KARL: Reinforcement Learning for LLM Agents on Multi-Turn Knowledge-Intensive Agentic Tasks

Xueqiao Sun[1*], **Xiao Liu**[12*], **Bowen Lv**[1†*], **Hanchen Zhang**[1†], **Bohao Jing**[2†], **Zehan Qi**[1†], **Yifan Xu**[1†], **Yuxiao Dong**[13], **Jie Tang**[13]

[1]Tsinghua University [2]Z. AI [3]Beijing National Research Center for Information Science and Technology

Abstract

Large Language Models have shown remarkable potential as autonomous agents, but their effectiveness in knowledge-intensive tasks remains limited by passive knowledge utilization. We introduce KARL (Knowledge-Augmented Reinforcement Learning), a framework that enables LLM agents to dynamically explore structured knowledge sources through multi-turn interactions. Unlike existing retrieval-augmented approaches, KARL empowers agents to proactively decide when and what knowledge to acquire during task execution. Our framework incorporates online reinforcement learning with curiosity-driven reward shaping, explicitly incentivizing knowledge exploration while optimizing tool-use behaviors end-to-end. Extensive evaluation across six structured knowledge benchmarks demonstrates that KARL achieves state-of-the-art performance, with our Qwen2.5-14B-based agent significantly outperforming GPT-4o, Claude-4, and o4-mini on both knowledge graph and database tasks. Source code is available at https://github.com/THUDM/KARL.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities as autonomous agents, capable of reasoning, planning, and executing complex tasks through iterative tool use and multi-turn interactions (Yao et al., 2023b; Liu et al., 2023; Schick et al., 2023). These agentic frameworks have shown promising results across diverse domains, from web navigation (Zhou et al., 2024) and database querying (Lei et al., 2025) to knowledge graph reasoning (Gu et al., 2024) and code generation (Jimenez et al., 2024). Recent advances in reinforcement learning (RL) have further enhanced agent performance (Schulman et al., 2017; DeepSeek-AI et al., 2025), enabling more effective policy optimization in multi-step decision-making scenarios. Meanwhile, retrieval-augmented generation (RAG) techniques have emerged as a paradigm for incorporating external knowledge into LLM reasoning process, particularly for knowledge-intensive question answering (QA) tasks (Huang et al., 2025a; Tan et al., 2026).

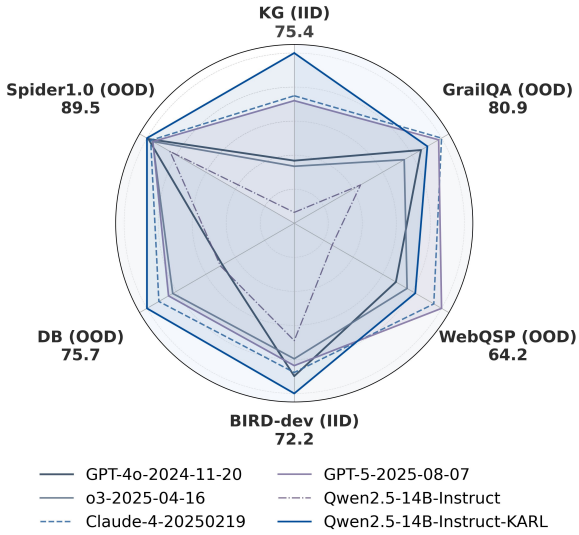


Figure 1: Scores across 6 knowledge-intensive agentic datasets based on Knowledge Graph and DataBase. KARL with 14B open models can surpass proprietary LLMs such as GPT-5 and Claude-4 in in-distribution datasets, and achieve strong generalization results on out-of-distribution datasets.

Despite these advances, training LLM agents for knowledge-intensive agentic tasks remains challenging due to several critical limitations:

- **Insufficient Knowledge–Decision Integration.** Existing work on RAG-based approaches and RL-enhanced methods either retrieve knowledge in a plug-and-play manner or focus on single-turn optimization focus in QA scenarios targeting retrieval of relevant documants and answer generation. However, real-world knowledge-intensive agentic tasks often require extended

[*]Xueqiao, Xiao, and Bowen contributed equally. Emails: snow10072740@gmail.com, shawliu9@gmail.com, lvbowen1228@gmail.com

[†]Work was partially done when interned at Z. AI.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 47539–47558

multi-turn tool use, iterative reasoning, and dynamic decision-making over long horizons beyond simply answering questions. Therefore, how to incentivize the agents to proactively explore knowledge during multi-turn interactions and efficiently utilize knowledge in its decision-making process remains underexplored.

- **Instability of Multi-Turn RL Systems** Moving from single-turn QA or supervised retrieval to interactive, knowledge-intensive RL requires agents to interact with heterogeneous environments, perform multi-step planning under real-time feedback, and maintain long-horizon dependencies during rollouts. However, existing training frameworks lack applicability to such complex multi-turn settings in several key aspects: system-level inefficiency due to synchronous and tightly coupled generation–training pipelines; training instability caused by delayed interactions and sparse reward signals; and limited scalability and extensibility hindering large-scale parallel rollouts across diverse environments.
- **Lack of exploration incentives.** In knowledge-intensive environments with structured knowledge sources (e.g., knowledge graphs, databases), agents must learn not only to complete tasks but also to actively explore and leverage external knowledge. Existing reward designs based solely on task success and formatting (DeepSeek-AI et al., 2025) provide insufficient supervision and fail to encourage systematic knowledge exploration, leading to suboptimal learning dynamics.

Motivation. To address these challenges, we propose KARL (Knowledge-Augmented Reinforcement Learning), a unified end-to-end RL framework that empowers LLM agents to reason, plan, and act through multi-turn tool use while dynamically exploring structured knowledge sources. Unlike previous RAG-based methods where knowledge retrieval is predetermined, KARL enables agents to proactively decide when and how to query external knowledge during task execution, allowing the knowledge buffer to expand dynamically as the agent progresses through multi-turn interactions. Notably, our framework implements an online, asynchronous RL training paradigm that optimizes both tool-use behavior and knowledge exploration strategies in an end-to-end fashion.

For the challenge of insufficient rewards in exploring and utilizing knowledge-intensive environments, we introduce a curiosity-driven reward mechanism that explicitly incentivizes knowledge exploration. The exploration bonus is integrated into trajectory-level rewards, encouraging agents to seek novel and relevant knowledge while maintaining focus on task objectives. Through this design, agents learn to balance between exploitation (using known knowledge to complete tasks) and exploration (discovering new knowledge that expands their reasoning capabilities).

Contributions. The main contributions of this work can be summarized as follows:

- We present KARL, an online multi-turn reinforcement learning framework specifically designed for training LLM agents on knowledge-intensive tasks, enabling stable and scalable long-horizon interaction, as well as dynamic knowledge exploration and utilization throughout extended task execution.
- We propose an end-to-end training paradigm that jointly optimizes knowledge acquisition and tool-use policies, allowing agents to proactively control when and what structured knowledge to explore during decision-making process, rather than relying on passive retrieval mechanisms.
- We introduce curiosity-driven reward shaping to explicitly incentivize knowledge exploration in structured knowledge environments, addressing the challenge of unstable RL training, sparse supervision and improving learning efficiency in knowledge-intensive scenarios.

Extensive experiments across six structured knowledge benchmarks, including both in-distribution and out-of-distribution datasets based on Knowledge Graph (KG) and DataBase (DB)(Gu et al., 2021; Yih et al., 2016; Yu et al., 2019; Li et al., 2023; Lei et al., 2025; Liu et al., 2023), demonstrate that KARL achieves state-of-the-art performance on in-distribution datasets and strong generalization results on out-of-distribution datasets, significantly outperforming both open-source baselines and strong proprietary models including GPT-4o, Claude-4, and o4-mini on tasks requiring complex reasoning over knowledge graphs and databases, as is shown in Figure 1.

2 Related Work

LLM Agents with Tool Use

Large language models increasingly serve as agentic frameworks, excelling in complex multi-turn interactions and external tool orchestra-

tion (Huang et al., 2025b; Li et al., 2025; Wang et al., 2025). Foundational works such as ReAct (Yao et al., 2023b) and Toolformer (Schick et al., 2023) demonstrated the ability of LLMs to perform reasoning and multi-tool coordination. More recent efforts, including ToolLLM (Qin et al., 2023) and RL-based agent optimization methods (DeepSeek-AI et al., 2025; Heng et al., 2025; Yao et al., 2023a; Gao et al., 2023; Qi et al., 2024; Schulman et al., 2017; Shao et al., 2024; Qian et al., 2025), show that RL effectively improves tool calling and decision-making in agentic settings.

However, in knowledge-intensive agent tasks, where agents must learn not only to complete tasks but also to actively explore and leverage external knowledge, it remains underexplored how to guide the model to effectively expand its capability boundary and perform efficient and proactive knowledge exploration. Our framework directly targets this scenario, and proposes a unified RL paradigm that jointly optimizes these behaviors.

Retrieval-based RL

Retrieval-based reasoning has been extensively explored for QA settings, where external knowledge supports the reasoning process. Methods such as RAG-RL (Huang et al., 2025a), RAG-R1 (Tan et al., 2026), R1-Searcher (Song et al., 2025), Search-R1 (Jin et al., 2025), and ReSearch (Chen et al., 2025) go a step further by optimizing search or retrieval behaviors via RL to improve answer accuracy and reasoning reliability.

These methods, however, remain fundamentally tied to single-turn or multi-hop QA tasks, where retrieval is regulated by outcome-guided signals and used solely to optimize the final answer.

In contrast, our KARL framework extends this paradigm to real-world, knowledge-intensive agentic tasks, which takes a step beyond and demands extended multi-turn tool use, iterative reasoning, as well as dynamic decision-making over long horizons. Our KARL framework proposes both efficient system-level framework design and curiosity-based optimization incentives, which encourage the agent to perform systematic exploration of structured knowledge sources. Overall, KARL efficiently guides the agent to proactively interact with the environment and jointly optimize knowledge exploration, reasoning, as well as decision-making across long multi-turn horizons.

3 Methodology

We present **KARL** — a knowledge-augmented reinforcement learning framework that empowers LLM agents to reason, plan, and act through multi-turn tool use and dynamic knowledge exploration in structured knowledge environments. The overview of our framework is shown in Figure 2.

3.1 Online End-to-End Asynchronous RL Framework

Agent tasks emphasize manipulation and decision-making. We target at structured knowledge tasks and adopt an **end-to-end asynchronous reinforcement learning framework** for agent training. Formally, we consider a sequential decision-making process in a deterministic, tool-augmented environment $\mathcal{E}$ defined by a goal G and an initial state s_0. A toolset $\mathcal{T} = \{fc_1, \ldots, fc_k\}$ is specified for each environment. At each timestep t, the agent executes an action $a_t = (r_t, fc_t)$, and environment consequently produces observation, reward signal and done flag:

$$o_{t+1}, r_{t+1}, d_{t+1} = \mathcal{E}(s_0, a_1, \ldots, a_t, G). \quad (1)$$

Each time, the agent decides on its next step given the action history, choosing which tool to call:

$$\pi_\theta(a_t \mid h_t, \mathcal{T}) = \prod_{i=1}^{n_t} p_\theta(a_{t,i} \mid a_{t,<i}, h_t, \mathcal{T}),$$
$$h_t = (s_0, G, o_1, a_1, \ldots, o_t). \quad (2)$$

An entire episode yields a trajectory $\tau = (s_0, G, (o_1, a_1), \ldots, (o_T, a_T))$, and the agent's objective is to maximize the expected return:

$$\max_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)], \quad R(\tau) = \sum_{t=1}^{T} r_t. \quad (3)$$

System-Level Asynchronous Design. Our framework features a fully **asynchronous multi-turn training architecture**, where rollout generation and policy optimization are decoupled and allocated to separate distributed resource groups, coordinated by a central controller. This separation eliminates training bottlenecks caused by unstable long-horizon environment interactions and significantly improves throughput stability in knowledge-intensive agentic settings.

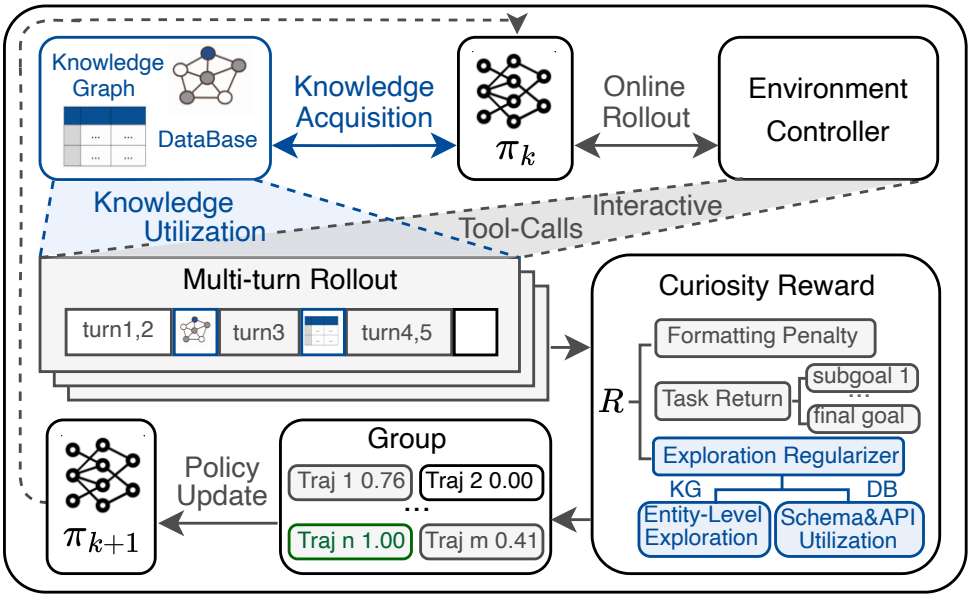


Figure 2: Illustration of KARL framework. Conditioned on the user task goal(G), initial state(S_0), the LLM agent extracts information from knowledge source($\mathcal{K}$), and engages in structured reasoning(r_t) and strategic tool invocation(fc_t) over multi-turn interactions. The environment obtains an interactive session with the agent, and provides fine-grained curiosity reward, which drives reinforcement learning to iteratively refine and optimize the agent's policy.

Containerized Environment Substrate. To support heterogeneous knowledge-intensive benchmarks (e.g., AgentBenchKG, Bird, Spider), we build a scalable containerized environment platform that standardizes tool interfaces across tasks. Each rollout is executed in an isolated interactive session, enabling safe and reproducible state transitions. The environment supports real-time observation updates, dynamic reward computation, and is easily extensible to diverse agentic domains.

3.2 Knowledge-Augmented RL

Agent tasks not only emphasize manipulation and decision-making, but also require factual or domain-specific knowledge beyond the model's innate capacity. We propose a knowledge-augmented framework, which involves automated knowledge construction process and dynamic knowledge exploration during execution of tasks.

Task-Specific Knowledge Construction. We construct a knowledge source $\mathcal{K}$ using automated pipelines. In Knowledge Graph (KG) tasks, we identify entities related to the query and retrieve Wikipedia-derived descriptions to encode type hierarchies and relations. In DataBase (DB) tasks, we extract tools or table names and retrieve structured specifications (e.g., schemas, API documentation) to support semantic understanding.

Relevance Filtering and Formatting. Extracted elements are filtered via heuristics (e.g., named entity recognition for KG, function/table alignment for DB) to ensure relevance, consistency, and scalability across domains.

Knowledge Augmented Optimization. Our KARL framework supports **dynamic** knowledge exploration process, where the agent decides on when and how to explore external structured knowledge during task completion. The agent's knowledge buffer correspondingly expands during task completion, expanding its capability boundary.

Therefore, the agent's policy can be redefined as:

$$\pi_\theta(a_t \mid h_t, \mathcal{K}, \mathcal{T}) = \prod_{i=1}^{n_t} p_\theta(a_{t,i} \mid a_{t,<i}, h_t, \mathcal{K}, \mathcal{T}),$$
$$h_t = (s_0, G, o_1, a_1, \ldots, o_t). \quad (4)$$

The policy π_θ is optimized within each environment using policy gradient methods. Our framework incorporates Group Relative Policy Optimization (GRPO) as advantage-based objectives. To regulate the exploration-exploitation trade-off, the policy objective is augmented with entropy and KL divergence regularization terms (see Appendix C for algorithmic details).

Our framework supports two distinct training

schemes: a standard single-task setup, and a multi-task regime. In the latter, the agent is jointly trained across a heterogeneous pool of tasks to promote generalization and structured exploration.

3.3 Curiosity Reward

To tackle the challenges of sparse supervision and credit assignment in knowledge-intensive environments, we define a composite, trajectory-level reward $R(\tau)$ that integrates task completion, syntactic correctness, and exploration incentives.

The overall reward is defined as:

$$R(\tau) = \lambda_{\text{explore}}(\tau) \cdot \min\left(R_{\text{task}}(\tau) + R_{\text{format}}(\tau),\ r_{\text{limit}}\right) \tag{5}$$

where $\lambda_{\text{explore}}(\tau) \in [0, 1]$ modulates the return based on exploration quality.

Task Return. Measures progress toward the final goal G and intermediate sub-goals $\{g_j\}_{j=1}^{M}$:

$$R_{\text{task}}(\tau) = w_{\text{final}} \cdot \mathbb{1}[\text{succeeds}(\tau, G)] + \sum_{j=1}^{M} w_j \cdot \mathbb{1}[\text{achieves}(\tau, g_j)] \tag{6}$$

Formatting Penalty. Penalizes syntactic or procedural errors in actions:

$$R_{\text{format}}(\tau) = -\alpha \sum_{t=1}^{T} \mathbb{1}[\text{is\_invalid}(a_t)] \tag{7}$$

Exploration Regularizer. Encourages diverse knowledge exploration behaviors:

$$\lambda_{\text{explore}}(\tau) = f\big(V_{\text{explore}}(\tau; \mathcal{K})\big) \tag{8}$$

where V_{explore} quantifies knowledge novelty along τ, and $f(\cdot)$ refers to a scaling function used for exploration incentive.

This structured reward signal promotes stable learning and optimizes knowledge exploration and utilization by aligning task success, procedural validity, and knowledge-guided exploration. It encourages the agent to effectively leverage external knowledge sources through continuous feedback in complex, knowledge-grounded environments. We validate this design through ablations in Section 5.

4 Experiments

4.1 Environment Setup

To support RL in complex, multi-turn tool-calling tasks representative of real-world scenarios, we adapt two diverse knowledge-intensive environments from AgentBench (Liu et al., 2023): **DataBase (DB)** and **Knowledge Graph (KG)**. All environments are integrated into a unified, asynchronous, session-based RL framework that supports long-horizon decision-making and realistic tool use with the following features:

- **Asynchronous Sessions:** Environments are implemented via a controller-worker architecture in order to support real-time, independent agent-environment episodes.
- **Dataset Construction:** For each environment, we construct tasks with corresponding initial states and ground-truth solutions. For the DB environment, we build upon the BIRD dataset (Li et al., 2023), extending it to include a wider range of SQL operations, namely `SELECT`, `UPDATE`, and `INSERT`. For KG, we create task instances based on the Freebase knowledge graph, further refined through expert annotation. See Appendix A.1 for detailed dataset statistics and composition.
- **Unified Tool-Calling Interface:** All tasks follow a standardized multi-turn tool-calling protocol (OpenAI function-call compatible), tracking complete tool-use histories and structured observations per episode. Details of tool sets can be found in Appendix B.2.
- **Streaming Reward Feedback:** Fine-grained, curiosity-based online reward signals are provided throughout each episode to both reflect task progress and evaluate the knowledge exploration behavior of the agent as shown in Section 3.3.

4.2 Experiment Settings

Training Paradigm. All RL experiments are conducted using the *AgentRL* framework (Zhang et al., 2025), which is further adapted to implement our KARL formulation (Section 3). We use Group Relative Policy Optimization (GRPO) as the default optimization algorithm. Unless otherwise noted, agents are initialized from scratch using the Qwen-2.5-14B-Instruct model and trained separately within each environment.

We also conduct SFT as a strategy ablation. Details of the SFT data collection process are provided in Appendix B.1.

To further examine generalization and exploration, we also performed joint training on DB and KG, and the results can be found at Appendix G.

Evaluation Framework. Evaluation is conducted on an extended version of the AgentBench framework (Liu et al., 2023), adapted to support multi-turn tool-calling protocols compatible with OpenAI

Model	Knowledge Graph (KG)			Database (DB)			Overall
	AgentBench KG (IID)	GrailQA (OOD)	WebQSP (OOD)	BIRD-dev (IID)	AgentBench DB (OOD)	Spider 1.0 (OOD)	
API LLMs (Prompting)							
gpt-4o-2024-11-20	46.7	73.8	53.6	67.9	52.3	88.9	63.9
o3-mini-2025-01-31	28.7	66.2	46.4	65.1	53.0	87.6	57.8
o3-2025-04-16	45.2	68.0	56.2	63.6	67.7	87.2	64.7
o4-mini-2025-04-16	59.2	79.3	60.3	67.3	60.0	88.8	69.2
claude-3-7-sonnet-2025-02-19	39.1	**81.9**	**67.3**	64.8	63.3	86.7	67.2
claude-3-7-sonnet-2025-02-19-thinking	47.6	61.7	42.9	64.7	64.0	86.2	61.2
cladue-4-sonnet-2025-05-14	64.0	<u>80.9</u>	62.4	67.0	<u>72.0</u>	88.0	<u>72.4</u>
cladue-4-sonnet-2025-05-14-thinking	62.0	80.7	63.1	66.8	69.8	89.0	71.9
gpt-5-2025-08-07	62.7	79.9	<u>64.2</u>	65.3	69.0	87.1	71.4
Open LLMs (Prompting)							
Deepseek-v3-2025-03-24	17.3	25.1	22.8	65.5	61.2	87.0	46.5
Deepseek-r1-2025-05-28	55.1	78.2	59.4	66.0	64.5	85.8	68.2
Qwen2.5-7B-Instruct	19.7	40.2	31.2	54.3	51.3	77.2	45.7
Qwen2.5-14B-Instruct	32.9	52.9	38.8	59.1	52.9	80.2	52.8
Open LLMs (Agent Training)							
AgentLM-7B (Zeng et al., 2024)	11.9	52.2	28.3	24.7	40.3	48.9	34.4
AgentLM-13B (Zeng et al., 2024)	13.4	47.9	27.9	26.6	40.6	52.7	34.9
AgentLM-70B (Zeng et al., 2024)	20.1	47.6	32.9	32.3	42.6	49.8	37.6
MiddleWare (w/ gpt-4o) (Gu et al., 2024)	35.1	64.4	27.9	45.2	-	72.9	-
Arctic-ExCoT-32B (Zhai et al., 2025)	-	-	-	<u>68.5</u>	56.0	86.6	-
XiYanSQL-32B-2504 (Gao et al., 2024)	-	-	-	67.1	44.3	<u>89.2</u>	-
KARL (ours)							
w/ Qwen2.5-7B-Instruct	<u>66.4</u>	66.8	50.7	64.3	57.1	86.2	65.3
w/ Qwen2.5-14B-Instruct	**75.4**	76.0	58.1	**72.2**	**75.7**	**89.5**	**74.5**

Table 1: Generalization results across six structured knowledge datasets. **Best** and <u>second-best</u> results per column are highlighted. Results are omitted (-) for datasets unsupported by model implementation or training configuration. KARL-trained model achieves state-of-the-art performance on most datasets, showing strong generalization ability.

function-call standard. All two core environments-**DataBase (DB)**, and **Knowledge Graph (KG)** are modified accordingly to align with this interface. We report mean performance of models.

We further augment the testbed with several structured knowledge datasets to evaluate generalization and reasoning. Specifically, **BIRD-dev** (Li et al., 2023), **Spider 1.0** (Yu et al., 2019), **GrailQA** (Gu et al., 2021), and **WebQSP** (Yih et al., 2016) are integrated into the tool-calling setting and filtered for solvability. The **Spider 2.0** benchmark (Lei et al., 2025), which features more complex compositional queries, is directly adapted into our OpenAI-compatible tool interface.

Importantly, transforming these datasets into multi-turn tool-use formats often increases task complexity and changes the structure of intermediate reasoning. As such, we standardize the testbed to ensure consistent evaluation conditions across both proprietary models and our trained agents.

We define two evaluation regimes:

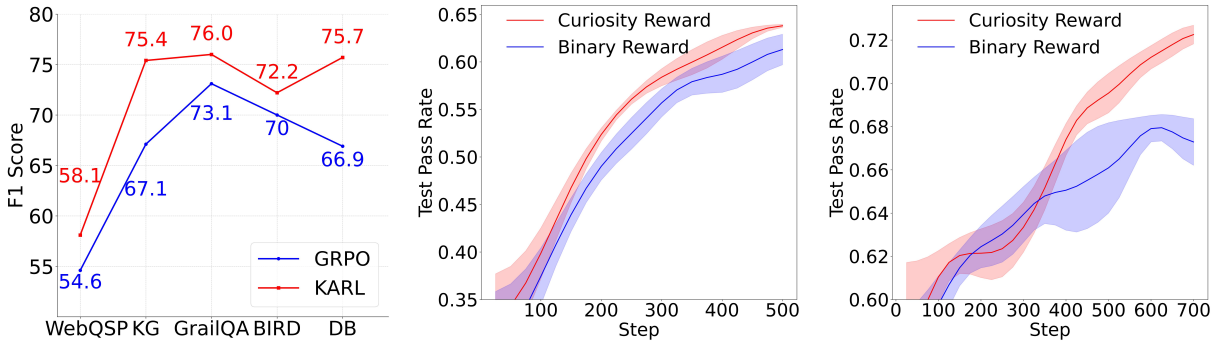
- **Held-in:** Tasks drawn from training-aligned environments (AgentBench KG and BIRD-dev), with additional task diversity to test robustness.
- **Held-out:** Out-of-distribution and transfer evaluation using unseen datasets, including Spider 1.0 (Yu et al., 2019), AgentBench DB (Liu et al., 2023), GrailQA (Gu et al., 2021), WebQSP (Yih et al., 2016), Spider 2.0-Snow, and Spider 2.0-Lite (Lei et al., 2025).

By normalizing interface and complexity, our testbed ensures fair comparison across diverse reasoning and tool-use capabilities.

We evaluate API LLMs, Open LLMs based on prompting or agent training techniques, as well as our KARL-trained agents on these benchmarks. Specifically, for API LLMs, we provide evaluation results in Table 1 as default evaluation method, and further evaluation results using RAG techniques are specified in Table 12.

4.3 Results and Analysis

Results of Knowledge Graph (KG) tasks. Table 1 showcases the strong performance of our method on knowledge-centric tasks. On AgentBench KG,



(a) Comparison of performance of agents trained with and without knowledge augmentation (KARL vs GRPO) on different structured knowledge settings

(b) Comparison of test set pass rate between different reward design strategies (Exploration-guided reward vs binary reward) on KG

(c) Comparison of test set pass rate between different reward design strategies (Exploration-guided reward vs binary reward) on DB

Figure 3: Overall ablation of models across different settings. The implementation of both knowledge-augmentation procedure and exploration-guided reward boost our model's performance.

Method	AgentBenchKG	GrailQA	WebQSP	Bird-dev	AgentBenchDB	Spider1.0	Overall
GRPO-14B-Binary Reward-Static Knowledge	65.3	69.1	42.4	69.6	67.3	**89.8**	67.3
GRPO-14B-Binary Reward-Dynamic Exploration	63.4	<u>75.6</u>	<u>53.1</u>	<u>70.0</u>	65.7	87.9	69.3
GRPO-14B-Curiosity Reward-Static Knowledge	<u>73.3</u>	73.5	52.9	68.2	<u>68.9</u>	88.7	<u>70.9</u>
KARL-14B-Curiosity Reward-Dynamic Exploration	**75.4**	**76.0**	**58.1**	**72.2**	**75.7**	<u>89.5</u>	**74.5**

Table 2: Finer-grained ablations on Qwen2.5-14B-Instruct model of the two main components of KARL framework: reward design and dynamic knowledge exploration method. **Best** and <u>second-best</u> results per column are highlighted. Our KARL-trained agent surpasses the other GRPO-trained variants in both KG and BIRD domains and shows great generalization ability.

our KARL-trained agent built on Qwen2.5-14B-Instruct achieves a new state-of-the-art F1 score of **75.4**, outperforming all open-source baselines, training-based methods on structured knowledge tasks, as well as the best closed-source baseline (claude-4-sonnet-2025-05-14), by a substantial margin of **+11.4** points. This highlights the effectiveness of reinforcement learning combined with structured knowledge augmentation in complex multi-hop reasoning settings.

Our approach also generalizes well to out-of-domain benchmarks. On GrailQA and WebQSP, KARL achieves new best open-model results: 76.0 on GrailQA and 58.1 on WebQSP—surpassing gpt-4o-2024-11-20 and the o3 series, and closely approaching o4-mini-2025-04-16 and claude-4-sonnet-2025-05-14.

These results demonstrate that dynamic exploration with knowledge source enhanced via reinforcement learning is the key to building robust agents capable of complex knowledge reasoning and generalization.

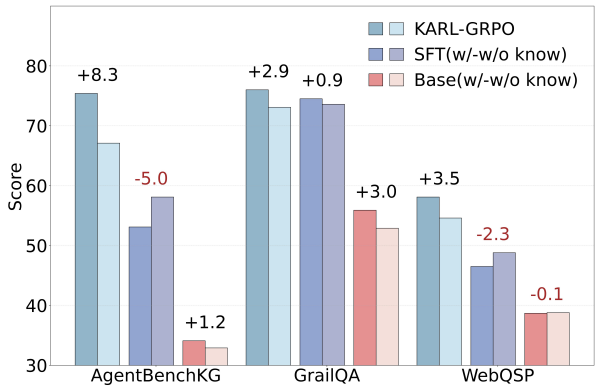
Results of DataBase (DB) tasks. Table 1 presents results on database-centric tasks. Training through KARL framework yields substantial gains: Qwen2.5-14B-Instruct trained with KARL achieves new state-of-the-art scores on both BIRD-dev (**72.2**), AgentBench DB(**75.7**) and Spider 1.0 (**89.5**), outperforming all evaluated closed-source models, open-source baselines, as well as trained methods targeting structured knowledge tasks.

Our framework yields strongest results—both on in-domain and out-of-distribution DB tasks. These findings underscore the power of combining reinforcement learning with structured knowledge exploration.

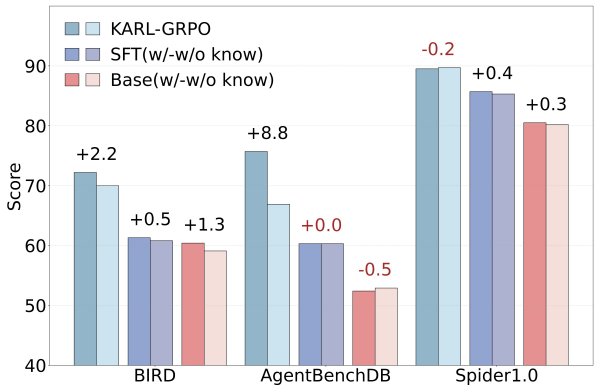
Detailed analysis of knowledge-intensive settings regarding sensitivity to noisy knowledge and benefit of knowledge augmentation across tasks can be found at Appendix D. We also provide additional RAG-based baseline results in Appendix F, generalization results in Appendix G, extensibility results of our framework in Appendix H and in-depth analysis of failure modes and case studies in Appendix E.

5 Ablation Study

We conduct ablation study on the key strategies and designs from KARL, including the knowledge augmentation itself and the effect of RL utiliza-



(a) Knowledge Leverage among different training algorithms on KG-related datasets.



(b) Knowledge Leverage among different training algorithms on DB-related datasets.

Figure 4: Knowledge Leverage among different training algorithms. Gain on performance is observed when RL is applied to knowledge-intensive context, while this is often not the case for SFT and base models, demonstrating effectiveness of RL on knowledge leverage.

Model	**AgentBenchKG**	**GrailQA**	**WebQSP**	**BIRD**	**AgentBenchDB**	**Spider1.0**	**Overall**
Qwen2.5-7B Variants							
Qwen2.5-7B	19.7	40.2	31.2	54.3	51.3	77.2	45.7
R1-Searcher-7B (Song et al., 2025)	5.6	52.9	23.9	40.2	47.0	79.1	41.5
ReSearch-7B (Chen et al., 2025)	9.9	57.2	27.9	41.8	51.7	82.0	45.1
ToolRL-7B (Qian et al., 2025)	8.5	54.6	25.8	43.1	52.2	82.2	44.4
RAG-R1-7B (Tan et al., 2026)	14.9	58.3	28.7	52.9	52.7	81.2	48.1
Qwen2.5-7B-KARL (Dynamic Exploration)	66.4	66.8	50.7	64.3	57.1	86.2	65.3
Qwen2.5-14B Variants							
Qwen2.5-14B	32.9	52.9	38.8	59.1	52.9	80.2	52.8
Qwen2.5-14B-RAG	34.1	55.9	38.7	60.4	52.4	80.5	53.7
Qwen2.5-14B-GRPO-RAG	60.9	70.8	51.3	70.7	66.3	**89.7**	68.3
Qwen2.5-14B-RL (Static Knowledge)	73.3	73.5	52.9	68.2	68.9	88.7	70.9
Qwen2.5-14B-KARL (Dynamic Exploration)	**75.4**	**76.0**	**58.1**	**72.2**	**75.7**	89.5	**74.5**

Table 3: Performance of models with different knowledge augmentation methods across structured knowledge datasets. **Best** and second-best results per column are highlighted. Our model trained with KARL achieves the best performance across all datasets.

tion, reward design, and comparisons with other knowledge augmentation methods.

Knowledge Augmentation. To better assess the role of knowledge augmentation, we conduct controlled ablations on environments between KARL and GRPO (RL alone without providing external knowledge source).

As shown in Figure 3a, knowledge augmentation consistently improves performance across both in-domain and out-of-domain evaluations. On AgentBench KG, for instance, incorporating knowledge during training boosts performance from 67.1 to 75.4. Similar trends are observed on the held-out KG and DB benchmarks: knowledge-augmented agents outperform their non-augmented counterparts by a large margin—achieving 58.1 vs. 54.6 on WebQSP, and 75.7 vs. 66.9 on AgentBench DB.

These results underscore the critical importance of knowledge augmentation in structured, knowledge-intensive domains. Providing agents with knowledge source while RL training not only improves task completion, but also strengthens their ability to reason over and dynamically explore knowledge — ultimately enhancing generalization and compositional reasoning across diverse knowledge-grounded tasks.

Effect of RL on Knowledge Leverage. We compare base models, SFT-only models, and RL-trained models with and without knowledge augmentation to demonstrate the effectiveness of RL in enhancing knowledge leverage abilities. Results across six structured knowledge datasets are sum-

marized in Figure 4.

While adding external knowledge source alone provides limited or no gain in both base models and SFT-trained models and in some cases slightly hurts performance, RL substantially improves results, indicating that RL is key to learning how to use external context effectively. This further underscores the importance of the interactive training framework in KARL, where both tool-using behavior and knowledge usage are optimized jointly in a task-driven manner.

Reward Shaping Strategy. To isolate the effect of different reward signal strategies, we conduct ablations on the reward design for RL training in KARL framework using Qwen2.5-14B-Instruct with GRPO. We compare our primary *Curiosity Reward* approach with the *binary reward* scheme (assigning $+1.0$, 0.0 for success and failure. The results are shown in Figure 3b and Figure 3c.

Curiosity reward: Providing more continuous reward encouraging knowledge exploration signal yields the best performance across KG and DB. Tasks with complex multi-step reasoning and tool us benefit substantially from richer, shaped signals.

Binary reward: This scheme generally underperforms, suggesting that richer feedback and sufficient exploration is important for stable RL optimization and improvement of agent reasoning in knowledge-intense scenarios.

We also provide finer-grained ablations of GRPO-trained Qwen2.5-14B-Instruct on different knowledge augmentation strategies to further validate our curiosity reward design. As is shown in Table 2, whether we provide static knowledge injection or dynamic knowledge exploration to the agent, curiosity reward design boosts the performance of agent across most scenarios, showing its effectiveness in knowledge-intensive tasks.

Comparison with Different Knowledge-Augmentation Methods. We compare our method with RAG baselines(with knowledge retrieval at the start of episode for base and GRPO-trained models), RL-based search and retrieval agents, and other knowledge augmentation methods(training the model with static knowledge injection at the start of episode with GRPO). As is shown in Table 3, compared to those baselines, our dynamic knowledge exploration support and reinforcement learning framework (KARL) consistently achieves the best performance across all benchmarks.

Furthermore, as shown in Table 2, agents trained with GRPO to dynamically explore knowledge consistently surpasses those trained with static knowledge injection under different reward designs. This result highlights the importance of coupling knowledge acquisition with policy optimization, rather than treating knowledge as a fixed input.

Overall, KARL enables dynamic expansion of the agent's capability boundary by jointly optimizing knowledge exploration and decision-making, leading to superior performance on structured knowledge-intensive tasks.

6 Conclusion

In this work, we introduced **KARL**, a unified Knowledge-Augmented Reinforcement Learning framework for LLM agents. Our KARL framework supports stable and scalable multi-turn reinforcement learning, facilitating real-time knowledge access, long-horizon decision-making, and instant curiosity-based feedback.

By combining multi-turn tool use with dynamic knowledge exploration through training process, KARL expands the operational boundaries of agents, enabling improved reasoning and decision-making in knowledge-intensive tasks. Extensive experiments across diverse AgentBench domains demonstrate that KARL achieves state-of-the-art performance on knowledge-based and database benchmarks (e.g., AgentBench KG, BIRD), while also enhancing generalization to unseen tasks such as GrailQA, WebQSP, AgentBench DB, Spider 1.0, Spider 2.0, and BFCLv3.

Overall, KARL establishes a robust paradigm for training intelligent, knowledge-driven LLM agents, demonstrating the practical impact of RL-enhanced knowledge exploration on both agent performance and generalization ability.

7 Acknowledgement

This work was supported by the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM101), Natural Science Foundation of China (624B2084, 62425601, 62495063), BNR2026RC01009 and the New Cornerstone Science Foundation through the XPLORER PRIZE.

8 Limitations

Our agent depends on a 14B backbone model. As a result, there remains performance headroom on OOD datasets such as GrailQA and WebQSP,

which can be attributed to both model scale limitations and linguistic mismatches across datasets as discussed in Appendix D.3. In future work, we plan to scale up the model size and incorporate more diverse training data to further reduce OOD discrepancies and improve generalization.

What's more, evaluation of KARL has focused primarily on tasks where structured knowledge is critical (e.g., KG and DB), leaving open questions regarding the trade-offs between intrinsic model capacity and knowledge integration across a wider range of agentic tasks.

Additionally, KARL is currently designed for text-based environments; extending it to multimodal domains, such as web interfaces, mobile systems, or robotics, by integrating Vision-Language Models (VLMs), represents an important avenue for further research.

References

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. 2025. Research: Learning to reason with search for llms via reinforcement learning. *Preprint*, arXiv:2503.19470.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. *Preprint*, arXiv:2211.10435.

Yingqi Gao, Yifu Liu, Xiaoxia Li, Xiaorong Shi, Yin Zhu, Yiming Wang, Shiqi Li, Wei Li, Yuntao Hong, Zhiling Luo, and 1 others. 2024. A preview of xiyan-sql: A multi-generator ensemble framework for text-to-sql. *arXiv preprint arXiv:2411.08599*.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, WWW '21, page 3477–3488. ACM.

Yu Gu, Yiheng Shu, Hao Yu, Xiao Liu, Yuxiao Dong, Jie Tang, Jayanth Srinivasa, Hugo Latapie, and Yu Su. 2024. Middleware for llms: Tools are instrumental for language agents in complex environments. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7646–7663.

Zen Kit Heng, Zimeng Zhao, Tianhao Wu, Yuanfei Wang, Mingdong Wu, Yangang Wang, and Hao Dong. 2025. Boosting universal llm reward design through heuristic reward observation space evolution. *Preprint*, arXiv:2504.07596.

Jerry Huang, Siddarth Madala, Risham Sidhu, Cheng Niu, Hao Peng, Julia Hockenmaier, and Tong Zhang. 2025a. Rag-rl: Advancing retrieval-augmented generation via rl and curriculum learning. *Preprint*, arXiv:2503.12759.

Xu Huang, Weiwen Liu, Xingshan Zeng, Yuefeng Huang, Xinlong Hao, Yuxian Wang, Yirong Zeng, Chuhan Wu, Yasheng Wang, Ruiming Tang, and Defu Lian. 2025b. Toolace-dev: Self-improving tool learning via decomposition and evolution. *Preprint*, arXiv:2505.07512.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? *Preprint*, arXiv:2310.06770.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *Preprint*, arXiv:2503.09516.

Sopan Khosla, Ritam Dutt, Vinayshekhar Bannihatti Kumar, and Rashmi Gangadharaiah. 2023. Exploring the reasons for non-generalizability of KBQA systems. In *Proceedings of the Fourth Workshop on Insights from Negative Results in NLP*, pages 88–93, Dubrovnik, Croatia. Association for Computational Linguistics.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. 2025. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. *Preprint*, arXiv:2411.07763.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Preprint*, arXiv:2305.03111.

Pengxiang Li, Zhi Gao, Bofei Zhang, Yapeng Mi, Xiaojian Ma, Chenrui Shi, Tao Yuan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, and Qing Li. 2025. Iterative tool usage exploration for multimodal agents via step-wise preference tuning. *Preprint*, arXiv:2504.21561.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng

Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2023. Agentbench: Evaluating llms as agents. *Preprint*, arXiv:2308.03688.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, and 1 others. 2024. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *Preprint*, arXiv:2504.13958.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *Preprint*, arXiv:2307.16789.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *Preprint*, arXiv:2503.05592.

Zhiwen Tan, Jiaming Huang, Qintong Wu, Hongxuan Zhang, Chenyi Zhuang, and Jinjie Gu. 2026. Rag-r1: Incentivizing the search and reasoning capabilities of llms through multi-query parallelism. *Preprint*, arXiv:2507.02962.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. 2025. Otc: Optimal tool calls via reinforcement learning. *Preprint*, arXiv:2504.14870.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

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

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2019. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *Preprint*, arXiv:1809.08887.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024. Agenttuning: Enabling generalized agent abilities for llms. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3053–3077.

Bohan Zhai, Canwen Xu, Yuxiong He, and Zhewei Yao. 2025. Excot: Optimizing reasoning for text-to-sql with execution feedback. *arXiv preprint arXiv:2503.19988*.

Hanchen Zhang, Xiao Liu, Bowen Lv, Xueqiao Sun, Bohao Jing, Iat Long Iong, Zhenyu Hou, Zehan Qi, Hanyu Lai, Yifan Xu, Rui Lu, Hongning Wang, Jie Tang, and Yuxiao Dong. 2025. Agentrl: Scaling agentic reinforcement learning with a multi-turn, multi-task framework. *Preprint*, arXiv:2510.04206.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. *Preprint*, arXiv:2307.13854.

A Additional Experimental Details

A.1 Dataset Construction Details

Dataset composition is shown in Table 4.

DataBase tasks contruction We extend the original BIRD dataset, which only contains SELECT queries, by constructing new instructions for INSERT and UPDATE operations using the self-instruct. To further enhance the task diversity and encourage exploration of the agent during reinforcement learning, we leverage GPT-4o to rewrite or paraphrase instructions into more ambiguous

Environment	#Tasks	Data Source	Construction Notes
DB	3531	BIRD with Self-Instruct	BIRD augmented (expanded types of SQL) with Self-Instruct; further cleaned by GPT-4o
KG	1214	Annotated KBQA	Expert annotations of Knowledge-based tasks of Freebase

Table 4: Dataset composition, size, and construction details for all environments.

or indirect forms, allowing the agent to generalize beyond templated patterns.

Knowledge Graph tasks construction We generate tasks based on Freebase. Human annotators are given generated tasks, and asked to finish and verify the completion of task.

B Additional Environment Details

B.1 SFT Data Collection Details

To construct high-quality supervised fine-tuning (SFT) data, we collect tool-augmented trajectories by rolling out strong reasoning agents that perform well in their respective environments. For Database (DB) tasks, in addition to generating trajectories with GPT-4o and Claude 3, we manually correct a portion of the outputs with expert supervision—especially to fix syntax issues such as misplaced backticks in column names. For Knowledge Graph (KG) tasks where ground-truth actions are available, we adopt a reverse construction strategy: using the Qwen model to generate plausible thoughts that logically lead to the known tool calls, thus forming coherent (thought, action) pairs. This multi-source, environment-adapted strategy allows us to build robust and semantically grounded SFT data across diverse tool-use scenarios.

B.2 Tool Set Details

Table 5 lists the set of tools available in each of the two environments used in our study: DB-related environments and KG-related ones. Each environment defines a specific set of tools tailored to its task space. These tools are exposed to the agent, and each tool is associated with a name and a short description indicating its functionality. This design enables agents to interact with diverse environments in a structured and interpretable way.

C RL Algorithms

We provide implementation details and loss formulations for the reinforcement learning (RL) algorithms used in our framework. As introduced in Section 3, we adopt a knowledge-augmented, multi-turn tool-calling setup, where agents are equipped with an external knowledge source. Within this setting, we apply one advantage-based policy gradient method— Group Relative Policy Optimization (GRPO)—to jointly optimize reasoning and tool-use behaviors. Full algorithmic details and training objectives are described below.

C.1 Group Relative Policy Optimization (GRPO)

GRPO (Shao et al., 2024) replaces value-based advantage estimation with a group-relative advantage, placing greater emphasis on the ranking of sequences within each sampled group. For each prompt q, G outputs $\{a_1, ..., a_G\}$ are sampled from roll-outs of the agent; for token $a_{i,t}$, a normalized group-based advantage $\widehat{A}_{i,t}$ is computed, yielding the following advantage-related loss:

$$\mathcal{L}_{\text{GRPO}} = -\mathbb{E}_{q,\,\{a_i\}\sim\pi_{\text{old}}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|a_i|}\sum_{t=1}^{|a_i|} I(a_{i,t})\cdot \min\left(\rho_{i,t}\widehat{A}_{i,t},\ \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\widehat{A}_{i,t}\right)\right] \tag{9}$$

where $\rho_{i,t} = \frac{\pi_\theta(a_{i,t}|q,a_{i,<t})}{\pi_{\text{old}}(a_{i,t}|q,a_{i,<t})}$. Compared to standard PPO, GRPO is more computational-efficient and leverages relative outcomes within each group.

C.2 Full Policy Loss

In order to balance among exploration, exploitation and stability of training, we use the following composed actor loss for GRPO in our setting:

$$\mathcal{L}_{\text{actor}} = \mathcal{L}_{\text{RL}} - \lambda\,\mathbb{E}\big[H[\pi_\theta(\cdot|c)]\big] + \beta\,\mathbb{E}\big[D_{\text{KL}}(\pi_\theta(\cdot|c)\ ||\ \pi_{\text{ref}}(\cdot|c))\big] \tag{10}$$

where $\mathcal{L}_{\text{RL}}$ denotes GRPO advantage-based loss for detail), c represents the policy context (i.e., the full generation history up to each token), λ is the entropy regularization coefficient, and β controls

Environment	Tool Name	Description
DB	execute_sql	Executes a given SQL statement on the database and returns the result.
	commit_final_answer	Commits the final answer after all operations are completed.
KG	get_relations	Fetches all relations associated with a given entity or variable to navigate the KB.
	get_neighbors	Returns all entities connected to a variable through a specified relation.
	intersection	Computes the intersection of two sets of entities or variables.
	get_attributes	Retrieves all numerical attributes of a given variable.
	argmax	Finds the entity with the maximum value of a specified attribute.
	argmin	Finds the entity with the minimum value of a specified attribute.
	count	Counts the number of entities within a given variable.

Table 5: Tool sets and their descriptions across different environments.

Model / Performance Gap	KG (F1/EM)	GrailQA (F1/EM)	WebQSP (F1/EM)
Qwen2.5-14B-Instruct-SFT	2.2 / 2.6	1.1 / 0.9	2.6 / 1.2
Qwen2.5-14B-Instruct-KARL (KG)	1.3 / 1.4	1.7 / 1.6	1.5 / 1.0

Table 6: Performance gaps under noisy knowledge context. The performance do not downgrade much when disturbed by mild knowledge corruption.

Model	AgentBench KG (F1)	GrailQA (F1)	WebQSP (F1)
Qwen2.5-14B-Instruct	32.9 ± 5.9	52.9 ± 2.6	38.8 ± 2.4
Qwen2.5-14B-Instruct-GRPO	67.1 ± 5.3	73.1 ± 3.92	54.6 ± 1.37
Qwen2.5-14B-Instruct-KARL	75.4 ± 5.1	76.0 ± 1.8	58.1 ± 2.0

Table 7: Performance on knowledgebase tasks (95% confidence intervals). Knowledge exploration is crucial for knowledgebase tasks.

Model	AgentBench DB	BIRD-dev	Spider-1.0
Qwen2.5-14B-Instruct	52.9 ± 4.3	59.1 ± 2.4	80.2 ± 1.96
Qwen2.5-14B-Instruct-GRPO	66.9 ± 2.35	70.0 ± 2.35	89.7 ± 0.98
Qwen2.5-14B-Instruct-KARL	75.7 ± 3.13	72.2 ± 0.98	89.5 ± 0.19

Table 8: Performance on Database tasks (95% confidence intervals). Knowledge exploration achieves great gain on complex database tasks like BIRD and AgentBench-DB, while on Spider1.0 where knowledge could be explored, the gains are modest.

the strength of KL penalty. What's more, the entropy and KL terms serve to promote exploration and policy stability.

D Additional Analysis on Knowledge Intensive Settings

D.1 Robustness under Noisy Knowledge

We further evaluate the robustness of our method under **noisy knowledge** settings, where a subset

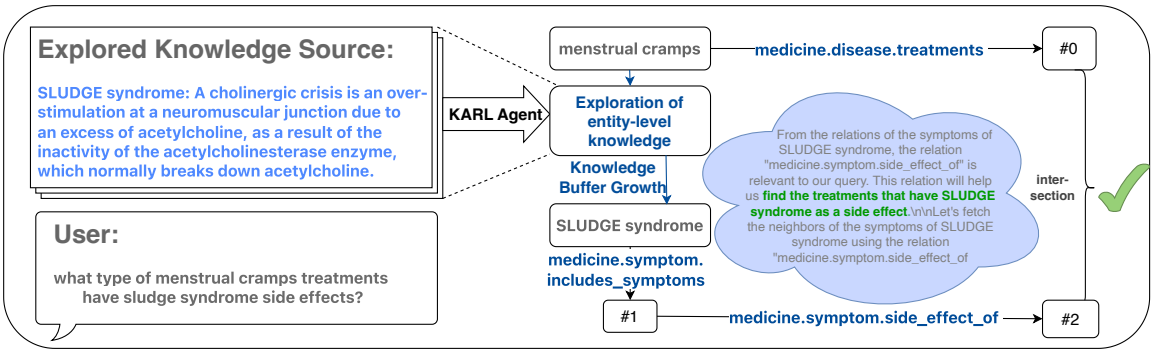


Figure 5: Correct Trajectory Operated by KARL-trained Agent

of entities are randomly assigned incorrect knowledge. Table 6 reports the performance gaps (correct knowledge minus noisy knowledge) for both SFT and KARL models.

Overall, our framework maintains strong performance even with mild knowledge corruption, highlighting its ability to leverage environmental feedback to recover critical relations.

D.2 Benefit of Knowledge Augmentation varies across Tasks

The benefit of knowledge augmentation varies across tasks, depending on how essential external knowledge is for task success. Table 7 and 8 report results with 95% Confidence Intervals for clarity.

Where knowledge helps most:

- **KG tasks (AgentBench-KG, GrailQA, WebQSP):** External knowledge helps the model understand unseen entities, their attributes, and relationships.
- **BIRD and AgentBench-DB:** In complex table navigation and tool use, explored knowledge of APIs and table schemas supports better understanding of callable tool behaviors and schema constraints.

Where gains are modest:

- **Spider:** Agents can already acquire task-relevant knowledge through exploration of the database itself, so augmentation is less impactful.

In summary, knowledge injection is most beneficial in environments where critical facts or structures are not directly observable and must be inferred or recalled, while our RL framework remains strong across all scenarios.

D.3 Explanation of Performance Gaps Across Datasets

Our KARL-trained 14B agents achieve SOTA performance on trained domains(AgentBench KG and BIRD-dev) and great generalization ability performance on out-of-distribution (OOD) domains. Though this is a considerable result for small models, our agents' performance on OOD datasets such as GrailQA and WebQSP remains lower than that of the strongest closed-source baselines relying on substantially larger models.

This performance gap is expected not only due to the model scale difference, but also the linguistic mismatch across datasets. Although AgentBenchKG, GrailQA, and WebQSP are grounded in the same underlying knowledge base, prior study (Khosla et al., 2023) has shown that they differ significantly in linguistic characteristics, including question explicitness, length, grammaticality, readability, and coherence due to distinct annotation pipelines. These variations introduce an additional dimension of distribution shift beyond structural or reasoning complexity and are known to contribute substantially to cross-dataset performance degradation.

Therefore, this gap opens an opportunity for futher improvement of our setting. By scaling the model size and incorporating more linguistically diverse and interaction-rich training data, we expect to further reduce OOD discrepancies and improve generalization across heterogeneous knowledge-intensive tasks.

E In-Depth Analysis

E.1 Tool Calls and Failure Modes

Tool Calls. Table 9 shows the average number of tool calls per episode for the base and KARL-

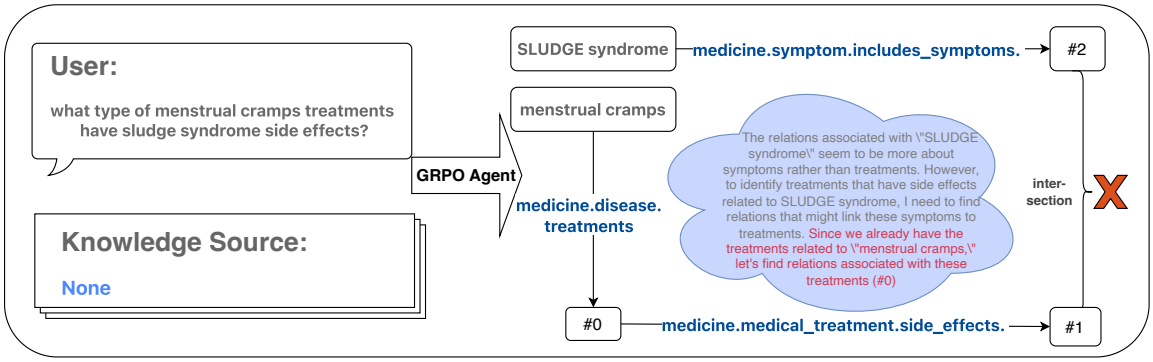


Figure 6: Wrong Trajectory Operated by GRPO-trained Agent without Knowledge Context

trained models across key environments. The results demonstrate that KARL reduces the average tool calls across all tasks, reflecting improved efficiency and reasoning capabilities.

Environment	Base	KARL
DB	4.34	3.35
KG	9.32	7.45

Table 9: Average number of tool calls per episode for base and KARL-trained models. Our agent finish tasks in fewer turns, demonstrating efficient tool-using ability.

Failure Modes. Tables 10 and 11 report the distribution of key failure modes across select environments for the base and KARL-trained models. Completed indicates successful task completion without forced termination. Results show that KARL significantly reduces task limit reached errors and increases completion rates, demonstrating enhanced robustness.

E.2 Case Study

In this section, we focus on a single representative question: **"What type of menstrual cramps treatments have sludge syndrome side effects?"** We compare the behaviors of different agents—a KARL-trained agent, a GRPO-trained agent, and the agent using the base Qwen2.5-14B-Instruct model—under varying knowledge context conditions. Our analysis demonstrates how incorporating structured knowledge during training leads to more coherent reasoning and more accurate task completion.

Analysis of correct trajectory operated by our KARL-Trained Agent As is shown in Figure 5, our KARL-trained agent successfully follows a correct reasoning trajectory to answer the question. In this case, the agent first identifies potential treatments for *menstrual cramps*. Then, it explores the knowledge about *sludge syndrome*, and leverages this to retrieve the symptoms associated with *sludge syndrome*. It subsequently infers which treatments might cause those symptoms as side effects. By intersecting this set with the treatments initially found for menstrual cramps, the agent arrives at the correct final answer.

This behavior demonstrates the KARL-trained agent's ability to effectively explore knowledge, utilize it and compose multi-hop reasoning steps.

Env	Completed	Task Limit	Invalid Act
DB	0.957	0.043	0.000
KG	0.747	0.213	0.040

Table 10: Failure mode distribution for the base model across environments.

Env	Completed	Task Limit	Invalid Act
DB	0.993	0.007	0.000
KG	0.947	0.033	0.020

Table 11: Failure mode distribution for the KARL-trained model across environments. Invalid actions and reaching task limits are significantly reduced.

Analysis of wrong trajectory performed by GRPO-Trained Agent In alignment with its training setup, the GRPO-trained agent was not provided with any knowledge source during this task. As a result, the agent ultimately failed to answer the question correctly, as shown in Figure 6.

Initially, the agent successfully retrieves the symptoms associated with *sludge syndrome* as well

Model	AgentBenchKG	GrailQA	WebQSP	Bird-dev	AgentBenchDB	Spider1.0	Overall
o4-mini-2025-04-16-RAG	43.7	75.8	59.0	57.8	64.3	87.1	64.6
claude-4-sonnet-2025-05-14-RAG	69.0	80.6	62.2	64.1	70.9	88.6	72.6
gpt-5-2025-08-07-RAG	68.1	**83.9**	63.0	59.0	69.3	87.1	71.7
claude-3-7-sonnet-2025-02-19-RAG	58.1	79.9	**66.7**	59.6	66.7	86.9	69.7
Deepseek-v3-2025-03-24-RAG	13.1	32.9	14.1	54.6	60.5	86.3	43.6
Deepseek-r1-2025-05-28-RAG	44.8	76.3	61.4	55.5	66.0	85.7	65.0
AgentLM-7B-RAG	8.7	45.5	22.1	18.3	40.6	48.8	30.7
KARL-7B (ours)	66.4	66.8	50.7	64.3	57.1	86.2	65.3
KARL-14B (ours)	**75.4**	76.0	58.1	**72.2**	**75.7**	**89.5**	**74.5**

Table 12: Performance comparison between KARL and RAG-based baselines across multiple knowledge-intensive benchmarks. Our KARL-trained model achieves the best overall performance.

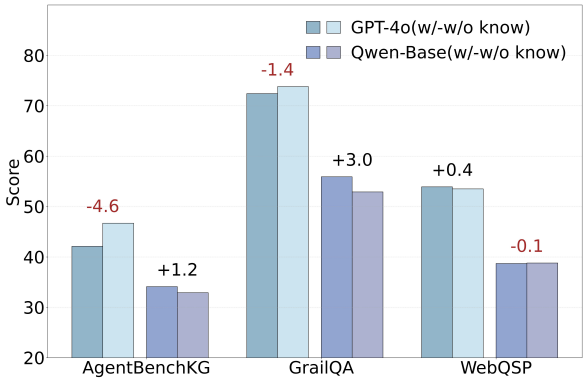


Figure 7: Performance Gap with and without Knowledge Injection during Test Time. Providing knowledge context without integrating into the training process does not improve performance in most cases.

as the treatments for *menstrual cramps*. However, the critical step in the reasoning process is to determine which of these treatments have *sludge syndrome* as a side effect. Instead of reasoning in this direction, the agent incorrectly attempts to identify whether the side effects of these treatments overlap with the symptoms of *sludge syndrome*.

Although the early part of the agent's exploration is reasonable, the agent ultimately fails to correctly interpret the question and to distinguish the semantic relationships between the entities involved. This leads to a flawed reasoning trajectory and an incorrect final answer.

Analysis of wrong trajectory of Base Qwen2.5-14B-Instruct Agent As is shown in Figure 8, the base agent also fails to correctly answer the question. While it begins by successfully identifying the treatments associated with *menstrual cramps*, the subsequent reasoning steps lack coherence and correctness.

Specifically, the agent demonstrates an incorrect understanding of *sludge syndrome*. Rather than retrieving the symptoms related to sludge syndrome and reasoning about which treatments might cause those symptoms, the agent attempts to intersect the side effects of the treatments previously retrieved with the term *sludge syndrome* itself. This reflects a misunderstanding of the question's semantic structure and an inability to decompose the query into its necessary multi-hop reasoning steps.

As a result, the agent fails to complete the task, highlighting the limitations of untrained or minimally trained agents when faced with questions that require precise knowledge interpretation and compositional reasoning.

E.3 Cost of frequent knowledge queries in real-time applications.

In real-time applications, a great concern about knowledge-intensive agents is the potential overhead of frequent knowledge queries. Our KARL setting largely reduces this issue through unique implementation of the framework and action efficiency of the agent.

Negligible Knowledge Exploration Cost in Our Setting. Firstly, our knowledge sources are pre-constructed and stored locally, enabling constant-time in-memory retrieval without external API calls or network communication. Consequently, the marginal latency of each knowledge query is effectively negligible.

What's more, although KARL explicitly encourages proactive knowledge exploration, it improves long-horizon decision-making and reduces redundant environment interactions. Empirically, as is shown in Table 9, this leads to fewer steps required to complete tasks. Therefore, knowledge exploration reduces downstream interaction cost rather than introducing additional runtime overhead.

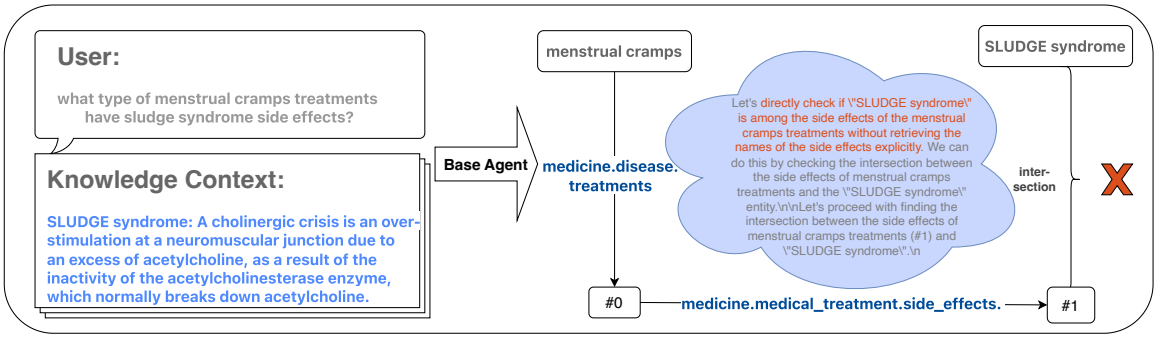


Figure 8: Wrong Trajectory Operated by Base Qwen2.5-14B-Instruct Agent with Knowledge Context Provided

Applicability of KARL to real-world deployments. In real-world settings where knowledge retrieval relies on external search engines, large-scale document stores, or API-based systems, query cost may become non-trivial. In such cases, our framework can be naturally extended by incorporating retrieval-cost-aware penalties into the reinforcement learning objective, enabling explicit control of the exploration–efficiency trade-off.

F Influence of Knowledge Injection during Test Time

Table 12 provides the result of baseline methods equipped with RAG to isolate the influence of external knowledge injection on the performance of existing APIs and training methods. As is shown in the table, our KARL-trained model consistently outperforms other RAG-enhanced representative baselines across most benchmarks, achieving the best overall performance.

Additionally, we provide a visualization of the gap in the ability of knowledge utilization of existing baseline models in Figure 7. We compare the performance of GPT-4o and Qwen2.5-14B-Instruct base models on a range of knowledge graph-related tasks, under two conditions: with and without knowledge context provided at test time. Interestingly, providing knowledge context alone does not consistently improve task performance. In fact, both models exhibit minor difference in performance or decreased accuracy on most datasets when knowledge is given directly without additional guidance. This highlights the value of our KARL framework, which not only supplies structured knowledge, but also explicitly teaches the model how to reason with it. By integrating external knowledge into both exploration and task completion processes, KARL enables more effective use of knowledge for complex reasoning tasks in knowledge-intensive environments.

G Additional Generalization Results

G.1 Cross-Task Generalization

We evaluate the generalization of SFT-trained and KARL-trained models (in KG settings) on unseen tools across AgentBenchDB, BIRD-dev, and Spider-1.0 tasks. Results are shown in Table 13.

While SFT-trained models experience notable performance drops due to distribution shift, KARL framework exhibits strong transfer from Knowledge Graph (KG) to previously unseen DataBase tasks, show generalization to unseen tool settings.

G.2 Generalization to Unseen, Complex Tool-Calling Benchmarks

We further evaluate our framework on the **BFCL-v3** benchmark (Patil et al., 2025), which features more complex tool uses. Table 15 reports agents' performance across single-turn non-live AST and live AST settings.

KARL outperforms the base Qwen2.5-14B-Instruct model on single-turn tasks (+4.1 on Nonlive-AST, +3.3 on Live-AST) In contrast, the SFT model shows a significant performance drop across all metrics, highlighting KARL's superior training framework.

Despite tackling significantly more complex tool-use patterns, our RL-based framework achieves promising performance in the single-turn (multi-step) setting, showing the robustness and generalization ability of our framework.

G.3 Generalization to challenging spider2.0 benchmark

Table 14 represents the performance of our model on our adapted function-calling version of Spider 2.0—a challenging benchmark that encompasses

Model	AgentBenchDB	BIRD-dev	Spider-1.0
Qwen2.5-14B-Instruct	52.9	59.1	80.2
Qwen2.5-14B-Instruct-SFT (KG only)	43.7	54.6	82.5
Qwen2.5-14B-Instruct-KARL (KG only)	**57.0**	**61.8**	**85.0**

Table 13: Performance of SFT-trained and KARL-trained models on unseen tools. Our model trained in KG with KARL generalizes well on unseen database tasks, while SFT model's performance downgrades.

Model/Setting	Spider 2.0-Snow	Spider 2.0-Lite
GPT-4o-11-20	4.9	6.4
o3-mini-2025-01-31	15.7	26.7
o3-2025-04-16	17.0	22.9
o4-mini-2025-04-16	**19.9**	**29.3**
Qwen2.5-14B-Instruct	2.5	4.4
Qwen2.5-14B-Instruct-GRPO	3.3	6.0
Qwen2.5-14B-Instruct-Multitask	4.4	6.4

Table 14: Generalization performance on the Spider 2.0 benchmark. Despite the highly-challenging nature of the benchmark, our agent trained with KG and DB jointly established stronger generalization performance, closing the gap with proprietary systems.

highly complex and diverse database tasks. Across both the snow and lite splits, reinforcement learning with KARL delivers 30%+ gains over the base Qwen2.5-14B-Instruct, while multitask optimization further elevates performance—reaching 4.4 on Snow and 6.4 on Lite. Notably, this brings our open-source model to within striking distance of GPT-4o (4.9 on Snow, 6.4 on Lite) and narrows the gap with the best proprietary systems like o4-mini. These results highlight how reinforcement learning and multitask strategies improve the generalization and compositional prowess of open models on the toughest database reasoning benchmarks.

H Framework Extensibility Across Diverse Agentic Domains

To further evaluate the extensibility of our framework, we conduct additional experiments on three diverse agentic domains from AgentBench: **AlfWorld**, **WebShop**, and **Operating System**.

As shown in Table 16, our framework demonstrates strong extensibility across heterogeneous agentic environments. Without any domain-specific adaptation, KARL-14B consistently achieves superior performance on AlfWorld and WebShop, and remains competitive on the Operating System domain against strong proprietary models. These results highlight that our framework can generalize effectively to diverse interaction paradigms, including embodied tasks, web-based decision making, and system-level operations.

I Hyperparameters

The detailed hyperparameters used during training is shown in Table 17.

J Training Resources

For training resources (taking multitask training for KG and DB as an example, which is most resource heavy), we used 2 nodes with 16 NVIDlA H800 GPUs. In multi-task training for KG and DB for Qwen-2.5-14B-Instruct, approximately 500 steps were needed for Rl curve convergence. Experimental results show an average update time of 250 seconds per step, totaling 556 GPU hours.

K AI Assistance

AI assisted in the writing process of this paper to ensure reading clarity, which is further augmented by the authors.

Model	bfcl-single-turn-nonlive-ast	bfcl-single-turn-live-ast
Qwen2.5-14B-Instruct	77.4	73.2
Qwen2.5-14B-Instruct-SFT	51.7	71.4
Qwen2.5-14B-Instruct-Multitask (KG&DB)	**81.5**	**76.5**

Table 15: Performance on BFCL-v3 benchmark. Multitask agent trained with KG&DB with our KARL framework demonstrate significant performance gain on much more complex tool-calling tasks, while SFT models suffer from great performance drop.

Model / Setting	**AlfWorld**	**Operating System**	**WebShop**
GPT-4o-2024-11-20	60.0	37.5	65.2
o3-2025-04-16	76.0	27.1	49.9
o4-mini-2025-04-16	61.5	35.4	54.8
Claude-3-7-sonnet-20250219	71.6	43.1	57.8
Claude-3-7-sonnet-20250219-thinking	52.3	**48.6**	63.8
Qwen2.5-14B-Instruct	7.0	18.0	60.6
KARL-14B (Ours)	**92.6**	43.8	**74.3**

Table 16: Performance comparison on additional agentic domains. The agents trained using the extension of KARL framework on three diverse agentic domains achieve competitive results on those environments.

Hyperparameter	Value
Data	
Max Prompt Length	2048
Max Response Length	16384
Train Batch Size	64
Validation Batch Size	64
Seed	42
Actor	
Strategy	FSDP
Mini Batch Size	32768
Micro Batch Size per GPU	1
Use Dynamic Batch Size	True
Max Token Length per GPU	16384
Gradient Clipping	1.0
Clip Ratio	0.2
Entropy Coefficient	0.0
Use KL Loss	True
KL Loss Coefficient	0.001
KL Loss Type	low_var_kl
Epochs	1
Learning Rate	1e-6
Warmup Style	constant
Ulysses Sequence Parallel Size	2
Multi-turn	True
Rollout	
Rollout Type	async
Prompt Length	2048
Response Length	16384
Data Type	bfloat16
GPU Memory Utilization	0.7
Enable Memory Saver	False
Free Cache Engine	True
Load Format	dummy_dtensor
Number of Responses (n)	8
Temperature	0.8
Max New Tokens	1024
Max Turns	40
Enable Chunked Prefill	True
Critic	
Strategy	FSDP
Learning Rate	1e-5
Clip Range Value	0.5
Ulysses Sequence Parallel Size	2
Algorithm	
Gamma	1.0
Lambda	1.0
Advantage Estimator	GRPO
KL Penalty	kl
KL Coefficient	0.01

Table 17: Hyperparameters for RL Algorithm