PRO-Step: Step-level Process Reward Optimization for Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) enhances Large Language Models (LLMs) by grounding responses in external knowledge. However, multi-hop reasoning remains vulnerable to error propagation, where early retrieval failures confound subsequent steps. Standard outcome-based optimization fails to address this, as rewarding only the final answer inadvertently reinforces spurious successes. While Process Reward Models offer step-level feedback, existing PRM approaches focus on formal logic and lack mechanisms to verify factual grounding against external documents. To bridge this gap, we introduce PRO-STEP, a framework integrating process-level supervision into RAG. First, we train a generative PRM to evaluate both logical validity and evidential grounding. To prevent the policy model from learning these spurious trajectories, we employ PRM-guided Monte Carlo Tree Search (MCTS) to construct preference pairs that contrast valid reasoning steps against flawed ones, systematically filtering out paths that evade outcome-based evaluation. Finally, we optimize the policy model using step-level Direct Preference Optimization (DPO). Experiments on single- and multi-hop QA datasets demonstrate that PRO-STEP mitigates error propagation and outperforms existing outcome- and process-based RAG baselines.

1 Introduction

Large Language Models (LLMs) (OpenAI, 2025; DeepSeek-AI, 2025) have demonstrated capabilities across a wide range of reasoning tasks, including mathematics and code generation. However, they still suffer from hallucinations, often generating plausible but unsupported claims when they lack relevant or up-to-date knowledge. Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) addresses these issues by equipping LLMs with external knowledge sources, improving factual accuracy and timeliness. However, traditional RAG follows a linear retrieve-then-generate workflow. This design struggles with multi-step reasoning tasks that require iterative retrieval and dynamic query reformulation. To overcome this, recent work has proposed agentic RAG systems (Li et al., 2025a; Wang et al., 2025a), enabling models to dynamically decide when and what to retrieve through iterative retrieval, query rewriting, and self-reflection mechanisms (Jiang et al., 2023; Su et al., 2024; Asai et al., 2024).

Early approaches (Wang et al., 2025a) relied on prompt engineering, while Supervised Fine-Tuning (SFT) methods (Li et al., 2025a) directly optimized model parameters for retrieval-augmented workflows. More recently, outcome-supervised Reinforcement Learning (RL) (Ouyang et al., 2022) has achieved improvements: Search-R1 (Jin et al., 2025a) and R1-Searcher (Song et al., 2025) incorporate search engines into the LLM's environment and use final-answer correctness as the reward signal, demonstrating that end-to-end RL can enhance agentic RAG capabilities.

However, outcome-based optimization presents limitations in multi-hop reasoning (Zhang et al., 2025). First, a minor retrieval error in the early steps, such as an imprecise query or an irrelevant document, propagates through subsequent reasoning steps, but outcome-based optimization lacks the means to detect or penalize these intermediate failures. Second, because the reward signal is derived solely from final-answer correctness, it is sparse. The model receives feedback only after completing the entire reasoning chain, requiring more training data to converge.

Process Reward Model (PRM) approaches (Lightman et al., 2024; Luo et al., 2024; Wang et al., 2024; Zheng et al., 2025a) address this limitation by providing step-level feedback that guides models along correct reasoning trajectories. However, applying existing PRM approaches to RAG presents

two challenges. First, they focus on logical consistency in domains like mathematics and cannot verify whether a claim is factually grounded in retrieved documents. Second, most PRM approaches produce only scalar scores without explaining why a step is flawed, limiting the richness of the supervision signal.

To address these challenges, we propose PRO-STEP, a framework that integrates process-level supervision into RAG optimization. PRO-STEP consists of two stages: (1) **Generative PRM training.** We generate diverse Chain-of-Thought trajectories where reasoning and retrieval are interleaved, and train a Generative PRM that evaluates each step with a correctness label and a rationale. (2) **Policy optimization.** We construct step-level preference pairs via Monte Carlo Tree Search (MCTS) guided by the PRM, and apply Direct Preference Optimization (DPO) to train the policy model to reason correctly at each step.

Our contributions are as follows:

- We analyze error propagation in multi-hop RAG and demonstrate the necessity of process supervision to overcome the limitations of outcome-based optimization.
- We propose PRO-STEP, a framework that combines Generative PRM training, PRM-guided MCTS for preference pair construction, and process-supervised DPO. By providing dense, step-level feedback, this approach effectively resolves the reward sparsity issue inherent in outcome-based methods, enabling efficient optimization for retrieval-augmented reasoning.
- Through experiments on various single-hop and multi-hop QA datasets, we demonstrate that PRO-STEP mitigates error propagation and consistently outperforms RAG baselines.

2 Related Work

2.1 Retrieval-Augmented Generation

Early RAG systems followed a retrieve-then-generate paradigm, appending fixed passages to the input prompt (Asai et al., 2024). While effective for single-hop queries, this approach struggles with multi-hop scenarios necessitating iterative retrieval and reasoning. Prompt-based methods introduced uncertainty-triggered retrieval (FLARE; Jiang et al. 2023), dynamic query reformulation (DRAGIN; Su et al. 2024), and interleaved reasoning-retrieval chains (Trivedi et al. 2023). RL-based frameworks subsequently enabled autonomous retrieval decisions: Search-R1, R1-Searcher, and related systems incorporate search environments using final-answer correctness as the reward, showing that end-to-end outcome-based RL improves agentic retrieval performance (Li et al., 2025a; Wang et al., 2025a).

Recently, process-level supervision has been applied to RAG through step-level rewards (Li et al., 2025b; Wang et al., 2025b; Zheng et al., 2025b; Wu et al., 2025) and tree search (Peng and Wei, 2025; Zhang et al., 2025). However, to evaluate intermediate steps, these methods predominantly rely on Monte Carlo rollouts grounded in final-answer correctness. Terminal rewards fail to penalize spurious reasoning, where flawed retrieval or logic accidentally produces the correct answer, thereby assigning high rewards to incorrect intermediate steps. Additionally, executing full-depth simulations for state evaluation incurs high computational costs.

2.2 Process Reward Model Approaches

PRM approaches provide step-level verification for multi-step reasoning. Lightman et al. (2024) first demonstrated that PRM-based verification improves mathematical reasoning by independently verifying steps. Subsequent works scaled PRM training using automated Monte Carlo rollouts (OmegaPRM; Luo et al. 2024) and step-level outcome estimation (Math-Shepherd; Wang et al. 2024).

Moving beyond scalar evaluation, GenPRM (Zhao et al., 2025) introduced a generative paradigm where the model articulates rationales before assigning a correctness label. By generating these rationales, the PRM not only scores steps but explicitly identifies errors, providing richer feedback for policy optimization.

However, existing PRM research remains concentrated on formal reasoning, where step correctness is logically verifiable. RAG environments present a different challenge: each step interleaves internal reasoning with external retrieval, requiring verification of both logical validity and evidential grounding in retrieved passages. While VersaPRM (Zeng et al., 2025) extends PRM approaches to multiple domains, it does not address the dependency between retrieval and reasoning. PRO-STEP addresses this gap by training a Generative PRM

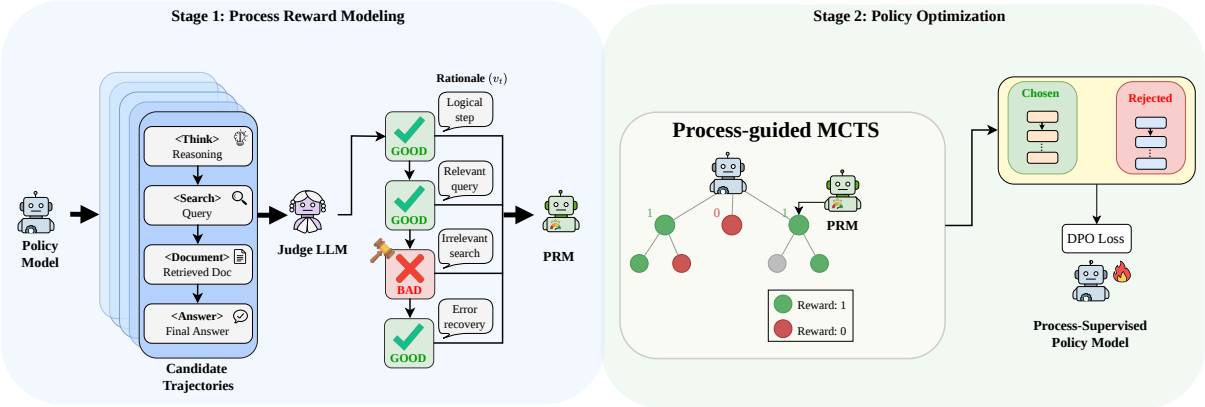

Figure 1: **PRO-STEP Overview:** Detailed architecture of the process-supervised optimization framework.

that evaluates both dimensions, using these rationales to guide MCTS for step-level preference pair construction and policy optimization. These limitations motivate a process-level approach specifically tailored for retrieval-augmented reasoning.

3 PRO-STEP

We introduce PRO-STEP, a two-stage framework. Section 3.1 describes the construction of process-supervised data, including trajectory generation, step annotation, and generative PRM training. Section 3.2 presents policy optimization using PRM-guided MCTS and step-level DPO.

3.1 Process Reward Modeling

Although existing PRM approaches evaluate logical steps in mathematical reasoning, they lack mechanisms for factual grounding in external evidence, failing to outperform majority voting in RAG contexts (Figure 3). To address this limitation, we construct a process-supervised dataset by generating reasoning trajectories, annotating step-level labels, and training a generative PRM specific to RAG.

Trajectory generation. We formulate trajectory generation as a multi-turn interaction between the policy model π_θ (using the system prompt detailed in Section A) and an external search engine $\mathcal{E}$. π_θ generates its internal reasoning within `<think>...</think>`. When external factual grounding is necessary, the model triggers retrieval by generating `<search>...</search>`. Upon detecting a search request, the system halts generation, queries $\mathcal{E}$ with the generated search string, and appends the retrieved documents back into the sequence as `<document>...</document>`. This interleaved generation-and-retrieval cycle continues until the model outputs a final prediction formatted as `<answer>...</answer>` or exhausts the maximum interaction steps. Using this procedure, we sample 16 candidate reasoning trajectories per question from 2,000 instances each of HotpotQA (Yang et al., 2018) and MuSiQue (Trivedi et al., 2022). After discarding incomplete paths or those lacking formatted answers, we retain 31,728 trajectories.

Step annotation. We use QwQ-32B (Qwen Team, 2024b) to annotate each model-generated step s_t based on the evaluation prompt provided in Section A. Given q and the preceding context $s_{\leq t}$, QwQ-32B generates a rationale v_t and a binary correctness label $r_t \in \{0, 1\}$. Unlike prior PRM methods that discard all subsequent steps once an error occurs, we retain the entire trajectory. In RAG contexts, because π_θ can recover from an incorrect step by issuing a revised search query, a local failure ($r_t = 0$) does not invalidate subsequent steps. All 31,728 trajectories are annotated to form $\mathcal{D}_{\text{PRM}}$ (see Appendix B for detailed dataset statistics).

Generative PRM training. We train a generative process reward model π_ψ (initialized from DeepSeek-R1-Distill-8B (DeepSeek-AI, 2025; Qwen Team, 2025)) on $\mathcal{D}_{\text{PRM}}$. The model is optimized to generate both the rationale v_t and the binary correctness label r_t by minimizing the following objective:

$$\mathcal{L}_{\text{PRM}} = -\mathbb{E}_{(q,\tau,v_t,r_t)\sim\mathcal{D}_{\text{PRM}}}\Big[\log \pi_\psi(v_t \mid q, s_{\leq t}) + \log \pi_\psi(r_t \mid q, s_{\leq t}, v_t)\Big] \quad (1)$$

where the expectation is computed over the model-generated steps. At inference, π_ψ first generates the

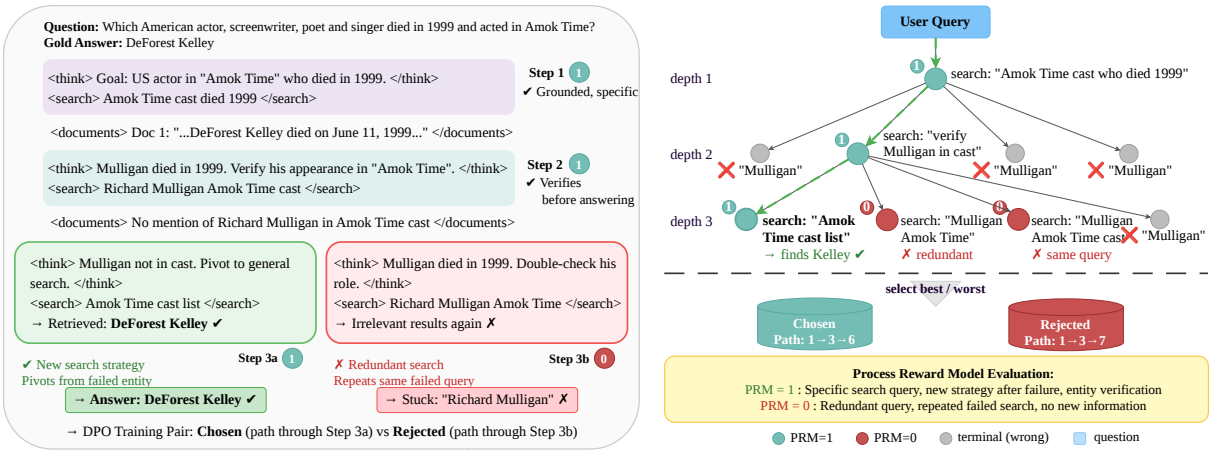

Figure 2: Example of MCTS-based data generation with a PRM. Left: reasoning trace showing PRM evaluation at each step. Right: MCTS tree where green nodes (PRM=1) indicate sound reasoning and red nodes (PRM=0) indicate flawed steps. The best-scoring path becomes chosen and the worst becomes rejected for DPO training.

rationale, and then outputs the binary prediction:

$$\hat{r}(s_t) = \arg\max_{r \in \{0,1\}} \pi_\psi(r \mid q, s_{\leq t}, v_t) \quad (2)$$

3.2 Policy Optimization

Standard process-level methods evaluate intermediate steps by simulating trajectories to completion and scoring the final outcome. However, this approach lacks mechanisms to penalize flawed reasoning that spuriously reaches the correct answer. To mitigate this issue, we construct preference pairs using MCTS guided by π_ψ.

Process-guided MCTS. Within the MCTS framework, each node represents a state $s_t = (q, s_{\leq t})$, and an action corresponds to sampling the next model-generated step from π_θ. If the sampled step is a `<search>` action, $\mathcal{E}$ appends the corresponding `<document>` block before the next action is taken. The tree is built via iterating selection, expansion, and evaluation using upper confidence bound (UCB)-based selection (Kocsis and Szepesvári, 2006) and back-propagation (detailed in Appendix C).

For a terminal node, the reward is defined as the depth-discounted F1 score $Q(s_T)$. For a non-terminal node s_t, we avoid full rollouts by using the binary prediction from π_ψ:

$$Q(s_t) = \hat{r}(s_t) \cdot \gamma^{d(s_t)} \quad (3)$$

where $\gamma \in (0, 1)$ penalizes long trajectories, and $d(\cdot)$ denotes the depth of the node. The prediction $\hat{r}(s_t)$ is computed after node expansion and propagated to all ancestor nodes, updating the expected value $\bar{Q}(s)$ without inflating visit counts.

Preference data construction. For each parent node, we rank its child nodes based on a value function that balances the outcome quality with the assessment from π_ψ:

$$V(s) = \bar{Q}(s) + \alpha \cdot \hat{r}(s) \quad (4)$$

Here, $\bar{Q}(s)$ aggregates the quality of descendant trajectories through back-propagation, while $\hat{r}(s) \in \{0, 1\}$ assesses the correctness of step s. For any pair of sibling nodes s_i and s_j, if the value difference $|V(s_i) - V(s_j)| > \delta$, the sibling with the higher value is designated as the chosen step (s^+) and the other as the rejected step (s^-). Formally, given $V(s_i) > V(s_j)$ and $|V(s_i) - V(s_j)| > \delta$, we assign $s^+ = s_i$ and $s^- = s_j$.

DPO training. We optimize π_θ using step-level DPO (Rafailov et al., 2023). First, we define the implicit utility $u_\theta(x, y_t)$:

$$u_\theta(x, y_t) = \beta \log \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\text{ref}}(y_t \mid x, y_{<t})} \quad (5)$$

Using this utility, we minimize the DPO loss, masking `<document>` blocks from the computation:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{\mathcal{D}_{\text{dpo}}}\Big[\log \sigma(u_\theta(x, y_t^w) - u_\theta(x, y_t^l))\Big] \quad (6)$$

where $\mathcal{D}_{\text{dpo}}$ is the step-level preference dataset of tuples $(x, y_{<t}, y_t^w, y_t^l)$. Here, x denotes the question with system prompt, $y_{<t}$ is the shared reasoning prefix, y_t^w and y_t^l are the chosen and rejected steps at depth t, and the hyperparameter β controls the KL constraint.

Table 1: Comparison of different RAG methods across 5 benchmarks using EM and F1 metrics. **Bold** denotes the best performance, and underline denotes the second-best.

Type	Method	PopQA		HotpotQA		2WikiMulti		Bamboogle		MuSiQue		Average	
		EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Zero-shot	Direct	14.8	19.0	18.7	26.6	24.6	29.3	12.0	19.2	4.0	11.9	14.8	21.2
	Standard RAG	37.4	44.1	31.6	41.9	28.3	34.6	16.8	25.1	4.8	12.3	23.8	31.6
Advanced	FLARE	15.5	19.6	18.8	26.6	24.6	29.0	12.0	19.2	4.0	11.6	15.0	21.2
	Self-RAG	30.0	38.6	18.2	29.6	13.2	25.0	5.6	18.0	4.7	11.8	14.3	24.6
	IRCoT	32.7	40.7	31.2	42.8	26.2	35.1	23.2	33.6	6.2	13.0	23.9	33.0
	Search-o1	10.7	15.0	18.5	26.5	19.6	23.4	25.6	38.5	4.5	11.8	15.8	23.0
RL-based	AutoRAG	24.1	31.2	14.8	24.1	11.0	20.4	23.2	34.3	6.5	13.1	16.1	24.6
	Search-R1	**40.6**	**46.7**	37.8	49.5	34.8	42.5	33.6	43.5	12.9	21.2	32.0	40.7
	ReasonRAG	37.7	44.8	36.3	47.5	**39.8**	46.3	**38.4**	**46.8**	10.5	19.2	32.5	40.9
Ours	**PRO-STEP**	39.5	46.5	**38.0**	**50.3**	39.6	**48.4**	32.8	42.5	**13.2**	**23.0**	**32.6**	**42.1**

4 Experiments

In this section, we present the experimental settings and results.

4.1 Experimental Setup

Datasets and Evaluation Metrics. We evaluate PRO-STEP on five benchmark datasets spanning single-hop and multi-hop question answering (QA). For single-hop QA, we use PopQA (Mallen et al., 2023). For multi-hop QA, we use HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), Bamboogle (Dong et al., 2024), and MuSiQue (Trivedi et al., 2022). We report Exact Match (EM) and F1 score using the FlashRAG evaluation framework (Jin et al., 2025b) for reproducibility.

Implementation Details. We employ Qwen2.5-7B-Instruct (Qwen Team, 2024a) as the primary backbone for PRO-STEP and most baselines. For specific baselines, we utilize their official model checkpoints to ensure a fair comparison: Self-RAG employs its specialized Llama-2-13B variant (Asai et al., 2024), and AutoRAG utilizes Llama-3-8B-Instruct (Yu et al., 2024). For retrieval, we use the 2018 Wikipedia dump consisting of 5.9M passages, indexed with the BGE-base-en-v1.5 (Xiao et al., 2024) retriever, and set the number of retrieved passages to $k = 3$ across all methods to ensure consistency. We train PRO-STEP on a curated mixture of 5,000 multi-hop QA questions, comprising 2,000 instances each from the *training splits* HotpotQA and MuSiQue, and 1,000 from 2WikiMulti. For policy optimization, we leverage 13,183 preference pairs derived through MCTS to facilitate robust step-level learning (Rafailov et al., 2023).

Baselines. We evaluate PRO-STEP against 9 baselines categorized into three groups: **Zero-shot Baselines** (Naive Generation, Standard RAG (Lewis et al., 2020)), **Advanced RAG** featuring adaptive retrieval mechanisms and iterative reasoning capabilities (FLARE (Jiang et al., 2023), Self-RAG, IRCoT (Trivedi et al., 2023), Search-o1 (Li et al., 2025a)), and **RL-based RAG** paradigms that leverage autonomous exploration and test-time scaling (AutoRAG, Search-R1 (Jin et al., 2025a), ReasonRAG (Zhang et al., 2025)). Refer to Section D for more implementation details regarding ReasonRAG and other baselines. All experiments were conducted on 2 NVIDIA A100 (80GB) GPUs.

4.2 Main Results

The main results comparing PRO-STEP with baseline methods across five datasets are presented in Table 1. From the results, we make the following key observations:

Overall results. PRO-STEP achieves the highest average performance (32.6 EM and 42.1 F1), outperforming inference-time scaling and process-level supervision methods like ReasonRAG without requiring multiple LLM calls. While PRO-STEP trails on Bamboogle, this dataset comprises only 125 instances, where a difference of a few questions disproportionately inflates metric gaps.

Limitations of heuristic adaptive retrieval. Advanced RAG methods relying on heuristic triggers

Table 2: Impact of different optimization strategies on PRO-STEP's effectiveness across five benchmarks. **Bold** denotes the best performance, and underline denotes the second-best.

Method	PopQA		HotpotQA		2WikiMulti		Bamboogle		MuSiQue		Average	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
PRO-STEP (SFT)	37.7	44.9	35.8	46.9	32.9	41.1	31.2	41.3	11.4	19.6	29.8	38.7
PRO-STEP (KTO)	38.4	44.8	34.4	45.5	30.4	38.9	**36.0**	**42.5**	11.0	19.0	30.0	38.2
PRO-STEP (DPO)	**39.5**	**46.5**	**38.0**	**50.3**	**39.6**	**48.4**	32.8	**42.5**	**13.2**	**23.0**	**32.6**	**42.1**

FLARE and Self-RAG average only 15.0 and 14.3 EM, significantly underperforming Standard RAG (23.8 EM). Even IRCoT, despite matching Standard RAG on average, underperforms it on complex multi-hop tasks like HotpotQA (31.2 vs. 31.6 EM) and 2WikiMulti (26.2 vs. 28.3 EM). These fixed, rule-based triggers often disrupt generation or fail to provide meaningful context in complex reasoning chains. In contrast, PRO-STEP learns optimal retrieval timing directly via preference optimization, bypassing the inefficiencies of heuristics.

Process vs. outcome in multi-hop reasoning. Process-level optimization significantly improves performance on datasets requiring extended logical chains. For instance, PRO-STEP outperforms the outcome-based baseline, Search-R1, by 4.73 EM points on 2WikiMulti and 1.77 F1 points on MuSiQue. While outcome-based RL struggles with sparse rewards and error propagation from early retrieval mistakes, our process-level supervision assigns immediate, step-by-step feedback. By replacing a single delayed reward with dense, granular learning signals, PRO-STEP not only effectively corrects intermediate logical deviations but also fundamentally resolves the reward sparsity problem. Consequently, this enhanced sample efficiency allows PRO-STEP to surpass the average performance of Search-R1 using only 5,000 training instances, compared to the 90,000 instances required by Search-R1.

4.3 Impact of Optimization Strategies

We evaluate the impact of different policy optimization strategies on PRO-STEP. We design this comparison to validate our hypothesis that DPO is particularly suited for process-level optimization. Specifically, its step-wise contrastive loss directly captures the fine-grained differences between valid reasoning and intermediate flaws. To ensure a fair comparison, all variants are trained using the exact same process-supervised data (Section 3.1). For PRO-STEP (SFT), we use only the chosen steps (y_t^w) from the preference pairs and apply supervised fine-tuning using next-token prediction. For PRO-STEP (KTO), we train on unpaired data using the absolute binary correctness labels ($r_t \in \{0, 1\}$) Kahneman-Tversky Optimization (KTO)(Ethayarajh et al., 2024). Finally, PRO-STEP (DPO) is our default approach, utilizing contrastive preference pairs (y_t^w, y_t^l). Table 2 summarizes the performance of these variants. From the results, we observe the following:

Effectiveness of contrastive pairs (DPO). PRO-STEP (DPO) achieves the highest average performance (32.6 EM and 42.1 F1). By explicitly contrasting successful steps against failed ones, DPO provides a clear gradient for the policy model to distinguish between valid reasoning and logical errors or unhelpful retrievals. This pair-wise contrast proves most effective for internalizing the PRM's step-level guidance.

Instability of unpaired binary signals (KTO). PRO-STEP (KTO) yields mixed results compared to SFT, achieving a slightly higher average EM (30.0) but a lower average F1 (38.2). While KTO leverages both positive and negative signals, the lack of direct relative comparison causes instability across varying datasets. For instance, it outperforms SFT on Bamboogle but suffers noticeable drops on multi-hop tasks like HotpotQA and 2WikiMulti. This indicates that absolute binary rewards are insufficient for consistent multi-step reasoning compared to relative preference pairs.

Limitations of positive-only training (SFT). PRO-STEP (SFT) exhibits limited generalization capability. Because SFT exclusively maximizes the likelihood of successful trajectories, it completely lacks the step-level contrastive signals necessary for process-level learning. Without exposure to

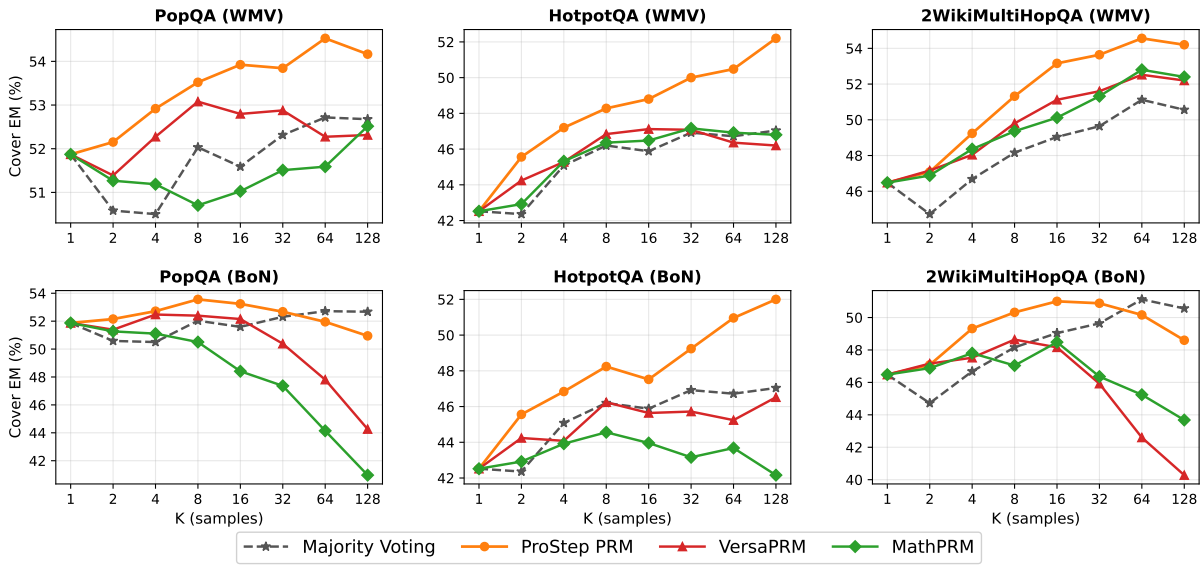

Figure 3: Comparison of WMV (top) and BoN (bottom) using a RAG-specific PRM against open-source PRM approaches on a multi-hop QA dataset.

intermediate flaws, the model cannot learn the fine-grained boundaries between valid reasoning and spurious paths. Consequently, it fails to internalize how to recover from early-stage errors or effectively filter out irrelevant documents, leading to diminished performance on complex multi-hop datasets like MuSiQue (11.4 EM vs. 13.2 EM for DPO).

4.4 Comparison of PRM Approaches on Trajectory Reranking

To validate the necessity of a RAG-specific PRM, we compare our model against existing process reward models. We sample 128 reasoning trajectories per question for 500 instances from PopQA, HotpotQA, and 2WikiMulti using Qwen2.5-7B-Instruct (Qwen Team, 2024a). We evaluate two models, VersaPRM (Zeng et al., 2025) and Qwen2.5-Math-PRM-7B (Yang et al., 2024), alongside a standard Majority Voting (MV) baseline. For selection, we apply best-of-N (BoN) and Weighted Majority Voting (WMV) using the minimum step score of each trajectory, scaling the size of the candidate pool K from 1 to 128. To evaluate the accuracy of the final selected trajectories, we report Cover Exact Match (Cover EM), which measures whether the exact ground truth answer string is successfully contained within the model's generated response.

The results in Figure 3 demonstrate that general-purpose and math-specific PRM approaches are ill-suited for RAG environments. Under the BoN strategy, both VersaPRM and MathPRM exhibit performance degradation as K increases, frequently falling below the MV baseline across all three datasets. Because they evaluate logical coherence without verifying factual grounding, these PRM approaches often assign high scores to hallucinated but plausible trajectories. Consequently, as the candidate pool expands, BoN increasingly selects these confidently flawed trajectories.

In contrast, our PRM consistently improves Cover EM as the candidate pool size increases in both WMV and BoN settings. By evaluating both logical validity and the factual grounding of the retrieved documents, our model filters out spurious trajectories. On 2WikiMulti with WMV ($K = 64$), our PRM achieves 53.1% Cover EM, outperforming MV (49.9% Cover EM) and VersaPRM (51.0% Cover EM). However, under the BoN strategy, our PRM slightly underperforms the MV baseline on PopQA and 2WikiMulti at larger K. Because these datasets inherently contain many valid answers for a single question, MV naturally benefits from counting the frequency of these abundant correct answers as K grows. In contrast, BoN relies solely on a single highest-scored trajectory, missing this frequency advantage.

In this section, we investigate the performance of PRO-STEP under different numbers of retrieved documents and retrieval steps to evaluate its context utilization and planning adaptivity.

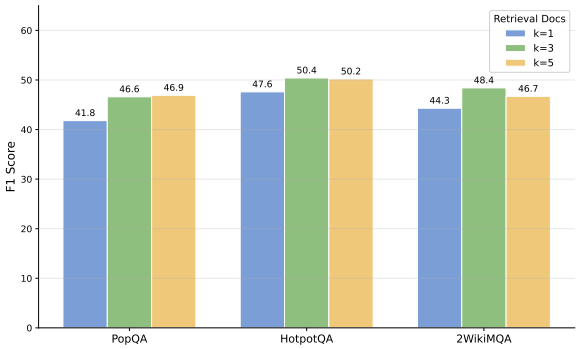

Figure 4: Impact of the number of retrieved documents (k) on performance. PopQA saturates early, while 2WikiMulti exhibits a performance drop at $k = 5$ due to noise.

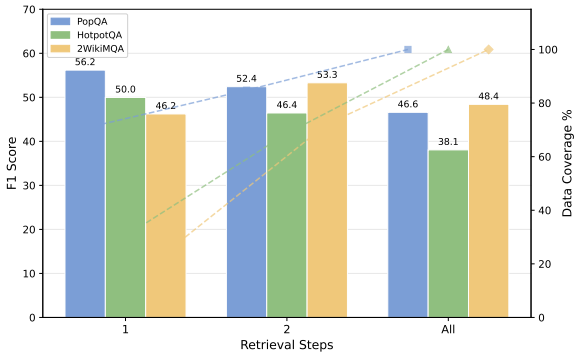

Figure 5: Cumulative average F1 score (bars) and data coverage (dashed lines) across different retrieval steps (1, 2, All) on PopQA, HotpotQA, and 2WikiMulti datasets

4.5 Impact of Retrieval

Impact of the number of retrieved documents. As shown in Figure 4, increasing the retrieval context size from $k = 1$ to $k = 3$ consistently improves performance across all evaluated datasets, yielding an average F1 gain of 3.91. This demonstrates that at $k = 1$, the retriever frequently fails to fetch the relevant documents, leaving the model without the critical evidence needed to answer the query. However, expanding the context further to $k = 5$ introduces noisy, irrelevant documents into the prompt, yielding mixed results. In some cases, the model is robust to this noise; for instance, performance on the single-hop PopQA dataset continues to increase slightly (+0.26 F1). Conversely, this noise can also actively degrade reasoning. On the multi-hop 2WikiMulti benchmark, performance drops at $k = 5$ (-1.76 F1), indicating that excessive irrelevant documents successfully confound the model during complex multi-hop reasoning.

Impact of the number of retrieval steps. Figure 5 illustrates the cumulative average F1 and data coverage across retrieval steps. Cumulative F1 is the average F1 over instances resolved within the given number of steps, and data coverage denotes the fraction of total instances resolved. PRO-STEP autonomously adjusts search depth based on task difficulty: on PopQA, ∼70% of instances are resolved in the first step (F1: 56.2), while on HotpotQA and 2WikiMulti only 30.4% and 24.2% are resolved at step one, with reasoning naturally extending to two or more steps. Notably, on 2WikiMulti, cumulative F1 increases from 46.2 to 53.3 as coverage expands from 24.2% to 72.2% at the second step, indicating that the model successfully acquires critical missing evidence during its second retrieval. Case studies in Section E further illustrate how PRO-STEP recovers from intermediate retrieval failures compared to baselines.

5 Conclusion

We introduced PRO-STEP, a framework that integrates process-level supervision into RAG to mitigate error propagation in multi-hop reasoning. Our approach trains a generative PRM to evaluate both the logical validity and the evidential grounding of intermediate steps. We employ this PRM to guide MCTS, proactively constructing step-level preference pairs by explicitly filtering out spurious trajectories that coincidentally reach correct final outcomes. Using these targeted preference pairs, we optimize the policy model using DPO to enhance its ability to navigate interleaved generation and retrieval. Experiments demonstrate that PRO-STEP effectively addresses intermediate reasoning flaws, consistently outperforming existing outcome-based and standard process-based RAG baselines across diverse QA datasets. By resolving the limitations of sparse outcome rewards, PRO-STEP offers an effective approach for fine-grained policy optimization in multi-hop QA tasks.

Limitations

Distributional mismatch in generative PRM-guided data regeneration. Inspired by generative process reward models (Zhao et al., 2025), we attempted to augment our DPO training data by utilizing our PRM not merely as a scalar scorer, but as a natural language rationale. However, as detailed in Section F, regenerating trajectories using the PRM's explicit natural language feedback led

to a degradation in downstream performance. We attribute this to a fundamental distributional mismatch that disrupted preference optimization. Because the regenerated chosen trajectories (y_w) were explicitly guided by the PRM's feedback, their token distribution shifted significantly away from the reference policy. The DPO objective suffered from reward confounding: the policy model learned to associate the externally injected, verbose feedback style with positive rewards, rather than internalizing the underlying multi-hop reasoning. During test-time inference without such explicit rationale guidance, the model generated verbose hallucinations instead of executing concise RAG actions. This suggests that maintaining action-type consistency between the reference and target distributions is crucial, and naive regeneration from natural language feedback introduces detrimental policy deviations.

Ethical Considerations

This work presents PRO-STEP, a framework for optimizing retrieval-augmented reasoning in large language models. All datasets used in this study (HotpotQA, MuSiQue, 2WikiMultiHopQA, PopQA, Bamboogle) are publicly available benchmarks intended for research purposes. Our training data is derived solely from these existing datasets, and no personally identifiable information is collected or processed. The proposed framework does not introduce new risks beyond those inherent to LLM-based systems, such as residual hallucination or retrieval of outdated information. Practitioners deploying this framework in domains should apply appropriate human oversight.

AI writing assistance was used solely for grammar and language checking of the manuscript.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations (ICLR)*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Zican Dong, Tianyi Tang, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. 2024. BAMBOO: A comprehensive benchmark for evaluating long text modeling capacities of large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Model alignment as prospect theoretic optimization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24.

Xanh Ho, Anh-Khoa Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qiao Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025a. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. In *Proceedings of the 2nd Conference on Language Modeling (COLM 2025)*.

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. 2025b. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In *Companion Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025 - 2 May 2025*. ACM.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European Conference on Machine Learning*. Springer.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025a. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5420–5438.

Yuan Li, Qi Luo, Xiaonan Li, Bufan Li, Qinyuan Cheng, Bo Wang, Yining Zheng, Yuxin Wang, Zhangyue Yin, and Xipeng Qiu. 2025b. R3-RAG: Learning step-by-step reasoning and retrieval for LLMs via reinforcement learning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John

Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*.

OpenAI. 2025. Gpt-5 system card. *arXiv preprint arXiv:2601.03267*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in neural information processing systems 35*. Neurips.

Xianshu Peng and Wei Wei. 2025. GRAT: Guiding retrieval-augmented reasoning through process rewards tree search. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 27861–27875, Vienna, Austria. Association for Computational Linguistics.

Qwen Team. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Qwen Team. 2024b. QwQ: Reflect deeply on the boundaries of the unknown. https://qwenlm.github.io/blog/qwq-32b-preview/. Blog post. Accessed: 2024.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. Dragin: Dynamic retrieval augmented generation based on the information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics (TACL)*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Y. Wang, Z. Liu, and W. X. Zhao. 2025a. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*.

Zhao Wang, Ziliang Zhao, and Zhicheng Dou. 2025b. Prorag: Process-oriented optimization for retrieval-augmented generation. *arXiv preprint arXiv:2601.21912*.

Peilin Wu, Mian Zhang, Kun Wan, Wentian Zhao, Kaiyu He, Xinya Du, and Zhiyu Chen. 2025. Hiprag: Hierarchical planning in retrieval-augmented generation. *arXiv preprint arXiv:2510.07794*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

An Yang and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tian Yu, Shaolei Zhang, and Yang Feng. 2024. Auto-rag: Autonomous retrieval-augmented generation for large language models. *arXiv preprint arXiv:2411.19443*.

Thomas Zeng, Shuibai Zhang, Shutong Wu, Christian Classen, Daewon Chae, Ethan Ewer, Minjae Lee, Heeju Kim, Wonjun Kang, Jackson Kunde, Ying Fan, Jungtaek Kim, Hyung Il Koo, Kannan Ramchandran,

Dimitris Papailiopoulos, and Kangwook Lee. 2025. Versaprm: Versatile process reward modeling across domains. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, Ruiming Tang, and Xiangyu Zhao. 2025. Process vs. outcome reward: Which is better for agentic RAG reinforcement learning. In *39th Conference on Neural Information Processing Systems (NeurIPS 2025)*.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. 2025. Genprm: Scaling test-time compute of process reward models via generative reasoning. *arXiv preprint arXiv:2504.00891*.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025a. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. 2025b. StepSearch: Igniting LLMs search ability via step-wise proximal policy optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

A Prompts and Models Reference

In this section, we detail the specific models and prompts used for both the policy model's trajectory generation and the PRM's step-level evaluation.

Policy Model (Trajectory Generation & Evaluation)

System Prompt: You are a helpful assistant who is good at answering questions with multi-turn search engine calling. To answer questions, you must first reason through the available information using `<think>` and `</think>`. If you identify missing knowledge, you may issue a search request using `<search> query </search>` at any time. The retrieval system will provide you with relevant documents enclosed in `<documents>` and `</documents>`. You can search as many times as you want. Once you have sufficient information or if you find no further external knowledge is needed, directly provide your final answer. Ensure your answer is concise, using nouns or short phrases whenever possible. Conclude with: "So the answer is `<answer>answer</answer>`".

User Prompt: Question: {question}

PRM (Training Data Generation) This prompt is used for single-turn, full trajectory evaluation to generate training data. Note that while the gold answer is provided as a reference, the labeling criteria are strictly evidence-based (R1-R6).

System Prompt: You are a strict process supervisor for a multi-hop question answering agent that uses retrieval-augmented generation (RAG). Your task is to evaluate each step of the agent's trajectory and assign a binary label: GOOD or BAD.
The agent operates using four XML-tagged actions: `<think>` (Internal reasoning), `<search>` (Retrieval query), `<documents>` (Retrieved passages), and `<answer>` (Final answer).

[Labeling Principles]
1. Default label is BAD. Assign GOOD only when a step makes a verifiable contribution toward answering.
2. No info gain means BAD (e.g., restating the question, repeating prior reasoning).
3. Do not use own world knowledge. Unverified claims not in `<documents>` are hallucination.

[Evaluation Criteria]
R1. Entity/Relation: BAD if `<think>` misidentifies an entity, reverses a relation, or drifts. GOOD if it maintains correct grounding.
R2. Search: GOOD if query is specific, returns relevant `<documents>`, or tries a new strategy after a failure. BAD if it repeats a failed attempt, is vague, or returns irrelevant data.
R3. Reasoning: GOOD if it extracts useful info from `<documents>` or identifies a concrete knowledge gap. BAD if it merely restates, repeats, or hallucinates.
R4. Answer: BAD if empty, uncertain, requires unsupported logical leaps, or contradicts `<documents>`. GOOD if specific and logically derivable.
R5. Recovery: GOOD if agent issues a new, meaningfully different `<search>` after a failure. BAD if it proceeds to `<answer>` ignoring the failure.
R6. Overconfidence: BAD if agent produces `<answer>` without prior successful search.

User Prompt:
Question: {question} | Ground Truth: {gold_answer}
Trajectory
{full trajectory with all steps}

Expected Output: Return a JSON array: `[{"step": 1, "label": "GOOD", "reasoning": "..."}, ...]`

PRM (MCTS Inline Scoring Prompt) This prompt is used for real-time evaluation during the MCTS expansion phase.

System Prompt: You are a step-level PRM for evaluating reasoning quality in multi-hop question answering. The trajectory uses XML tags: `<think>`, `<search>`, `<answer>`, `<documents>`. Analyze each step's logical soundness and evidence grounding. First explain your reasoning inside `[REASONING]` tags, then output a label (1=good, 0=bad).

User Prompt:
Question: {question}
Previous Steps:
Step 1
`<think>...</think>` `<search>...</search>`
`<documents>...</documents>`
...
Current Step to Evaluate:
Step N
`<think>...</think>` `<search>...</search>`

Expected Output Format:
[REASONING] {PRM reasoning} [/REASONING]
Label: {1 or 0}

B Data Statistics

We present detailed statistics of the datasets used for training both the PRM and the policy model via DPO.

PRM Training Data. To train the PRM, we sampled 31,728 reasoning trajectories across 2,000 questions, resulting in 109,664 individual steps. To validate the reliability of these automatically assigned labels, we manually inspected 50 randomly sampled trajectories, achieving a high human agreement rate of 84% (95% CI: [72.0%, 92.0%]). Table 3 summarizes the label distribution. The majority of the steps are search actions (71.2%), reflecting the highly interactive nature of our multi-hop RAG environment. Notably, the `reason` action (internal thought) accounts for only 0.5% of the steps and has a high rejection rate (73.7% BAD), indicating that the PRM penalizes unsupported internal hallucinations, forcing the model to rely on external search.

Step Type	Count (%)		GOOD		BAD	
Search	78,055	(71.2%)	55,710 (71.4%)		22,345	(28.6%)
Answer	31,062	(28.3%)	14,436 (46.5%)		16,626	(53.5%)
Reason	547	(0.5%)	144 (26.3%)		403	(73.7%)
Total	**109,664**	**(100%)**	**70,290 (64.1%)**		**39,374**	**(35.9%)**

Table 3: Statistics of the PRM training dataset (v8). The dataset consists of 109,664 total steps generated from 2,000 questions, averaging 3.5 steps per trajectory.

DPO Preference Pairs. For policy optimization, we constructed 13,183 contrastive pairs from 4,114 unique questions, yielding approximately 3.2 pairs per question. As detailed in Table 4, the DPO pairs exhibit rich diversity in action types, tree depths, and reward margins. Over 80% of the comparisons pit actions of the same type against each other (e.g., search vs. search), allowing the policy to learn fine-grained strategic differences rather than mere format adherence. Furthermore, the step-level distribution peaks at depth 2 (36.8%) and depth 3 (33.1%), directly aligning with the multi-hop reasoning requirements. The reward difference between chosen and rejected steps ensures a robust gradient, with 41.1% of pairs having a substantial margin (≥ 0.20).

Pair Type	%	Reward Diff Margin	%
Search vs. Search	54.9	< 0.05 (Low)	26.4
Answer vs. Answer	25.9	$0.05 - 0.10$	14.9
Search vs. Answer	12.1	$0.10 - 0.20$	17.6
Answer vs. Search	6.0	$0.20 - 0.50$	30.3
–	–	≥ 0.50 (High)	10.8
Step Level (Depth)	**%**	**Action (Ch / Rej)**	**%**
Step 1	19.3	Search	67.2 / 61.0
Step 2	36.8	Answer	32.4 / 38.2
Step 3	33.1	Reason	0.4 / 0.8
Step 4	9.4	–	–
Step 5+	1.4	–	–

Table 4: Detailed statistics of the 13,183 DPO preference pairs. Action percentages represent the proportion of Chosen / Rejected actions respectively.

C MCTS Implementation Details

This section provides the detailed mathematical formulations and algorithmic mechanics for the MCTS procedure introduced in Section 3.2. Specifically, we detail how the step-level feedback from our generative PRM is integrated into the tree search to actively balance exploration and exploitation, ultimately guiding the generation of high-quality, process-supervised trajectories for PROSTEP.

Selection.

$$\mathrm{UCB}(s_t) = \bar{Q}(s_t) + c\sqrt{\frac{\ln N(\mathrm{parent}(s_t))}{N(s_t)}}$$

where $N(s_t)$ is the visit count of s_t, $\bar{Q}(s_t) = W(s_t)/N(s_t)$ is its mean backed-up value, $W(s_t)$ is its cumulative backed-up value, and c is the exploration constant. Unvisited nodes ($N = 0$) are selected immediately.

Terminal reward.

$$Q(s_T) = \mathrm{F1}(s_T,\, y^*) \cdot \gamma^{d(s_T)}$$

where $\gamma \in (0, 1)$ discourages unnecessarily long trajectories and $d(\cdot)$ denotes node depth.

Back-propagation. After evaluating a terminal node, we propagate $Q(s_T)$ to all ancestors:

$$N(s) \leftarrow N(s) + 1, \qquad W(s) \leftarrow W(s) + Q(s_T)$$

When π_ψ evaluates a non-terminal node s_t, we apply a retroactive update to its ancestors without changing their visit counts:

$$W(s) \leftarrow W(s) + \hat{r}(s_t) \cdot \gamma^{d(s_t)}$$

MCTS Parameters. During the process-supervised data generation phase, we utilize MCTS to explore reasoning trajectories. Table 5 details the specific hyperparameter values used for the tree search. To balance the reward scale, the terminal F1 score is discounted by 0.9 per depth level, and the intermediate step score is computed as a weighted sum of the backed-up F1 reward and the PRM's feedback ($\alpha = 0.3$). Because the PRM provides binary labels $\{0, 1\}$ whereas the F1 score is continuous in $[0, 1]$, this weight allows the process reward to break ties among steps with similar outcomes without overriding substantial F1 differences.

Parameter	Value
Branching factor (K)	2
Max depth	7
Iterations per question	64
F1 discount (γ)	0.9 (Reward $= \text{F1} \cdot 0.9^d$)
PRM weight (α)	0.3 (Combined $= \text{Reward} + 0.3 \cdot \text{PRM}$)
PRM scoring interval	Every 4 iterations
Retrieved docs per search	3
Preference margin (δ)	0.01

Table 5: Hyperparameters for the MCTS rollout phase.

D Implementation Details and Hyperparameters

Training Hyperparameters. Table 6 summarizes the hyperparameter configurations for training the policy model across different optimization strategies (SFT, DPO, KTO). All policy models share the same base model (Qwen2.5-7B-Instruct) and apply document masking during training to prevent the model from memorizing the retrieved context. We utilize LoRA for parameter-efficient fine-tuning across all models, targeting all linear layers.

Parameter	SFT	DPO	KTO
Base Model		Qwen2.5-7B-Instruct	
Data Size	13,183 (Chosen)	13,183 pairs	13,183 pairs
Objective	Next-token	Sigmoid ($\beta = 0.1$)	KTO ($\beta = 0.1$)
Learning Rate	2e-5	2e-5	5e-6
Effective Batch Size	16	16	16
Epochs	1	1	1
Max Length	8,192	8,192	8,192
Document Masking	Yes	Yes	Yes
LoRA Configuration			
Rank (r)	64	64	64
Alpha (α)	128	128	128
Dropout	0.05	0.05	0.05
Target Modules		all-linear	

Table 6: Training hyperparameters for the Policy models. The effective batch size of 16 is achieved via 1×8 accumulation $\times$ 2 GPUs (for SFT/DPO) and 2×4 accumulation $\times$ 2 GPUs (for KTO).

E Case Study

We present qualitative examples to highlight the fundamental limitation of outcome-based learning: even when generating reasoning step-by-step, models trained solely on final answers fail to penalize intermediate logical gaps. In contrast, PRO-STEP's PRM evaluates the validity of every single step, effectively preventing early mistakes from ruining the entire trajectory.

Case Study 1: Resolving Entity Confusion. As shown in Figure 6, the baseline confuses the target "Vernon Sewell" with "John Sewell." Blind to this semantic drift, it prematurely outputs an incorrect answer at Step 2. Conversely, PRO-STEP's PRM immediately detects this entity mismatch and assigns a BAD signal. Forced to reject the spurious path, the model persistently explores alternative queries until finding the correct entity at Step 9.

Case Study 2: Preventing Error Propagation. Figure 7 demonstrates the danger of error propagation in multi-hop queries. The baseline (Reason-RAG) incorrectly decomposes the first hop (e.g., targeting "NBC" instead of "CBS"). Without step-level correction, it stubbornly repeats the same futile search four times before hallucinating. PRO-STEP, however, receives step-level PRM verification. By recognizing the invalid intermediate conclusion early, it pivots its strategy and successfully constructs the correct logical chain (NBC $\rightarrow$ CBS $\rightarrow$ Date).

F Analysis of PRM-Guided Data Regeneration

To explore the potential of scaling process-supervised data, we conducted an ablation study on data regeneration using our generative PRM. During the MCTS rollout, many trajectories inevitably lead to incorrect final answers or accumulate low step-level rewards. Rather than discarding these suboptimal trajectories, we sought to salvage them by prompting the policy model with the PRM's natural language feedback at the points of failure, instructing it to synthesize corrected reasoning steps. Table 7 summarizes the impact of these augmented datasets on the HotpotQA benchmark.

The inclusion of regenerated pairs consistently led to severe performance degradation. Expanding the dataset to 17,639 pairs (V4) caused a 7.18% absolute drop in the F1 score, and adding even a

Dataset Setup	Pairs	Description	EM	F1
Original (V1)	13,183	MCTS baseline data	**38.06**	**50.36**
+ Regenerated (Merged)	17,639	+ 4,456 PRM-guided pairs	32.36	43.18
+ Partial Subset (V4b)	14,239	+ 1,056 PRM-guided pairs	24.02	32.73

Table 7: Impact of generative PRM-guided data regeneration on HotpotQA performance. The inclusion of regenerated trajectories guided by the PRM's natural language feedback consistently degrades both EM and F1 scores, highlighting the vulnerability of DPO to distributional shifts.

partial subset of the regenerated data (V4b) drastically reduced the EM score to 24.02%.

This failure exposes a critical vulnerability in DPO. DPO relies on the subtle contrast between chosen (y_w) and rejected (y_l) trajectories generated by the same base policy. By forcing the model to regenerate y_w using an external PRM's natural language prompt, we inadvertently introduced a severe distributional shift. The regenerated pairs internalized the PRM's descriptive language—explaining *why* an action should be taken—rather than simply taking the concise action (e.g., `Search[query]`). Consequently, the optimization process minimized the DPO loss not by improving the model's actual retrieval or reasoning logic, but by overfitting to this verbose, out-of-distribution style. This style-reward confounding fundamentally disrupted the model's ability to autonomously execute iterative RAG workflows during inference.

Code Availability

The implementation is available at https://anonymous.4open.science/r/PRO-Step-8778.

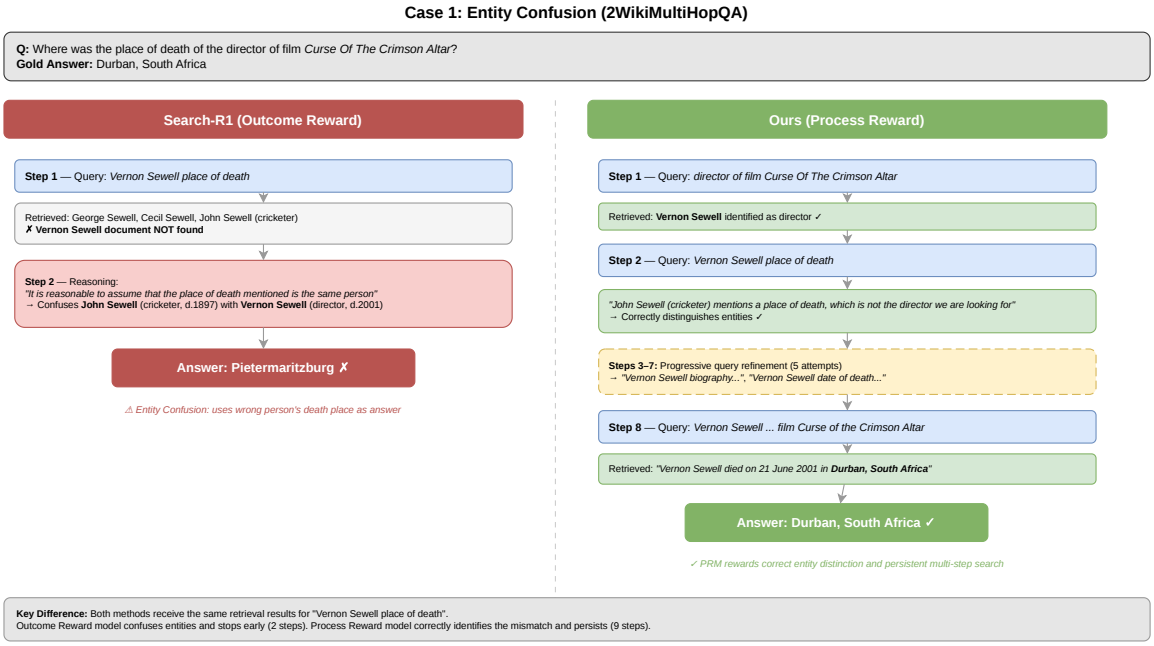

Figure 6: Case Study 1: Entity confusion. The baseline prematurely answers based on a mismatched entity (John Sewell), whereas PRO-STEP rejects the flawed step via PRM feedback and persistently searches for the correct target.

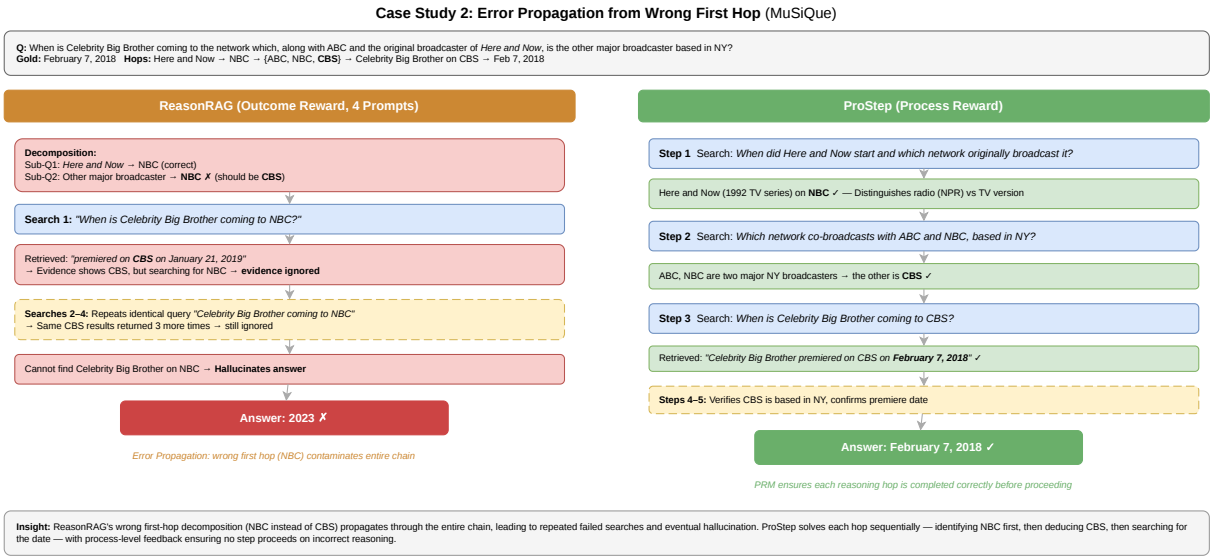

Figure 7: Case Study 2: Error propagation. The baseline repeats useless searches due to a failed first hop. PRO-STEP evaluates each hop, recovers from the initial error, and completes the logical chain.