GAPD: Gold-Action Policy Distillation for Agentic Reinforcement Learning in Knowledge Base Question Answering

Anonymous ACL submission

Abstract

Reinforcement learning (RL) is a natural fit for agentic knowledge base question answering (KBQA), where a model must issue executable actions, observe knowledge-base feedback, and eventually return an answer. However, current RL-based KBQA systems mainly optimize sparse rewards from the final answer, leaving intermediate action errors weakly supervised. This is especially limiting for logical-form annotated KBQA benchmarks: gold logical forms can be converted into executable action sequences, but existing pipelines use them mainly for warm-start data construction rather than for on-policy RL updates. We propose **GAPD**, a training-time **G**old-**A**ction **P**olicy **D**istillation framework that adds dense token-level guidance to outcome-based RL. To align gold actions with on-policy student rollouts, GAPD uses ENTITY-ANCHOR MATCHING: it treats the intermediate entities reached during student exploration and gold execution as state anchors, and matches student states to gold states through these explored entity sets. The current policy conditioned on this aligned gold action serves as a stop-gradient teacher, whose token distribution is distilled back to the ordinary student policy over matched student-generated token spans. GAPD consistently surpasses the current state of the art on WebQSP, GrailQA, and GraphQ. The anonymous code repository is available at `https://anonymous.4open.science/r/GAPD-860A`.

1 Introduction

Knowledge Base Question Answering (KBQA) requires a model to answer natural language questions by navigating a structured knowledge base (KB), such as Freebase (Bollacker et al., 2008), and producing executable logical operations over entities, relations, and constraints (Berant et al., 2013; Yih et al., 2016; Gu et al., 2021).

Prior KBQA work spans semantic parsing (Berant et al., 2013), subgraph-based reasoning (He et al., 2021), and LLM-based logical-form generation (Luo et al., 2024a). Recent agentic formulations instead decompose KBQA into interactions with an executable environment (Xiong et al., 2024; Luo et al., 2025; Chen et al., 2025; Sun et al., 2025). In this setting, the model emits an action, the KB executor returns an observation, and the model continues until it can produce an answer. Compared with one-shot logical-form generation, this action-based formulation can improve robustness because intermediate actions can be executed and checked against the KB environment.

Reinforcement learning (RL) is a natural fit for such agentic KBQA, and has recently become a central post-training method for improving LLM reasoning with verifiable rewards (Schulman et al., 2017; Shao et al., 2024; Guo et al., 2025; Yu et al., 2025). Systems such as KBQA-R1 (Sun et al., 2025) and KnowCoder-A1 (Chen et al., 2025) train policies with outcome-based rewards, typically using answer F1 and format validity, and update the policy with algorithms such as GRPO (Shao et al., 2024). This shifts learning from static trace imitation to verifiable interaction optimization. **However, in outcome-reward GRPO, supervision is attached to the whole rollout: generated tokens in the same response inherit the same group-relative advantage computed from the final answer reward.** When a trajectory fails, this rollout-level signal gives little information about which intermediate action caused the failure or which token should have been changed.

Logical-form annotated KBQA benchmarks already provide executable programs or semantic parses, from which gold action sequences can be derived (Berant et al., 2013; Yih et al., 2016; Gu et al., 2021; Su et al., 2016). Prior KBQA pipelines have used this information for few-shot logical-form demonstrations (Li et al., 2023; Nie et al., 2024), offline program or logical-form supervision (Cao et al., 2022; Shu et al., 2022; Luo et al., 2024a), tra-

jectory synthesis (Luo et al., 2025), or supervised warm-starting (Sun et al., 2025). However, prior RL-based KBQA methods typically discard this supervision during on-policy updates and optimize only final outcome reward (Sun et al., 2025; Chen et al., 2025).

We address this gap by introducing **GAPD** (**G**old-**A**ction **P**olicy **D**istillation), a framework for RL-based KBQA. The core idea is simple: when the student policy reaches an intermediate state, we first decide which gold action is valid for that state, use the aligned action to condition the current model as a teacher, and distill the teacher's token-level preference back into the student policy. This turns gold actions into dense guidance without replacing RL exploration with direct behavior cloning. The teacher is not a separate frozen oracle; it is the current policy conditioned on additional gold-action information. Thus, the distillation signal asks the model to move from its unguided behavior toward its own behavior under valid procedural conditioning. The alignment step is necessary because on-policy rollouts can deviate from the gold trajectory: the next gold action should be determined by the current execution state, not by the raw turn index. GAPD therefore uses ENTITY-ANCHOR MATCHING inside the distillation procedure. It represents each student and gold intermediate state by the entities reached during execution and matches the student state to the most compatible gold state.

Our contributions are:

❶ We identify the token-level credit-assignment problem in agentic KBQA RL policies.

❷ We propose GAPD, which uses ENTITY-ANCHOR MATCHING to align gold actions with student states, constructs gold-action-conditioned self-teacher distributions, and supplies token-level distillation guidance during RL.

❸ We show that GAPD surpasses the current state of the art on WebQSP, GrailQA, and GraphQ.

2 Background

2.1 Agentic KBQA

We follow the action-based, agentic KBQA setting explored by KBQA-O1, KBQA-R1, and KnowCoder-A1 (Luo et al., 2025; Sun et al., 2025; Chen et al., 2025). Given a question q, linked topic entities, and a KB executor, the model generates a sequence of actions such as `Find_relation`, `Merge`, `Order`, `Compare`, `Time_constraint`, and `Count`. Appendix E provides the full action templates and their corresponding logical-form updates. Each action updates an intermediate S-expression state. To obtain KB feedback, the executor translates the current S-expression into SPARQL and executes the SPARQL query against the KB. The environment returns observations containing retrieved entities, relation candidates, execution status, and diagnostic messages. The trajectory terminates when the model emits an answer.

2.2 Outcome-Only RL

Let π_θ be the student policy and $\tau = (a_1, o_1, \ldots, a_T, o_T, \hat{y})$ be a rollout. Prior RL-based KBQA training assigns a trajectory-level reward:

$$R(\tau) = \lambda_{\text{out}} r_{\text{out}}(\hat{y}, y^*) + \lambda_{\text{fmt}} r_{\text{fmt}}(\tau), \qquad (1)$$

where r_{out} measures final answer correctness and r_{fmt} rewards valid output format. GRPO estimates the advantage by comparing multiple sampled rollouts for the same prompt:

$$\hat{A}_i = R(\tau_i) - \frac{1}{n} \sum_{j=1}^{n} R(\tau_j). \qquad (2)$$

Here i indexes a sampled multi-turn response, where a response denotes the whole rollout rather than a single KBQA turn. This objective can improve final answers, but its credit assignment is coarse: all generated tokens are weighted by the same trajectory advantage.

2.3 Gold Actions in Existing Pipelines

Logical-form annotated KBQA datasets provide a gold logical form z^*. An action parser can convert z^* into a gold action sequence $\mathbf{g} = (g_1, \ldots, g_M)$. Prior KBQA pipelines use this supervision in several offline ways. Supervised semantic parsers and LLM-based generators learn from question–program or question–logical-form pairs (Cao et al., 2022; Shu et al., 2022; Ye et al., 2022; Hu et al., 2022; Zhang et al., 2023; Luo et al., 2024a). Few-shot prompting methods instead use labeled logical forms as demonstrations without fine-tuning (Li et al., 2023; Nie et al., 2024). Agentic methods use stepwise traces more directly: KBQA-O1 initializes policy and reward models from annotated trajectories and further fine-tunes them with MCTS-generated annotations (Luo et al., 2025), while KBQA-R1 uses referenced rejection sampling (RRS), where gold actions extracted from

S-expressions guide trajectory synthesis before SFT (Sun et al., 2025). KnowCoder-A1 instead emphasizes outcome supervision and curriculum RL to avoid process-supervised trajectory imitation (Chen et al., 2025). These strategies improve prompting, warm-start quality, or outcome-supervised exploration, but gold actions do not directly affect the later RL policy gradient. Our work uses gold actions during RL itself.

3 Method

3.1 Overview

GAPD augments outcome-based RL for KBQA (Sun et al., 2025; Chen et al., 2025) with dense supervision derived from gold action sequences. For each training question, the dataset provides a gold logical form or its corresponding executable action list, as in standard KBQA semantic-parsing benchmarks (Berant et al., 2013; Yih et al., 2016; Gu et al., 2021; Su et al., 2016). During RL, the student policy still performs on-policy KB interaction and receives the original outcome reward. In addition, for matched intermediate states, GAPD constructs a teacher-side prompt by conditioning the current policy on the aligned gold action, obtains a self-teacher distribution, and distills this distribution back to the student tokens.

Figure 1 contrasts the two supervision signals used in training. The blue path shows the original outcome-RL signal, where the final answer reward is broadcast as a coarse GRPO advantage. The orange and green paths show GAPD: ENTITY-ANCHOR MATCHING first aligns a student state with a gold execution state, the aligned gold action conditions the teacher-side context, the gold-action-conditioned self-teacher scores the sampled student tokens, and the resulting token-level guide is fused with the GRPO advantage before the policy update. The detailed training procedure is provided in Appendix B as Algorithm 1.

3.2 Gold-Action-Conditioned Self-Teacher

As in prior action-based KBQA pipelines (Luo et al., 2025; Sun et al., 2025), we parse the gold logical form z^* to obtain a ground-truth action sequence $\mathbf{g} = (g_1, \ldots, g_M)$, where each g_m is an atomic operation in the action space. We execute this sequence once to collect the corresponding gold execution states u_m and their reached entity sets $E(u_m)$. These gold states are used only for training-time alignment and supervision construction; they are not exposed to the student at inference time. The orange self-teacher path in Figure 1 visualizes this construction: a gold action sequence is paired with intermediate entity anchors, ENTITY-ANCHOR MATCHING selects the aligned state, and the next gold action is inserted only into the teacher-side prompt for scoring.

At a student turn t, let x_t be the ordinary student context containing the original question, previous student actions, and previous KB observations. Let $m^*(t)$ denote the accepted gold-state index selected by ENTITY-ANCHOR MATCHING for the student state s_t. When this match is accepted, the next gold action $g_{m^*(t)+1}$ provides a candidate procedural conditioning signal. We construct a teacher-side prompt $\tilde{x}_t$ by adding this gold action to the same student context:

$$\tilde{x}_t = \mathrm{GoldCond}(x_t, g_{m^*(t)+1}). \tag{3}$$

The added action describes the next valid operation in the action language. It is intentionally procedural: it may reveal the next relation, merge target, or constraint operation, but it does not directly reveal the final answer.

The self-teacher is the current policy conditioned on the gold-action prompt:

$$p_\theta^{\mathrm{teach}}(\cdot \mid \tilde{x}_t) = \mathrm{stopgrad}\left(\pi_\theta(\cdot \mid \tilde{x}_t)\right). \tag{4}$$

The student distribution is the same policy conditioned on the original prompt:

$$p_\theta^{\mathrm{stud}}(\cdot \mid x_t) = \pi_\theta(\cdot \mid x_t). \tag{5}$$

This design follows the on-policy distillation view that the student should receive feedback on tokens sampled from its own current rollout (Agarwal et al., 2024; Gu et al., 2024; Song and Zheng, 2026). Unlike hard behavior cloning, the target is not a single gold string but the model's own distribution under valid procedural conditioning. The student is therefore encouraged to move toward a gold-action-conditioned self-teacher while still being optimized by outcome reward, preserving softer preferences among generated-token alternatives.

3.3 Gold-Action Selection with ENTITY-ANCHOR MATCHING

Gold actions are useful for distillation only when they correspond to the student's current execution state. In an executable KBQA environment, the

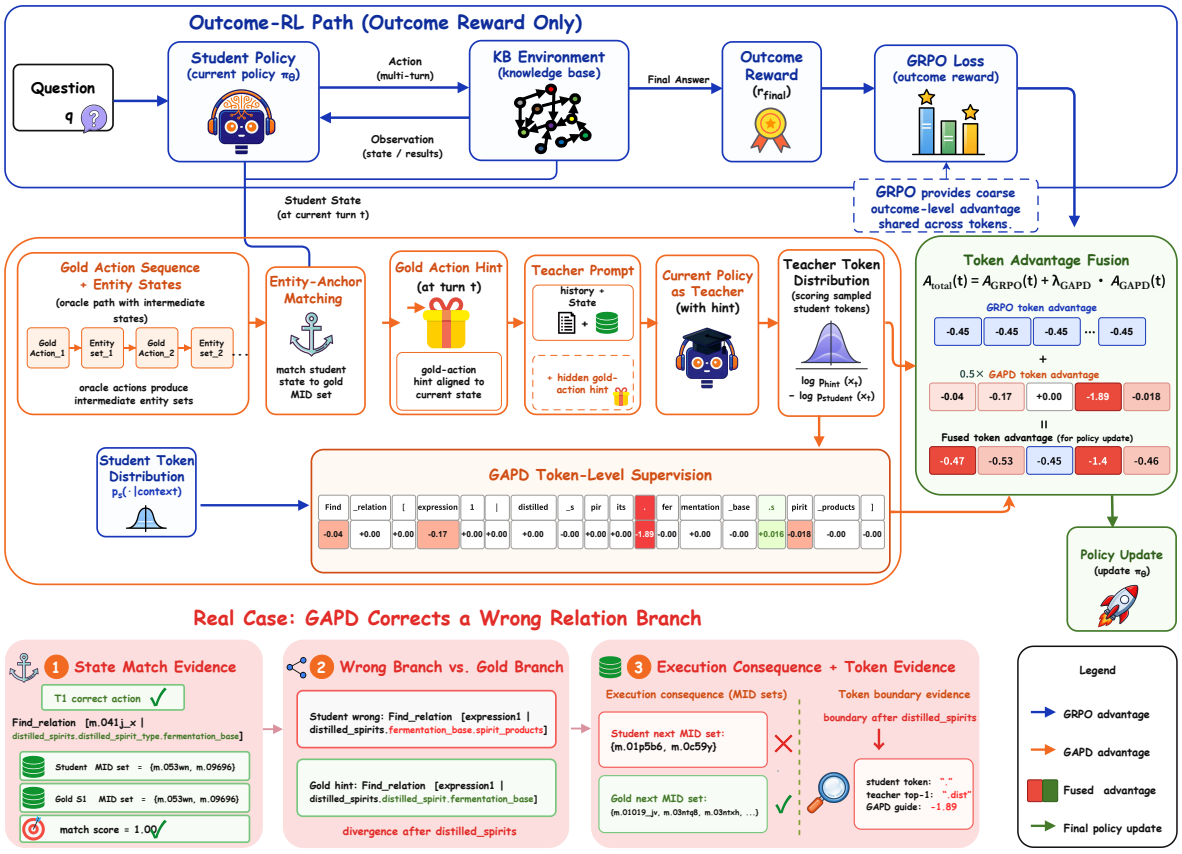


Figure 1: Overview of GAPD. GRPO supplies supervision mainly through the final answer reward. GAPD uses ENTITY-ANCHOR MATCHING to align an on-policy student state with a gold execution state, conditions the current policy on the aligned gold action to obtain a self-teacher distribution, assigns dense guidance to the selected student tokens, and fuses this guidance with the original GRPO advantage.

state after each action determines which next operations are valid: a relation, merge target, or constraint that is correct for one intermediate entity set may be invalid for another. Thus, GAPD must align gold actions by state rather than blindly applying the gold action at the same turn index. ENTITY-ANCHOR MATCHING performs this alignment using the intermediate entity sets reached during student exploration and gold execution. The state-alignment box and the lower real-case panel in Figure 1 show this principle: when the student and gold entity anchors match, the next gold action is treated as the action that should be valid from that state; when execution later diverges, the teacher can penalize the specific relation branch that led to the wrong next entity set. For a student state s_t and a gold state u_m, ENTITY-ANCHOR MATCHING computes the Jaccard similarity over their reached entity sets:

$$\mathrm{sim}(s_t, u_m) = \frac{|E(s_t) \cap E(u_m)|}{|E(s_t) \cup E(u_m)|}. \tag{6}$$

The matched gold state is selected as

$$m^*(t) = \arg\max_m \mathrm{sim}(s_t, u_m). \tag{7}$$

If this match is accepted, the next gold action $g_{m^*(t)+1}$ becomes the conditioning action for the teacher-side prompt. ENTITY-ANCHOR MATCHING is part of training-time teacher construction, not an additional reasoning module at inference time. We apply two conservative filters before assigning token-level guidance. First, we skip answer turns because the teacher conditioning uses a next reasoning action, while an answer turn is already supervised by the final outcome reward. Second, we use a same-next-state filter: if the executed student action reaches the same non-empty next entity set as the aligned gold action, we drop the distillation signal for that turn. This preserves exploration of equivalent KB paths instead of forcing every successful state transition onto the annotated gold surface path.

3.4 Token-Level Fusion with GRPO

GAPD does not optimize a standalone JSD regularizer. Instead, it converts the gold-action-conditioned self-teacher distribution into a dense token-level guide and fuses this guide with the original GRPO policy update (Shao et al., 2024). Let $y_{i,k}$ be the sampled token at position k of response i, and let $\mathcal{I}$ denote the set of selected student-generated token positions after ENTITY-ANCHOR MATCHING matching. In Figure 1, this is the transition from the orange "GAPD Token-Level Supervision" strip to the green "Token Advantage Fusion" box: GAPD produces token-specific scores only on matched generated spans, while the blue GRPO advantage remains the outcome-level signal shared across response tokens. For $k \in \mathcal{I}$, we score the sampled token under the gold-action-conditioned self-teacher and the ordinary student context:

$$d_{i,k} = \log p_\theta^{\text{teach}}(y_{i,k} \mid \tilde{x}_{i,k}) - \log p_\theta^{\text{stud}}(y_{i,k} \mid x_{i,k}). \tag{8}$$

We use this score as a detached, clipped token advantage:

$$A_{i,k}^{\text{gapd}} = \text{clip}\left(\text{sg}(d_{i,k}), -c, c\right), \tag{9}$$

and $A_{i,k}^{\text{gapd}} = 0$ for tokens outside $\mathcal{I}$. A positive value reinforces a sampled token that the gold-action-conditioned self-teacher assigns higher probability than the ordinary student context, while a negative value suppresses a token that the self-teacher disfavors.

Let A_i^{grpo} be the standard group-relative advantage derived from the final answer reward for response i. We form a token-level fused advantage

$$\widehat{A}_{i,k} = A_i^{\text{grpo}} + \lambda_{\text{gapd}} A_{i,k}^{\text{gapd}}. \tag{10}$$

The policy is updated with the GRPO objective over generated response tokens, using the standard PPO-style clipped surrogate (Schulman et al., 2017; Shao et al., 2024):

$$\mathcal{L}_{\text{fuse}} = -\frac{1}{Z} \sum_{i,k} m_{i,k} \min\Big(r_{i,k}(\theta)\widehat{A}_{i,k}, \text{clip}(r_{i,k}(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_{i,k} \Big),$$
$$r_{i,k}(\theta) = \frac{\pi_\theta(y_{i,k} \mid x_{i,k})}{\pi_{\text{old}}(y_{i,k} \mid x_{i,k})}. \tag{11}$$

where $m_{i,k}$ is the response-token mask and $Z = \sum_{i,k} m_{i,k}$. Thus the final-answer reward supplies the shared GRPO signal, while GAPD adds local guidance only on aligned student-generated tokens. The numeric example in Figure 1 illustrates this fusion: a wrong relation segment receives both the coarse negative GRPO advantage and a dense negative GAPD advantage on the divergent relation tokens, yielding a stronger update on the actual error span.

4 Experiments

4.1 Experimental Setup

4.1.1 Datasets

We evaluate on three widely used KBQA benchmarks. **GrailQA** (Gu et al., 2021) is designed to evaluate compositional generalization in KBQA. It contains 64,331 questions, with 44,337 training questions, 6,763 validation questions, and 13,231 test questions. Following prior work (Luo et al., 2024a, 2025), we report results on the dev set and break down performance into *i.i.d.*, *compositional*, and *zero-shot* splits. **WebQSP** (Yih et al., 2016) contains 4,737 questions with semantic parses, split into 3,098 training questions and 1,639 test questions. **GraphQ** (Su et al., 2016) contains 5,166 graph-structured questions, with 2,508 training questions and 2,658 test questions.

4.1.2 Baselines

We organize baselines into prompting and fine-tuning based KBQA methods. **Prompting methods** include KB-BINDER (Li et al., 2023), KB-Coder (Nie et al., 2024), ARG-KBQA (Tian et al., 2024), and Interactive-KBQA (Xiong et al., 2024). These methods rely on in-context learning or tool interaction with strong LLMs. We further include a KBQA-R1 Harness comparison, where GLM-5 (GLM-5 Team, 2026), Kimi-K2.5 (Kimi Team, 2026), gpt-5.3-codex (OpenAI, 2026), and Claude 4.6 Sonnet (Anthropic, 2026) are plugged into the KBQA-R1 interaction harness (Sun et al., 2025). To control the token cost of evaluating recent proprietary models in multi-turn KB interaction, we randomly sample 500 queries for this comparison: 200 GrailQA, 200 WebQSP, and 100 GraphQ examples. We report these results as KBQA-R1 Harness rows to make the comparison with current commercial LLMs explicit. **Fine-tuning based methods** include RnG-KBQA (Ye et al., 2022), DecAF (Yu et al., 2022), TIARA (Shu et al., 2022), MCTS-KBQA (Xiong et al., 2025), KBQA-o1 (Luo et al., 2025), and KnowCoder-A1 (Chen

Table 1: Performance on GrailQA. Bold numbers indicate the best result, and underlined numbers indicate the second-best result.

Method	LLM	I.I.D.		Compositional		Zero-shot		Overall	
		EM	F1	EM	F1	EM	F1	EM	F1
Prompting Methods									
KB-BINDER (Li et al., 2023)	Codex-davinci-002	40.0	43.3	33.9	36.6	40.1	44.0	38.7	42.2
KB-Coder (Nie et al., 2024)	GPT-3.5-turbo	40.6	45.5	34.5	38.6	42.2	47.3	40.1	44.9
ARG-KBQA (Tian et al., 2024)	GPT-3.5-turbo	46.6	51.5	36.4	41.8	46.6	52.1	43.8	48.5
KBQA-R1 Harness (Sun et al., 2025)	GLM-5	46.0	62.1	51.4	64.4	68.7	75.5	60.0	70.2
	Kimi-K2.5	64.0	70.1	57.1	65.7	73.9	78.0	68.5	73.9
	gpt-5.3-codex	55.6	64.3	56.2	65.6	77.4	79.6	68.6	73.5
	Claude 4.6 Sonnet	62.0	68.6	62.9	68.8	79.1	81.9	72.0	76.3
Fine-tune-based Methods									
RnG-KBQA (Ye et al., 2022)	T5-large	86.7	89.0	61.7	68.9	68.8	74.7	69.5	76.9
DecAF (Yu et al., 2022)	T5-large	88.7	92.4	71.5	79.8	65.9	77.3	72.5	81.4
TIARA (Shu et al., 2022)	T5-large	88.4	91.2	66.4	74.8	73.3	80.7	75.3	81.9
KBQA-o1 (Luo et al., 2025)	Llama-3.1-8B	77.8±0.5	85.5±0.4	76.3±0.6	77.6±0.5	68.1±0.8	76.1±0.4	71.9±0.3	78.5±1.0
KnowCoder-A1 (Chen et al., 2025)	Qwen2.5-Coder-7B	–	81.1	–	70.6	–	84.1	–	80.5
KBQA-R1 (Sun et al., 2025)	Llama-3.1-8B	90.0±0.3	91.5±0.2	78.0±0.4	82.5±0.3	83.6±0.3	85.2±0.3	83.9±0.2	86.1±0.3
GAPD	Llama-3.1-8B	**92.8**±0.2	**93.8**±0.2	**82.4**±0.3	**87.0**±0.3	**87.8**±0.2	**88.7**±0.2	**87.8**±0.2	**89.5**±0.2
Rel. improv. over KBQA-R1		*+3.1%*	*+2.5%*	*+5.6%*	*+5.5%*	*+5.0%*	*+4.1%*	*+4.6%*	*+3.9%*

Table 2: Results on WebQSP. Bold numbers indicate the best result, and underlined numbers indicate the second-best result.

Method	LLM	F1
Prompting Methods		
KB-BINDER (Li et al., 2023)	Codex-davinci-002	52.6
KB-Coder (Nie et al., 2024)	GPT-3.5-turbo	55.7
ARG-KBQA (Tian et al., 2024)	GPT-3.5-turbo	58.8
Interactive-KBQA (Xiong et al., 2024)	GPT-4-turbo	71.2
KBQA-R1 Harness (Sun et al., 2025)	GLM-5	64.5
	Kimi-K2.5	73.3
	gpt-5.3-codex	71.1
	Claude 4.6 Sonnet	76.7
Fine-tune-based Methods		
RnG-KBQA (Ye et al., 2022)	T5-large	75.6
DecAF (Yu et al., 2022)	T5-large	76.7
TIARA (Shu et al., 2022)	T5-large	78.9
MCTS-KBQA (Xiong et al., 2025)	Llama-3.1-8B	76.0
KBQA-o1 (Luo et al., 2025)	Llama-3.1-8B	57.8
KnowCoder-A1 (Chen et al., 2025)	Qwen2.5-Coder-7B	77.2
KBQA-R1 (Sun et al., 2025)	Llama-3.1-8B	83.4±0.3
GAPD	Llama-3.1-8B	**86.1**±0.2
Rel. improv. over KBQA-R1		*+3.2%*

Table 3: Results on GraphQ. Bold numbers indicate the best result, and underlined numbers indicate the second-best result.

Method	LLM	F1
Prompting Methods		
KB-BINDER (Li et al., 2023)	Codex-davinci-002	27.1
KB-Coder (Nie et al., 2024)	GPT-3.5-turbo	31.1
KBQA-R1 Harness (Sun et al., 2025)	GLM-5	48.5
	Kimi-K2.5	53.3
	gpt-5.3-codex	51.5
	Claude 4.6 Sonnet	51.8
Fine-tune-based Methods		
SPARQA (Sun et al., 2020)	BERT-base	21.5
BERT+Ranking (Gu et al., 2021)	BERT-base	25.0
ArcaneQA (Gu and Su, 2022)	BERT-base	31.8
CoTKR (Wu et al., 2024)	Llama-3-8B	47.5
KBQA-o1 (Luo et al., 2025)	Llama-3.1-8B	48.7
KBQA-R1 (Sun et al., 2025)	Llama-3.1-8B	53.8±0.7
GAPD	Llama-3.1-8B	**55.1**±0.4
Rel. improv. over KBQA-R1		*+2.4%*

et al., 2025). For GraphQ, we additionally include SPARQA (Sun et al., 2020), BERT+Ranking (Gu et al., 2021), ArcaneQA (Gu and Su, 2022), and CoTKR (Wu et al., 2024). Detailed baseline descriptions are provided in Appendix D.

4.1.3 Evaluation Metrics

We use standard KBQA metrics. **Exact Match (EM)** measures whether the predicted answer set exactly matches the gold answer set. **F1** computes entity-level precision and recall over executed answers. Training configuration details are provided in Appendix A.

4.2 Main Results

Tables 1–3 show that GAPD achieves the best result on every reported benchmark metric. On GrailQA, it raises overall EM/F1 from 83.9/86.1 to 87.8/89.5, corresponding to relative gains of 4.6%/3.9%, and improves every split; the gains are especially clear on compositional and zero-shot questions, where local errors in intermediate relations often determine the final answer. On WebQSP, GAPD improves F1 from 83.4 to 86.1 (+3.2% relative), and on GraphQ from 53.8 to 55.1 (+2.4% relative). The margins are not uniform across datasets, but the direction is consistent: adding token-level guidance during RL improves executed-answer accuracy beyond the outcome-reward base-

Table 4: Ablation study for GAPD.

Variant	WebQSP	GrailQA	GraphQ
GAPD	**86.1**	**89.5**	**55.1**
Outcome reward only	83.4	86.1	53.8
Teacher Variants (Yang et al., 2025)			
Qwen3-32B	84.8	87.2	54.1
Qwen3-14B	83.9	86.2	52.9
Alignment Ablations			
Turn-index alignment (w/o ENTITY-ANCHOR MATCHING)	79.3	81.2	49.2
First-turn only	85.0	87.9	54.7

line. Compared with the strongest commercial LLMs plugged into the KBQA-R1 interaction harness, GAPD is substantially stronger on WebQSP, GrailQA, and GraphQ, despite using an 8B open backbone.

4.3 Ablations and Training Analysis

We next isolate which components support the gains in Tables 1–3. Table 4 compares full GAPD with outcome-only RL, teacher variants, and alignment ablations. Figure 2 provides optimization and interaction-efficiency analysis.

Table 4 shows three ablation patterns. First, outcome reward alone is consistently lower, showing that token-level GAPD supervision provides useful optimization signal beyond the rollout-level reward. Second, Qwen3-32B and Qwen3-14B teacher variants (Yang et al., 2025), both trained with the SFT dataset, improve over outcome-only RL on most datasets but remain below full GAPD. This suggests that policy-consistent gold-action-conditioned self-teacher distributions are more useful than simply using a larger external teacher. Because the self-teacher reuses the current policy under gold-action conditioning, it also avoids loading an additional teacher model and requires much less GPU memory than the Qwen3 teacher variants. Third, alignment is critical: turn-index alignment without ENTITY-ANCHOR MATCHING sharply reduces F1, while first-turn-only supervision recovers much of the score but still trails full GAPD. The latter result is expected because first turns are naturally aligned, but later turns need state-based matching to assign the token-level guide to the correct branch.

Figure 2(a) supports the same interpretation from the optimization side. The reward curves show that adding GAPD leads to higher training rewards while preserving smooth on-policy optimization.

Figure 2(b) evaluates each method as a complete KBQA system by plotting average F1 against the average number of interactions or calls per query. GAPD occupies the upper-left region with 76.9 average F1 and 2.68 interactions per query. This is more accurate and slightly cheaper than the KBQA-R1 baseline (74.4 F1, 2.91 interactions), and it is also substantially stronger than the best recent LLM harness run, Claude 4.6 Sonnet (71.6 F1, 5.45 interactions). Thus the gain is not obtained by spending more environment calls, but by making the learned policy more accurate within a compact action trajectory.

Coefficient sensitivity is reported in Appendix C; performance is stable across $\lambda_{\text{gapd}} = 0.1$–$1.0$.

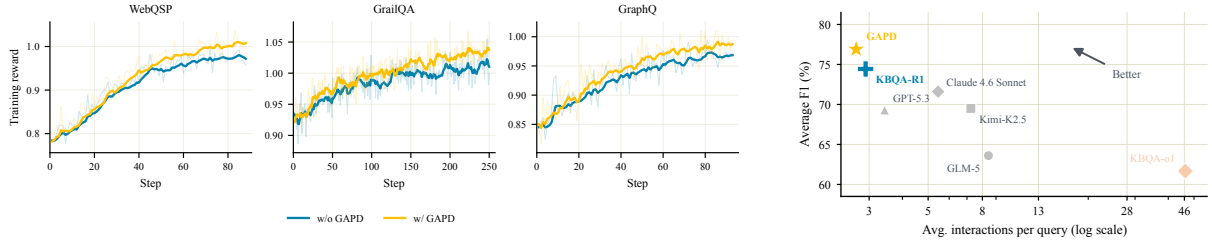
5 Case Study

Figure 3 compresses three dashboard examples into a paper-facing view; full traces and token tables are in Appendix H. Each example shows the student action, the aligned gold action, and the token-level guide on the matched student-generated span.

The three cases illustrate complementary failure modes. In the time-zone example, the student samples a non-canonical daylight-saving relation surface that is repaired by relation retrieval, and the guide concentrates on the divergent relation tokens rather than uniformly across the whole generated turn. In the island comparison example, the question asks for a strict *larger-than* constraint, while the student uses a non-strict comparator; the strongest negative score lands exactly on the comparator token. In the grain-whisky example, ENTITY-ANCHOR MATCHING first matches the intermediate entity state before applying a gold action, so the penalty is assigned to the wrong branch rather than to the already-correct first action.

6 Related Work

Knowledge Base Question Answering. Classic KBQA systems map questions to executable logical forms such as SPARQL or S-expressions and execute them over a KB (Berant et al., 2013; Gu et al., 2021). LLM prompting methods use demonstrations or code-style representations to generate and bind logical forms (Li et al., 2023; Nie et al., 2024). Fine-tuned KBQA generators combine schema retrieval, ranking, or auxiliary grounding objectives with supervised logical-form generation (Shu et al., 2022; Ye et al., 2022; Luo et al., 2024a). Interactive and agentic KBQA systems expose KB operations as tool calls, search actions, or RL rollouts (Xiong et al., 2024; Luo et al., 2025; Sun et al., 2025; Chen



(a) Training reward curves

(b) Interaction efficiency

Figure 2: Training dynamics and interaction efficiency. Left: reward curves comparing w/o GAPD and w/ GAPD runs; each curve plots mean reward (outcome reward + format reward), with faint per-step values and EMA-smoothed values using $\alpha = 0.2$. Right: interaction-efficiency comparison, reporting average F1 over three benchmarks against average interactions or calls per query on a log scale.

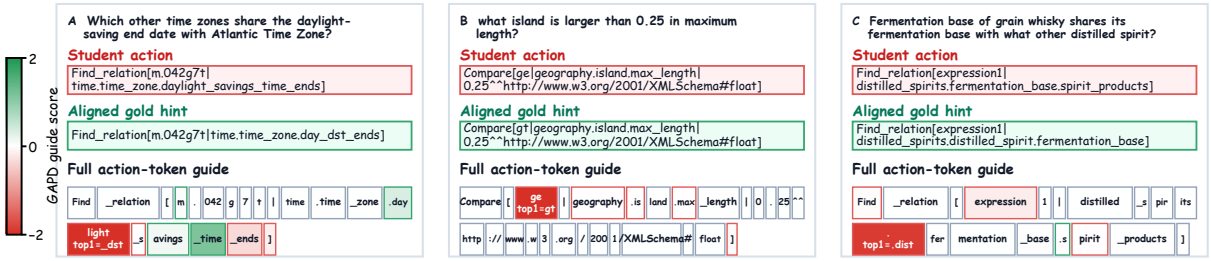


Figure 3: Case studies of GAPD. Each example shows the student action aligned to a gold action and highlights where token-level guidance is applied on the divergent span.

et al., 2025). Subgraph-based methods retrieve a question-specific KB neighborhood before reasoning: earlier systems use graph neural reasoning or iterative retrieval (Sun et al., 2018, 2019; He et al., 2021), while recent retrievers constrain the graph context passed to the reasoner or LLM (Zhang et al., 2022; Li et al., 2025b). Our work focuses on the RL training stage of action-based KBQA and asks how gold logical forms can provide dense supervision during on-policy updates.

RL for LLM Reasoning. RL methods such as PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024) optimize LLM policies using scalar rewards. In reasoning and tool-use tasks, these rewards are often sparse and outcome-based. Recent agentic KBQA work also explores outcome-supervised training without process-level trajectory supervision (Chen et al., 2025). We complement outcome-based RL with token-level supervision derived from executable gold actions.

On-Policy Distillation. Knowledge distillation transfers soft teacher distributions to a student model (Hinton et al., 2015). Classical distillation usually trains on fixed teacher-generated or dataset prefixes, whereas on-policy distillation samples from the student's current policy and obtains feedback on the trajectories the student actually visits (Song and Zheng, 2026). This idea is closely related to interactive imitation learning, where querying an expert on learner-induced states reduces distribution shift between training and deployment (Ross et al., 2011). For LLMs, GKD trains on student-generated outputs with teacher feedback and supports several divergence choices (Agarwal et al., 2024), MiniLLM optimizes reverse KL through an on-policy objective (Gu et al., 2024). GAPD follows the same on-policy premise but targets RL-based KBQA: GRPO samples the student rollout, executable gold logical forms are aligned to the student's execution state, and the added distillation signal is applied only to matched turns.

7 Conclusion

We proposed GAPD, a framework for using gold KBQA actions during on-policy RL. Instead of relying only on final outcome reward, the method conditions the current model on aligned gold actions to produce self-teacher distributions and distills token-level guidance back into the student policy. This provides a path toward dense and stable RL training for agentic KBQA.

Limitations

One practical limitation is training cost. Compared with outcome-reward-only RL, GAPD training is roughly 1.5× slower because the gold-action-conditioned self-teacher must score generated student tokens on supervised spans; Appendix Table 5 summarizes this train-time and test-time tradeoff. This overhead is a training-time tradeoff for finer token-level supervision. It does not introduce an external teacher model or any additional computation at inference time. Instead, the learned policy navigates the knowledge graph more precisely and uses fewer LLM calls per query on average in our experiments, which can make testing faster despite the higher training cost.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. 2024. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*.

Anthropic. 2026. Claude Sonnet 4.6 system card. Official system card.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

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

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

Zhuo Chen, Fei Wang, Zixuan Li, Zhao Zhang, Weiwei Ding, Chuanguang Yang, Yongjun Xu, Xiaolong Jin, and Jiafeng Guo. 2025. Knowcoder-a1: Incentivizing agentic reasoning capability with outcome supervision for kbqa. *arXiv preprint arXiv:2510.25101*.

GLM-5 Team. 2026. GLM-5: From vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*.

Yu Gu, Sue Kase, Michelle T. Vanni, Brian M. Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, WWW '21, page 3477–3488, New York, NY, USA. Association for Computing Machinery.

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

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. Minillm: On-policy distillation of large language models. In *International Conference on Learning Representations*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 553–561, New York, NY, USA. Association for Computing Machinery.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

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

Kimi Team. 2026. Kimi K2.5: Visual agentic intelligence. *Preprint*, arXiv:2602.02276.

Kun Li, Tianhua Zhang, Xixin Wu, Hongyin Luo, James R. Glass, and Helen M. Meng. 2025a. Decoding on graphs: Faithful and sound reasoning on knowledge graphs through generation of well-formed chains. In *Proceedings of the 63rd Annual Meeting of*

the Association for Computational Linguistics (Volume 1: Long Papers), pages 24349–24364, Vienna, Austria. Association for Computational Linguistics.

Mufei Li, Siqi Miao, and Pan Li. 2025b. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. In *International Conference on Learning Representations*.

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

Haoran Luo, Haihong E, Yikai Guo, Qika Lin, Xiaobao Wu, Xinyu Mu, Wenhao Liu, Meina Song, Yifan Zhu, Anh Tuan Luu, and 1 others. 2025. KBQA-o1: Agentic knowledge base question answering with monte carlo tree search. *arXiv preprint arXiv:2501.18922*.

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

Linhao Luo, Yuan-Fang Li, Reza Haffari, and Shirui Pan. 2024b. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*.

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

Zhijie Nie, Richong Zhang, Zhongyuan Wang, and Xudong Liu. 2024. Code-style in-context learning for knowledge-based question answering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):18833–18841.

OpenAI. 2026. Introducing GPT-5.3-Codex. Official product announcement.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 627–635.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv 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. *arXiv preprint arXiv:2402.03300*.

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

Mingyang Song and Mao Zheng. 2026. A survey of on-policy distillation for large language models. *arXiv preprint arXiv:2604.00626*.

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

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2380–2390, Hong Kong, China. Association for Computational Linguistics.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, Brussels, Belgium. Association for Computational Linguistics.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations*.

Xin Sun, Zhongqi Chen, Xing Zheng, Qiang Liu, Shu Wu, Bowen Song, Zilei Wang, Weiqiang Wang, and Liang Wang. 2025. Kbqa-r1: Reinforcing large language models for knowledge base question answering. *arXiv preprint arXiv:2512.10999*.

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. 2020. Sparqa: Skeleton-based semantic parsing for complex questions over knowledge bases. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8952–8959.

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

Yike Wu, Yi Huang, Nan Hu, Yuncheng Hua, Guilin Qi, Jiaoyan Chen, and Jeff Z. Pan. 2024. CoTKR: Chain-of-thought enhanced knowledge rewriting for complex knowledge graph question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3501–3520, Miami, Florida, USA. Association for Computational Linguistics.

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

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

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

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 (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2022. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases. *arXiv preprint arXiv:2210.00063*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

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

Lingxi Zhang, Jing Zhang, Yanling Wang, Shulin Cao, Xinmei Huang, Cuiping Li, Hong Chen, and Juanzi Li. 2023. FC-KBQA: A fine-to-coarse composition framework for knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1002–1017, Toronto, Canada. Association for Computational Linguistics.

A Training Configuration

Backbone. All reported GAPD runs use Llama-3.1-8B-Instruct as the policy backbone.

SFT warm-start and RL. Training uses a two-stage recipe. First, the policy is warm-started by SFT on action trajectories generated from logical-form annotations, including referenced rejection sampling (RRS). Second, the policy is optimized with executable KB feedback using GRPO and token-level GAPD guidance on aligned student-generated token spans. The SFT stage uses a learning rate of 5×10^{-6} with cosine decay. The GRPO stage uses an actor learning rate of 1×10^{-6} and 30% linear warmup. GrailQA is trained for 1 epoch due to its larger training set, while WebQSP and GraphQ are trained for 8 epochs. The train batch size is 256, and we report the final training-step checkpoint.

GRPO configuration. The rollout policy samples $n = 5$ responses per prompt with temperature 1.0 and top-$p = 0.99$. The PPO-style clipping ratios are $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$, following DAPO (Yu et al., 2025). The KL coefficient is $\beta = 0.001$. The reward weights are $\lambda_{\text{outcome}} = 1.0$, $\lambda_{\text{format}} = 0.1$, and $\lambda_{\text{gapd}} = 0.5$; raw GAPD guide scores are clipped to $[-2, 2]$ before scaling. For ENTITY-ANCHOR MATCHING, we accept a matched gold state only when the Jaccard state-match score is at least $\tau_{\text{state}} = 0.95$. The PPO mini-batch size is 128 with dynamic micro-batching enabled.

Infrastructure. Training is conducted on 8 NVIDIA A100-80GB GPUs with FSDP model sharding. The Freebase backend uses Virtuoso with

Method	Train time	Avg. F1	Avg. test calls
Outcome-only RL	8.2h	74.4	2.91
GAPD	12.4h	**76.9**	**2.68**

Table 5: Training-time and test-efficiency tradeoff. GAPD incurs additional training cost because the gold-action-conditioned self-teacher scores student tokens on supervised spans. At test time, no teacher or gold action is used; the trained policy requires fewer LLM calls per query on average while achieving higher average F1. Training time is measured on WebQSP for 88 RL steps; average F1 and test calls are computed over WebQSP, GrailQA, and GraphQ.

Algorithm 1 GAPD Training Step

Require: prompt q, gold actions $\mathbf{g}$, policy π_θ, KB executor $\mathcal{E}$

Roll out the student policy to obtain trajectory τ
Execute student actions and collect entity-anchor states $\{E(s_t)\}$
Execute gold actions and collect entity-anchor states $\{E(u_m)\}$
Initialize distillation mask $\mathcal{I} \leftarrow \emptyset$
for each student turn t **do**
    Match s_t to gold state $u_{m^*(t)}$ using reached entity sets
    Build teacher-side prompt $\tilde{x}_t$ by adding gold action $g_{m^*(t)+1}$
    Add student-generated token positions at turn t to $\mathcal{I}$
end for
Compute GRPO advantages from final answer reward
Score student tokens under the stop-gradient self-teacher
Convert teacher-student log-prob gaps into token-level GAPD advantages on $\mathcal{I}$
Update θ with a GRPO-style clipped loss using fused token advantages

ODBC connection pooling, pool size 48, and query timeout 600 seconds.

B Training Algorithm

Algorithm 1 summarizes one on-policy training step of GAPD. The procedure starts from an ordinary student rollout under the current policy, executes both the sampled student actions and the gold action sequence, and then uses ENTITY-ANCHOR MATCHING to identify student turns whose reached entity states can be aligned to gold states. Only these aligned turns receive gold-action-conditioned self-teacher scores.

C Coefficient Sensitivity

The coefficient sweep shows a useful range around $\lambda_{\text{gapd}} = 0.1$–$1.0$: very small weights underuse the guide, while the gains taper at $\lambda_{\text{gapd}} = 1.0$ compared with the stronger $\lambda_{\text{gapd}} = 0.5$ setting. Overall, performance remains stable across a broad co-

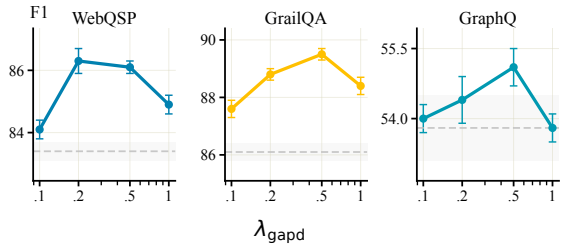


Figure 4: Sensitivity to the GAPD coefficient λ_{gapd}. Each panel reports F1 after varying λ_{gapd}; the dashed gray horizontal line marks $\lambda_{\text{gapd}} = 0$, i.e., the KBQA-R1 outcome-only RL baseline without GAPD.

efficient range, indicating that GAPD is not overly sensitive to this hyperparameter.

Figure 5 provides an additional guide-strength analysis. It reports the share of supervised generated tokens with strong negative raw guide scores. Divergent actions consistently have heavier negative tails, indicating that the token-level guide mainly acts as a localized suppressive correction on generated spans that depart from the gold-action-conditioned self-teacher.

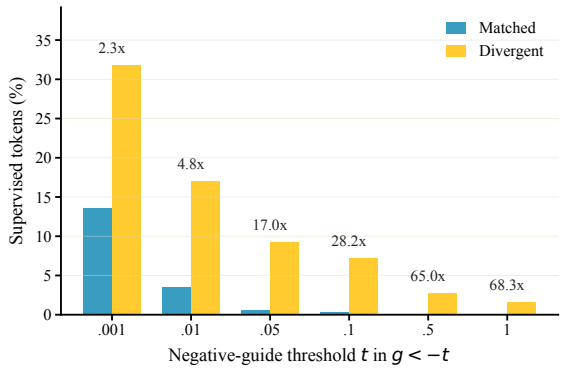


Figure 5: Negative-tail mass of the token-level GAPD guide. Bars show the share of supervised generated tokens whose raw guide score g falls below $-t$.

D Baseline Details

We compare with representative KBQA systems under the settings reported by their original papers or by the corresponding interaction harness. The baselines cover prompting-only systems, fine-tuned semantic parsers, and recent action-based or outcome-supervised KBQA agents.

Prompting methods. These systems use strong LLMs with in-context examples, retrieved KB context, or tool interaction, but do not train a

Action	Arguments	Target Function	Equivalent Logical Form
`Find_relation`	*entity* | *relation*	*expression* = JOIN(‘*relation*’, START(*entity*))	(JOIN *relation entity*)
`Merge`	*expression1* | *expression*	*expression* = AND(*expression1*, *expression*)	(AND (*expression1*) (*expression*))
`Order`	*MAX/MIN* | *expression* | *relation*	*expression* = ARG(‘*mode*’, *expression*, ‘*relation*’)	(*mode* (*expression*) *relation*)
`Compare`	*le/lt/ge/gt* | *relation* | *number*	*expression* = CMP(‘*mode*’, ‘*relation*’, *number*, *expression*)	(*mode relation number* (*expression*))
`Time_constraint`	*relation* | *time*	*expression* = TC(*expression*, ‘*relation*’, ‘*time*’)	(TC (*expression*) *relation time*)
`Count`	*expression*	*expression* = COUNT(*expression*)	(COUNT (*expression*))

Table 6: Action space used in the action-based KBQA setting. Each action is mapped to a functional update over the intermediate expression state and an equivalent logical-form template; the resulting S-expression is later translated into SPARQL for execution.

task-specific policy with executable outcome reward. **KB-BINDER** (Li et al., 2023) uses few-shot logical-form demonstrations to produce a draft program and then binds the draft to executable KB items. **KB-Coder** (Nie et al., 2024) converts logical-form generation into a code-style in-context learning problem to reduce format errors. **ARG-KBQA** (Tian et al., 2024) augments LLM prompting with question-related graph structures retrieved from the KB. **Interactive-KBQA** (Xiong et al., 2024) exposes KB interaction APIs to the LLM and decomposes complex questions through multi-turn tool use. **ToG** (Sun et al., 2024) and **PoG** (Chen et al., 2024) are prompting-based graph reasoning agents that perform test-time graph exploration or planning; their reported Hits@1 results are not mixed into our F1/EM main tables.

KBQA-R1 Harness with commercial LLMs. To test whether recent general-purpose LLMs can solve the task without KBQA-specific RL, we plug GLM-5 (GLM-5 Team, 2026), Kimi-K2.5 (Kimi Team, 2026), gpt-5.3-codex (OpenAI, 2026), and Claude 4.6 Sonnet (Anthropic, 2026) into the KBQA-R1 interaction harness (Sun et al., 2025). These runs use the same action-style KB interface and executor, but the underlying LLMs are not fine-tuned on our training data. Because multi-turn KB execution with proprietary models is expensive, we evaluate this harness comparison on a 500-example subset containing 200 GrailQA, 200 WebQSP, and 100 GraphQ queries.

Fine-tuning and agentic KBQA methods. **RnG-KBQA** (Ye et al., 2022) first ranks candidate logical forms retrieved from KB search and then generates the final logical form conditioned on the ranked candidates. **DecAF** (Yu et al., 2022) jointly decodes answers and logical forms for KBQA. **TIARA** (Shu et al., 2022) combines multi-grained retrieval over entities, schema items, and exemplary logical forms with constrained decoding. **MCTS-KBQA** (Xiong et al., 2025) uses Monte Carlo tree search with step-wise rewards to explore KBQA reasoning paths with open-source instruction LLMs. **KBQA-o1** (Luo et al., 2025) is an agentic KBQA framework that uses Monte-Carlo tree search with policy and reward models to explore KB action trajectories. **KnowCoder-A1** (Chen et al., 2025) trains an agentic KBQA policy with outcome supervision and curriculum-style optimization. **KBQA-R1** (Sun et al., 2025) is our primary outcome-reward-only RL baseline: it uses the same Llama-3.1-8B backbone, action space, Freebase executor, and SFT warm-start recipe, but optimizes GRPO with final-answer and format rewards without token-level GAPD guidance.

GraphQ-specific baselines. For GraphQ, we additionally include established semantic-parsing and graph-reasoning baselines reported in prior KBQA work. **SPARQA** (Sun et al., 2020) uses skeleton-based semantic parsing for complex KB questions. **BERT+Ranking** (Gu et al., 2021) ranks candidate parses with a BERT-based model. **ArcaneQA** (Gu and Su, 2022) combines generation with dynamic program induction and contextual schema encoding. **CoTKR** (Wu et al., 2024) uses chain-of-thought knowledge rewriting to convert retrieved KB evidence into reasoning-friendly text.

E Action Space

Following the action-based KBQA setting, we decompose logical-form construction into a small set of executable operations over an evolving intermediate expression. At each turn, the model emits an action inside the structured action block; the executor parses the action, updates the corresponding functional expression, and returns KB feedback that is appended to the next-turn context. This design avoids requiring the policy to generate a complete nested S-expression in one shot, while still keeping every intermediate step convertible to an executable logical form. Table 6 lists the action templates used in our rollouts. The first

four columns specify the surface action, its arguments, the functional state update, and the equivalent logical-form fragment. In training, GAPD applies token-level guidance only to matched student-generated spans; environment feedback remains context rather than a supervised target.

F Prompt Templates

Tables 7 and 8 show the two prompt fragments used during training. The student rollout prompt defines the action protocol, while the teacher-only reference block is inserted only when computing GAPD token guidance for a matched student state. At inference time, the teacher-side block is never used.

You are an expert assistant for querying the Freebase knowledge base using structured reasoning actions.
Answer the given question about Freebase knowledge base.
You MUST conduct reasoning inside `<think>...</think>` before emitting actions.
After reasoning, provide structured actions inside `<action>...</action>`.
The system will return query results between `<information>...</information>`.
When ready, return the final answer inside `<answer>...</answer>` using MIDs or literal values.
`Available Actions : {The Description of Actions}`
Question: {QUESTION}.

Table 7: Action-based reasoning prompt template for the student policy.

Student context before the matched generation turn.
`. . . dialogue history, actions, and KB observations . . .`

Teacher-only reference block
`[Reference action sequence. Do not mention or repeat this in your response, just silently use it as guidance.]`
`{matched_gold_action}`
`[End reference]`

Assistant-turn prefix from the chat template; teacher logits are evaluated on the original student response tokens after this point.

Table 8: Teacher-only prompt augmentation for computing GAPD token guidance. The full dataset-specific action descriptions are inserted at the `{The Description of Actions}` placeholder and summarized in Table 6.

G Additional Subgraph-Based Baselines

Table 9 provides a focused comparison with subgraph-based KBQA baselines, including RoG, GNN-RAG, and SubgraphRAG. These methods retrieve or construct question-specific graph neighborhoods and perform reasoning over the resulting subgraph, so we report them separately as additional graph-context baselines. We keep the comparison in the appendix because the available comparable results are WebQSP-only and therefore do not match the cross-dataset F1/EM setting of the main tables. Among the listed subgraph-based baselines, SubgraphRAG with GPT-4o reports the strongest WebQSP F1 of 78.2; the outcome-only KBQA-R1 baseline reaches 83.4, and GAPD further improves to 86.1. DoG (Li et al., 2025a) is discussed in related work but omitted from the table because we could not identify a directly comparable WebQSP F1 result.

Method	Backbone	WebQSP F1
RoG (Luo et al., 2024b)	Llama2-7B	70.8
GNN-RAG (Mavromatis and Karypis, 2025)	Llama2-7B	71.3
SubgraphRAG (Li et al., 2025b)	Llama-3.1-8B	70.6
SubgraphRAG (Li et al., 2025b)	GPT-4o	78.2
KBQA-R1 (Sun et al., 2025)	Llama-3.1-8B	83.4
GAPD	Llama-3.1-8B	**86.1**

Table 9: Additional WebQSP comparison with graph/-subgraph reasoning baselines.

H Full Student Traces and Teacher Token Scores

We expand the three dashboard case studies into full appendix trajectory views. The aligned gold action is visible only to the self-teacher and is used only to compute token-level guide scores for safely matched spans; red and green cells mark negative and positive guide values. Figures 6–8 provide the full traces for the time-zone relation, strict-comparison, and distilled-spirits cases from the main text.

I AI Assistant Use

We used AI assistants to help modify experimental and analysis code and to polish manuscript wording. The authors reviewed and verified all generated code, experiments, analyses, and final paper text.

C1. could you provide me with the time zones in which day dst ends on the same day as that in atlantic time zone?

Turn 1 | matched gold step 0

Student output

```
<think> To find the time zones where the day of DST (Daylight Saving Time) ends is the same as in the Atlantic Time Zone, I need to first
identify the days when DST ends in the Atlantic Time Zone. </think><action> Find_relation [ m.042g7t | time.time_zone.daylight_savings_time_ends
] </action>
```

Gold-conditioned hint

```
Find_relation [ m.042g7t | time.time_zone.day_dst_ends ]
```

Full matched-turn token scores (think + action)

| | | |

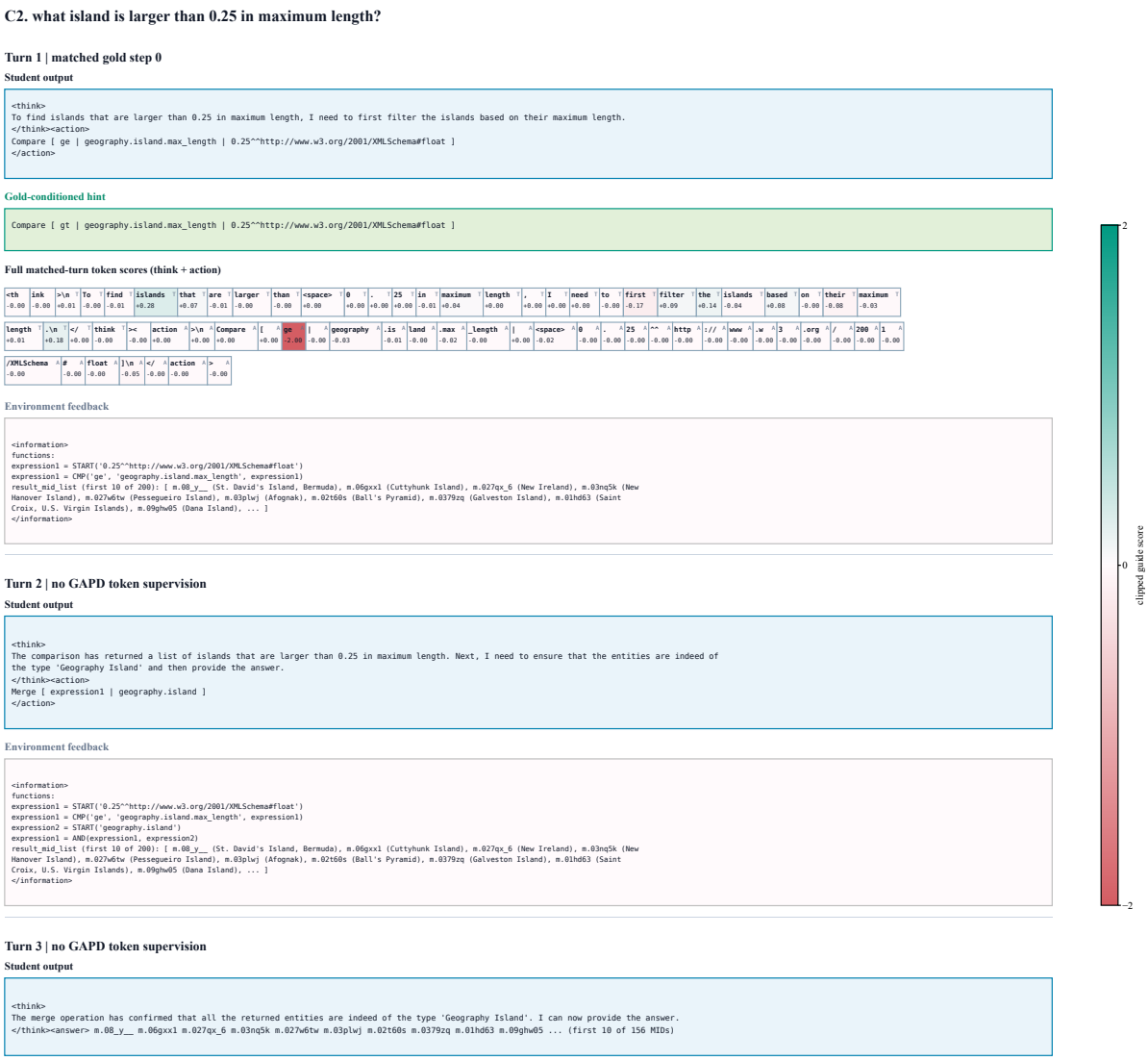


Figure 7: Full trajectory view for the island comparison case. The figure preserves the complete student–environment interaction chain and overlays think-plus-action token scores for the supervised comparison turn. For readability, long `result_mid_list` fields and answer MID lists are truncated to the first ten entries, with ellipses marking omitted MIDs.

C3. fermentation base of grain whisky shares it's fermation base with what other distilled spirit?

Turn 1 | matched gold step 0

Student output

```
<think> To find the fermentation base of grain whisky and check if it shares the fermentation base with any other distilled spirits, I need to
first identify the fermentation base of grain whisky. </think><action> Find_relation [m.041j_x |
distilled_spirits.distilled_spirit_type.fermentation_base] </action>
```

Gold-conditioned hint

```
Find_relation [ m.041j_x | distilled_spirits.distilled_spirit_type.fermentation_base ]
```

Full matched-turn token scores (think + action)

| | | |