


Fig. 2. The RL training process under two settings: with and without history resampling (RQ4). The figure shows how reward, response length, and retrieval count change over training steps.

to explore multi-hop paths [3], [26], though prior methods often struggle with efficiency and error accumulation. To address this, recent works incorporate LLMs for sub-question decomposition or as priors for guiding RL [20], [27].

Recent progress demonstrates that explicitly prompting LLMs to perform thinking before give an answer significantly enhances their problem-solving capabilities [28], [29]. Reinforcement Learning has further strengthened these abilities by enabling LLMs to iteratively reflect and reason effectively [8], [22]. While RL-augmented LLMs succeed in unstructured multi-hop retrieval [30], KG reasoning remains more constrained, requiring alignment to discrete graph paths. This limits the applicability of open-ended reasoning and calls for structural-aware LLM integration.

VI. CONCLUSION

In this paper, we introduced KG-Hopper, a novel RL-based framework designed to empower compact open LLMs with enhanced multi-hop reasoning capabilities over KGs. To the best of our knowledge, KG-Hopper is the first framework to apply reinforcement learning to enhance multi-hop knowledge graph

reasoning in large language models. Beyond this, KG-Hopper integrates KG reasoning directly into the intrinsic "thinking" process of the reasoning LLM, enabling autonomous multi-hop traversal and answer generation within a single inference round. Specifically, we propose a two-stage training paradigm: we begin with cold-start data to teach basic KG retrieval skills, and then apply reinforcement learning to substantially improve multi-hop reasoning capability. To support this, we design tailored reward functions that explicitly guide retrieval and reasoning behaviors, introduce masking techniques to control exposure to retrieved knowledge, and implement a history resampling strategy to improve training efficiency. Experimental results on eight benchmark datasets, along with comprehensive ablation studies, demonstrate the effectiveness of KG-Hopper in enabling efficient and robust KBQA.

ACKNOWLEDGMENT

This work was partially funded by the Autonomous Systems and Software Program (WASP), supported by the Knut and Alice Wallenberg Foundation, and the Chalmers Artificial Intelligence Research Centre (CHAIR).

REFERENCES

- [1] X. Wang, M. Costa, J. Kovaceva, S. Wang, and F. C. Pereira, “Plugging schema graph into multi-table qa: A human-guided framework for reducing llm reliance,” *arXiv preprint arXiv:2506.04427*, 2025.
- [2] S. Wang and Y. Yu, “iQUEST: An iterative question-guided framework for knowledge base question answering,” in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Jul. 2025, pp. 15 616–15 628. [Online]. Available: <https://aclanthology.org/2025.acl-long.760/>
- [3] W. Xiong, T. Hoang, and W. Y. Wang, “DeepPath: A reinforcement learning method for knowledge graph reasoning,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 564–573.
- [4] S. Wang, D. Parthasarathy, R. Feldt, and Y. Yu, “Domagent: Leveraging knowledge graphs and case-based reasoning for domain-specific code generation,” *arXiv preprint arXiv:2603.21430*, 2026.
- [5] X. V. Lin, R. Socher, and C. Xiong, “Multi-hop knowledge graph reasoning with reward shaping,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3243–3253.
- [6] A. Jaech, A. Kalai, A. Lerer, A. Richardson, A. El-Kishky, A. Low, A. Helyar, A. Madry, A. Beutel, A. Carney *et al.*, “Openai o1 system card,” *arXiv preprint arXiv:2412.16720*, 2024.
- [7] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, “Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning,” *arXiv preprint arXiv:2501.12948*, 2025.
- [8] F. Zhu, P. Wang, and Z. Sui, “Chain-of-thought tokens are computer program variables,” *arXiv preprint arXiv:2505.04955*, 2025.
- [9] X. Zhang, J. Wang, Z. Cheng, W. Zhuang, Z. Lin, M. Zhang, S. Wang, Y. Cui, C. Wang, J. Peng *et al.*, “Srpo: A cross-domain implementation of large-scale reinforcement learning on llm,” *arXiv preprint arXiv:2504.14286*, 2025.
- [10] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, “Curriculum learning for reinforcement learning domains: A framework and survey,” *Journal of Machine Learning Research*, vol. 21, no. 181, pp. 1–50, 2020.
- [11] A. Talmor and J. Berant, “The web as a knowledge-base for answering complex questions,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 641–651.
- [12] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh, “The value of semantic parse labeling for knowledge base question answering,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2016, pp. 201–206.
- [13] J. Berant, A. Chou, R. Frostig, and P. Liang, “Semantic parsing on freebase from question-answer pairs,” in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1533–1544.
- [14] Y. Gu, S. Kase, M. Vanni, B. Sadler, P. Liang, X. Yan, and Y. Su, “Beyond iid: three levels of generalization for question answering on knowledge bases,” in *Proceedings of the Web Conference 2021*, 2021, pp. 3477–3488.
- [15] A. Perevalov, D. Diefenbach, R. Usbeck, and A. Both, “Qald-9-plus: A multilingual dataset for question answering over dbpedia and wikidata translated by native speakers,” in *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*. IEEE, 2022, pp. 229–234.
- [16] H. El Sahar, P. Vougiouklis, A. Remaci, C. Gravier, J. Hare, F. Laforest, and E. Simperl, “T-rex: A large scale alignment of natural language with knowledge base triples,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.
- [17] F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. De Cao, J. Thorne, Y. Jernite, V. Karpukhin, J. Maillard *et al.*, “Kilt: a benchmark for knowledge intensive language tasks,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 2523–2544.
- [18] Y. Onoe, M. J. Zhang, E. Choi, and G. Durrett, “Creak: A dataset for commonsense reasoning over entity knowledge,” in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- [19] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, L. Ni, H.-Y. Shum, and J. Guo, “Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph,” in *The Twelfth International Conference on Learning Representations*, 2024.
- [20] R. Zhao, F. Zhao, L. Wang, X. Wang, and G. Xu, “KG-CoT: Chain-of-thought prompting of large language models over knowledge graphs for knowledge-aware question answering,” in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)*. International Joint Conferences on Artificial Intelligence, 2024, pp. 6642–6650.
- [21] G. Xiong, J. Bao, and W. Zhao, “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)*, L.-W. Ku, A. Martins, and V. Srikumar, Eds., Aug. 2024, pp. 10 561–10 582.
- [22] P. Wang, L. Li, Z. Shao, R. Xu, D. Dai, Y. Li, D. Chen, Y. Wu, and Z. Sui, “Math-shepherd: Verify and reinforce llms step-by-step without human annotations,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 9426–9439.
- [23] H. Song, J. Jiang, Y. Min, J. Chen, Z. Chen, W. X. Zhao, L. Fang, and J.-R. Wen, “R1-searcher: Incentivizing the search capability in llms via reinforcement learning,” *arXiv preprint arXiv:2503.05592*, 2025.
- [24] H. Zhang, J. Cai, J. Xu, and J. Wang, “Complex question decomposition for semantic parsing,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4477–4486.
- [25] Y. Gu and Y. Su, “ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering,” in *Proceedings of the 29th International Conference on Computational Linguistics*, N. Calzolari, C.-R. Huang, H. Kim, J. Pustejovsky, L. Wanner, K.-S. Choi, P.-M. Ryu, H.-H. Chen, L. Donatelli, H. Ji, S. Kurohashi, P. Paggio, N. Xue, S. Kim, Y. Hahm, Z. He, T. K. Lee, E. Santus, F. Bond, and S.-H. Na, Eds., 2022, pp. 1718–1731.
- [26] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, “Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning,” in *International Conference on Learning Representations*, 2018.
- [27] Z. Zhang and W. Zhao, “A collaborative reasoning framework powered by reinforcement learning and large language models for complex questions answering over knowledge graph,” in *Proceedings of the 31st International Conference on Computational Linguistics*, 2025, pp. 10 672–10 684.
- [28] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.
- [29] Z.-Z. Li, D. Zhang, M.-L. Zhang, J. Zhang, Z. Liu, Y. Yao, H. Xu, J. Zheng, P.-J. Wang, X. Chen *et al.*, “From system 1 to system 2: A survey of reasoning large language models,” *arXiv preprint arXiv:2502.17419*, 2025.
- [30] X. Li, J. Jin, G. Dong, H. Qian, Y. Zhu, Y. Wu, J.-R. Wen, and Z. Dou, “Webthinker: Empowering large reasoning models with deep research capability,” *arXiv preprint arXiv:2504.21776*, 2025.