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

Anonymous ACL submission

## Abstract

Retrieval-Augmented Generation enhances Large Language Models by grounding responses in external knowledge, but multi-hop reasoning remains vulnerable to error propagation, where early retrieval failures confound subsequent steps. Standard outcome-based optimization only rewards the final answer, leaving intermediate retrieval and reasoning errors undetected. While existing process-based methods introduce step-level signals, they still score each step against the final answer, rewarding spurious successes where flawed retrieval coincidentally produces the correct answer. Step-level supervision in RAG requires evaluating both logical validity and evidential grounding at each step. We introduce PRO-STEP: we train a generative PRM that evaluates both dimensions, employ PRM-guided value tree search to construct preference pairs contrasting valid steps against flawed ones, and optimize the policy via step-level Direct Preference Optimization. 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 value tree search guided (VTS) 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 identify that existing process-level RAG methods still optimize against final-answer outcomes, failing to penalize spurious reasoning because they lack mechanisms to evaluate both logical validity and evidential grounding at each step.
- We propose PRO-STEP, a framework that combines Generative PRM training, PRM-guided VTS for preference pair construction, and process-supervised DPO. By providing dense step-level feedback, this approach mitigates the reward sparsity issue inherent in outcome-based methods, enabling efficient optimization for retrieval-augmented reasoning.
- Through experiments on single-hop and multi-hop QA datasets, we demonstrate that PRO-STEP outperforms both outcome- and process-based 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). Despite this success, outcome-only rewards remain sparse and provide no signal on intermediate retrieval or reasoning quality, motivating recent shifts toward process-level supervision.

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, these methods still score intermediate steps against final-answer correctness, failing to filter out spurious successes where flawed retrieval coincidentally produces the correct answer.

### 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 that evaluates both dimensions, providing step-level signals that guide value tree search for preference pair construction and policy optimization.

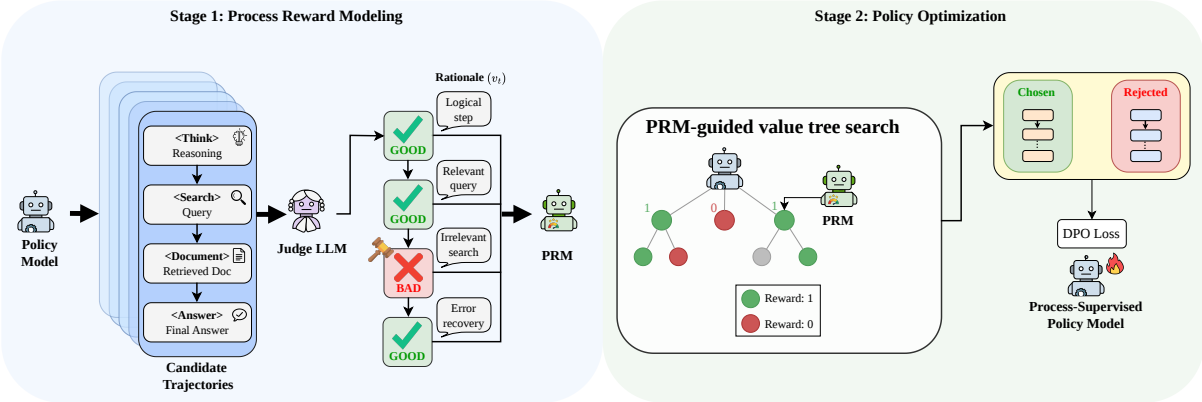


Figure 1: **PRO-STEP Overview:** Detailed architecture of the process-supervised optimization framework.

## 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 a PRM-guided value tree search 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 $\pi_\theta$ (using the prompt in Section A) and an external search engine $\mathcal{E}$. $\pi_\theta$ 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 `<documents>...</documents>`. 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 $s_{<t}$, 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 $\pi_\theta$ 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 $\mathcal{D}_{\text{PRM}}$ (dataset statistics in Section B, labeling rubric in Section A, and annotation quality audit in Section E).

**Generative PRM training.** We train a generative process reward model $\pi_\psi$ (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:

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

where the expectation is computed over the model-generated steps. At inference, $\pi_\psi$ first generates the rationale, and then outputs the binary prediction:

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

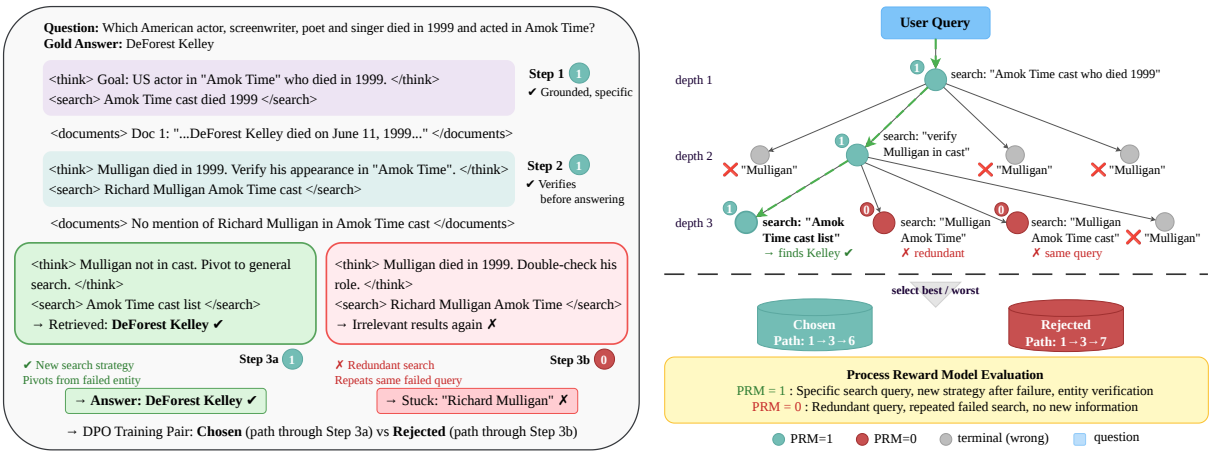


Figure 2: Example of tree-search data generation with a PRM. Left: reasoning trace showing PRM evaluation at each step. Right: search tree where green nodes (PRM=1) indicate sound reasoning and red nodes (PRM=0) indicate flawed steps. The best-scoring path becomes chosen and the worst becomes rejected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

### 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. Unlike prior process-based methods that still rely heavily on final outcomes, PRO-STEP actively filters out these spurious successes by using a PRM-guided value tree search to explicitly penalize flawed intermediate steps.

**PRM-guided Value Tree Search.** Following modern tree search paradigms such as AlphaZero (Silver et al., 2017), we adopt a value tree search framework that explicitly replaces standard random rollouts with direct value estimation. Within this framework, each node represents a state $s_t = (q, s_{\leq t})$, and an action corresponds to sampling the next model-generated step from $\pi_\theta$. 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 the depth-discounted F1 score $Q(s_T)$. For a non-terminal node $s_t$, we directly use the binary prediction from $\pi_\psi$:

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

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 $\pi_\psi$:

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

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 sibling pair $s_i$ and $s_j$ where $V(s_i) - V(s_j) > \delta$, we assign the higher-valued node as the chosen step ($s^+ = s_i$) and the other as the rejected step ($s^- = s_j$).

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

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

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

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

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 $\beta$ 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 underlining denotes the second-best. † indicates statistically significant improvements ($p < 0.05$) over representative RL baselines (Search-R1, ReasonRAG) using a two-sided paired $t$-test. Full significance tests, including comparisons with StepSearch, are detailed in Appendix F.

| 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 | **40.7** | 46.8 | 37.9 | 49.6 | 34.9 | 42.5 | 33.6 | 43.6 | 13.0 | 21.2 | 32.0 | 40.7 |
| | ReasonRAG | 37.8 | 44.9 | 36.4 | 47.5 | 39.8 | 46.3 | **38.4** | 46.9 | 10.6 | 19.2 | 32.6 | 41.0 |
| | StepSearch | 39.2 | 45.0 | **38.7** | 50.7 | 40.4 | 47.1 | 33.6 | 44.2 | **13.8** | **23.1** | 33.2 | 42.0 |
| **Ours** | **PRO-STEP** | 40.5 | **47.4**† | **38.7** | **51.6**† | **44.1**† | **51.4**† | 36.8 | **47.6** | 12.5 | 22.4 | **34.5**† | **44.1**† |

## 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* of HotpotQA and MuSiQue, and 1,000 from 2WikiMultiHopQA. For PRM training, we sample candidate trajectories per question to construct step-level annotations. For policy optimization, we then construct step-level preference pairs via PRM-guided VTS. Detailed data statistics and filtering procedures are deferred to Section B.

**Baselines.** We evaluate PRO-STEP against 10 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, including *outcome-reward* methods (AutoRAG, Search-R1 (Jin et al., 2025a)) and *process-reward* methods (ReasonRAG (Zhang et al., 2025), StepSearch (Zheng et al., 2025c)). See Section D for implementation details.

### 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 across all evaluated methods, recording 34.5 EM and 44.1 F1, robustly outper-

Table 2: Component ablation study evaluating the impact of the Process Reward Model (PRM). The table reports EM and F1 scores across 5 benchmarks. **Bold** denotes the better performance.

| Variant | PopQA | | HotpotQA | | 2WikiMulti | | Bamboogle | | MuSiQue | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EM | F1 | EM | F1 | EM | F1 | EM | F1 | EM | F1 | EM | F1 |
| **PRO-STEP** | **40.5** | **47.4** | **38.7** | **51.6** | **44.1** | **51.4** | **36.8** | **47.6** | **12.5** | **22.4** | **34.5** | **44.1** |
| Without PRM | 38.5 | 45.0 | 36.8 | 49.3 | 41.4 | 48.4 | 34.4 | 44.2 | 12.3 | **22.4** | 32.7 | 41.9 |

Table 3: Impact of different optimization strategies on PRO-STEP's effectiveness across five benchmarks. **Bold** denotes the best performance, and underlining 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) | 38.8 | 45.7 | 36.7 | 48.3 | 32.9 | 41.3 | 31.2 | 40.4 | 11.3 | 20.0 | 30.2 | 39.1 |
| PRO-STEP (KTO) | **41.8** | 46.3 | 32.3 | 43.5 | 36.1 | 42.0 | 20.8 | 32.7 | 8.9 | 18.5 | 28.0 | 36.6 |
| **PRO-STEP (DPO)** | 40.5 | **47.4** | **38.7** | **51.6** | **44.1** | **51.4** | **36.8** | **47.6** | **12.5** | **22.4** | **34.5** | **44.1** |

forming zero-shot, advanced RAG, and RL-based RAG baselines (significance tests in Section F, generalization across backbones in Section G).

**Limitations of heuristic adaptive retrieval.** Advanced RAG methods relying on heuristic triggers, such as FLARE and Self-RAG, underperform the much simpler Standard RAG on average (approximately 15.0 vs. 23.8 EM). Even IRCoT falls behind Standard RAG on complex multi-hop tasks such as HotpotQA and 2WikiMultiHopQA. In contrast, PRO-STEP learns 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. On average, PRO-STEP outperforms the outcome-based Search-R1 by 2.5 EM and 3.4 F1 points. This gap widens substantially on complex multi-hop tasks such as 2WikiMultiHopQA, where our method achieves a 9.2 EM and 8.9 F1 improvement. The dense step-level feedback alleviates the reward sparsity problem and improves sample efficiency: PRO-STEP surpasses Search-R1 using 5k seed questions, whereas Search-R1 reports roughly 17k training instances.

**Comparison with process-reward baselines.** PRO-STEP demonstrates superior average performance compared to other process-level methods like ReasonRAG and StepSearch. While these baselines operate at the process level, they optimize each step against the final F1 score, which fails to penalize spurious reasoning where flawed retrieval accidentally produces the correct answer. In contrast, our generative PRM evaluates the logical and evidential validity of each step, filtering out such false positives. Notably, prior methods distill reasoning trajectories from frontier models (e.g., GPT-4o (Hurst et al., 2024)), and some additionally invoke multiple LLM calls during inference for document refinement. PRO-STEP avoids both, internalizing step-level guidance into the policy through PRM-guided VTS, yielding higher average results with a single model at inference.

### 4.3 Effectiveness of Process Reward Model

To isolate the contribution of our generative PRM, we compare PRO-STEP against a variant without PRM scoring that constructs preference pairs using only the depth-discounted F1 reward. As shown in Table 2, removing the PRM degrades average performance from 34.5 to 32.7 EM and from 44.1 to 41.9 F1, with the largest drops on multi-hop datasets such as 2WikiMultiHopQA (-2.7 EM) and Bamboogle (-2.4 EM). Without PRM evaluation, preference pairs rely solely on final-answer correctness and cannot penalize spurious reasoning that coincidentally leads to correct answers, which the PRM filters out by scoring step-level validity. This validates that considering both logical validity and evidential grounding is essential for RAG.

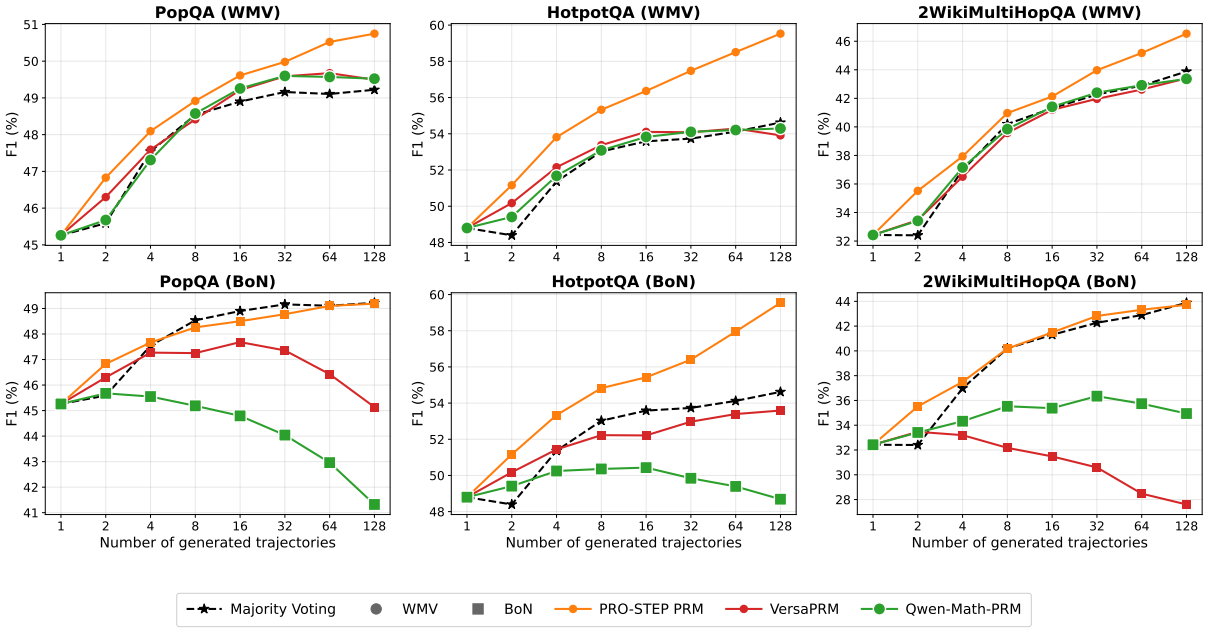


Figure 3: Comparison of WMV (top) and BoN (bottom) using a RAG-specific PRM against open-source PRM approaches on a multi-hop QA dataset.

### 4.4 Impact of Optimization Strategies

We compare three optimization strategies on the same process-supervised data to validate that DPO is best suited for process-level learning. PRO-STEP (SFT) trains only on the chosen steps $y_t^w$ via next-token prediction. PRO-STEP (KTO) uses unpaired binary correctness labels $r_t \in \{0, 1\}$ with Kahneman-Tversky Optimization (KTO) (Ethayarajh et al., 2024). PRO-STEP (DPO) is our default approach using contrastive pairs $y_t^w, y_t^l$.

As shown in Table 3, DPO achieves the highest average performance with 34.5 EM and 44.1 F1 by providing explicit step-wise contrast between valid and flawed reasoning. SFT lacks contrastive signals and cannot learn to recover from intermediate errors, leading to weaker performance on complex multi-hop tasks such as 11.3 vs. 12.5 EM on MuSiQue. KTO underperforms even SFT on average with 28.0 EM and 36.6 F1, as absolute binary rewards without relative comparison cause instability across datasets such as 20.8 vs. 31.2 EM on Bamboogle. These results confirm that pairwise contrast, rather than positive-only or unpaired signals, is essential for internalizing the PRM's step-level guidance.

### 4.5 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 2WikiMultiHopQA 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 two trajectory-selection baselines: majority voting (MV), which ignores PRM scores, and weighted majority voting (WMV), which weights each candidate by the minimum step score assigned by a PRM. We also report best-of-$N$ (BoN), which directly selects the highest-scoring trajectory, while scaling the candidate pool size $K$ from 1 to 128. To evaluate the quality of the final selected trajectories, we report token-level F1.

The results in Figure 3 demonstrate that general-purpose and math-specific PRM approaches struggle in RAG environments. Under the BoN strategy, both VersaPRM and MathPRM degrade 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 F1 as the candidate pool size increases under WMV. By evaluating both logical validity and the factual

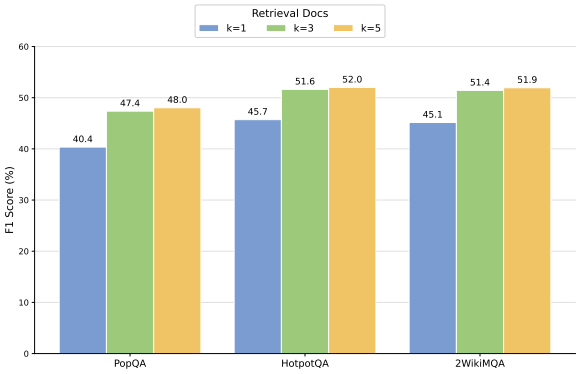


Figure 4: Impact of the number of retrieved documents ($k$) on performance.

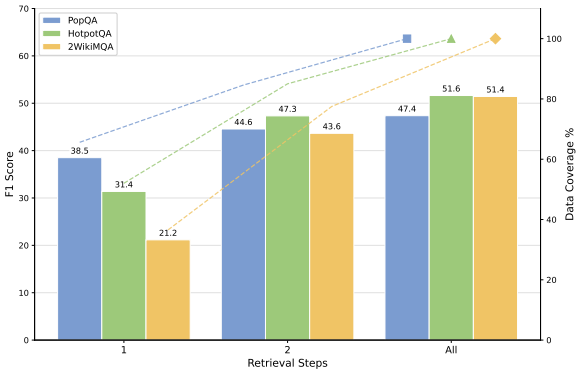


Figure 5: Cumulative average F1 score (bars) and data coverage (dashed lines) across different retrieval steps (1, 2, All) on PopQA, HotpotQA, and 2WikiMultiHopQA.

grounding of the retrieved documents, our model filters out spurious trajectories. At $K = 128$ with WMV, our PRM achieves **46.5 F1** on 2WikiMultiHopQA, **59.5 F1** on HotpotQA, and **50.8 F1** on PopQA, outperforming both MV (43.9 / 54.6 / 49.2) and VersaPRM (43.4 / 53.9 / 49.5) across all three datasets. Under the BoN strategy, our PRM remains stable as $K$ grows, performing comparably to MV on PopQA and 2WikiMultiHopQA (within 0.2 F1) and substantially outperforming it on HotpotQA (+4.9 F1), while VersaPRM and MathPRM degrade sharply at larger $K$. These results validate our motivation for building a RAG-specific PRM that jointly evaluates logical validity and evidential grounding, rather than relying on general-purpose reasoning critics.

### 4.6 Impact of Retrieval

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.

**Impact of the number of retrieved documents.** Figure 4 compares PRO-STEP under different top-$k$ retrieval settings. Increasing $k = 1$ to $k = 3$ consistently improves performance across all datasets (average +6.4 F1), and expanding to $k = 5$ yields further gains. These findings highlight PRO-STEP's capacity to effectively leverage additional retrieved documents, particularly in complex multi-hop scenarios.

**Impact of the number of retrieval steps.** Figure 5 illustrates the cumulative F1 score and data coverage at retrieval-step cap $K$. PRO-STEP autonomously adjusts search depth to task difficulty: on single-hop PopQA, 65.6% of instances are resolved within the first step (cumulative F1 = 38.5), while on multi-hop HotpotQA and 2WikiMultiHopQA only 52.0% and 36.6% are resolved at step one, with reasoning naturally extending to additional steps. On 2WikiMultiHopQA, cumulative F1 climbs from 21.2 ($K = 1$) to 43.6 ($K = 2$) as coverage grows from 36.6% to 77.5%, indicating that the model successfully acquires critical missing evidence during its second retrieval. Cumulative F1 increases monotonically across all datasets, demonstrating that PRO-STEP mitigates the error-propagation degradation typical of iterative RAG frameworks (recovery analysis in Section H, qualitative examples in Section I).

## 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 VTS, 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 addresses intermediate reasoning flaws, significantly outperforming existing outcome-based and standard process-based RAG baselines across diverse QA datasets. By mitigating the limitations of sparse outcome rewards, PRO-STEP offers an effective approach for fine-grained policy optimization in multi-hop QA tasks.

## Limitations

**Limitations of QwQ-32B as a PRM labeler.** While frontier closed-source models (e.g., GPT-4o) might marginally reduce potential label noise, we utilize QwQ-32B to ensure our pipeline is fully reproducible without restrictive API licenses. To validate this choice and quantify reliability, we conducted a 500-sample audit against Claude Opus 4.7 (detailed in Section E). The results demonstrated substantial agreement ($\kappa = 0.6104$) without systematic permissive bias, confirming that our open-source approach democratizes process-supervised training without sacrificing critical step-level accuracy.

**Trade-offs of open-source, small-data supervision.** While PRO-STEP achieves the highest average EM and F1, it does not uniformly dominate every dataset. The only non-negligible gap appears on MuSiQue, where PRO-STEP trails StepSearch by 1.3 EM. We attribute this to our deliberate choice to avoid closed-source teachers and train on a smaller pool (5,000 seed questions, 15,877 pairs) than StepSearch's GPT-4o-distilled 19k corpus. This is a transparent trade-off: PRO-STEP is reproducible without API access, at the cost of some headroom on the hardest tasks.

**Bamboogle's small test size limits statistical reliability.** Bamboogle contains only 125 test instances, yielding wide confidence intervals. Our paired $t$-test against ReasonRAG returns $p = 0.66$ (Section F), indicating the apparent gap is statistically indistinguishable from noise.

**PRM-feedback-guided regeneration disrupts policy alignment.** We explored augmenting preference pairs via PRM-feedback-guided regeneration (Zhao et al., 2025), but the augmented variant degrades evaluation by 10.7 EM and 9.8 F1 on average (Section J). Regenerated trajectories drift from the natural search-and-reason pattern of inference-time generation, and the outcome-margin filter cannot detect this stylistic drift. This motivates future work on action-format-consistent augmentation.

## 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.

To support reproducibility and open science, we will release the full implementation, including training and evaluation code, the curated preference dataset, the trained PRM, and the final policy model checkpoints upon publication. An anonymized version of our implementation is available at `https://anonymous.4open.science/r/PRO-Step-Anonymous-E4F4`. Our framework relies exclusively on open-source models (Qwen2.5-7B-Instruct, DeepSeek-R1-Distill-8B, QwQ-32B) and publicly available datasets, ensuring that the entire pipeline can be reproduced without API access or proprietary resources.

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)*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow,

Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

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)*.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, and 1 others. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

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. *In Proceedings of the 2026 International Conference on Learning Representations (ICLR)*.

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.

Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. 2025c. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21816–21841.

## Appendix Overview

The appendix includes the following sections:

- **Section A: Prompts and Models Reference.** Details the prompts used for the policy model's trajectory generation, the PRM's training-data annotation, and the PRM's VTS inline scoring. This serves as a supplement to Section 3.
- **Section B: Data Statistics.** Provides detailed statistics of the PRM training corpus and the DPO preference pairs, along with an ablation on the outcome-margin filter. This serves as a supplement to Sections 3 and 4.
- **Section C: PRM-guided value tree search** Presents the mathematical formulations and hyperparameters for the PRM-guided value tree search. This serves as an extended explanation of Section 3.2.
- **Section D: Implementation Details and Hyperparameters.** Reports the full PRM weight ($\alpha$) sweep and training hyperparameters for SFT, DPO, and KTO. This serves as a supplement to Sections 4 and 4.4.
- **Section E: Direct Validation of the PRM.** Reports a 500-sample audit of QwQ-32B-generated labels against an independent frontier-class annotator. This serves as a direct complement to the indirect reranking evaluation in Section 4.5.
- **Section F: Statistical Significance of Main Results.** Provides bootstrap confidence intervals and per-dataset paired $t$-tests against prior baselines. This serves as a supplement to Table 1.
- **Section G: Generalization Across Model Scales and Alignments.** Evaluates PRO-STEP on a non-instruction-tuned 7B base model and a smaller 3B instruction-tuned model. This serves as a supplement to Section 4.
- **Section H: Recovery from Retrieval Failure.** Quantifies how PRO-STEP handles trajectories with early retrieval failures, with a depth-stratified breakdown. This serves as an extended analysis of Section 4.6.
- **Section I: Case Study.** Presents two qualitative examples illustrating how PRM step-level supervision prevents entity confusion and error propagation. This serves as an extended example of Section 4.
- **Section J: PRM-Feedback-Guided Regeneration.** Documents a failed augmentation strategy in which PRM rationales were used to regenerate training trajectories, along with a diagnosis of the resulting distributional mismatch. This serves as an extended discussion of the limitations.

## A Prompts and Models Reference

In this section, we detail the specific 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 Prompt)** This prompt is used by the QwQ-32B annotator for single-turn, full-trajectory evaluation to generate PRM training labels. While the gold answer is provided as 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 information gain means BAD (e.g., restating the question, repeating prior reasoning).

3. Do not use the model's 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 (VTS Inline Scoring Prompt)** This prompt is used for real-time evaluation during the VTS expansion phase by the deployed 8B PRM.

**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 for the datasets used in the two training stages. To avoid ambiguity, we distinguish between the original seed-question pool, the subset retained after filtering, and the final number of step-level supervision instances.

**PRM Training Data.** The PRM is trained on 109,664 step-level decisions extracted from 31,728 reasoning trajectories sampled across 2,000 source multi-hop QA questions. Table 4 summarizes the action-type distribution and the QwQ-32B GOOD/BAD verdicts. The majority of the steps are search actions (71.2%), reflecting the highly interactive nature of our multi-hop RAG environment. 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 and forces the model to rely on external search. We validate the reliability of these QwQ-generated annotations through an independent frontier-LLM audit reported in Section E.

| 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 4: Statistics of the PRM training corpus: 109,664 steps generated from 31,728 trajectories over 2,000 source questions (averaging 3.5 steps per trajectory). Labels assigned by QwQ-32B under the R1–R6 rubric (Section A).

**DPO Preference Pairs.** For policy optimization, we began from the full pool of 5,000 seed multi-hop QA questions described in Section 4. After running VTS, we retained only preference pairs satisfying an outcome-margin filter (chosen $F_1 \geq 0.2$ and $\Delta F_1 \geq 0.2$), yielding 2,866 unique questions and 15,877 contrastive pairs, or approximately 5.5 pairs per retained question.

We adopt this filter for training and data efficiency rather than for performance. As shown in Table 5, removing the filter yields 39,578 pairs (2.5× more data) with comparable average performance (34.65 vs. 34.51 EM), indicating that the discarded low-margin pairs contribute little additional signal relative to their training cost.

As detailed in Table 6, the DPO pairs exhibit rich diversity in action types, tree depths, and reward margins. About three-quarters (75.2%) of the comparisons pit actions of the same type against each other (Search vs. Search: 57.5%, Answer vs. Answer: 17.5%), allowing the policy to learn fine-grained strategic differences rather than mere format adherence. The step-level distribution peaks at depth 2 (45.7%) and depth 3 (29.5%), directly

aligning with the multi-hop reasoning requirements. The reward difference between chosen and rejected steps ensures a robust gradient: $98.4\%$ of pairs have a substantial margin ($\geq 0.20$), with over half ($52.0\%$) exhibiting a high margin ($\geq 0.50$).

## C VTS Implementation Details

This section provides the detailed mathematical formulations and algorithmic mechanics for the VTS 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 PRO-STEP.

**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 $\pi_\psi$ evaluates a non-terminal node $s_t$, we apply a retroactive update to its ancestors without changing their visit counts:

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

**VTS Parameters.** During the process-supervised data generation phase, we utilize VTS to explore reasoning trajectories. Table 7 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.

## D Implementation Details and Hyperparameters

**PRM Weight Sweep.** Table 8 reports the full sweep over $\alpha$ for the combined value $V(s) = \bar{Q}(s) + \alpha \cdot \hat{r}(s)$. The results justify our use of PRM-based step evaluation: every configuration that incorporates PRM signals ($\alpha > 0$, except the over-weighted $\alpha = 0.5$) outperforms the no-PRM baseline ($\alpha = 0$, 32.68 EM), with $\alpha = 0.3$ achieving the best average (34.51 EM, 44.09 F1). This confirms that PRM signals provide supervision beyond what outcome rewards alone can offer, and that process and outcome rewards function as complementary signals.

**Training Hyperparameters.** Table 9 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.

## E Direct Validation of the PRM

We complement the indirect PRM evaluation in Section 4.5 (trajectory reranking) with a *direct* validation of label quality. We audit 500 stratified samples drawn from QwQ-32B-labeled trajectories (held-out from PRM training) by having an independent frontier-class LLM (Claude Opus 4.7) re-evaluate them under the *identical* labeling rubric (R1–R6, Section A). Both labelers operate on the same trajectory context and the rubric explicitly bars use of external world knowledge, so the audit isolates rule-application disagreement from knowledge differences.

**Findings.** $\kappa = 0.61$ falls in the *substantial* agreement range under the standard Landis–Koch interpretation ($\kappa \in [0.61, 0.80]$ = substantial). The disagreement direction is mild (frontier slightly stricter, $\sim$7% fewer GOOD calls), with no systematic permissive bias from QwQ. Per step type, agreement is near-perfect on answer steps ($\kappa = 0.82$) and substantial on search steps ($\kappa = 0.52$); the lower agreement on intermediate reasoning steps ($\kappa = 0.12$) reflects the inherent subjectivity of judging whether a ⟨think⟩ block provides "new useful information" — but reasoning steps account

Table 5: Effect of the outcome-margin filter on data scale and downstream performance. Removing the filter increases the pair count by $2.5\times$ but yields comparable average performance.

| Variant | Pairs | HotpotQA | PopQA | 2WikiMulti | Bamboogle | MuSiQue | Average |
|---|---|---|---|---|---|---|---|
| PRO-STEP (with margin filter) | 15,877 | 38.73 / 51.63 | 40.47 / 47.37 | 44.07 / 51.43 | 36.80 / 47.63 | 12.49 / 22.41 | **34.51 / 44.09** |
| w/o margin filter | 39,578 | 37.62 / 50.58 | 39.20 / 46.40 | 44.47 / 52.62 | 39.20 / 47.70 | 12.78 / 23.16 | 34.65 / 44.09 |

| Pair Type | % | Reward Diff Margin | % |
|---|---|---|---|
| Search vs. Search | 57.5 | $< 0.05$ (Low) | 0.3 |
| Answer vs. Answer | 17.5 | 0.05–0.10 | 0.6 |
| Answer vs. Search | 16.5 | 0.10–0.20 | 0.6 |
| Search vs. Answer | 4.1 | 0.20–0.50 | 46.4 |
| Other | 4.4 | $\geq 0.50$ (High) | 52.0 |
| **Step Level (Depth)** | **%** | **Action (Ch / Rej)** | **%** |
| Step 1 | 24.0 | Search | 61.9 / 74.1 |
| Step 2 | 45.7 | Answer | 37.7 / 21.7 |
| Step 3 | 29.5 | Reason | 0.4 / 4.2 |
| Step 4 | 0.8 | – | – |

Table 6: Detailed statistics of the 15,877 DPO preference pairs under the outcome filter (chosen $F_1 \geq 0.2$ AND $\Delta F_1 \geq 0.2$). Reward margin is computed on $V(s) = \bar{Q}(s) + \alpha \cdot \hat{r}(s)$ with $\alpha = 0.3$. Action percentages represent the proportion of Chosen / Rejected actions respectively.

| Parameter | Value |
|---|---|
| Branching factor ($K$) | 3 |
| Max depth | 7 |
| Iterations per question | 64 |
| F1 discount ($\gamma$) | 0.9 (Reward $= \mathrm{F1} \cdot 0.9^d$) |
| PRM weight ($\alpha$) | 0.3 (Combined $= \mathrm{Reward} + 0.3 \cdot \mathrm{PRM}$) |
| PRM scoring interval | Every 4 iterations |
| Retrieved docs per search | 3 |
| Preference margin ($\delta$) | 0.01 |

Table 7: Hyperparameters for the VTS rollout phase.

for only 10% of the audit, and the labels that drive PRM signal most directly (answer correctness and search quality) are precisely where frontier agreement is strongest. This confirms QwQ-32B as a trustworthy open-source labeler.

## F Statistical Significance of Main Results

We supplement Table 1 with bootstrap confidence intervals and per-dataset paired $t$-tests, computed on per-question results against PRO-STEP. All evaluations are conducted with temperature 0 (greedy decoding), which removes sampling stochasticity and yields deterministic per-question outputs. We therefore report significance over the per-question score distribution rather than across multiple seeds.

**Findings.** PRO-STEP significantly outperforms Search-R1 and ReasonRAG in macro-AVG EM and F1 (95% CI). Against StepSearch, F1 improvements are significant; while the EM CI marginally includes 0, PRO-STEP leads significantly on PopQA and 2WikiMultiHopQA. The decisive 2WikiMultiHopQA lead ($p < 10^{-75}$) confirms our multi-hop advantage is genuine, not an artifact of averaging.

## G Generalization Across Model Scales and Alignments

We test whether the PRO-STEP framework transfers beyond our main 7B instruction-tuned backbone by applying the same pipeline to two distinct settings: a non-instruction-tuned 7B base model, and a smaller 3B instruction-tuned model. All evaluations use the same pipeline as the main experiments.

**Findings.** PRO-STEP shows superior data efficiency, achieving 31.04 average EM at the 3B scale to outperform Search-R1 3B (+4.82 EM) despite using less training data. Consistent with 7B results, the largest gains occur on complex multi-hop benchmarks (+8.14 EM on 2WikiMultiHopQA), confirming step-level PRM effectively distills deep exploration. Finally, scale transfer strictly requires off-policy trajectories and SFT warmup. Direct DPO on the 3B base fails (12.40 EM); SFT is required to learn the format, and since on-policy VTS collapses at 3B, leveraging 7B trajectories is essential.

## H Recovery from Retrieval Failure

**Definitions.** We measure how each method handles *bad prefixes* — trajectories that initially fail to retrieve gold evidence but later succeed.

**Retrieval failure.** A retrieval failure at step $i$ means that the gold passage (defined per dataset by FlashRAG's gold annotation) is *not* present in any of the top-$K$ documents returned by the BGE retriever at step $i$ ($K = 3$ in our setting). A trajectory may issue multiple retrieval steps, and each step is evaluated independently against the same gold annotation.

Table 8: Performance comparison across different Process Reward Model (PRM) weight ($\alpha$) values. The table reports EM / F1 scores across 5 benchmarks.

| $\alpha$ | HotpotQA | PopQA | 2WikiMultiHopQA | Bamboogle | MuSiQue | Average |
|---|---|---|---|---|---|---|
| 0.0 (no PRM) | 36.75 / 49.31 | 38.48 / 45.01 | 41.44 / 48.40 | 34.40 / 44.20 | 12.33 / 22.35 | 32.68 / 41.85 |
| **0.3** $\star$ | **38.73 / 51.63** | **40.47 / 47.37** | **44.07 / 51.43** | **36.80 / 47.63** | 12.49 / 22.41 | **34.51 / 44.09** |
| 0.5 | 34.95 / 47.53 | 36.62 / 44.33 | 41.28 / 48.21 | 33.60 / 41.91 | 10.47 / 20.41 | 31.38 / 40.48 |
| 1.0 | 37.97 / 50.60 | 39.94 / 46.10 | 42.68 / 50.38 | 32.80 / 41.39 | **13.49 / 23.46** | 33.38 / 42.39 |

| Parameter | SFT | DPO | KTO |
|---|---|---|---|
| **Base Model** | Qwen2.5-7B-Instruct | | |
| **Data Size** | 15,877 (Chosen) | 15,877 pairs | 31,754 (15,877 +/15,877 -) |
| **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** ($\alpha$) | 128 | 128 | 128 |
| **Dropout** | 0.05 | 0.05 | 0.05 |
| **Target Modules** | all-linear | | |

Table 9: Training hyperparameters for the policy models. The effective batch size of 16 is achieved via $1 \times 8$ gradient accumulation $\times$ 2 GPUs across all three objectives. KTO uses both chosen and rejected from the outcome filter pair set as desirable / undesirable samples (hence 31,754 total).

Table 10: Agreement between QwQ-32B (our annotator) and Claude Opus 4.7 on the 500-sample audit.

| Metric | Value |
|---|---|
| Audited samples | 500 |
| Agreement | **80.56%** (402 / 499 with valid labels) |
| Cohen's $\kappa$ | **0.6104** (substantial) |
| QwQ-labeled GOOD | 274 |
| Frontier-labeled GOOD | 255 |

Table 11: Per-step-type agreement on the 500-sample audit.

| Step type | N | Agreement | Cohen's $\kappa$ |
|---|---|---|---|
| answer | 200 | 91.00% | **0.82** |
| search | 249 | 77.11% | 0.52 |
| reason | 50 | 56.00% | 0.12 |

Table 12: Macro-AVG bootstrap 95% confidence intervals ($B = 10^4$) for $\Delta$ vs. each prior baseline. Significance: $\star\star\star$ if the 95% CI excludes 0.

| vs. Baseline | $\Delta$ EM (95% CI) | $\Delta$ F1 (95% CI) |
|---|---|---|
| Search-R1 | +2.51 [+1.01, +4.06] $\star\star\star$ | +3.37 [+1.94, +4.82] $\star\star\star$ |
| ReasonRAG | +1.93 [+0.46, +3.36] $\star\star\star$ | +3.14 [+1.56, +4.65] $\star\star\star$ |
| StepSearch | +1.36 [-0.29, +3.00] NS | +2.10 [+0.38, +3.77] $\star\star\star$ |

**Recovery.** Recovery at step $k$ means that the first retrieval step that *succeeds* (gold appears in top-$K$) is step $k$, while every earlier step $0, 1, \ldots, k-1$ failed by the criterion above. Recovery depth therefore equals the number of consecutive failed retrievals before the first hit.

Table 13: Per-dataset paired $t$-test on EM. $n$ = test split size.

| Dataset | $n$ | vs. Search-R1 | vs. ReasonRAG | vs. StepSearch |
|---|---|---|---|---|
| HotpotQA | 7,405 | +0.85, $p$=0.10 NS | +2.36, $p < 10^{-6}$ $\star\star\star$ | +0.01, $p$=0.98 NS |
| PopQA | 14,267 | -0.18, $p$=0.56 NS | +2.69, $p < 10^{-15}$ $\star\star\star$ | +1.23, $p < 10^{-3}$ $\star\star\star$ |
| **2WikiMultiHopQA** | 12,576 | **+9.20**, $p < 10^{-75}$ $\star\star\star$ | +4.27, $p < 10^{-18}$ $\star\star\star$ | +3.69, $p < 10^{-13}$ $\star\star\star$ |
| Bamboogle | 125 | +3.20, $p$=0.40 NS | -1.60, $p$=0.66 NS | +3.20, $p$=0.43 NS |
| MuSiQue | 2,417 | -0.50, $p$=0.46 NS | +1.90, $p$=0.004 $\star\star$ | -1.32, $p$=0.054 NS |

**Recovered trajectories.** Recovered trajectories are those for which step 0 (the initial retrieval, issued before any in-context reflection by the agent) failed but at least one later retrieval step succeeded. This subset therefore directly stresses the model's ability to issue corrective queries after early retrieval mistakes.

**Key observation.** PRO-STEP's lead over Search-R1 *widens* with recovery depth ($k$=1: +4.0; $k \geq 4$: +25.0 EM). This is direct prefix-level evidence that step-level supervision keeps the policy productive after early retrieval mistakes — exactly where outcome-only RL accumulates the most error propagation. Per-bucket counts ($n$) also show PRO-STEP terminates earlier on average ($n = 2{,}562$ at $k$=1 vs. 3,184 for Search-R1), avoiding unnecessary deep iterations.

# I Case Study

We present qualitative examples to illustrate the 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

Table 14: PRO-STEP on Qwen2.5-7B **base** versus the instruction-tuned variant. The base variant incurs a moderate drop but remains competitive.

| Dataset | Base | Instruct (Main) |
|---|---|---|
| HotpotQA | 34.69 / 47.47 | **38.73 / 51.63** |
| PopQA | 40.42 / 46.04 | **40.47 / 47.37** |
| 2WikiMultiHopQA | 41.09 / 48.57 | **44.07 / 51.43** |
| Bamboogle | 36.00 / 45.38 | **36.80 / 47.63** |
| MuSiQue | 12.08 / **22.61** | **12.49** / 22.41 |
| **AVG** | 32.85 / 42.01 | **34.51 / 44.09** |

Table 15: Full 5-dataset results at the 3B scale (EM / F1, %). Bold marks the best result per row; "*" denotes $p < 0.05$ against the best baseline. PRO-STEP outperforms Search-R1 3B while using $34\times$ less training data.

| Dataset | Base 3B | Search-R1 3B | Ours |
|---|---|---|---|
| HotpotQA | 5.31 / 17.36 | 31.82 / 41.30 | **32.82 / 44.33*** |
| PopQA | 7.24 / 22.23 | 36.87 / 42.14 | **39.88* / 46.10*** |
| 2WikiMultiHopQA | 2.27 / 15.28 | 35.96 / 41.86 | **44.10* / 50.77*** |
| Bamboogle | 4.00 / 10.64 | 18.40 / 27.97 | **28.00* / 38.96*** |
| MuSiQue | 1.24 / 6.27 | 8.07 / 14.27 | **10.38* / 19.85*** |
| **AVG** | 4.01 / 14.35 | 26.22 / 33.51 | **31.04 / 40.00** |

Table 16: Final EM on **recovered trajectories** (step-0 retrieval failed; some later step succeeded). $n$ = number of recovered trajectories per dataset (same retriever, same gold annotation; failure defined as gold passage not in top-3).

| Dataset | $n$ | PRO-Step | Search-R1 |
|---|---|---|---|
| HotpotQA | 1,080 | **54.4%** | 51.2% |
| PopQA | 622 | **49.5%** | 44.5% |
| **2WikiMulti** | **2,940** | **52.8%** | 47.5% |
| Bamboogle | 25 | **84.0%** | 82.4% |
| MuSiQue | 416 | **38.5%** | 38.2% |

Table 17: Recovery-depth distribution on 2WikiMulti (the dataset with the largest bad-prefix population). Trajectories are bucketed by the *first* successful retrieval step $k$, given that all previous retrieval steps $0, 1, \ldots, k-1$ failed (gold not in top-3). Cells show final EM within each bucket; numbers in parentheses are $n$.

| Method | $k=1$ | $k=2$ | $k=3$ | $k \geq 4$ |
|---|---|---|---|---|
| PRO-Step | **53.1%** | **51.1%** | **44.1%** | **60.0%** |
| | ($n$=2,562) | ($n$=329) | ($n$=34) | ($n$=15) |
| Search-R1 | 49.1% | 42.8% | 38.9% | 35.0% |
| | ($n$=3,184) | ($n$=575) | ($n$=149) | ($n$=80) |

"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 → CBS → Date).

## J Critic-Feedback-Guided Regeneration: A Failed Augmentation Strategy

To test whether augmenting our preference set with critic-feedback-guided regeneration would improve downstream performance, we ran the following pipeline. For each of the 1,456 questions where the policy failed to produce a correct answer (final $F_1 = 0$), we injected the PRM's `critic_feedback` from the failure point into the user prompt and had the base Qwen2.5-7B-Instruct regenerate from the last sound prefix. The regenerated trajectories were re-scored by the PRM, and we applied the same outcome-margin filter (chosen $F_1 \geq 0.2$ and $\Delta F_1 \geq 0.2$) as in our main pipeline. This yielded 20,567 preference pairs over 3,128 questions (compared to 15,877 pairs over 2,866 questions in the original outcome filter set). We then trained DPO under identical hyperparameters ($\beta = 0.1$, LoRA $r = 64/\alpha = 128$) on the augmented set.

**Training metrics improve while evaluation collapses.** Table 18 shows that the augmented training run yields substantially stronger DPO statistics: the loss roughly halves (0.42 → 0.22), the preference margin grows by 60% (2.17 → 3.48), and training accuracy climbs from 81% to 91%. However, downstream evaluation collapses on every dataset, with average EM dropping from 34.51 to 23.83 and average F1 from 44.09 to 34.27 (Table 19).

**Diagnosis: action-format mismatch between training and inference.** The regenerated chosen trajectories were produced by the base policy under explicit critic-feedback prompting, which alters

Table 18: Training metrics for the outcome filter main set vs. the regen-augmented set under identical DPO hyperparameters.

| Variant | Loss | Margin | Train Acc |
|---|---|---|---|
| outcome filter main ($\beta = 0.1$) | 0.42 | 2.17 | 81% |
| outcome filter + regen ($\beta = 0.1$) | 0.22 | 3.48 | 91% |

the model's output distribution: the regenerated ⟨think⟩ blocks become verbose meta-commentary referencing the feedback (e.g., reasoning about why the previous step was flawed), and the action selection becomes biased toward what the feedback suggests rather than what the trajectory state requires. At inference time, the deployed policy operates without any such feedback channel, so the distribution it must produce diverges from the regenerated distribution it was trained to match.

The outcome-margin filter, which enforces $F_1 \geq 0.2$ and $\Delta F_1 \geq 0.2$, can verify answer-level quality but is blind to this stylistic drift in intermediate tokens. As a result, the augmented training set rewards the policy for matching feedback-conditioned outputs that never appear at test time, and the strong DPO metrics reflect overfitting to this artificial distribution rather than genuine improvement in multi-hop reasoning. We view this as a concrete failure mode of naive PRM-rationale-guided augmentation, and a meaningful constraint that future work in this direction must address.

Table 19: Downstream evaluation: the regen-augmented variant degrades on every benchmark despite stronger training metrics.

| Variant | PopQA | HotpotQA | 2WikiMulti | Bamboogle | MuSiQue | Average |
|---|---|---|---|---|---|---|
| outcome filter main | 40.47 / 47.37 | 38.73 / 51.63 | 44.07 / 51.43 | 36.80 / 47.63 | 12.49 / 22.41 | **34.51 / 44.09** |
| outcome filter + regen | 28.39 / 37.55 | 26.02 / 39.08 | 33.25 / 42.63 | 24.80 / 35.30 | 6.70 / 16.80 | 23.83 / 34.27 |
| $\Delta$ | $-12.08$ / $-9.82$ | $-12.71$ / $-12.55$ | $-10.82$ / $-8.80$ | $-12.00$ / $-12.33$ | $-5.79$ / $-5.61$ | $-10.68$ / $-9.82$ |

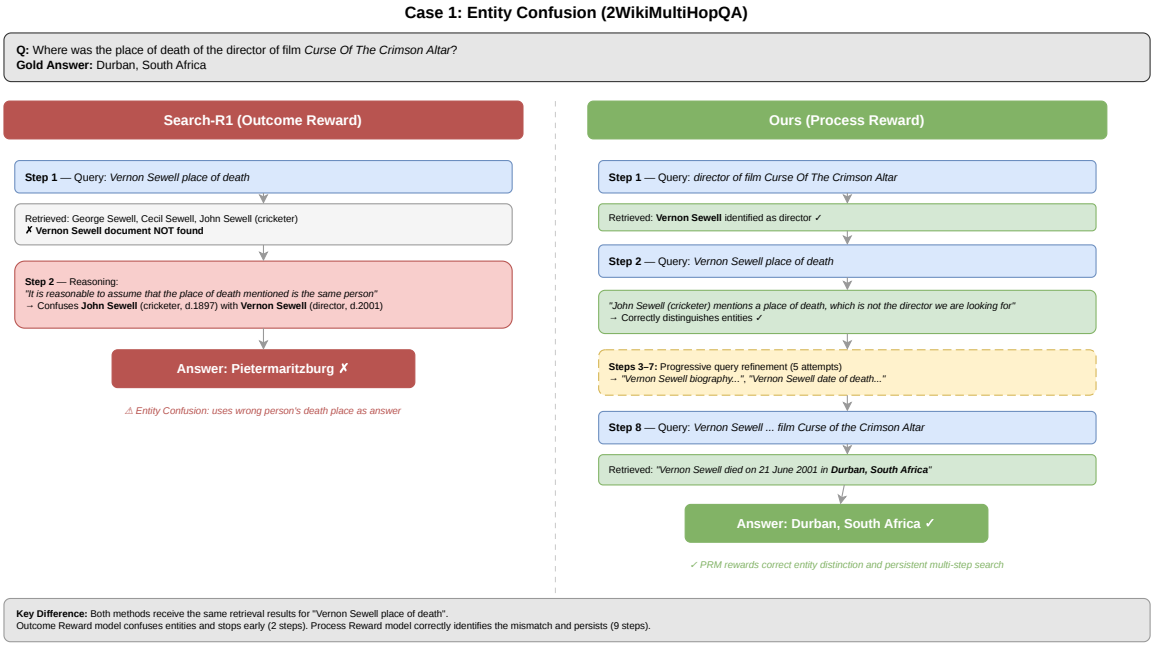


Figure 6: Case Study 1: Entity confusion. The baseline prematurely answers based on a mismatched entity (John Sewell), whereas PRO-STEP rejects the flawed step via PRM feedback and persistently searches for the correct target.

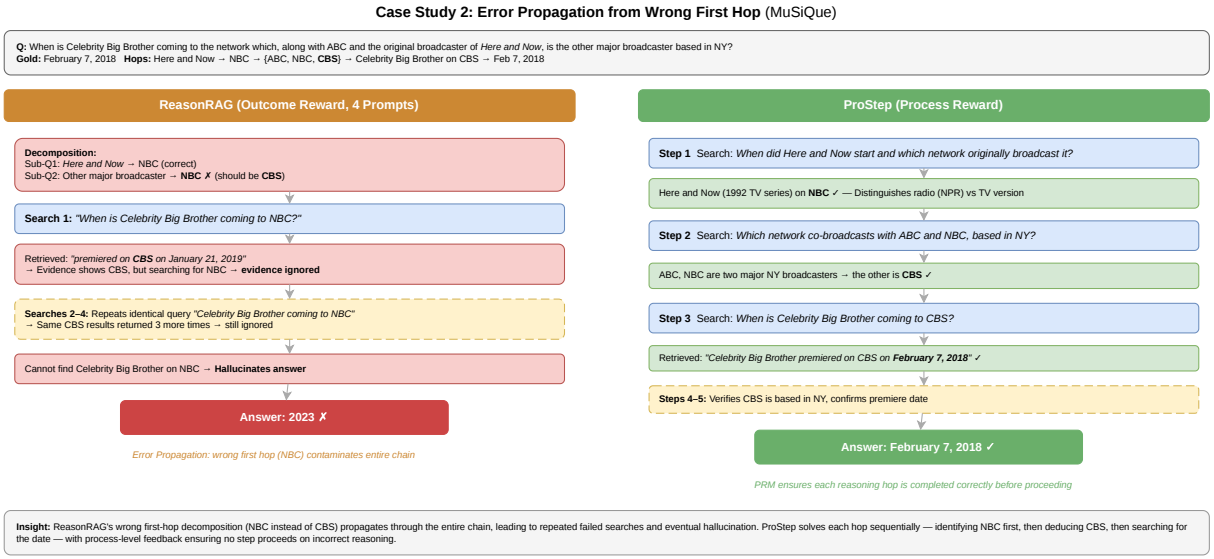


Figure 7: Case Study 2: Error propagation. The baseline repeats useless searches due to a failed first hop. PRO-STEP evaluates each hop, recovers from the initial error, and completes the logical chain.