REWARD ON PATH: LEARNING INTERMEDIATE SUPERVISION SIGNALS FOR KNOWLEDGE GRAPH QUESTION ANSWERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge Graph Question Answering (KGQA) aims to answer user questions by reasoning over Knowledge Graphs (KGs). Recent methods use supervision derived from answer labels or refined by Large Language Models (LLMs) to train models that retrieve KG evidence for LLM-based answer reasoning. However, answer-derived supervision treats every answer-reaching path as correct and thus yields noisy training signals, whereas LLM-refined supervision mitigates this noise at substantial cost. To address these limitations, we propose Reward on Path (ROP), a framework to learn a lightweight, question-conditioned path reward from answer labels with an asymmetric objective. Paths reaching the same answer are supervised jointly as a bag, allowing the reward model to learn their relative contributions, while each path is penalized individually for retrieving non-answer entities. The learned reward then trains an LLM-based relation path generator in two stages: reward distillation transfers reward-induced preferences over candidate paths into the generator, and on-policy optimization with GRPO further refines the policy on self-generated paths. Generated paths are grounded in the KG to retrieve evidence for answer reasoning. Experiments on multiple KGQA datasets show that ROP improves F1 over answer-derived methods by at least 2.6% while outperforming LLM-refined methods without costly supervision construction.

1 INTRODUCTION

Knowledge Graph Question Answering (KGQA) aims to answer users' natural language questions by reasoning over Knowledge Graphs (KGs) (Lan et al., 2023; Pan et al., 2024). It provides a user-friendly interface for accessing abundant factual knowledge in structured formats stored in large-scale KGs such as Freebase (Bollacker et al., 2008), Wikidata (Vrandečić & Krötzsch, 2014), and DBpedia (Auer et al., 2007). Recent KGQA methods augment Large Language Models (LLMs) with retrieved KG evidence through KG-based Retrieval-Augmented Generation (KG-RAG) methods.

To obtain relevant evidence, existing KG-RAG methods train models to retrieve KG components directly (Li et al., 2025; Mavromatis & Karypis, 2025) or to generate reasoning paths that guide evidence retrieval (Luo et al., 2024; 2025b). Training such models typically requires intermediate supervision that identifies relevant KG evidence, such as supporting triples, reasoning paths, or subgraphs. This exposes a fundamental challenge: **the lack of reliable intermediate supervision signals**. Obtaining gold annotations for such evidence is time- and resource-intensive, making reliable intermediate supervision difficult to obtain at scale (Gu et al., 2021; Li et al., 2023a; 2024).

One line of work derives intermediate supervision directly from answer labels (Li et al., 2025; Luo et al., 2024; 2025b; Ma et al., 2025b; Lu et al., 2026). Reasoning paths that reach ground-truth answers are typically treated as positive training signals. While scalable, this criterion can retain *spurious* paths that reach the correct answer without reflecting the reasoning process required by the question (Lin et al., 2018; Yao et al., 2025). For example, in Figure 1, both relation paths "*nationality*" and "*sibling* $\rightarrow$ *nationality*" reach the correct answers, although the former does not fully reflect the relations expressed in the question. Consequently, treating all answer-reaching paths as positive can introduce noisy supervision for training retrieval models, thereby degrading evidence retrieval and downstream answer reasoning.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

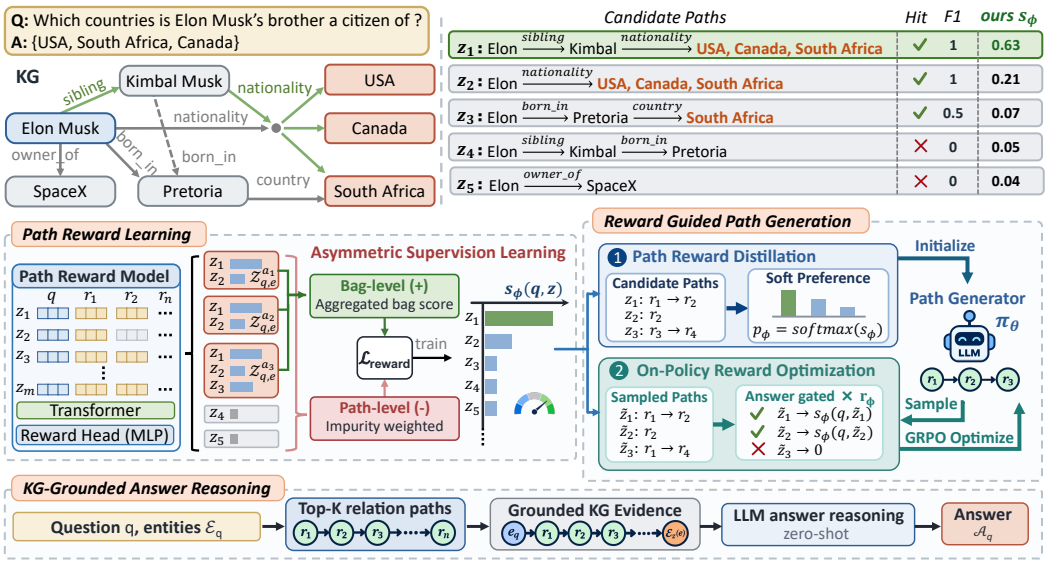


Figure 1: Motivating example and overview of ROP. Top: an illustrative KG subgraph shows that several relation paths reach the correct answers, but only one of them reflects the relations expressed in the question. Bottom: ROP consists of three stages: (1) path reward learning from answer labels, (2) reward-guided path generation through path reward distillation and on-policy reward optimization, and (3) KG-grounded answer reasoning over evidence retrieved by the generated paths.

Another line of work employs LLMs to construct or refine intermediate supervision. For example, LLMs can be used to select relevant paths from answer-reaching paths or assign rewards to generated paths (Yao et al., 2025; Zou et al., 2025; Wang et al., 2026; Luo et al., 2025a; Wang & Yu, 2026). Although LLM-refined supervision can reduce noise in answer-derived signals and provide more informative intermediate supervision, constructing such supervision typically requires repeated LLM inference over training instances, leading to substantial computational cost and limiting scalability.

To this end, we propose a framework called **R**eward **o**n **P**ath (ROP), which learns path rewards from answer labels to train an LLM-based relation path generator. ROP trains a lightweight question-conditioned reward model with an asymmetric objective that combines bag-level positive supervision with path-level penalties. Intuitively, the bag-level objective allows the reward model to learn and score each candidate path, *without* making the assumption that all answer-reaching paths are equally valid like prior methods. The path-level penalty, on the other hand, penalizes a path if it produces answers that are not part of the ground-truth answer set. Together, these signals enable the reward model to learn the relative utility of candidate paths for answering the question.

The learned reward model provides intermediate supervision for path generation training in two stages. *Path reward distillation* aligns the generator with the soft preferences induced by the learned rewards over candidate paths. *On-policy reward optimization* applies the same learned reward to self-generated paths and updates the policy through Group Relative Policy Optimization (GRPO). Intuitively, these two training schemes are a variation of the standard supervised fine-tuning (SFT) and GRPO training, where we use the learned reward to derive soft targets over candidate paths for the former and to score generated paths in the latter. The resulting path generator produces relation paths that guide KG evidence retrieval, providing grounded context for downstream answer reasoning. The main contributions of this work are summarized as follows:

- We propose ROP, which learns question-conditioned path rewards from answer labels using an asymmetric objective that combines bag-level positive supervision and path-level penalties.
- We develop a two-stage training scheme that applies the learned reward to reward distillation and on-policy optimization, extending reward guidance from candidate paths to self-generated paths.
- Experiments on multiple datasets show that ROP improves F1 over answer-derived methods by at least 2.6%, while outperforming LLM-refined methods. Further retrieval, ablation, and supervision quality analyses validate the learned path rewards and the two-stage training strategy.

2 Related Work

Knowledge Graph Question Answering (KGQA). Recent KGQA methods combine LLMs with structured KG evidence through different retrieval and reasoning strategies. KG-RAG methods either employ graph retrievers to directly identify relevant KG components (Li et al., 2025; Mavromatis & Karypis, 2025), or generate relation paths that serve as retrieval plans for subsequently grounding relevant evidence from the KG (Luo et al., 2024; 2025b; Ma et al., 2025b). Agent-based methods iteratively interact with the KG, dynamically exploring relevant entities and relations through multi-step reasoning and graph traversal (Sun et al., 2024; Chen et al., 2024; Luo et al., 2025a; Wang et al., 2026). Our work follows the KG-RAG paradigm with relation path generation and focuses on learning reliable intermediate supervision used to train the generator.

Intermediate Supervision for KGQA. Existing methods obtain intermediate supervision either from answer labels or from LLM feedback. The former line treats every relation path connecting a question entity to an annotated answer as a positive training instance (Li et al., 2025; Luo et al., 2024; 2025b; Ma et al., 2025b); RPO-RAG (Um et al., 2026) further ranks such answer-reaching paths by their similarity to the question under a pretrained encoder. The latter line invokes LLMs at training time: RAPL (Yao et al., 2025) and ReG (Zou et al., 2025) use LLM feedback to filter spurious paths, DAMR (Wang et al., 2026) and KBQA-o1 (Luo et al., 2025a) use LLM-guided search to construct or refine reasoning trajectories, and KG-Hopper (Wang & Yu, 2026) uses an LLM to reward generated paths during GRPO training.

RoP also learns from answer labels but differs from answer-derived approaches in how intermediate supervision is derived and used. It learns a question-conditioned path reward with an asymmetric objective to estimate the relative utility of candidate paths, and uses it to supervise both path reward distillation and on-policy reward optimization, reducing supervision noise without LLM feedback.

3 Preliminary

Knowledge Graphs (KGs) are composed of relational facts in the form of a set of triples: $\mathcal{G} = \{\langle e_h, r, e_t\rangle \mid e_h \in \mathcal{E}, r \in \mathcal{R}, e_t \in \mathcal{E}\}$, where $\mathcal{E}$ denotes the set of entities and $\mathcal{R}$ denotes the set of relations. In each triple $\langle e_h, r, e_t\rangle$, $e_h, e_t \in \mathcal{E}$ denote the head and tail entities, respectively, and $r \in \mathcal{R}$ represents the relation between them.

Knowledge Graph Question Answering (KGQA) is a reasoning task over KGs. Given a KG $\mathcal{G}$ and a question $q = (w_1, w_2, \ldots, w_n)$ with linked entities $\mathcal{E}_q \subseteq \mathcal{E}$, where w_i is the i-th token, the objective is to learn a function f that maps $(q, \mathcal{E}_q, \mathcal{G})$ to an answer set $\mathcal{A}_q \subseteq \mathcal{E}$, i.e., $\mathcal{A}_q = f(q, \mathcal{E}_q, \mathcal{G})$.

Relation Path is an ordered sequence of relations $z = \langle r_1, r_2, \ldots, r_l\rangle$, where $r_i \in \mathcal{R}$ denotes the i-th relation and l denotes the path length. It defines a compositional relation that connects a source entity to a set of target entities by following the relations in order (example provided in Appendix A). Given an entity e and a relation path z, we denote the set of entities reachable from e via z:

$$\mathcal{E}_z(e) = \left\{ e_l \in \mathcal{E} \mid e \xrightarrow{r_1} e_1 \xrightarrow{r_2} \cdots \xrightarrow{r_l} e_l \text{ in } \mathcal{G} \right\}.$$

4 Methodology

As illustrated in Figure 1, RoP consists of three main stages: (1) *path reward learning*, which learns a lightweight question-conditioned reward model s_ϕ from answer labels; (2) *reward-guided relation path generation*, which uses the learned reward to train an LLM-based relation path generator π_θ through path reward distillation and on-policy reward optimization; and (3) *KG-grounded answer reasoning*, which grounds the generated paths in the KG to retrieve evidence and employs an LLM answer generator p_ψ to answer the question via zero-shot prompting.

4.1 Path Reward Learning

To obtain reliable intermediate supervision, we learn a lightweight path reward model from answer labels to estimate the utility of candidate relation paths for answering the question. Formally, given a question q and a candidate relation path z, the model assigns a scalar reward score $s_\phi(q, z) \in \mathbb{R}$.

For each question entity $e \in \mathcal{E}_q$, we construct a set $\mathcal{Z}_{q,e} = \{z_i\}_{i=1}^{N}$ by breadth-first search from e up to L hops, retaining all answer-reaching paths and randomly sampling from the remaining paths. We exclude question–entity pairs with no answer-reaching candidate paths. Each path z_i retrieves a set of end entities $\mathcal{E}_{z_i}(e)$ when grounded in the KG. Sampling details are provided in Appendix E.

Path Reward Modeling. We parameterize $s_\phi(q, z)$ with a lightweight Transformer encoder that captures question-dependent path utility by jointly encoding the question and the relation sequence. For a path $z = \langle r_1, r_2, \ldots, r_l \rangle$, a frozen text encoder produces a question embedding $\mathbf{h}_q \in \mathbb{R}^d$ and relation embeddings $\{\mathbf{h}_{r_i}\}_{i=1}^{l}$. We add learnable positional embeddings $\{\mathbf{p}_j\}_{j=0}^{l}$ and feed the resulting sequence into the Transformer. The output at the question position serves as a question-conditioned path representation, which an MLP reward head maps to a scalar reward score:

$$\mathbf{H}_{q,z} = \mathrm{Transformer}_\phi([\mathbf{h}_q + \mathbf{p}_0,\, \mathbf{h}_{r_1} + \mathbf{p}_1,\, \ldots,\, \mathbf{h}_{r_l} + \mathbf{p}_l])\,, \qquad s_\phi(q, z) = \mathrm{MLP}_\phi\Big(\mathbf{H}^{(0)}_{q,z}\Big)\,. \tag{1}$$

The trainable parameters ϕ include the Transformer encoder, positional embeddings, and MLP, while the text encoder remains frozen.

Asymmetric Reward Learning. Paths reaching the same answer may not all reflect the relations expressed in the question. We therefore group these paths into an answer-specific bag and supervise its aggregate prediction rather than each path individually, allowing the reward model to learn their relative contributions. For each annotated answer $a \in \mathcal{A}_q$, we define $\mathcal{Z}^a_{q,e} = \{z_i \in \mathcal{Z}_{q,e} \mid a \in \mathcal{E}_{z_i}(e)\}$ and denote the answers with nonempty bags by $\mathcal{A}^+_{q,e} = \{a \in \mathcal{A}_q \mid \mathcal{Z}^a_{q,e} \neq \emptyset\}$.

For each $a \in \mathcal{A}^+_{q,e}$, the reward scores define attention weights over the paths in its bag, and the bag prediction is the attention-weighted average of the individual path predictions:

$$\omega^a_{q,e}(z_i) = \frac{\exp\left(s_\phi(q, z_i)\right)}{\sum_{z_j \in \mathcal{Z}^a_{q,e}} \exp\left(s_\phi(q, z_j)\right)}, \qquad \hat{y}^a_{q,e} = \sum_{z_i \in \mathcal{Z}^a_{q,e}} \omega^a_{q,e}(z_i)\, \sigma\left(s_\phi(q, z_i)\right). \tag{2}$$

Here, $\sigma(\cdot)$ denotes the sigmoid function, and $\sigma(s_\phi(q, z_i))$ is the individual prediction for path z_i.

To complement this aggregate supervision, we penalize individual paths according to their *answer impurity*, defined as the fraction of retrieved entities outside the annotated answer set:

$$\xi_{z_i} = 1 - \frac{|\mathcal{E}_{z_i}(e) \cap \mathcal{A}_q|}{|\mathcal{E}_{z_i}(e)|}. \tag{3}$$

We use answer impurity to weight the penalties on individual paths and combine them with positive supervision on answer-specific bags, yielding the following asymmetric objective:

$$\mathcal{L}_{\mathrm{score}} = \underbrace{-\frac{1}{|\mathcal{A}^+_{q,e}|} \sum_{a \in \mathcal{A}^+_{q,e}} \log \hat{y}^a_{q,e}}_{\text{Bag-level positive supervision}} \underbrace{- \frac{1}{|\mathcal{Z}_{q,e}|} \sum_{z_i \in \mathcal{Z}_{q,e}} \xi_{z_i} \log\left(1 - \sigma\left(s_\phi(q, z_i)\right)\right)}_{\text{Path-level penalty}}. \tag{4}$$

The objective is asymmetric in its supervision granularity: positive supervision applies to answer-specific bags, whereas impurity-weighted penalties apply to individual paths. This formulation avoids treating every answer-reaching path as a positive instance while penalizing the retrieval of non-answer entities. By learning question–path interactions across training examples and the relative contributions of paths within each bag, the shared reward model can effectively distinguish paths with otherwise identical answer-derived supervision and provide more informative rewards for path generation. Section 5.4 empirically validates the quality of the learned supervision.

4.2 Reward-Guided Relation Path Generation

The learned reward model s_ϕ provides supervision for training the relation path generator π_θ in two stages and remains frozen throughout. *Path reward distillation* fits the generator to soft preferences induced by the reward model over answer-reaching candidate paths. *On-policy reward optimization* then uses the same model to provide graded feedback on paths sampled from the current policy, guiding further refinement through GRPO. In both stages, the generator is trained to produce a relation path given the question and source entity, following the template in Figure 3.

Path Reward Distillation. For each question–entity pair (q, e), we convert the learned reward scores into a soft target distribution over candidate paths that reach at least one answer:

$$p_\phi(z_i \mid q, e) = \frac{\exp\left(s_\phi(q, z_i)\right)}{\sum_{z_j \in \mathcal{Z}_{q,e}^{\mathcal{A}_q}} \exp\left(s_\phi(q, z_j)\right)}, \qquad \text{where} \quad \mathcal{Z}_{q,e}^{\mathcal{A}_q} = \bigcup_{a \in \mathcal{A}_{q,e}^{+}} \mathcal{Z}_{q,e}^{a}. \tag{5}$$

Let $\pi_\theta(z \mid q, e) = \prod_{t=1}^{T_z} \pi_\theta(x_t \mid q, e, x_{<t})$ denote the autoregressive probability of a relation path z, where $(x_1, \ldots, x_{T_z})$ is the token sequence representing z. With p_ϕ defined as zero outside the answer-reaching candidate set, we fit the generator to the reward-induced target distribution by minimizing the forward KL divergence:

$$\begin{aligned} \mathcal{L}_{\text{distill}} &= D_{\text{KL}}(p_\phi(\cdot \mid q, e) \,\|\, \pi_\theta(\cdot \mid q, e)) \\ &= -\sum_{z_i \in \mathcal{Z}_{q,e}^{\mathcal{A}_q}} p_\phi(z_i \mid q, e) \log \pi_\theta(z_i \mid q, e) + \text{const}. \end{aligned} \tag{6}$$

Here, the constant is the negative entropy of p_ϕ, which is independent of θ and can therefore be omitted during optimization. Path reward distillation trains the generator on relation sequences extracted from the KG, weighting each sequence by its reward-induced probability rather than uniformly as in standard SFT over answer-reaching candidates. This provides a supervised initialization for subsequent GRPO training that incorporates both KG relation patterns and learned path preferences.

On-Policy Reward Optimization. Starting from the distilled generator, we further optimize path generation with Group Relative Policy Optimization (GRPO) (Shao et al., 2024). We construct the reward by scaling a binary answer-reaching signal using the same learned reward model, providing graded feedback on generated paths. For each question–entity pair (q, e), we sample a group of relation paths $\{\tilde{z}_i\}_{i=1}^{G}$ using the old policy $\pi_{\theta_{\text{old}}}$ and assign each path the following reward:

$$u_i = \mathbb{I}\left[\mathcal{E}_{\tilde{z}_i}(e) \cap \mathcal{A}_q \neq \emptyset\right] \sigma\left(s_\phi(q, \tilde{z}_i)\right), \tag{7}$$

where $\mathbb{I}[\cdot]$ denotes the indicator function. It restricts positive rewards to paths reaching an annotated answer, while the learned scores provide graded feedback on their utility for answering the question.

Let $\tilde{x}_{i,t}$ denote the t-th token of the generated path $\tilde{z}_i$. Following GRPO, we normalize rewards within each group to obtain relative advantages and compute token probability ratios between the updated and old policies as follows:

$$\hat{A}_i = \frac{u_i - \text{mean}\left(\{u_j\}_{j=1}^{G}\right)}{\text{std}\left(\{u_j\}_{j=1}^{G}\right) + \epsilon_{\text{std}}}, \qquad \rho_{i,t}(\theta) = \frac{\pi_\theta(\tilde{x}_{i,t} \mid q, e, \tilde{x}_{i,<t})}{\pi_{\theta_{\text{old}}}(\tilde{x}_{i,t} \mid q, e, \tilde{x}_{i,<t})}. \tag{8}$$

The generator is then updated by minimizing the clipped GRPO objective without KL regularization:

$$\mathcal{L}_{\text{GRPO}} = -\mathbb{E}_{(i,t)}\left[\min\left(\rho_{i,t}(\theta)\hat{A}_i,\ \text{clip}\left(\rho_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_i\right)\right], \tag{9}$$

where the expectation averages over all generated tokens in the training batch, $\epsilon_{\text{std}} > 0$ ensures numerical stability when normalizing group rewards, and ε controls the clipping range of the policy probability ratio. Sampling settings and optimization hyperparameters are detailed in Appendix E.

Overall, the two stages apply the same learned reward to different path sources: path reward distillation weights KG-extracted candidate paths by their reward-induced probabilities to initialize the generator, and on-policy reward optimization scores self-generated paths to refine it. The learned reward thus guides both the initialization over candidate paths and the subsequent refinement over self-generated ones, prioritizing paths by their estimated utility rather than by answer outcomes.

4.3 KG-Grounded Answer Reasoning

At inference time, the trained generator π_θ produces K candidate relation paths for each question entity $e \in \mathcal{E}_q$ using beam search. We ground each path in the KG from e to retrieve its reachable end entities:

$$\widetilde{\mathcal{Z}}_{q,e}^{K} = \text{BeamSearch}_K(\pi_\theta(\cdot \mid q, e)), \qquad c_{e,\tilde{z}} = (e, \tilde{z}, \mathcal{E}_{\tilde{z}}(e)), \quad \tilde{z} \in \widetilde{\mathcal{Z}}_{q,e}^{K}. \tag{10}$$

Each evidence item $c_{e,\tilde{z}}$ contains the source entity, relation sequence, and retrieved end entities. We aggregate these items across question entities and generated paths into an evidence context $\mathcal{C}_q$.

The question q and evidence context $\mathcal{C}_q$ are provided to the LLM answer generator p_ψ through zero-shot prompting, using the template in Figure 4. The model is instructed to select answers from the retrieved end entities and generates an answer sequence $y = (y_1, \ldots, y_T)$ according to

$$p_\psi(y \mid q, \mathcal{C}_q) = \prod_{t=1}^{T} p_\psi(y_t \mid q, \mathcal{C}_q, y_{<t}). \tag{11}$$

The resulting evidence context preserves question-relevant relational structure in a compact form. The answer generator then uses this context to jointly reason over multiple relation paths and their retrieved candidate answers for answering multi-hop and multi-constraint questions.

5 Experiments

We conduct experiments to answer the following research questions: **RQ1:** How does ROP compare with existing KGQA methods in effectiveness and efficiency? **RQ2:** How does reward model design affect KGQA performance? **RQ3:** Can the learned path rewards improve other retrieval models? **RQ4:** How does the two-stage reward-guided training strategy contribute to KGQA performance? **RQ5:** How reliable are the learned path rewards as intermediate supervision?

5.1 Experimental Setup

Datasets. We evaluate on WebQuestionsSP (WebQSP) (Yih et al., 2016) and ComplexWebQuestions (CWQ) (Talmor & Berant, 2018), using the preprocessed Freebase subgraphs adopted by prior work (Luo et al., 2024; Mavromatis & Karypis, 2025; Luo et al., 2025b). WebQSP contains 1- to 2-hop questions from Google search logs with annotated SPARQL queries, while CWQ focuses on compositional and multi-constraint questions of up to 4 hops. To evaluate ROP on a different KG, we additionally report results on MetaQA (Zhang et al., 2018) (Appendix F.6). Detailed dataset statistics are provided in Appendix C.

Evaluation Metrics. For answer generation, we adopt F1, Hit, and Hits@1 as the evaluation metrics. F1 measures answer set correctness using the harmonic mean of answer precision and recall. Hit assesses whether any ground-truth answer appears in the predicted answer set. Hits@1 measures the proportion of questions for which the top-1 predicted answer is correct. For retrieval, we report SPARQL path Hits@1 and Hit using paths extracted from annotated SPARQL queries as references, together with precision, recall, and F1 for retrieved entities against the annotated answer set. For supervision quality, we rank answer-reaching paths and report SPARQL path Hits@1 and MRR.

Baselines. We compare ROP against recent KG-RAG baselines grouped by their supervision paradigms: (1) *prompting-based methods*, (2) *training with answer-derived supervision*, and (3) *training with LLM-refined supervision*. Detailed descriptions are provided in Appendix D.

Implementation Details. The reward model is implemented by a two-layer Transformer encoder and an MLP scoring head, using gte-large-en-v1.5 (Li et al., 2023b) as the frozen text encoder for questions and relations. We train the LoRA parameters (Hu et al., 2022) of a LLaMA3.1-8B-Instruct relation path generator, keeping its base weights and the learned reward model frozen. At inference, we generate paths with beam size $K = 5$ and use GPT-4o as the default answer LLM. Further configurations and parameter studies are provided in Appendices E and G.

5.2 Main Results (RQ1)

KGQA Performance. We first compare ROP with other KG-RAG baselines on WebQSP and CWQ to evaluate its overall KGQA performance. The results for ROP are averaged over three independent runs, with standard deviations reported in Appendix F.1. As shown in Table 1, ROP achieves the strongest overall accuracy across the two datasets. Among trained methods, ROP consistently outperforms those using answer-derived supervision, with relative gains of 3.0% and 2.6% in F1 and up to 7.6% in Hits@1 across WebQSP and CWQ. These consistent gains show that more reliable intermediate supervision can improve KGQA accuracy. Although LLM-refined baselines invoke LLMs to enhance intermediate supervision, ROP still achieves higher F1 and Hits@1 on both datasets without requiring additional LLM calls for supervision construction. Overall, these results validate the effectiveness of learning path rewards directly from answer labels for improving KGQA reasoning.

Table 1: Comparison of KGQA performance (%) on WebQSP and CWQ. `Model` denotes the model used for answer generation. We highlight the best performance in **bold** and underline the second-best. † indicates that the model is fine-tuned. * indicates results taken from the original paper.

Method	Model	WebQSP			CWQ		
		F1	Hit	Hits@1	F1	Hit	Hits@1
Prompting with in-context learning							
ToG (Sun et al., 2024)	GPT-4o	50.9	78.5	72.1	42.3	56.9	51.8
DoG (Ma et al., 2025a)	GPT-4o	56.1	91.2	66.9	47.5	58.4	43.0
FiDeLiS (Sui et al., 2025)	GPT-4o	54.8	77.6	70.0	46.8	58.1	50.2
Training with answer-derived supervision							
RoG (Luo et al., 2024)	LLaMA2-Chat-7B†	67.0	81.9	78.3	53.6	59.2	53.7
GNN-RAG (Mavromatis & Karypis, 2025)	LLaMA2-Chat-7B†	69.2	81.1	76.2	58.3	65.3	60.5
GCR (Luo et al., 2025b)	GPT-4o	69.8	86.9	82.9	57.1	64.6	59.1
SubgraphRAG (Li et al., 2025)	GPT-4o	76.1	89.9	84.3	59.2	67.3	60.7
DP (Ma et al., 2025b)	GPT-4o	75.8	88.9	81.6	57.3	63.2	60.1
RPO-RAG (Um et al., 2026)	LLaMA3.1-8B-Instruct†	79.5	88.6	80.7	61.2	69.3	60.3
Training with LLM-refined supervision							
KG-Hopper* (Wang & Yu, 2026)	Qwen-2.5-7B†	-	-	83.2	-	-	61.1
RAPL* (Yao et al., 2025)	GPT-4o	80.7	**93.3**	-	58.8	69.0	-
ReG* (Zou et al., 2025)	GPT-4o	78.8	91.4	-	62.5	68.9	-
Ours							
RoP	GPT-4o	**81.9**	91.0	**86.0**	**62.8**	**71.1**	**65.3**

Table 2: Comparison of retrieval performance (%) on WebQSP and CWQ.

Method	WebQSP					CWQ				
	SPARQL Path		Answer			SPARQL Path		Answer		
	Hits@1	Hit	F1	Recall	Precision	Hits@1	Hit	F1	Recall	Precision
RoG	62.3	76.2	48.1	86.1	42.5	45.1	62.3	21.1	71.1	17.4
GCR	72.0	87.6	42.5	**91.9**	34.7	54.4	62.7	18.9	**75.6**	14.3
SubgraphRAG	-	-	10.3	90.6	7.2	-	-	3.7	74.0	2.0
DP	69.3	87.7	62.5	84.3	58.5	44.8	**81.0**	26.9	73.4	22.3
RPO-RAG	46.6	82.8	46.1	89.1	38.5	40.8	73.7	23.1	72.6	18.9
RoP	**83.1**	**88.7**	**69.8**	85.7	**67.0**	**68.6**	76.2	**29.1**	74.0	**25.0**

Retrieval Performance. Table 2 provides a closer view of the retrieval behavior underlying the KGQA results. For path generation, RoP improves SPARQL path Hits@1 over all baselines by at least 15.4% on both datasets, while attaining the highest SPARQL path Hit on WebQSP and a competitive Hit on CWQ. For answer retrieval, RoP also achieves the highest F1 and precision on both datasets, outperforming the second-best method, DP, by 11.7% and 8.2% in F1 and by 14.5% and 12.1% in precision on WebQSP and CWQ, respectively. Although GCR attains higher answer recall and DP a higher SPARQL path Hit on CWQ, RoP yields substantially higher Hits@1, precision, and F1, suggesting that its gains arise from more precise path generation rather than broader retrieval coverage. This pattern is consistent with the learned path rewards mitigating noise in answer-derived supervision and providing more reliable intermediate supervision for path generation.

Efficiency. We further compare inference cost and the cost of constructing training supervision. As shown in Table 3, RoP achieves the lowest runtime on both datasets while requiring substantially fewer tokens for answer reasoning. This efficiency follows from the precise path generation observed above. The generated paths yield a compact evidence context that reduces the input processed at inference time and shields the answer LLM from irrelevant evidence that would otherwise mislead its reasoning. For supervision construction, LLM-refined methods incur additional LLM inference over the training set. RAPL, the LLM-refined baseline in Table 3, requires one GPT-4o call and over 2K tokens per training question, amounting to roughly 6M and 93M tokens over the WebQSP and CWQ training sets, respectively.[1] In contrast, RoP learns path rewards directly from answer labels without any LLM calls for supervision construction, taking only 2.6 and 3.2 seconds per training question on WebQSP and CWQ (including data processing and reward model training).

[1] Approximately $22 and $318 under the GPT-4o API pricing as of 2026-09-14, assuming uncached inputs.

Table 3: Efficiency comparison on WebQSP and CWQ. All results are averages per question. `Train Sup.` denotes training supervision construction; `# Tokens` is the sum of input and output tokens.

Stage	Method	WebQSP			CWQ		
		Runtime (s)	# Calls	# Tokens	Runtime (s)	# Calls	# Tokens
Inference	GCR	3.5	2.0	686.6	3.6	2.0	741.3
	SubgraphRAG	2.3	**1.0**	2,699.9	2.8	**1.0**	2,607.7
	DP	4.4	2.5	2,799.5	6.6	3.1	3,985.7
	ROP	**1.8**	2.0	**382.0**	**2.1**	2.6	**593.8**
Train Sup.	RAPL	4.7	1.0	2,122.4	5.2	1.0	3,359.7
	ROP	**2.6**	**0**	**0**	**3.2**	**0**	**0**

5.3 ABLATION STUDY (RQ2–RQ4)

We conduct ablation studies with several variants on the test sets (GPT-4o as the answer LLM) to assess the path reward design and the two-stage training strategy, and whether the learned supervision transfers to other retrieval models. Table 4 summarizes the results, with Hits@1 and the implementation details of all variants provided in Appendix F.4.

Reward Model Design (RQ2). We compare ROP with three reward model variants sharing its architecture and training setup, the first two of which treat every answer-reaching path as positive: (1) *Pointwise BCE*, which adapts SubgraphRAG's pointwise classification objective to candidate paths (Li et al., 2025); (2) *Pairwise ranking*, which uses a margin loss to rank answer-reaching paths above non-answer-reaching paths (Lin et al., 2025); and (3) *w/o question*, which removes question conditioning while retaining the asymmetric objective.

Table 4: Ablation and transferability analysis on WebQSP and CWQ.

Variant	WebQSP		CWQ	
	F1	Hit	F1	Hit
ROP	**81.9**	91.0	**62.8**	**71.1**
Path Reward Design				
Pointwise BCE	75.9	89.4	57.5	65.7
Pairwise ranking	76.1	89.3	57.9	65.5
w/o question	62.6	80.7	46.8	58.4
Reward Transferability				
RoG	67.0	81.9	53.6	59.2
+ ROP Reward Sup.	72.4	82.8	57.7	64.9
SubgraphRAG	76.1	89.9	59.2	67.3
+ ROP Reward Sup.	78.2	89.5	60.5	68.6
Generator Optimization				
Distillation only	80.8	90.9	61.5	70.0
GRPO only	44.9	61.7	37.2	54.6
Distillation + GRPO-Ans	80.6	**91.2**	61.0	69.7
Distillation + GRPO-F1	80.9	90.6	60.4	67.9
SFT + GRPO-Ans	77.8	90.1	59.5	65.8

Among these alternatives, *Pairwise ranking* performs best, but still consistently underperforms ROP on both WebQSP and CWQ. Removing the question from reward modeling leads to a much larger drop, indicating that the learned reward captures question-dependent path utility rather than a question-agnostic preference. The consistent gains over both conventional objectives further suggest that the asymmetric objective yields more reliable path supervision and, in turn, improved downstream KGQA performance.

Reward Transferability (RQ3). The learned supervision also transfers to other retrieval models. Replacing RoG's uniform shortest-path targets with the highest-scoring answer-reaching path and reweighting SubgraphRAG's path supervision by the learned reward, without changing either architecture, improves F1 by 5.4 and 2.1 points on WebQSP and by 4.1 and 1.3 points on CWQ. Consistent gains on both an LLM-based path generator and a lightweight triple retriever indicate that the learned reward provides reusable intermediate supervision across retrieval architectures.

Reward-Guided Path Generation (RQ4). We consider five variants to characterize the roles of reward distillation and on-policy optimization: (1) *Distillation only*, which removes on-policy optimization; (2) *GRPO only*, which applies GRPO with the learned reward to the base model, without distillation; (3) *Distillation + GRPO-Ans*, which replaces the learned GRPO reward with a binary answer-reaching reward; (4) *Distillation + GRPO-F1*, which uses answer-set F1 as the GRPO reward; and (5) *SFT + GRPO-Ans*, which replaces distillation with uniform SFT over answer-reaching paths and uses the binary answer reward in GRPO.

Distillation alone provides a strong initialization, whereas applying GRPO with the learned reward directly to the base model, without distillation, leads to a substantial performance drop. Starting from the distilled policy, on-policy optimization with the learned reward further improves F1 by 1.1

and 1.3 points on WebQSP and CWQ. The reward used in this stage is decisive: replacing it with the binary answer reward or answer-set F1 lowers F1 by 1.3 and 1.8 points and by 1.0 and 2.4 points relative to the full model, and neither variant improves over distillation alone. Removing the learned reward from both stages, which reduces to the standard SFT-then-GRPO recipe, causes the largest decline among the distilled variants, 4.1 and 3.3 points below the full model, confirming that the learned reward is essential to both initialization and policy refinement.

5.4 Analysis of Learned Path Rewards (RQ5)

We further analyze the learned path rewards from two angles: how accurately they identify the reference path among answer-reaching candidates, and how this accuracy holds up as the number of such candidates grows. Both analyses use training questions with more than one answer-reaching candidate path, the setting in which answer-derived supervision is ambiguous. To limit the cost of LLM-based scoring, we randomly sample 200 such questions per dataset and report results averaged over five runs, with results on the development sets provided in Appendix F.5.

Training Supervision Quality. Table 5 ranks the answer-reaching candidate paths of each question with different scorers and reports SPARQL path Hits@1 and MRR against the reference paths. We compare two heuristic scorers, cosine similarity between question and path embeddings and Answer F1, which scores each path by the F1 of its retrieved entities against the ground-truth answers. We also include the two conventional learning objectives from Section 5.3 and GPT-4o as an LLM-based scorer. RoP outperforms all of them on both datasets. Notably, Answer F1, despite using the ground-truth answers directly, trails RoP by 18.9 and 49.6 Hits@1 points, showing that answer overlap alone cannot identify the reference path and explaining why answer-derived labels are noisy. GPT-4o comes within 1.6 Hits@1 points of RoP on WebQSP, but on CWQ, with more hops and larger candidate sets, RoP leads by 15.3 Hits@1 and 10.6 MRR points. These results support the effectiveness of the learned reward as intermediate supervision without LLM feedback.

Table 5: SPARQL path Hits@1 and MRR (%).

Scoring Method	WebQSP		CWQ	
	Hits@1	MRR	Hits@1	MRR
Cosine sim.	56.9	70.2	29.3	44.3
Answer F1	71.3	81.6	22.2	38.9
Pointwise BCE	55.7	70.4	25.7	37.2
Pairwise ranking	72.8	81.4	30.0	41.5
GPT-4o	88.6	92.3	56.5	68.1
RoP Reward	**90.2**	**93.1**	**71.8**	**78.7**

Effect of Supervision Ambiguity. Figure 2 groups the pooled WebQSP and CWQ samples by the number of answer-reaching candidate paths M, a proxy for supervision ambiguity, with a larger M indicating more paths consistent with the answer labels. Hits@1 decreases for all scorers as M grows, but RoP degrades most gradually, and its lead over every baseline, including the GPT-4o scorer, widens accordingly. The widening gap further highlights the advantage of RoP in distinguishing among answer-reaching paths as supervision ambiguity increases.

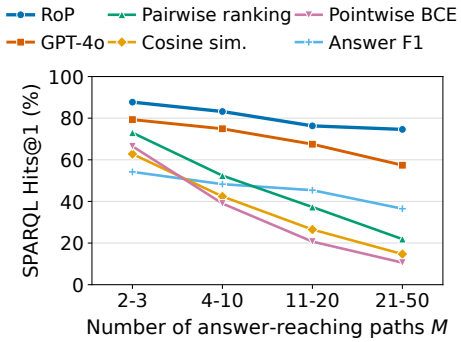


Figure 2: SPARQL path Hits@1 under varying levels of supervision ambiguity.

Supplementary results, case studies, and error analyses are provided in Appendices F–I.

6 Conclusion

We proposed RoP, a framework for learning path rewards from answer labels to train an LLM-based relation path generator for KGQA. By combining asymmetric path reward learning with reward distillation and on-policy optimization, RoP provides more reliable intermediate supervision for path generation without costly LLM-refined supervision construction. Experiments on multiple KGQA datasets demonstrate strong KGQA and retrieval accuracy, while further analyses show that the learned rewards remain effective under ambiguous answer-derived supervision and transfer across KG retrieval models. These results highlight the effectiveness of learned path rewards as reliable and scalable intermediate supervision for KG-grounded reasoning.

AI USE STATEMENT

We used generative AI tools to assist in implementing methods, specifically to write and debug parts of the experimental code, and did not use them for any other task requiring disclosure, including developing the method, formulating claims, designing experiments, or interpreting results. Additionally, we used generative AI tools to edit the paper for readability. All AI-generated code was verified and tested by the authors, and all AI-edited text was checked against the methods and results it describes. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses existing KGQA benchmarks and knowledge graph resources and does not involve human participants, user studies, or the collection of new personal data. We do not identify additional ethical concerns specific to the proposed methodology beyond those associated with the underlying data and models.

REPRODUCIBILITY STATEMENT

Implementation details and hyperparameters are provided in Appendix E, prompt templates in Appendix B, dataset statistics in Appendix C, and baseline descriptions and protocols in Appendix D. Anonymous source code and experimental configurations are available at `https://anonymous.4open.science/r/Reward-on-Path-0CB2`.

REFERENCES

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*, pp. 722–735, 2007.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pp. 1247–1250, 2008.

Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. Plan-on-Graph: Self-correcting adaptive planning of large language model on knowledge graphs. In *NeurIPS*, pp. 37665–37691, 2024.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond I.I.D.: Three levels of generalization for question answering on knowledge bases. In *WWW*, pp. 3477–3488, 2021.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. Complex knowledge base question answering: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(11):11196–11215, 2023.

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

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. Few-shot in-context learning for knowledge base question answering. In *ACL*, pp. 6966–6980, 2023a.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023b.

Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. FlexKBQA: A flexible LLM-powered framework for few-shot knowledge base question answering. In *AAAI*, pp. 18608–18616, 2024.

Can Lin, Zhengwang Jiang, Ling Zheng, Qi Zhao, Yuhang Zhang, Qi Song, and Wangqiu Zhou. RJE: A retrieval-judgment-exploration framework for efficient knowledge graph question answering with LLMs. In *EMNLP*, pp. 17277–17294, 2025.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. Multi-hop knowledge graph reasoning with reward shaping. In *EMNLP*, pp. 3243–3253, 2018.

Yuyin Lu, Ziran Liang, Yanghui Rao, Wenqi Fan, Fu Lee Wang, and Qing Li. Double-Calibration: Towards reliable LLMs via calibrating knowledge and reasoning confidence. *arXiv preprint arXiv:2601.11956*, 2026.

Haoran Luo, Haihong E, Yikai Guo, Qika Lin, Xiaobao Wu, Xinyu Mu, Wenhao Liu, Meina Song, Yifan Zhu, and Anh Tuan Luu. KBQA-o1: Agentic knowledge base question answering with Monte Carlo Tree Search. In *ICML*, pp. 41177–41199, 2025a.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *ICLR*, 2024.

Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Yuan-Fang Li, Chen Gong, and Shirui Pan. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. In *ICML*, pp. 41540–41565, 2025b.

Jie Ma, Zhitao Gao, Qi Chai, Wangchun Sun, Pinghui Wang, Hongbin Pei, Jing Tao, Lingyun Song, Jun Liu, Chen Zhang, et al. Debate on graph: A flexible and reliable reasoning framework for large language models. In *AAAI*, pp. 24768–24776, 2025a.

Jie Ma, Ning Qu, Zhitao Gao, Rui Xing, Jun Liu, Hongbin Pei, Jiang Xie, Linyun Song, Pinghui Wang, Jing Tao, and Zhou Su. Deliberation on priors: Trustworthy reasoning of large language models on knowledge graphs. In *NeurIPS*, 2025b.

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

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yuan Sui, Yufei He, Nian Liu, Xiaoxin He, Kun Wang, and Bryan Hooi. FiDeLiS: Faithful reasoning in large language models for knowledge graph question answering. In *Findings of ACL*, pp. 8315–8330, 2025.

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

Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In *NAACL*, pp. 641–651, 2018.

Kaehyun Um, KyuHwan Yeom, Haerim Yang, Minyoung Choi, Hyeongjun Yang, and Kyong-Ho Lee. RPO-RAG: Aligning small LLMs with relation-aware preference optimization for knowledge graph question answering. In *WWW*, pp. 2513–2523, 2026.

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

Shuai Wang and Yinan Yu. KG-Hopper: Empowering compact open LLMs with knowledge graph reasoning via reinforcement learning. *arXiv preprint arXiv:2603.21440*, 2026.

Yingxu Wang, Shiqi Fan, Mengzhu Wang, Siyang Gao, Chao Wang, and Nan Yin. DAMR: Efficient and adaptive context-aware knowledge graph question answering with LLM-guided MCTS. In *ICLR*, 2026.

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

Tianjun Yao, Haoxuan Li, Zhiqiang Shen, Pan Li, Tongliang Liu, and Kun Zhang. Learning efficient and generalizable graph retriever for knowledge-graph question answering. *arXiv preprint arXiv:2506.09645*, 2025.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In *ACL*, pp. 201–206, 2016.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. Variational reasoning for question answering with knowledge graph. In *AAAI*, pp. 6069–6076, 2018.

Deyu Zou, Yongqiang Chen, Mufei Li, Siqi Miao, Chenxi Liu, Bo Han, James Cheng, and Pan Li. Weak-to-strong GraphRAG: Aligning weak retrievers with large language models for graph-based retrieval augmented generation. *arXiv preprint arXiv:2506.22518*, 2025.

A Relation Path Example

We provide an illustrative example on how a relation path is grounded over a KG. Consider a relation path $z = \langle \text{sibling} \rightarrow \text{nationality} \rangle$. Given a starting entity $e = \text{Elon Musk}$, the reachable entity set $\mathcal{E}_z(e)$ contains all entities that can be reached by following the relations in z from e. Traversing the path z from the entity Elon Musk can ground a KG reasoning path:

$$\text{Elon Musk} \xrightarrow{sibling} \text{Kimbal Musk} \xrightarrow{nationality} \{\text{USA, Canada, South Africa}\}.$$

Thus,

$$\mathcal{E}_z(\text{Elon Musk}) = \{\text{USA, Canada, South Africa}\}.$$

B Prompt Template

In this section, we illustrate all prompt templates used in this work.

Relation Path Generation Prompt. As shown in Figure 3, the prompt template for relation path generation takes a question and its question entity as input, and instructs the LLM to generate a relation path consisting of a sequence of relations, i.e., $z = \langle r_1, r_2, \ldots, r_l \rangle$. The same template is used in both training stages and at inference.

KG-grounded Answer Reasoning Prompt. As shown in Figure 4, the answer reasoning prompt takes the question and the evidence context $\mathcal{C}_q$, i.e., the grounded paths of all question entities, each listed with its question entity, relation path, and retrieved end entities.

Relation Path Generation Prompt

Instruction: Reasoning path is a sequence of relations in the Knowledge Graph that connects the topic entity in the question to answer entities. Given a question, please generate a reasoning path in the Knowledge Graph starting from the topic entity to answer the question.

Input:
Question: `<Question>`
Topic entity: `<Topic Entity>`

Output:
`<PATH>` $r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_l$ `</PATH>`

Figure 3: The prompt template for relation path generation.

KG-Grounded Answer Reasoning

Instruction: You are a helpful and precise assistant for answering questions based on the provided reasoning paths on a knowledge graph. Please return all the possible answers from the entities mentioned in the reasoning paths. Please return each answer at a new line.

Input:

Reasoning Paths:

[PATH1]
Topic Entity: <Topic Entity>,
Relation Path: $r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_l$
End Entities: <End Entity$_1$> <SEP> <End Entity$_2$> <SEP> $\cdots$

[PATH2]
$\cdots$

Question: <Question>

Output:

Answer$_1$
Answer$_2$
$\cdots$
Answer$_n$

Figure 4: The prompt template for KG-grounded inductive reasoning.

C Datasets Statistics

Table 6 reports the overall scale of each dataset, including the associated KG, numbers of entities, relations, triples, training and test questions, and the maximum reasoning hop. Table 7 provides the distribution of answer set sizes for each dataset, which helps characterize the answer sparsity and complexity of these datasets.

Table 6: Statistics of the datasets.

Dataset	KG	Entities	Relations	Triples	Train	Test	Max Hop	License
WebQSP	Freebase	2,566,291	7,058	8,309,105	2,826	1,628	2	CC BY 4.0
CWQ	Freebase	2,566,291	7,058	8,309,105	27,639	3,531	4	-
MetaQA	Wiki-Movie	43,234	9	133,592	329,282	30,903	3	-

Table 7: Statistics of number of answers on WebQSP and CWQ.

Dataset	#Ans = 1	$2 \leq$ #Ans≤ 4	$5 \leq$ #Ans≤ 9	#Ans≥ 10
WebQSP	51.2%	27.4%	8.3%	12.1%
CWQ	70.6%	19.4%	6%	4%
MetaQA	37.9%	29.6%	12.6%	19.9%

D Baselines

We group the baselines according to how they obtain supervision for KG reasoning: (1) *prompting-based methods*, which perform KG reasoning through LLM prompting without task-specific training; (2) *training with answer-derived supervision*, which derives training signals from answer labels; and (3) *training with LLM-refined supervision*, which uses LLMs to construct, refine, or evaluate intermediate reasoning signals. We describe each baseline below.

Prompting-based Methods. These methods rely on prompted LLMs to explore or reason over the KG without task-specific model training.

- ToG (Sun et al., 2024) treats the LLM as an agent that performs beam search over the KG, iteratively expanding relations and entities until sufficient evidence is collected for answering.
- DoG (Ma et al., 2025a) employs multiple prompted LLM roles to debate over candidate subgraphs, with an answer-trying module determining whether to stop or continue graph exploration.
- FiDeLiS (Sui et al., 2025) combines training-free path retrieval with deductive verification, verifying each reasoning step against the KG before proceeding or producing the final answer.

Training with Answer-derived Supervision. These methods derive supervision from ground-truth answer labels alone.

- RoG (Luo et al., 2024) adopts a planning–retrieval–reasoning pipeline in which an LLM generates relation paths, retrieves matching KG reasoning paths, and reasons over the retrieved evidence for answer generation.
- GNN-RAG (Mavromatis & Karypis, 2025) uses a GNN to score answer candidates over a question-specific subgraph and verbalizes shortest paths from question entities to predicted candidates for LLM-based answer reasoning.
- GCR (Luo et al., 2025b) incorporates KG structure into LLM decoding through a KG-Trie, using a KG-specialized LLM to generate KG-grounded reasoning paths and a general LLM to aggregate them for answer prediction.
- SubgraphRAG (Li et al., 2025) retrieves compact evidence subgraphs by scoring KG triples with a lightweight model enhanced by directional distance encoding, and prompts an LLM to reason over the retrieved evidence.
- DP (Ma et al., 2025b) distills KG structural priors into an LLM through supervised fine-tuning and preference optimization, and employs reasoning introspection to verify constraints and trigger backtracking when necessary.
- RPO-RAG (Um et al., 2026) ranks answer-reaching paths by their similarity to the question under a pretrained semantic encoder and incorporates the resulting preferences into relation-aware preference optimization.

Training with LLM-refined Supervision. These methods use LLMs to refine intermediate supervision or to provide reward signals for training KG reasoning models.

- KG-Hopper (Wang & Yu, 2026) trains a compact LLM for KG reasoning with reinforcement learning, using LLM-based rewards to evaluate generated reasoning paths during policy optimization.
- RAPL (Yao et al., 2025) trains a lightweight graph retriever with LLM-rationalized path labels and triple-level message passing to retrieve compact KG evidence for downstream reasoning.
- ReG (Zou et al., 2025) refines weak supervision with LLM feedback and reorganizes retrieved graph evidence into logic-preserving chains for downstream LLM reasoning.

E Implementation Details

All experiments are conducted on a machine with two NVIDIA H100 GPUs and 320 GB RAM. Unless otherwise specified, we use the preprocessed Freebase subgraphs described in Section 5.

E.1 Path Reward Learning

We construct candidate relation paths by breadth-first search from each question entity, with the maximum path length set to $L = 2$ for WebQSP and $L = 3$ for CWQ. We retain all candidate

paths that reach at least one ground-truth answer. To control the candidate set size and reduce the computational cost of reward-model training, we randomly sample 50 non-answer-reaching paths per training question on WebQSP and 200 on CWQ.

For the path reward model, we use gte-large-en-v1.5 (Li et al., 2023b) as the frozen text encoder for both questions and relations. The resulting embeddings are processed by a two-layer Transformer encoder with a hidden dimension of 128 and an MLP scoring head. The text encoder remains frozen, while the Transformer encoder, positional embeddings, and MLP are trainable, resulting in 0.5M trainable parameters.

We optimize the asymmetric reward-learning objective with AdamW using a learning rate of 4×10^{-4}, weight decay of 0.01, and gradient clipping at 1.0. The batch size is 4 question–entity pairs per optimization step. We linearly warm up the learning rate for the first 10% of training steps and then apply cosine decay to zero. The reward model is trained for 500 epochs on WebQSP and 200 epochs on CWQ.

E.2 Reward-Guided Relation Path Generation

We use LLaMA3.1-8B-Instruct as the relation path generator and optimize only its LoRA parameters (Hu et al., 2022), while keeping the base LLM weights frozen. The LoRA rank and scaling factor are set to 16 and 32, respectively. The learned path reward model remains frozen throughout both stages of generator training.

Path Reward Distillation. For each training instance, we retain all answer-reaching candidate paths and construct the target distribution by applying a softmax over their learned reward scores. The generator is optimized using the complete autoregressive likelihood of each relation path, without length normalization or truncating the candidate set by reward rank.

We train the generator for 10 epochs with AdamW using a learning rate of 5×10^{-5}, weight decay of 0.01, and an effective batch size of 16. The learning rate is kept constant without warmup.

On-Policy Reward Optimization. Starting from the distilled generator, we further optimize the policy with GRPO. For each question–entity pair, we sample $G = 8$ relation paths using rollout temperatures $\{1.0, 1.0, 1.15, 1.15, 1.3, 1.3, 1.45, 1.45\}$. Policy probabilities used for computing the importance ratios are evaluated at temperature 1, without additional importance-sampling correction for the rollout temperatures.

Rewards are normalized within each rollout group with $\epsilon_{\text{std}} = 10^{-6}$. We use a PPO clipping coefficient of $\varepsilon = 0.2$ and perform one policy-update epoch per rollout batch. The loss is averaged over all generated tokens in the batch, without additional KL-divergence or entropy regularization. We optimize the LoRA parameters with AdamW using a learning rate of 1×10^{-5}, weight decay of 0.01, and gradient clipping at 1.0; the learning rate is kept constant without warmup. Each update uses 16 prompts with 8 rollouts per prompt, corresponding to 128 generated sequences. We train for approximately 3 epochs over the prompt pool, reporting the checkpoints at step 540 on WebQSP and step 7,893 on CWQ.

KG-Grounded Answer Reasoning. At inference time, we generate the top-K relation paths with beam search using $K = 5$. Each generated path is grounded from the corresponding question entity in the KG, and the resulting evidence is provided to the answer LLM through zero-shot prompting with decoding temperature 0.

F Supplementary Results

This section provides additional experimental results that complement the main evaluation, including run-to-run stability, robustness across model backbones, performance across reasoning hops, additional path-ranking analyses, and extended ablation results.

F.1 Standard Deviation of KGQA Performance

Following prior work (Ma et al., 2025b), we conduct three independent runs and report the mean and standard deviation in Table 8.

Table 8: Stability analysis of ROP with answer LLMs. Results are reported as mean values with standard deviations shown as subscripts.

LLM	WebQSP			CWQ		
	F1	Hit	Hits@1	F1	Hit	Hits@1
GPT-4o	$81.9_{\pm 0.6}$	$91.0_{\pm 0.5}$	$86.0_{\pm 0.7}$	$62.8_{\pm 0.5}$	$71.1_{\pm 0.4}$	$65.3_{\pm 0.6}$
GPT-4.1	$82.2_{\pm 0.6}$	$91.2_{\pm 0.5}$	$85.2_{\pm 0.7}$	$62.7_{\pm 0.5}$	$73.2_{\pm 0.5}$	$64.3_{\pm 0.6}$

F.2 PERFORMANCE WITH DIFFERENT BACKBONES

Table 9: KGQA performance (%) of ROP with different text encoders, relation path generators, and answer generators.

Configuration	WebQSP			CWQ		
	F1	Hit	Hits@1	F1	Hit	Hits@1
ROP (default)	81.9	91.0	**86.0**	62.8	71.1	**65.3**
Text Encoder						
all-MiniLM-L6-v2	81.7	90.3	85.5	**63.2**	70.6	65.1
Relation Path Generator						
Qwen2.5-7B-Instruct	**82.4**	**91.3**	**86.0**	62.2	69.9	64.8
Qwen3-8B	81.4	90.6	85.3	62.3	70.3	64.5
Answer Generator						
GPT-4.1	82.2	91.2	85.2	62.7	**73.2**	64.3

To examine the effect of backbone choices, we evaluate ROP with alternative text encoders, relation path generators, and answer generators, varying one component at a time while keeping the remaining configuration unchanged. Specifically, we replace gte-large-en-v1.5 with all-MiniLM-L6-v2, LLaMA3.1-8B-Instruct with Qwen2.5-7B-Instruct or Qwen3-8B (Yang et al., 2025), and GPT-4o with GPT-4.1; the first two alternatives are adopted by recent KGQA baselines (Um et al., 2026; Wang & Yu, 2026).

As shown in Table 9, ROP maintains comparable performance across all choices. Replacing gte-large-en-v1.5 with all-MiniLM-L6-v2 slightly improves F1 on CWQ, Qwen2.5-7B-Instruct achieves the highest F1 and Hit on WebQSP, and Qwen3-8B performs close to the default configuration. Replacing GPT-4o with GPT-4.1 as the answer generator yields comparable F1 (82.2 and 62.7) and a higher Hit on CWQ. Across all alternatives, F1 differs from the default configuration by at most 0.6 points on either dataset, showing that the gains of ROP do not depend on a particular encoder, generator, or answer LLM.

F.3 PERFORMANCE ACROSS REASONING HOPS

Table 10: Breakdown of KGQA performance (%) over # hops.

Model	WebQSP-sub						CWQ-sub							
	1		2		Overall		1		2		≥3		Overall	
	F1	Hit	F1	Hit	F1	Hit	F1	Hit	F1	Hit	F1	Hit	F1	Hit
SubgraphRAG	80.3	89.7	74.4	83.0	78.3	87.5	57.0	59.7	63.3	68.0	51.6	52.0	60.8	64.7
GCR	71.7	90.1	69.7	88.5	71.0	89.6	60.6	67.3	65.6	74.5	47.7	51.5	63.1	71.1
ROP	**84.5**	**92.4**	**80.7**	**92.3**	**83.2**	**92.4**	**66.4**	**75.5**	**75.0**	**83.8**	**56.2**	**64.7**	**71.4**	**80.3**

Following SubgraphRAG (Li et al., 2025), we further evaluate ROP across questions with different reasoning depths on WebQSP-sub and CWQ-sub, where questions whose answer entities are absent from the KG are excluded, and compare with SubgraphRAG and GCR. As shown in Table 10, ROP achieves the best F1 and Hit in every hop group on both datasets. The gains do not shrink with

reasoning depth: on WebQSP-sub, ROP improves F1 over SubgraphRAG by 4.2 and 6.3 points on 1-hop and 2-hop questions, respectively. On CWQ-sub, the F1 improvement over GCR reaches 9.4 points on 2-hop questions, and the Hit improvement over SubgraphRAG reaches 12.7 points on questions requiring three or more hops. The benefit of reward-guided path generation thus extends to questions requiring longer relational reasoning.

F.4 ABLATION STUDY

Table 11: Complete ablation results on WebQSP and CWQ. We report F1, Hit, and Hits@1 (%).

Variant	WebQSP			CWQ		
	F1	Hit	Hits@1	F1	Hit	Hits@1
Reward Model Design						
ROP	**81.9**	91.0	**86.0**	**62.8**	**71.1**	**65.3**
Pointwise BCE	75.9	89.4	82.6	57.5	65.7	59.8
Pairwise ranking	76.1	89.3	83.1	57.9	65.5	60.4
w/o question	62.6	80.7	71.9	46.8	58.4	50.9
Transferability of Learned Reward Supervision						
RoG (Luo et al., 2024)	67.0	81.9	78.3	53.6	59.2	53.7
+ ROP Reward Sup.	72.4	82.8	79.5	57.7	64.9	59.1
SubgraphRAG (Li et al., 2025)	76.1	89.9	84.3	59.2	67.3	60.7
+ ROP Reward Sup.	78.2	89.5	84.9	60.5	68.6	62.3
Reward-Guided Path Generation						
Distillation only	80.8	90.9	84.8	61.5	70.0	63.8
GRPO only	44.9	61.7	52.8	37.2	54.6	42.1
Distillation + GRPO-Ans	80.6	**91.2**	85.1	61.0	69.7	63.6
Distillation + GRPO-F1	80.9	90.6	84.5	60.4	67.9	61.8
SFT + GRPO-Ans	77.8	90.1	82.7	59.5	65.8	61.2

Reward Model Baselines. We compare our asymmetric reward-learning objective with two existing path-scoring objectives while keeping the scoring architecture $s_\phi(q, z)$ unchanged.

Pointwise BCE follows the path-scoring objective used in SubgraphRAG (Li et al., 2025). Each candidate path is assigned the answer-derived label

$$y_i = \mathbb{I}\left[z_i \in \mathcal{Z}_{q,e}^{\mathcal{A}_q}\right], \tag{12}$$

and optimized with binary cross-entropy:

$$\mathcal{L}_{\mathrm{BCE}}^{q,e} = -\frac{1}{|\mathcal{Z}_{q,e}|} \sum_{z_i \in \mathcal{Z}_{q,e}} \left[y_i \log \sigma(s_\phi(q, z_i)) + (1 - y_i) \log(1 - \sigma(s_\phi(q, z_i)))\right]. \tag{13}$$

Pairwise ranking follows the margin-based path-ranking objective (Lin et al., 2025). For each question–entity pair, we use all answer-reaching and non-answer-reaching path pairs:

$$\mathcal{P}_{q,e} = \mathcal{Z}_{q,e}^{\mathcal{A}_q} \times \left(\mathcal{Z}_{q,e} \setminus \mathcal{Z}_{q,e}^{\mathcal{A}_q}\right), \tag{14}$$

and optimize

$$\mathcal{L}_{\mathrm{rank}}^{q,e} = \frac{1}{|\mathcal{P}_{q,e}|} \sum_{(z^+, z^-) \in \mathcal{P}_{q,e}} \left[1 - s_\phi(q, z^+) + s_\phi(q, z^-)\right]_+, \tag{15}$$

where $[x]_+ = \max(0, x)$. We use the standard margin of 1 and enumerate the full Cartesian product without pair sampling. Question–entity pairs without both positive and negative paths are excluded. For both baselines, the per-pair losses are averaged across the batch.

Both baselines use the same reward-model architecture, candidate paths, optimizer, learning rate, batch size, and training epochs as ROP.

The *w/o question* variant removes the question representation from the reward model while keeping the asymmetric reward-learning objective and all other settings unchanged.

Transferability Settings. We further evaluate whether the learned path rewards can supervise other KG retrieval architectures. We consider RoG (Luo et al., 2024), which generates relation paths with an LLM, and SubgraphRAG (Li et al., 2025), which trains a lightweight triple retriever.

For *RoG* + ROP *Reward Sup.*, we retain the RoG architecture and use the highest-scoring answer-reaching path for each question–entity pair as training supervision:

$$z^*_{q,e} = \arg\max_{z \in \mathcal{Z}^{\mathcal{A}_q}_{q,e}} s_\phi(q, z). \tag{16}$$

For *SubgraphRAG* + ROP *Reward Sup.*, we retain its original shortest-path supervision and reweight each supervised path using the learned reward. For a path z with reward score $c = s_\phi(q, z)$, we assign

$$w(z) = \exp(c - c_{\max}), \qquad c_{\max} = \max_{z' \in \mathcal{Z}_{q,e}} s_\phi(q, z'), \tag{17}$$

where $c_{\max}$ is computed over the forward paths scored for the same question–entity pair. This is equivalent to the temperature-1 softmax score normalized by its maximum. All other training and answer-generation settings follow the corresponding original baselines.

Reward-Guided Path Generation Variants. We consider five variants to examine the two stages of reward-guided path generation.

Distillation only retains path reward distillation and removes GRPO.

GRPO only removes path reward distillation and performs GRPO with the learned path reward directly on the model.

Distillation + GRPO-Ans retains path reward distillation but replaces the learned reward in GRPO with the binary answer-reaching reward

$$u_i^{\text{ans}} = \mathbb{I}\left[\mathcal{E}_{\tilde{z}_i}(e) \cap \mathcal{A}_q \neq \emptyset\right]. \tag{18}$$

Distillation + GRPO-F1 instead uses the answer-set F1 of the retrieved entities as the GRPO reward,

$$u_i^{\text{F1}} = \text{F1}(\mathcal{E}_{\tilde{z}_i}(e),\, \mathcal{A}_q). \tag{19}$$

For both variants, the same output-format, hop-length, and KG-grounding validity constraints as the full model are applied before reward assignment, and all other GRPO settings remain unchanged.

Finally, *SFT + GRPO-Ans* removes the learned reward from both stages. The first stage performs uniform SFT over answer-reaching paths,

$$p_{\text{uni}}(z_i \mid q, e) = \frac{1}{|\mathcal{Z}^{\mathcal{A}_q}_{q,e}|}, \qquad \mathcal{L}_{\text{uniform}} = -\sum_{z_i \in \mathcal{Z}^{\mathcal{A}_q}_{q,e}} p_{\text{uni}}(z_i \mid q, e) \log \pi_\theta(z_i \mid q, e), \tag{20}$$

and the second stage performs GRPO with the binary answer reward in Eq. 18. This variant corresponds to the standard SFT-then-GRPO recipe.

Results. The complete results in Table 11 are consistent with Section 5.3. On Hits@1, the asymmetric, question-conditioned reward again outperforms both conventional objectives, and its benefits transfer to RoG and SubgraphRAG. For path generation, distillation provides the initialization, whereas GRPO with the binary answer reward or answer-set F1 brings no consistent gain over distillation alone on Hits@1 either. Only GRPO with the learned reward improves all three metrics, and the standard SFT-then-GRPO recipe remains the weakest distilled variant.

F.5 Additional SPARQL Path Ranking Results

Table 12 reports SPARQL path ranking results on the development sets, using the full WebQSP development set and 200 randomly sampled CWQ development questions with more than one answer-reaching candidate path, selected as in Section 5.4. The results are consistent with the main analysis: ROP Reward achieves the best hits@1 and MRR on both datasets, leading GPT-4o by 5.0 and 10.6 Hits@1 points on WebQSP and CWQ and the remaining scorers by larger margins.

Table 12: SPARQL path ranking performance (%) on the development sets of WebQSP and CWQ.

Scoring Method	WebQSP		CWQ	
	Hits@1	MRR	Hits@1	MRR
Cosine sim.	58.6	71.6	26.7	42.7
Answer F1	75.9	84.9	21.3	38.2
Pointwise BCE	69.8	79.1	33.9	46.2
Pairwise ranking	69.1	78.9	37.2	48.4
GPT-4o	83.3	88.2	59.7	69.4
RoP Reward	**88.3**	**91.9**	**70.3**	**78.0**

Table 13: KGQA performance (%) on MetaQA. † denotes a fine-tuned LLM answer reasoner, and * denotes results reported in the original paper.

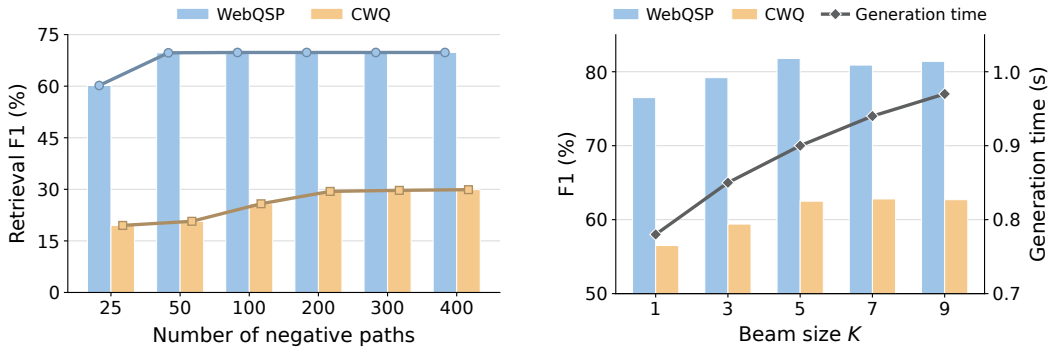
Method	LLM	F1	Hit	Hits@1
ToG (Sun et al., 2024)	GPT-4o	86.3	87.7	80.1
DoG (Ma et al., 2025a)	GPT-4o	93.3	98.3	90.4
RoG (Luo et al., 2024)	LLaMA2-Chat-7B†	88.1	92.7	90.3
DP* (Ma et al., 2025b)	GPT-4o	94.4	96.5	95.2
RoP	GPT-4o	**96.2**	**99.8**	**97.5**

F.6 Results on MetaQA

To evaluate RoP on a different KG, we additionally conduct experiments on MetaQA (Zhang et al., 2018), a 1- to 3-hop KGQA benchmark built on WikiMovies. Following prior work (Ma et al., 2025b;a), we evaluate on 200 questions for each hop.

As shown in Table 13, RoP achieves the best F1, Hit, and Hits@1 among all methods. The effectiveness of reward-guided path generation thus extends to a different KG.

G Parameter Study



(a) Effect of the number of negative paths on grounded entity retrieval F1.

(b) Effect of beam size on answer F1 and generation runtime.

Figure 5: Parameter study of RoP.

In this section, we conduct the parameter study on the development sets of WebQSP and CWQ. Figure 5 summarizes the sensitivity of RoP to the number of sampled negative paths during reward-model training and the beam size used at inference.

Number of Negative Paths. As shown in Figure 5a, increasing the number of sampled negative paths improves grounded entity retrieval F1, with the gains saturating after 50 negatives on WebQSP

Table 14: Example of path reward learning on a CWQ training question. Green denotes the SPARQL-consistent path, while red denotes a spurious answer-reaching path.

Question: What language would be spoken in the region where the Khorshid newspaper is circulated?
Question Entity: Khorshid

Candidate Answer-Reaching Paths:
Path 0: book.periodical.language
Path 1: book.periodical.language →language.human_language.countries_spoken_in →location.country.official_language
Path 2: book.newspaper.circulation_areas →film.film_subject.films →film.film.language
Path 3: book.newspaper.circulation_areas →location.country.languages_spoken
Path 4: book.periodical.language →language.human_language.language_family →language.language_family.languages

Top-ranked Path by RAPL (GPT-4o): book.periodical.language
Top-ranked Path by ROP Reward: book.newspaper.circulation_areas →location.country.languages_spoken

and around 200 negatives on CWQ. We therefore use 50 and 200 negative paths for WebQSP and CWQ, respectively, which provide a favorable trade-off between retrieval quality and training cost.

Beam Size. Figure 5b shows that increasing the beam size improves answer F1 initially, while also increasing generation time. Performance largely saturates beyond $K = 5$, with only marginal changes at larger beam sizes. We therefore set $K = 5$ for all experiments to balance answer quality and inference efficiency.

H CASE STUDY

We provide two examples to illustrate how the learned path rewards distinguish ambiguous answer-reaching paths and how reward-guided path generation affects downstream KG-grounded answer reasoning.

Path Reward Learning.

Table 14 presents a CWQ training example in which multiple candidate paths reach plausible answers. The question asks for the language spoken in the region where the *Khorshid* newspaper is circulated. Path 0 directly retrieves the newspaper language but bypasses the circulation-region constraint in the question. In contrast, Path 3 first reaches the newspaper's circulation region and then retrieves the languages spoken there, matching the annotated SPARQL relation path.

RAPL ranks the shortcut Path 0 highest, whereas the learned ROP reward assigns the highest score to the SPARQL-consistent Path 3. Figure 6 further shows how the reward-induced preferences over the candidate paths evolve during training. The preference gradually shifts toward Path 3, illustrating how the asymmetric objective can differentiate paths that are indistinguishable under binary answer-reaching supervision.

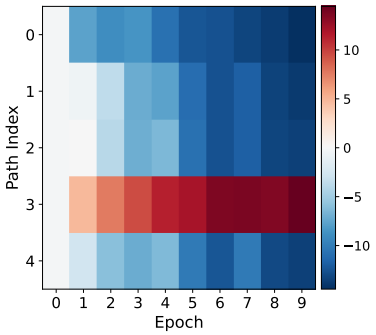


Figure 6: Evolution of learned reward scores for candidate paths in the example from Table 14.

KG-Grounded Answer Reasoning. Table 15 illustrates how path supervision affects downstream KG-grounded answer reasoning. For the WebQSP question "Who were Demeter's siblings?", the variant without learned path rewards generates a spurious parent–children path and an incomplete sibling path, resulting in unresolved Freebase entities in the final prediction. In contrast, ROP generates a grounded relation path that reaches the correct answer, Zeus.

This example complements the quantitative ablations: the learned path reward helps the generator avoid shortcut or incomplete relation paths and provides KG-grounded evidence that reflects the reasoning required by the question.

I ERROR ANALYSIS

Table 15: Example of KG-grounded answer reasoning on a WebQSP test question.

Question	Who were Demeter's siblings?
Answer	Zeus
SFT + GRPO-Ans	***Generated Paths:*** [PATH 1] Start Entity: Demeter Path: fictional_universe.fictional_character.parents →fictional_universe.fictional_character.children End Entities: Hera, Poseidon, Hestia, Hades, Zeus [PATH 2] Start Entity: Demeter Path: fictional_universe.fictional_character.siblings End Entities: m.0gwhv5j, m.0j85m5t
	Predicted Answer: m.0gwhv5j, m.0j85m5t
RoP	***Generated Paths:*** [PATH 1] Start Entity: Demeter Path: fictional_universe.fictional_character.siblings → fictional_universe.sibling_relationship_of_fictional_characters.siblings End Entities: Zeus
	Predicted Answer: Zeus

We analyze test questions with F1 $<$ 1 and categorize the failures into path generation errors, where none of the generated paths matches the SPARQL reference path, and reasoning errors, where reference paths are generated but the answer LLM fails to return the correct answers. As shown in Figure 7, 490 WebQSP questions (30%) and 1,616 CWQ questions (46%) fall into this category. On WebQSP, reasoning errors account for 57.6% of the failures and path generation errors for 42.4%, while on CWQ the two sources are more evenly split, at 51.5% and 48.5%. Neither source dominates on either dataset. Path generation errors propagate by producing incomplete or irrelevant evidence for answer reasoning, whereas reasoning errors arise even when the reference path is retrieved, as the answer LLM fails to identify or compose the correct answers. We discuss the implications in Appendix J.

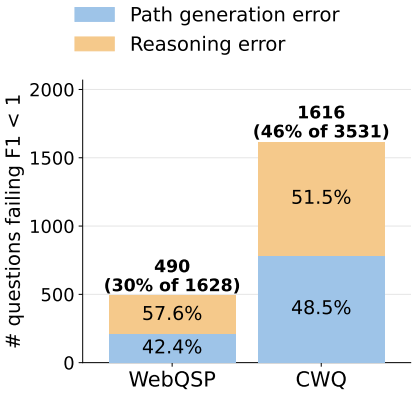


Figure 7: Breakdown of error types among cases with F1 $<$ 1 on WebQSP and CWQ.

J Limitations

RoP follows a staged pipeline of relation path generation, KG grounding, and answer reasoning. As shown in Appendix I, failures occur in both path generation and answer reasoning, and errors in an earlier stage propagate to the final prediction. A unified agentic framework that coordinates path generation, evidence retrieval, and answer reasoning, allowing feedback across stages, may mitigate such errors. Second, RoP relies on the KG to provide sufficient evidence. When relevant facts or connecting paths are missing, the generated paths cannot recover the information needed for reasoning, a limitation shared by all KG-grounded approaches.