DELTAPRM: ROLLOUT-FREE AUTOMATIC PROCESS REWARDS VIA ANSWER LIKELIHOOD GAIN FOR RETRIEVAL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-augmented language agents have become a central paradigm for knowledge-intensive question answering, yet they are typically trained with outcome rewards alone, leaving individual search interactions unsupervised. Process reward models (PRMs) address this gap, but their step-level labels are expensive to obtain: human annotation and LLM-as-a-judge labels both demand substantial resources, while Monte-Carlo (MC) estimation, the dominant annotation-free alternative, becomes prohibitive for retrieval agents, since each per-step continuation triggers additional interaction with the search environment. We introduce DeltaPRM, a process reward that requires neither extra rollouts nor extra environment interaction. DeltaPRM scores each retrieval call and the reasoning that follows it by how much the information it brings raises the model's likelihood of the correct final answer–the delta between steps. Labeling thus reduces to a single forward pass per step and relies only on outcome supervision already available in any QA dataset. We train a PRM to predict the discounted return of this label signal from the reasoning prefix alone, no longer requiring access to the ground truth final answer. Across multi-hop agentic search benchmarks, we test DeltaPRM on best-of-N answer selection, step-level guided generation (greedy decoding, beam search, and MC tree search), step-level verification on AgentProcessBench, and online RL, where it consistently matches or surpasses substantially more expensive baselines, including LLM-as-a-judge PRMs. At the same time, DeltaPRM reduces labeling cost by $278\times$ and $217\times$ relative to MC estimation and LLM-as-a-judge, respectively.[1]

1 INTRODUCTION

Language agents that interleave reasoning with retrieval have become the dominant approach to knowledge-intensive question answering. To resolve multi-hop queries, agents must decide when their own knowledge suffices, when to query, and how to fold retrieved evidence into their reasoning. Previous approaches primarily rely on outcome supervision, only rewarding agents for producing the correct final answer. Outcome reward models (ORMs), however, assign a single scalar to an entire trajectory where successful trajectories are reinforced in full even when they contain irrelevant or redundant queries, and failed trajectories are penalized in full even when their early retrievals were on the right track. This mismatch is especially costly for search agents, where answering often demands longer, multi-hop trajectories, and each additional retrieval decision includes an interaction with the search environment.

Process reward models (PRMs) address this gap, but their step-level labels are expensive to obtain. Human annotation is costly and hard to scale, and LLM-as-a-judge pipelines replace human effort with substantial inference expense, typically requiring a model an order of magnitude larger than the PRM being trained to read every trajectory, retrieved passages included. Monte-Carlo estimation, the dominant annotation-free alternative, is particularly computationally ill-suited to retrieval agents. As each sampled continuation issues queries against the search environment, labeling cost scales with the product of trajectory length, rollout count, and per-call retrieval latency.

We take a different view of what makes a retrieval step good. A search call is valuable to the extent that the information it brings into context raises the likelihood of the correct final answer.

[1] Code and models will be publicly released upon acceptance.

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

A query that retrieves what the model already knew contributes little, while one that supplies a missing fact contributes substantially. Queries that lead to the correct final answer are rewarded while those that lead the model astray are penalized. We operationalize this notion of progress through the model's own likelihood, measuring the change in the probability it assigns to the correct final answer immediately before and after taking a step, conditioned on the reasoning and retrieved evidence accumulated so far. We call this per-step quantity the delta, and we introduce DeltaPRM, a process reward model built on the discounted return of the delta. Labeling a full dataset reduces to one forward pass per step and relies only on outcome supervision that is already available, with no additional sampling and no additional interaction with the search environment. After training our PRM, DeltaPRM no longer requires access to the correct final answer $a^\star$ and can be used for test-time scaling at inference time or to shape rewards in reinforcement learning (RL).

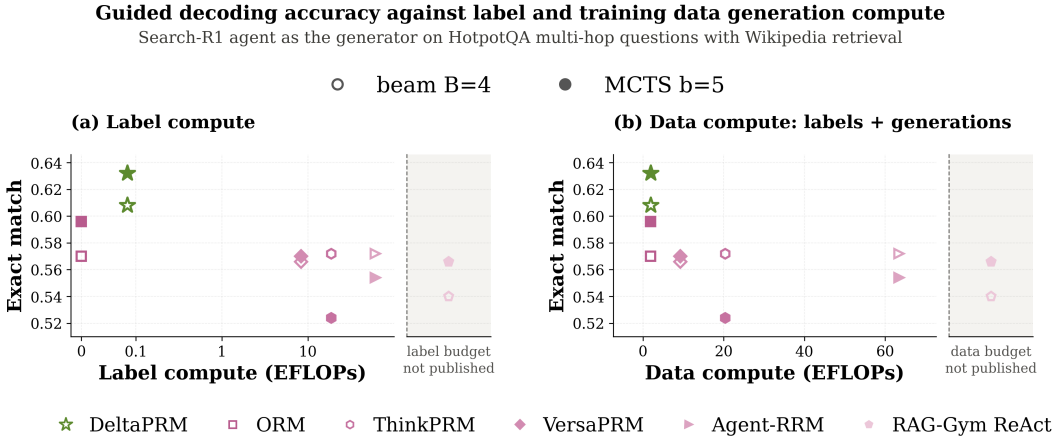


Figure 1: Multi-hop agentic search, with Search-R1 generator on 500 HotpotQA questions and Wikipedia corpus, guided decoding accuracy against label generation compute (a) and total compute with label and training data generation combined (b). Open marks denote beam search and solid marks Monte-Carlo tree search (MCTS).

We evaluate DeltaPRM on multi-hop agentic search across the four settings in which process rewards are typically deployed: best-of-N selection, greedy decoding, beam search and Monte-Carlo tree search (MCTS), step-level verification on AgentProcessBench, and online RL. Across all four, DeltaPRM matches or surpasses PRM baselines that require orders of magnitude more compute to construct, including LLM-as-a-judge PRMs. DeltaPRM also transfers, improving guided generation for agents from a different model family and at larger scale than the one it was trained on. Finally, we show that the DeltaPRM labeling approach is not specific to retrieval. Applying it to mathematical reasoning and multi-domain benchmarks such as MMLU-Pro yields comparable gains, suggesting that likelihood-gain-based process rewards offer an efficient path to PRM construction beyond the agentic setting.

2 Related Work

Process reward models and their labels. PRMs score each step of a solution rather than only its final answer (Cobbe et al., 2021; Uesato et al., 2022; Lightman et al., 2023), and are used to rerank pools, guide step-level beam search and MCTS (Snell et al., 2024), and shape dense RL rewards (Wang et al., 2024a; Cui et al., 2025). Human annotation does not scale, so steps are labeled by *Monte-Carlo estimation*—the fraction of sampled continuations reaching the correct answer (Wang et al., 2024a), refined by tree search (Luo et al., 2024; Chen et al., 2024; Guan et al., 2025b), denoising (Ding et al., 2025), and judge filtering (Zhang et al., 2025b)—or by an *LLM judge* that locates the first incorrect step (Zeng et al., 2025; Yun et al., 2025), ranks candidate actions (Xiong et al., 2026), or writes critiques (Fan et al., 2026a); PAV argues the useful target is a step's *advantage*, its change in success probability (Setlur et al., 2024).

Process rewards from outcome supervision alone. Implicit PRM derives per-step rewards from partial sums of a policy-to-reference log-likelihood ratio trained on outcomes (Yuan et al., 2024), PRIME updates such a reward online in RL (Cui et al., 2025), and AutoPSV differences an outcome PRM's confidence across steps (Lu et al., 2024).

Reference-answer likelihood as a signal. Verifier-free RL uses $\pi(a^\star \mid x, z)$ directly as reward (Zhou et al., 2025), token-averaged (Yu et al., 2025), as a length-normalized perplexity (Liu et al., 2025), or as a log-probability (Kwiatkowski et al., 2026)—while VR-CLI rewards the *improvement* from conditioning on the reasoning (Gurung & Lapata, 2025) and P2S assigns step rewards from a self-synthesized reference chain of thought (Zhong et al., 2026). Retrieval work likewise supervises retrievers by the likelihood a document lends the correct output (Shi et al., 2023; Izacard et al., 2022) or its gain (Jiang et al., 2025). All use the likelihood online and require $a^\star$ at use time; DeltaPRM distills it offline into step labels—consecutive differences with a discounted return—and trains a PRM that can transfer to questions without a reference.

Process supervision for search agents. Search-R1-style agents train with outcome reward only (Jin et al., 2025; Song et al., 2025); Search-o1 adds retrieval at inference time without training (Li et al., 2025). Process-supervised variants label steps with Monte-Carlo or tree-search rollouts (Zhang et al., 2025a; Wang et al., 2026; Zhang et al., 2026), binary-tree trajectory synthesis (Guan et al., 2025a), LLM-judge preferences (Xiong et al., 2026), or checks against gold evidence and external PRMs (Wang et al., 2025; Wu et al., 2026); outcome-trained agents are repeatedly found to over-search (Wu et al., 2025; 2026). DeltaPRM needs neither rollouts, a judge, nor gold evidence—only the final answer during PRM training.

3 METHOD

DeltaPRM trains a PRM using the likelihood gain of the correct final answer before and after a step to provide a step label without sampling a single additional rollout.

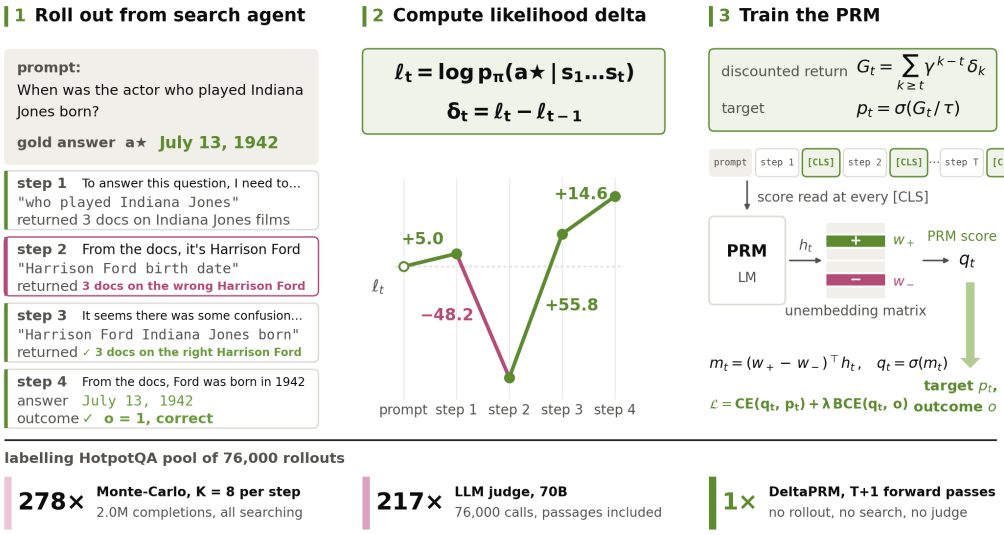


Figure 2: **DeltaPRM on an example multi-hop agentic search rollout. (1)** A frozen generator π answers the question in steps that interleave reasoning and search. Here, two actors share the name and the name-only search retrieves the wrong one; the next query is more specific and retrieves the correct person, resulting in the correct final answer. **(2)** Each step is labeled with the likelihood gain of the correct final answer $a^\star$ before and after the step $\delta_t = \ell_t - \ell_{t-1}$, $\ell_t = \log p_\pi(a^\star \mid x, s_1 \ldots s_t)$. Labeling a T-step rollout costs $T{+}1$ passes with no extra rollouts, no search calls and no judge model. Credit is signed and graded: the search that found the wrong actor is charged -48.2 nats, while the next step that finds the correct one recovers by $+55.8$ nats, a distinction an outcome reward cannot make. The example is illustrative; the ℓ_t, δ_t and p_t shown are those of a real HotpotQA training rollout (Figure 7, left). **(3)** PRM readout occurs at the delimiters spliced in after every step trained toward the target of the gain, discounted forward into G_t and squashed into a soft target $p_t = \sigma(G_t/\tau)$ with an outcome auxiliary. **Bottom:** labelling the HotpotQA 76k-rollout pool in our way costs 278× less compute than Monte-Carlo estimation with $K{=}8$ rollouts per step and 217× less than an LLM judge.

3.1 SETTING

We consider an agent solving a knowledge-intensive question x by interacting with a retrieval environment $\mathcal{E}$ over multiple turns. At turn t, conditioned on the question x and the interaction his-

tory $s_{<t} = (s_1, \ldots, s_{t-1})$, the frozen generator π produces a step s_t, comprising of a reasoning segment, a query issued to $\mathcal{E}$, and the passages $\mathcal{E}$ retrieved in response. The generator then transitions to state $s_{\leq t}$ and either continues to turn $t{+}1$ or terminates with a final answer, yielding a trajectory $y = (s_1, \ldots, s_T)$ of T steps. Each completed trajectory y is assigned an outcome grade $o = r(y, a^\star) \in \{0, 1\}$, obtained by verifying the trajectory's final answer against the correct final answer $a^\star$.

3.2 Answer likelihood gain

For each step t, we compute the likelihood of the correct final answer $a^\star$ where the context includes the trajectory thus far with prompt x and steps $s_{0:t}$:

$$\ell_t \;=\; \log p_\pi(a^\star \,|\, x,\, s_{0:t})\,, \qquad t = 0, \ldots, T, \tag{1}$$

where $s_0 := \emptyset$, so that $\ell_0 = \log p_\pi(a^\star \mid x)$ conditions on the question alone. The label of step t is the gain it produced: the *delta* between the likelihoods of the ground truth answer with and without step t,

$$\delta_t \;=\; \ell_t - \ell_{t-1}. \tag{2}$$

δ_t is positive when step t made the correct final answer more probable, in nats, and negative when it made it less so. Since the trajectory and $a^\star$ are fixed, each ℓ_t is a deterministic quantity from one forward pass. Computing $\ell_0, \ldots, \ell_T$ thus costs a single causal forward pass over the trajectory $(x, s_1, \ldots, s_T)$ to populate the KV cache, plus, at each step t, one short forward pass that appends $a^\star$ to the cached prefix $(x, s_{0:t})$ and is discarded before extending the cache with s_{t+1}: no continuation is sampled, no retrieval is re-executed, and no second model is called.

A step's effect on the final answer is not necessarily immediate; it may instead manifest as a change in the likelihood of a correct final answer several steps later. To capture this, we score each step by the discounted return of the gains that follow it, and turn this return into a probability:

$$G_t \;=\; \sum_{k \geq t} \gamma^{\,k-t}\, \delta_k, \qquad p_t \;=\; \sigma(G_t/\tau)\,. \tag{3}$$

The discount $\gamma \in [0, 1]$ controls how far a step's effect is allowed to propagate: a small γ attributes a step's score mainly to the gain it produces immediately, while a larger γ lets a step's score also reflect gains realized several steps later. The scale τ fixes the sensitivity of $\sigma(\cdot)$ to the magnitude of G_t. The resulting $p_t \in (0, 1)$ is the target we use to train the PRM.

3.3 PRM training

Following standard practice for step-level PRMs (Zhang et al., 2025b; Zeng et al., 2025), we append a delimiter token after each step and train on the logits of two fixed tokens, '$+$' and '$-$', at that position: their difference gives a margin $m_t \;=\; (w_+ - w_-)^\top h_t$, and $q_t = \sigma(m_t)$ is the PRM's predicted probability that step t is on the path to a correct final answer. The training loss

$$\mathcal{L} \;=\; -\frac{1}{M} \sum_t \Big[p_t \log q_t + (1{-}p_t) \log(1{-}q_t) \Big] + \lambda \left(-\frac{1}{M} \sum_t \Big[o \log q_t + (1{-}o) \log(1{-}q_t) \Big] \right), \tag{4}$$

combines a cross-entropy loss toward p_t with a weighted term toward the trajectory's outcome grade o, so that q_t is trained to predict both. At decode time this trained PRM can score a candidate prefix without any access to $a^\star$: it appends the same delimiter and computes q_t from the resulting margin. The same score q_t can then serve any downstream use of a process reward, such as best-of-N selection, beam search, or reinforcement learning.

4 Experiments

We evaluate DeltaPRM in four settings: (i) best-of-N re-ranking of complete trajectories, (ii) step-level guided generation with beam search and MCTS, (iii) step-level verification against human labels on AgentProcessBench, and (iv) online RL with GRPO.

4.1 Experimental setup

Dataset and generator. We study a search agent that performs multi-hop reasoning, using the publicly released 7B checkpoint of Search-R1 (Jin et al., 2025)[2] as the generator. The generator plays two roles: it produces the training trajectories, and its answer likelihoods define the supervision signal from which DeltaPRM derives step labels ℓ_t (Eq. 1). We roll out the generator on

[2]https://huggingface.co/PeterJinGo/SearchR1-nq_hotpotqa_train-qwen2.5-7b-it-em-ppo

HotpotQA (Yang et al., 2018), sampling 8 trajectories for each of 9,500 questions comprising our PRM training data. For retrieval, we use E5 over the 2018 Wikipedia corpus and return the top 3 passages per query. Each trajectory is a sequence of *steps*, where a step is one agent turn consisting of a reasoning trace, a search query, and the retrieved passages.

PRM training details. We initialize the fully-finetuned training of DeltaPRM with the generator model, the 7B checkpoint of Search-R1. The training objective (Eq. 4) has three hyperparameters: the outcome weight λ, label threshold τ, and discount factor γ. We set $\lambda = 0.5$ and τ to the standard deviation of G_t over the training pool. We select $\gamma = 0.3$ on a held-out validation set of 500 HotpotQA questions.[3] Full training details are provided in Appendix C.

Baselines. We compare with the following models (details in Table 8, Appendix A):

- **RAG-Gym** (Xiong et al., 2026): an 8B PRM trained with a pairwise ranking loss on 1,000 HotpotQA training questions to prefer GPT-4o's chosen action among 10 sampled actions per step. We use its three released HotpotQA PRMs (ReAct, ReSearch, and Search-o1 scaffolds).
- **Agent-RRM** (Fan et al., 2026a): an 8B generative reward model that writes a critique of the trajectory before emitting a score, trained with SFT on 28K GPT-OSS-120B judgments and GRPO on 90K agent trajectories.
- **VersaPRM** (Zeng et al., 2025): an 8B discriminative PRM built for reasoning domains beyond math, LoRA-trained on 84K multi-domain MMLU-Pro chains of thought labeled by Llama-3.1-70B-Instruct under a first-error rule.

For a fair comparison, we additionally train the following two baselines with the same recipe, the same 76,000 Search-R1 rollouts, and the same 7B backbone (the generator itself) as DeltaPRM; they differ from DeltaPRM primarily in the label source:

- **ThinkPRM** (Khalifa et al., 2025): a CoT-based PRM, obtained by fine-tuning the backbone to generate the step-level critiques written by a judge (Qwen2.5-72B-Instruct) up to the first error. Since Search-R1 rollouts have no gold step labels, we omit ThinkPRM's gold-label filtering.
- **ORM**: an outcome reward model trained on the same rollouts, with a single label per trajectory—its final-answer correctness—read out at the trajectory's final delimiter. (A per-step variant that broadcasts the outcome label to every step is studied as an ablation in Section 5 and Appendix J.)

4.2 Test-time scaling

4.2.1 Tasks

Test-time scaling tasks include Best-of-N, and guided decoding via greedy decoding, beam search, and MCTS. An illustration for the test-time scaling tasks is shown in Figure 3.

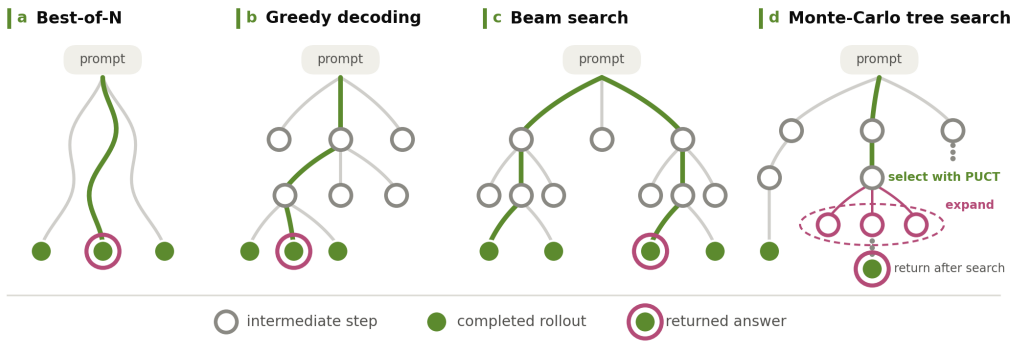


Figure 3: Illustration of test-time scaling methods: (a) best-of-N, (b) greedy decoding, (c) beam search, and (d) Monte-Carlo tree search.

Best-of-N. Best-of-N evaluates a PRM's ability to select the most accurate of N candidate responses to a question: the PRM's per-step scores are aggregated into a score per candidate, the highest-scoring candidate is selected, and the metric is the accuracy of the selected responses.

Greedy decoding. Greedy decoding samples n continuations per step and selects the step with the highest PRM score until a final answer is reached.

[3] A sweep over $\gamma \in [0, 1]$ (Table 25) shows that performance is not sensitive to this choice.

Beam search. Beam search (Snell et al., 2024) maintains the B partial trajectories with the highest mean PRM score along their path, expanding each with n sampled next steps, scoring all $B \cdot n$ candidates with the PRM, and setting aside any candidate that emits a final answer, whose best-scoring member is the output (Algorithm 1, Appendix L).

Monte-Carlo tree search (MCTS). MCTS builds a search tree over k simulations per question with the standard select (PUCT), expand, rollout, and backup phases (Hao et al., 2023; Feng et al., 2023), backing up the mean PRM score of the trajectory completed through the selected leaf and returning the highest-value completed trajectory (Algorithm 2, Appendix L).

4.2.2 Results

Best-of-N. Figure 4 and Table 12 report Best-of-N results on HotpotQA with Search-R1 7B as the generator. DeltaPRM matches the final-answer accuracy of the ORM and significantly outperforms majority voting (self-consistency) and all PRM baselines.

As Best-of-N judges complete trajectories solely by their final answers, the ORM, trained on exactly this criterion, performs strongly. This reflects a known limitation of Best-of-N for evaluating PRMs (Zhang et al., 2025b): it rewards final-answer correctness and cannot distinguish a PRM that assesses the reasoning process from one that primarily tracks the outcome. We therefore also evaluate DeltaPRM in guided decoding, where intermediate steps must be scored directly.

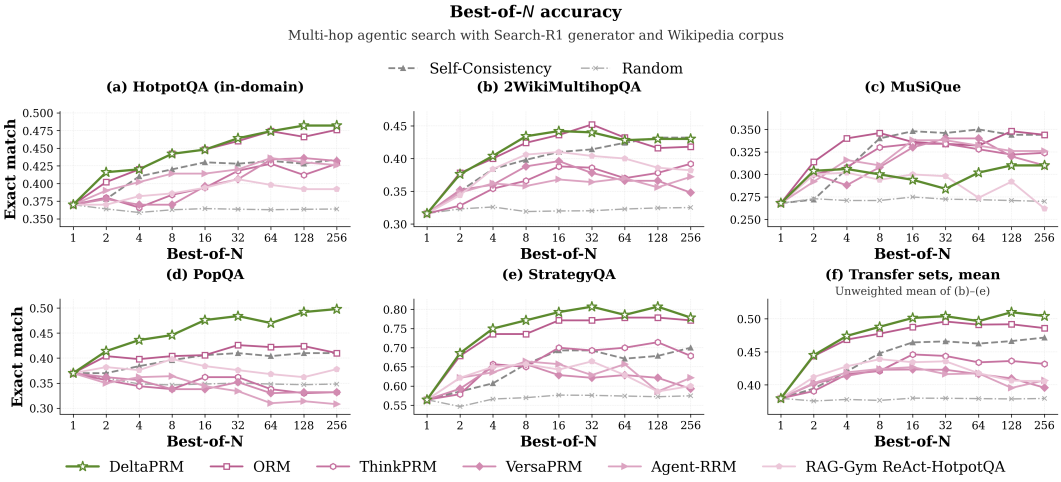


Figure 4: Best-of-N on multi-hop agentic search. Each PRM reranks N completions generated by Search-R1 with a Wikipedia corpus on multi-hop datasets and single-hop dataset (PopQA) alongside aggregated performance; 500 questions per panel (140 on StrategyQA).

Guided decoding. Table 1 reports final-answer accuracy when each PRM guides greedy, beam, and MCTS decoding. In nearly all settings, DeltaPRM outperforms the baselines, both on HotpotQA, the dataset it was trained on, and on unseen multi-hop (2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), StrategyQA (Geva et al., 2021)) and single-hop (PopQA (Mallen et al., 2023)) datasets.

Moreover, accuracy under DeltaPRM increases consistently from greedy to beam search to MCTS. Since these strategies optimize the PRM score with increasing search effort, this trend confirms that higher DeltaPRM scores reliably translate into better trajectories rather than into reward-hacking behavior. MCTS guided decoding shows statistically significant improvements against all baselines on HotpotQA, 2WikiMultiHopQA, PopQA, and StrategyQA.

4.3 Step-level verification

AgentProcessBench (Fan et al., 2026b) is a step-level verification task with human-annotated ground truth labels. On the HotpotQA subset (250 human-labeled agent trajectories, 734 steps), the first erroneous step is located from the PRM's predicted step-wise scores following the benchmark's protocol, and performance is reported via first-error accuracy, step accuracy, and AUROC.

Our method DeltaPRM correctly ranks correct steps higher than incorrect steps at a higher rate, as measured by AUROC, than baselines. DeltaPRM attains the highest AUROC (78.4), comparable to Agent-RRM at 72.5 and significantly outperforming remaining baselines. The ORM column

Table 1: Guided decoding on multi-hop agentic search. Within a column, bold marks the best value; a dagger (†) marks every value not significantly worse than it by an exact paired McNemar test (α=0.05), and values significantly worse than the best are unmarked; this convention holds for all tables unless marked otherwise.

	HotpotQA			2WikiMultiHopQA			MuSiQue			PopQA			StrategyQA		
Model	Greedy	Beam	MCTS	Greedy	Beam	MCTS	Greedy	Beam	MCTS	Greedy	Beam	MCTS	Greedy	Beam	MCTS
Open-source Process Reward Models															
VersaPRM	0.536	0.566	0.570	0.370	0.384	0.410	**0.287**	0.313†	0.302†	0.354	0.346	0.348	0.622	0.650	0.614
Agent-RRM	0.530	0.572	0.554	0.376†	0.366	0.390	**0.287**	0.312†	0.315†	0.346	0.334	0.340	0.596	0.608	0.614
RAG-Gym ReAct	0.528	0.540	0.566	0.376†	0.410†	0.418	**0.287**	0.308†	0.282	0.358	0.360	0.344	0.628	0.618	0.616
RAG-Gym ReSearch	0.542	0.550	0.574	0.372†	0.412†	0.400	0.283†	0.298†	0.301†	0.390	0.364	0.364	0.586	0.628	0.608
RAG-Gym Search-o1	0.570†	0.560	0.564	0.354	0.360	0.382	0.260	0.274	0.277	0.344	0.362	0.356	0.628	0.646	0.630
Reward Models Trained in Our Setting															
ThinkPRM	0.562†	0.572	0.524	0.382†	0.386	0.376	0.284†	0.316†	0.304†	0.342	0.370	0.348	0.592	0.658	0.618
ORM	0.544†	0.570	0.596	0.358	0.426†	0.402	0.258	0.282	0.302†	0.386	0.394	0.414	0.626	0.710	0.710
DeltaPRM	**0.578**	**0.608**	**0.632**	**0.414**	**0.444**	**0.464**	0.279†	**0.319**	**0.324**	**0.458**	**0.480**	**0.480**	**0.714**	**0.750**	**0.782**

confirms that an outcome score is ill fitted to step-level discrimination (AUROC below chance, 39.4): an outcome-supervised score on a partial trajectory estimates the probability of eventually reaching a correct answer rather than the correctness of the steps so far (Uesato et al., 2022).

Table 2: Step-level process quality on AgentProcessBench HotpotQA subset. Within a row, bold marks the best value; a dagger (†) marks every value a paired cluster bootstrap over the query index cannot separate from it (α=0.05), and values significantly worse than the best are unmarked. ± is half the width of the value's marginal 95% CI from the same bootstrap.

	DeltaPRM	Agent-RRM	VersaPRM	RG ReAct	RG ReSearch	RG Search-o1	ORM
AUROC	**78.4** ±6.3	72.5 ±5.8†	70.1 ±6.2	44.2 ±7.8	64.6 ±5.1	34.1 ±6.8	39.4 ±5.9
Step acc.	**68.4** ±8.7	**68.4** ±9.4	67.8 ±8.8†	68.0 ±9.3†	68.0 ±9.3†	68.0 ±9.3†	68.0 ±9.3†
First-error acc.	**61.6** ±7.6	59.6 ±9.8†	60.4 ±9.6†	58.4 ±9.8†	58.4 ±9.8†	58.4 ±9.8†	58.8 ±9.6†

4.4 GRPO

Beyond test-time scaling, DeltaPRM also serves as a dense reward for online RL with GRPO (Shao et al., 2024). The PRM is incorporated as a second, separately scaled advantage. PRM scores are aggregated across steps and centered among same-group rollouts that agree on both outcome and search count. Given the cost of online RL, the policy is initialized from Qwen2.5-7B and compared against the outcome reward alone and VersaPRM, the best-performing PRM baseline in the test-time scaling experiments (Section 4.2.2). Training details are provided in Appendix H.

Table 3: GRPO checkpoint transfer evaluation, pooled over two training seeds. ± is half the width of the 95% question-cluster bootstrap CI. Bold marks the column's best arm; † marks an arm the paired question-cluster bootstrap cannot separate from it (two-sided, $\alpha = 0.05$); unmarked arms are significantly worse.

Reward	HotpotQA	2WikiMultiHopQA	MuSiQue	PopQA	StrategyQA
DeltaPRM	**59.5** ± 1.6	37.4† ± 1.5	**19.2** ± 1.4	**44.0** ± 1.6	71.8 ± 1.4
Outcome	51.6 ± 1.6	34.2 ± 1.4	17.5 ± 1.3	42.2 ± 1.6	70.3 ± 1.4
VersaPRM	56.6 ± 1.6	**38.1** ± 1.5	18.6† ± 1.4	43.7† ± 1.6	**73.8** ± 1.4

GRPO. Table 3 reports policies trained on HotpotQA and evaluated in-domain and on four additional QA datasets. On HotpotQA, DeltaPRM improves accuracy by 7.9 points over the outcome reward alone and by 2.9 points over VersaPRM. The gains carry over to the multi-hop dataset MuSiQue and single-hop dataset PopQA, where DeltaPRM again outperforms both baselines. On the multi-hop 2WikiMultiHopQA, DeltaPRM is comparable to VersaPRM. For the yes/no StrategyQA, DeltaPRM is neither the strongest nor comparable, suggesting that the benefit of step-level rewards is concentrated in multi-hop reasoning.

5 Analysis

We additionally examine transfer to unseen datasets, generator families, and generator sizes, and test whether our training recipe generalizes to non-search domains, including math (MATH (Hendrycks

et al., 2021)) and general question answering (MMLU-Pro (Wang et al., 2024b)), and to other search domains (PaperSearchQA (Burgess et al., 2026); Appendix F).

Labeling cost comparison. Table 4 compares label costs on 76,000 Search-R1 rollouts on HotpotQA (256,164 steps) against the two dominant annotation strategies: MC estimation, which we instantiate for search agents following Math-Shepherd (Wang et al., 2024a) (K=8), and LLM-as-a-judge designs (Khalifa et al., 2025; Zeng et al., 2025; Fan et al., 2026a; Xiong et al., 2026). MC decodes K continuations per step, each re-entering the search environment (6.3 extra calls per step). LLM-as-a-judge avoids the environment but relies on an external model an order of magnitude larger than the generator, and decodes roughly 500 tokens of rationale per step. DeltaPRM needs neither: it computes the gold answer's likelihood at each prefix in a single forward pass, reusing the sampler's KV cache. Estimating compute at two FLOPs per parameter per token (Kaplan et al., 2020), DeltaPRM costs 0.08 EFLOPs, 278× less than MC and 217× less than LLM-as-a-judge (217–359× across the three judge designs).[4] Pipelines that combine MC and judge labels for their complementary strengths (Zhang et al., 2025b) cost 502× ours (MC plus one 72B-judge read of every rollout). Details of the cost accounting are provided in Appendix I.

Table 4: Labeling cost on 76,000 HotpotQA rollouts (256,164 steps). *Labeler calls*: number of passes of the label-source model. *Tokens / step*: tokens processed by the labeler per step; label compute is totalled over the pool. Agent-RRM's judge (GPT-OSS-120B) is a sparse MoE charged at its total parameter count; at its 5.1B active parameters per token it costs 1.2 EFLOPs (16×). Full accounting and additional baselines are in Table 20.

Method	Label source	Labeler calls	Tokens / step	Extra search	EFLOPs	× ours
Math-Shepherd (MC, K=8)	Policy (7.6B) + retriever	K decodes/step	5,687	6.3	22.1	278×
ThinkPRM (judge, per step)	External LLM (72B)	1 decode/step	498	0	18.5	233×
VersaPRM (judge, per rollout)	External LLM (70B)	1 decode/rollout	478	0	17.3	217×
Agent-RRM (judge, per rollout)	External MoE LLM (120B)	1 decode/rollout	478	0	28.6	359×
DeltaPRM	**Policy (7.6B)**	**1 forward/step**	**20**	**0**	**0.08**	**1×**

Cross-model transfer. DeltaPRM is trained on trajectories from a single generator, Search-R1-Qwen-7B, so we examine whether its PRM learns a generator-agnostic notion of step quality. We test this with Search-R1 checkpoints from other model families and scales (Table 11, Appendix D). Without retraining, the PRM improves over the unguided baseline by $+16.4$ points on Search-R1-Llama-3B, a generator from a different model family, and by $+17.4$ points on Search-R1-Qwen-3B, a smaller generator from the same family. Both are on par with the $+15.4$ gain on its own training generator. Training DeltaPRM with Search-R1-Llama-3B used for all training components (trajectories, labels, and backbone) yields $+17.4$ points on Llama itself and $+13.8$ points on Search-R1-Qwen-7B, so transfer holds regardless of which model the PRM is trained from. Replacing only the labeling model with Qwen2.5-14B gives $+14.8$ points, indicating that the labels are not tied to the model that computes them. All gains are significant under an exact paired McNemar test ($p < 0.001$). The same protocol applied to the MATH checkpoint is in Table 17.

Training target ablation. We ablate the design choices of our training target: the discount factor γ, the outcome weight λ, regressing on the raw log-likelihood ℓ_t instead of the gain δ_t, and training with outcome supervision alone. All variants are evaluated on guided decoding via beam search, with other training settings held fixed (full results in Table 25). Performance does not differ significantly across $\gamma \in [0, 1)$, whereas $\gamma = 1$ is significantly worse. This suggests that DeltaPRM is robust to γ, with the useful signal lying mainly in the immediate information gain. However, the undiscounted return ($\gamma = 1$) lets future steps dominate the target, blurring credit assigned to the current step. Directly regressing on the raw log-likelihood ℓ_t performs worse, because this does not take the delta into consideration and conflate model's prior knowledge with information acquired during search. Finally, we isolate each term of the loss in Eq. 4. Removing the outcome term ($\lambda = 0$) and training on the outcome label alone both significantly underperform the full model. This suggests that the

[4]These ratios are lower bounds particularly for MC, as FLOPs price neither retrieval nor memory-bound decoding.

step-level gain and the outcome signal carry complementary information and should be used together (a detailed analysis is in Appendix J).

Table 5: Our DeltaPRM against the ORM and $\lambda = 0$ ablation on HotpotQA: strict EM under best-of-N, greedy, beam, and MCTS decoding, and AUROC on AgentProcessBench. Bold marks the best value in a column and a dagger (†) every value not significantly worse than it (exact paired McNemar; a paired cluster bootstrap over queries for AUROC). The best-of-N column carries no test (as in Appendix E), so bold there is the column maximum.

	Best-of-N, N=256	**Greedy**	**Beam**	**MCTS**	**APB AUROC**
DeltaPRM	**0.482**	**0.578**	**0.608**	**0.632**	**78.4**
$\lambda = 0$ ablation	0.408	0.496	0.508	0.526	63.0
ORM	0.476	0.544†	0.570	0.596	39.4

Beyond search agents. The label needs only a correct final answer and a step boundary, so the same recipe applies to any step-structured generation. We extend our PRM training recipe to math (MATH) and to general question answering without retrieval (MMLU-Pro). Table 6 reports guided decoding on both: DeltaPRM is on par with the strongest trained PRM of each domain (Qwen2.5-Math-PRM-7B and VersaPRM, respectively, both in domain there), taking the best MCTS value on MATH-500 and the best beam value on MMLU-Pro, and not significantly worse than the best in every remaining column except MMLU-Pro under MCTS—at orders of magnitude lower label cost (Appendix I). Guided decoding across nine further math benchmarks is reported in Appendix F (Table 16): the MATH checkpoint transfers at parity with Qwen2.5-Math-PRM-7B, separated from the best method only on Olympiad-level sets under MCTS. The recipe also extends to a second search domain, single-hop retrieval over scientific abstracts (PaperSearchQA); those results, together with a biomedical transfer battery, are likewise collected in Appendix F.

Table 6: Task transfer to math (MATH-500) and general non-retrieval question answering (MMLU-Pro), guided decoding accuracy. Bold marks the best value in a column of its block and a dagger (†) every value not significantly worse than it, by an exact paired McNemar test on the per-question outcomes, two-sided at α=0.05.

Method	**Greedy**	**Beam**	**MCTS**	**Method**	**Greedy**	**Beam**	**MCTS**
MATH-500				*MMLU-Pro*			
DeltaPRM	0.838†	0.882†	**0.894**	DeltaPRM	0.558†	**0.603**	0.612
Qwen2.5-Math-PRM-7B	**0.862**	**0.890**	0.886†	VersaPRM	**0.586**	0.590†	**0.652**
Math-Shepherd	0.836	0.828	0.850	ORM	0.513	0.554	0.537
ORM	0.840†	0.860	0.880†				

6 Discussion and conclusion

While DeltaPRM's clearest advantage is in multi-hop agentic search, on other domains it achieves performance comparable to the strongest in-domain PRMs at a fraction of their labeling compute. Extending its advantage beyond the agentic setting remains an area for future research.

Our step labels are derived from a model's teacher-forced likelihoods. As with LLM-as-a-judge approaches, this presumes the model can reason over the information in the observed trajectory alone. The assumption may fail on a counterfactual corpus, whose contents can conflict with the model's prior knowledge and be misinterpreted. The robustness of PRMs given unreliable or unfamiliar information remains an open question.

DeltaPRM builds its PRM from privileged task information: a correct final answer and identifiable step boundaries. It therefore applies directly to the many tasks that satisfy these requirements, including QA and mathematical reasoning, but not to settings without a known correct answer or those requiring verification of intermediate states. Extending the method to such settings remains an important direction for future work.

We introduce an automatic process reward for multi-hop agentic search DeltaPRM whose label source is the likelihood gain of the correct final answer between steps. Our PRM is rollout-free, does not require LLM-as-a-judge labels, and thus provides labels that are compute-efficient by orders of magnitude. DeltaPRM shows strong or comparable performance in test-time scaling tasks, step-level verification, and online RL in multi-hop agentic search, where generations are especially expensive.

AI USE STATEMENT

In this work, we used generative AI tools to design and provide feedback on research methodology and experiments, and to implement methods. We have not used generative AI tools to develop theoretical models or conceptual frameworks, propose or refine hypotheses, support qualitative or thematic data analysis, or interpret results; the research idea and its conceptual development are the authors' own. The remaining tasks with required disclosure—generating synthetic data sets, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, assisting with translation, and cleaning or reformatting datasets—are not applicable to this work. Additionally, we used generative AI tools for tasks with recommended disclosure: creating and modifying scientific figures, suggesting experimental parameters, creating and editing software code, creating artifacts, drafting parts of the paper, suggesting the structure of the paper, summarizing and analyzing existing literature, identifying relevant literature, sourcing and searching for information, editing the paper to improve readability, and proposing a title and keywords. While generative AI tools assisted in identifying relevant literature, the citations in this paper were produced entirely by hand: every reference was read, verified against the original source, and entered and formatted by the authors. We have reviewed all AI-assisted work: AI-implemented code was verified and tested for correctness by the authors before use in any reported experiment, and AI-drafted text was reviewed, fact-checked against our experimental records, and edited by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

James Burgess, Jan N. Hansen, Duo Peng, Yuhui Zhang, Alejandro Lozano, Min Woo Sun, Emma Lundberg, and Serena Yeung-Levy. Papersearchqa: Learning to search and reason over scientific papers with rlvr, 2026. URL https://arxiv.org/abs/2601.18207.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Alphamath almost zero: Process supervision without process, 2024. URL https://arxiv.org/abs/2405.03553.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards, 2025. URL https://arxiv.org/abs/2502.01456.

Yuyang Ding, Xinyu Shi, Juntao Li, Xiaobo Liang, Zhaopeng Tu, and Min Zhang. Scan: Self-denoising monte carlo annotation for robust process reward learning, 2025. URL https://arxiv.org/abs/2509.16548.

Kaixuan Fan, Kaituo Feng, Manyuan Zhang, Tianshuo Peng, Zhixun Li, Yilei Jiang, Shuang Chen, Peng Pei, Xunliang Cai, and Xiangyu Yue. Exploring reasoning reward model for agents, 2026a. URL https://arxiv.org/abs/2601.22154.

Shengda Fan, Xuyan Ye, Yupeng Huo, Zhi-Yuan Chen, Yiju Guo, Shenzhi Yang, Wenkai Yang, Shuqi Ye, Jingwen Chen, Haotian Chen, Xin Cong, and Yankai Lin. Agentprocessbench: Diagnosing step-level process quality in tool-using agents, 2026b. URL https://arxiv.org/abs/2603.14465.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*, 2023.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies, 2021. URL https://arxiv.org/abs/2101.02235.

Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. Deeprag: Thinking to retrieve step by step for large language models, 2025a. URL https://arxiv.org/abs/2502.01142.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking, 2025b. URL https://arxiv.org/abs/2501.04519.

Alexander Gurung and Mirella Lapata. Learning to reason for long-form story generation, 2025. URL https://arxiv.org/abs/2503.22828.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 8154–8173, 2023.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps, 2020. URL https://arxiv.org/abs/2011.01060.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models, 2022. URL https://arxiv.org/abs/2208.03299.

Yi Jiang, Sendong Zhao, Jianbo Li, Haochun Wang, and Bing Qin. Gainrag: Preference alignment in retrieval-augmented generation through gain signal synthesis, 2025. URL https://arxiv.org/abs/2505.18710.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think, 2025. URL https://arxiv.org/abs/2504.16828.

Ariel Kwiatkowski, Natasha Butt, Ismail Labiad, Julia Kempe, and Yann Ollivier. Likelihood-based reward designs for general llm reasoning, 2026. URL https://arxiv.org/abs/2602.03979.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models, 2025. URL https://arxiv.org/abs/2501.05366.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Wei Liu, Siya Qi, Xinyu Wang, Chen Qian, Yali Du, and Yulan He. Nover: Incentive training for language models via verifier-free reinforcement learning, 2025. URL https://arxiv.org/abs/2505.16022.

Jianqiao Lu, Zhiyang Dou, Hongru Wang, Zeyu Cao, Jianbo Dai, Yingjia Wan, and Zhijiang Guo. Autopsv: Automated process-supervised verifier, 2024. URL https://arxiv.org/abs/2405.16802.

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. Improve mathematical reasoning in language models by automated process supervision, 2024. URL https://arxiv.org/abs/2406.06592.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories, 2023. URL https://arxiv.org/abs/2212.10511.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning, 2024. URL https://arxiv.org/abs/2410.08146.

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

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. Replug: Retrieval-augmented black-box language models, 2023. URL https://arxiv.org/abs/2301.12652.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2503.05592.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multi-hop questions via single-hop question composition, 2022. URL https://arxiv.org/abs/2108.00573.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022. URL https://arxiv.org/abs/2211.14275.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024a. URL https://arxiv.org/abs/2312.08935.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024b. URL https://arxiv.org/abs/2406.01574.

Zhao Wang, Ziliang Zhao, and Zhicheng Dou. Prorag: Process-supervised reinforcement learning for retrieval-augmented generation, 2026. URL https://arxiv.org/abs/2601.21912.

Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization, 2025. URL https://arxiv.org/abs/2505.15107.

Peilin Wu, Mian Zhang, Xinlu Zhang, Xinya Du, and Zhiyu Zoey Chen. Search wisely: Mitigating sub-optimal agentic searches by reducing uncertainty, 2025. URL https://arxiv.org/abs/2505.17281.

Peilin Wu, Mian Zhang, Kun Wan, Wentian Zhao, Kaiyu He, Xinya Du, and Zhiyu Chen. Hiprag: Hierarchical process rewards for efficient agentic retrieval augmented generation, 2026. URL https://arxiv.org/abs/2510.07794.

Guangzhi Xiong, Qiao Jin, Xiao Wang, Yin Fang, Haolin Liu, Yifan Yang, Fangyuan Chen, Zhixing Song, Dengyu Wang, Minjia Zhang, Zhiyong Lu, and Aidong Zhang. Supervising the search process produces reliable and generalizable information-seeking agents, 2026. URL https://arxiv.org/abs/2502.13957.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018. URL https://arxiv.org/abs/1809.09600.

Tianyu Yu, Bo Ji, Shouli Wang, Shu Yao, Zefan Wang, Ganqu Cui, Lifan Yuan, Ning Ding, Yuan Yao, Zhiyuan Liu, Maosong Sun, and Tat-Seng Chua. Rlpr: Extrapolating rlvr to general domains without verifiers, 2025. URL https://arxiv.org/abs/2506.18254.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels, 2024. URL https://arxiv.org/abs/2412.01981.

Jaehoon Yun, Jiwoong Sohn, Jungwoo Park, Hyunjae Kim, Xiangru Tang, Yanjun Shao, Yonghoe Koo, Minhyeok Ko, Qingyu Chen, Mark Gerstein, Michael Moor, and Jaewoo Kang. Med-prm: Medical reasoning models with stepwise, guideline-verified process rewards, 2025. URL https://arxiv.org/abs/2506.11474.

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. Versaprm: Multi-domain process reward model via synthetic reasoning data, 2025. URL https://arxiv.org/abs/2502.06737.

Tianhua Zhang, Kun Li, Junan Li, Yunxiang Li, Hongyin Luo, Xixin Wu, James Glass, and Helen Meng. Treeps-rag: Tree-based process supervision for reinforcement learning in agentic rag, 2026. URL https://arxiv.org/abs/2601.06922.

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. Process vs. outcome reward: Which is better for agentic rag reinforcement learning, 2025a. URL https://arxiv.org/abs/2505.14069.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning, 2025b. URL https://arxiv.org/abs/2501.07301.

Wenlin Zhong, Chengyuan Liu, Yiquan Wu, Bovin Tan, Changlong Sun, Yi Wang, Xiaozhong Liu, and Kun Kuang. P2s: Probabilistic process supervision for general-domain reasoning question answering, 2026. URL https://arxiv.org/abs/2601.20649.

Xiangxin Zhou, Zichen Liu, Anya Sims, Haonan Wang, Tianyu Pang, Chongxuan Li, Liang Wang, Min Lin, and Chao Du. Reinforcing general reasoning without verifiers, 2025. URL https://arxiv.org/abs/2505.21493.

A Domains, generators and baselines

Table 7 lists the four instantiations of the recipe. The HotpotQA column is the main experiment of Section 4.1; the other three are the domains of Section 5 and Appendix F. Table 8 details the HotpotQA baselines of Section 4.1, and Table 9 the external PRMs the other three domains are compared against.

Table 7: The four instantiations of the recipe. Only γ, τ, and the context window differ across columns (τ is $\mathrm{std}(G)$ over the pool; the delimiter, readout, outcome weight λ=0.5 and optimization are shared (Appendix C).

	HotpotQA	**PaperSearchQA**	**MATH**	**MMLU-Pro**
Generator	Search-R1 PPO Qwen2.5-7B-Instruct	Search-R1 PPO Qwen2.5-7B-Instruct	Qwen2.5-Math -7B-Instruct	Qwen2.5-7B -Instruct
Environment	wiki-18, E5 dense, top-3	16.2M PubMed abstracts, BM25, top-3	—	—
Step unit	turn + <information>	turn + <information>	paragraph	paragraph
Answer probe	<answer>$a^\star$</answer>	<answer>$a^\star$</answer>	\boxed{$a^\star$}	<answer>$a^\star$</answer>
Pool questions	9,500	9,500	7,201	5,468
Samples / question	8	8	16	16
Pool steps	256,164	191,206	1,196,183	740,232
γ	0.3	0.3	0.8	0.8
τ	15.11	13.81	8.94	2.60
Context	16,384	16,384	4,096	4,096
Evaluation	500 q (val)	500 q (test)	500 q (MATH-500)	534 q (val)
Grader	strict EM	strict EM	boxed verifier	strict EM

Table 8: External baselines for multi-hop agentic search. †: in domain (trained on the evaluation task). The last two rows are our own recipe with a different label source.

Baseline	**Checkpoint**	**Backbone**	**Label rule**
RAG-Gym ReAct†	RAG-Gym/ReAct-HotpotQA-PRM	Llama-3.1-8B-Instruct	action ranking by a GPT-4o judge
RAG-Gym ReSearch†	RAG-Gym/ReSearch-HotpotQA-PRM	Llama-3.1-8B-Instruct	action ranking by a GPT-4o judge
RAG-Gym Search-o1†	RAG-Gym/Search-o1-HotpotQA-PRM	Llama-3.1-8B-Instruct	action ranking by a GPT-4o judge
Agent-RRM	bunny127/Agent-RRM	Qwen3-8B	generative critique (GPT-OSS-120B)
VersaPRM	UW-Madison-Lee-Lab/VersaPRM	Llama-3.1-8B	LLM judge, first error
ThinkPRM†	ours	Search-R1 7B	LLM judge, first error (no true label rejection)
ORM†	ours	Search-R1 7B	final answer correctness

B Evaluation protocol

Decoding harness. All guided-decoding cells share one harness: 5 candidates per step, temperature 0.9, top-p 0.95, at most 512 new tokens per step, and at most 8 agent turns on the retrieval domains (question sets in Table 7); on MATH-500 and MMLU-Pro a step is a paragraph, capped at

Table 9: External baselines for the three domains beyond HotpotQA, by comparison block. "Label rule" is how each baseline's own step supervision was produced. †: in domain. VersaPRM is initialised from Llama-PRM800K, a Llama-3.1-8B-Instruct fine-tuned on PRM800K.

Baseline	Checkpoint	Backbone	Label rule
PaperSearchQA and its biomedical transfer battery (PubMedQA, MMLU-Med, BioASQ factoid)*			
Med-PRM†	`dmis-lab/llama-3.1-medprm-reward-v1.0`	Llama-3.1-8B-Instruct	retrieval-augmented LLM judge (Gemini-2.0-flash), first error
RAG-Gym ReAct-MedQA-PRM†	`RAG-Gym/ReAct-MedQA-PRM`	Llama-3.1-8B-Instruct	action ranking by a GPT-4o judge
VersaPRM	`UW-Madison-Lee-Lab/VersaPRM`	Llama-PRM800K (LoRA r=16)	LLM judge, first error
MATH-500			
Qwen2.5-Math-PRM-7B	`Qwen/Qwen2.5-Math-PRM-7B`	Qwen2.5-Math-7B	Monte-Carlo estimation + 72B judge
Math-Shepherd	`peiyi9979/math-shepherd-mistral-7b-prm`	Mistral-7B	Monte-Carlo estimation
MMLU-Pro			
VersaPRM†	`UW-Madison-Lee-Lab/VersaPRM`	Llama-PRM800K (LoRA r=16)	LLM judge, first error

1,024 new tokens, sampled at top-p 0.9 (Appendix L). *Greedy* is step-wise best-of-5 (5 PRM calls per step). *Beam* keeps the B=4 highest-valued partial trajectories and re-ranks all 20 expansions on the path aggregate (20 PRM calls per step). *MCTS* (b=5, 16 simulations, PUCT with c_{puct}=1.5) rolls each selected leaf out under step-wise best-of-5, expands it with the rollout's first-step candidates, backs up the path value (mean per-step score), and returns the answered trajectory with the highest path value. On MATH-500 and MMLU-Pro, greedy ran at a 25-step depth cap for ours and the ORM and at 50 for the external PRMs; beam and MCTS ran at 50 for every arm. The search columns are not budget-matched: an MCTS number above a beam number bought it with more PRM calls. External PRMs score the same step boundaries and retrieved passages; the RAG-Gym and Agent-RRM rows on HotpotQA run through a second decode file whose random-pick control differs from ours by at most 1.4 points in any column (Table 10). *Best-of-N* reranks the first N of one shared pool of 256 generations per question, so every comparison is paired. Each arm is read at its own best aggregation of the per-step scores over {mean, sum, last, min} (chosen by its mean curve over $N \geq 2$, granted to every PRM alike; the in-sample inflation is 1–3 points for all arms). The whole-trajectory ORM has one score per completion, so all its aggregations coincide.

Graders. Retrieval, MMLU-Pro and biomedical cells use strict exact match on the extracted, normalised answer; MATH-500 uses the boxed verifier. Every number is a regrade of the stored predictions, not the runner's containment score (3–10 points higher). BioASQ factoid is graded against a single-spelling gold (the official gold is registration-gated), which understates every arm equally.

Statistics. Arms in a column decode the same questions with the same frozen generator, so they are compared with an exact paired McNemar test on the per-question outcomes (two-sided, α=0.05, uncorrected; a Holm-adjusted p changes no quoted verdict). Bold marks the best value in a column, a dagger (†) every value the test cannot separate from it; unmarked values are significantly worse. The generator-transfer tables (Section 5 and Table 17) instead test each row against its own random-pick control, since rows carry different generators. AgentProcessBench uses a paired cluster bootstrap (Appendix G); the best-of-N tables carry no test (Appendix E).

Controls. Table 10 records the decoding controls not drawn in the main tables: random pick (the same budget with the step chosen at random) and, where run, the untrained backbone with the same readout.

C Training recipe details

Sequence and readout. A training sequence is

$$x \;\; s_1 \texttt{<extra_0>} \;\; s_2 \texttt{<extra_0>} \;\; \ldots \;\; s_T \texttt{<extra_0>},$$

where `<extra_0>` is an atomic added token (Qwen2.5's reserved id 151651, embedding seeded from "\n\n") spliced in after each step, with the text otherwise untouched. Atomicity makes the readout position identical in training text and decode-time prefixes. At each delimiter the frozen LM head's rows for "+" (id 10) and "–" (id 12) give the margin m_t of Section 3.3; full vocabulary logits are never materialised. Neither token appears in the input, and causal masking gives step t's delimiter only steps $1 \ldots t$.

Answer probe and prompt suffix. ℓ_t sums the correct final answer's token log-probabilities under the generator, teacher-forced after the prefix $x, s_1 \ldots s_t$ and a short elicitation. On the `<answer>`-tag domains the scored span is `<answer>`$a^\star$`</answer>` after "Therefore, the answer is"; on the boxed domains `\boxed{` is part of the elicitation and only $a^\star$} is scored. Questions are shown through the backbone's own chat template with a format instruction appended, which defines "a step" and makes the final answer machine-readable:

Table 10: Decoding controls, strict exact match (greedy / beam $B{=}4$ / MCTS $b{=}5$). "Ours" and "external" are the two HotpotQA decode files. The MuSiQue and StrategyQA controls use the original column sizes (500 and 140 questions); MATH-500 uses the boxed verifier.

Set	Control	Greedy	Beam	MCTS
HotpotQA	random pick, ours harness	0.494	0.504	0.478
HotpotQA	random pick, external harness	0.490	0.494	0.492
2WikiMultiHopQA	random pick	0.336	0.344	0.356
MuSiQue (500 q)	random pick	0.278	0.290	0.278
PopQA	random pick	0.380	0.338	0.352
StrategyQA (140 q)	random pick	0.636	0.557	0.593
PaperSearchQA	random pick	0.442	0.464	0.456
PaperSearchQA	untrained backbone	0.458	0.430	0.412
PubMedQA*	random pick	0.668	0.670	0.670
PubMedQA*	untrained backbone	0.618	0.630	0.628
MMLU-Med	random pick	0.745	0.736	0.755
MMLU-Med	untrained backbone	0.734	0.756	0.743
BioASQ factoid	random pick	0.262	0.282	0.266
BioASQ factoid	untrained backbone	0.248	0.250	0.244
MATH-500	random pick	0.788	0.710	0.848
MMLU-Pro	random pick	0.534	0.479	0.541

```
\nThink step by step and denote the start of each step with
"Step i:" and return the final answer in <answer></answer>
tags.
```

(on the boxed domains the same sentence ends "... and place your final answer inside \boxed{}").

Optimization. Full fine-tune of the generator's own 7B backbone: one epoch, learning rate 2×10^{-5}, cosine schedule with 5% warmup, batch of 32 sequences. 10% of *questions* are held out (so no generation of a training question appears in validation) and the reported checkpoint is the best by validation loss. Per-checkpoint constants (γ, τ, context, grader) are the lower rows of Table 7.

D Model and recipe transfer

Table 11 carries the generator- and recipe-transfer results of Section 5. Each row decodes a different generator, so it is tested against its own random-pick control at the same budget; a gain is bold when an exact paired McNemar test separates it from zero. The MATH analogue is Table 17 (Appendix F).

Table 11: Transfer of the HotpotQA PRM under guided decoding, against a random-pick control. *Training*: the models that generate the training trajectories, compute their labels, and initialize the backbone; *Inference*: the generator the PRM guides. (a) The frozen PRM guides a different generator. (b) All training components replaced with Llama-3.2-3B. (c) Only the labeling model replaced, threshold fixed to the no-transfer value. Every gain is bold: each is separated from zero (exact paired McNemar, $p < 0.001$).

	Training			Inference	Accuracy		
Setting	**Generator**	**Labeling model**	**Backbone**	**Generator**	**PRM-guided**	**Random**	**Gain (pts)**
No transfer	Qwen2.5-7B	Qwen2.5-7B	Qwen2.5-7B	Qwen2.5-7B	0.632	0.478	**+15.4**
(a) Different generator at inference							
Family and scale	Qwen2.5-7B	Qwen2.5-7B	Qwen2.5-7B	Llama-3.2-3B	0.584	0.420	**+16.4**
Scale only	Qwen2.5-7B	Qwen2.5-7B	Qwen2.5-7B	Qwen2.5-3B	0.598	0.424	**+17.4**
(b) All training components changed							
No transfer	Llama-3.2-3B	Llama-3.2-3B	Llama-3.2-3B	Llama-3.2-3B	0.594	0.420	**+17.4**
Family and scale	Llama-3.2-3B	Llama-3.2-3B	Llama-3.2-3B	Qwen2.5-7B	0.616	0.478	**+13.8**
(c) Labeling model changed							
Larger model	Qwen2.5-7B	Qwen2.5-14B	Qwen2.5-7B	Qwen2.5-7B	0.626	0.478	**+14.8**

E Best-of-N tables

Tables 12 and 13 give the HotpotQA, MATH-500 and MMLU-Pro best-of-N curves of Figures 4 and 5 as numbers, under the protocol of Appendix B. No paired test is run on these curves; bold marks the best reward model in each column (self-consistency, random pick and pass@N are not eligible), with no dagger drawn.

Table 12: Best-of-N on multi-hop agentic search. Bold is the best reward model at that depth.

method	**agg.**	N=1	2	4	8	16	32	64	128	256
HotpotQA (test)										
DeltaPRM	mean	0.370	**0.416**	**0.420**	**0.442**	**0.448**	**0.464**	**0.474**	**0.482**	**0.482**
VersaPRM	last	0.370	0.378	0.370	0.370	0.396	0.406	0.434	0.436	0.432
Agent-RRM	mean	0.370	0.390	0.402	0.414	0.414	0.420	0.436	0.430	0.426
RAG-Gym ReAct-HotpotQA	sum	0.370	0.370	0.382	0.386	0.394	0.406	0.398	0.392	0.392
ORM	—	0.370	0.402	**0.420**	**0.442**	**0.448**	0.460	**0.474**	0.466	0.476
Self-consistency	—	0.370	0.378	0.410	0.420	0.430	0.428	0.432	0.428	0.434
Random pick	—	0.370	0.364	0.359	0.363	0.365	0.364	0.363	0.364	0.364
pass@N	—	0.370	0.444	0.494	0.540	0.588	0.618	0.650	0.682	0.698

Table 13: Best-of-N on MATH-500 (500 questions, boxed verifier, per-generation accuracy 0.809) and MMLU-Pro (534, strict EM, 0.452). "agg." is the aggregation of the per-step scores each arm is read at. Bold is the best reward model at that depth.

method	**agg.**	N=1	2	4	8	16	32	64	128	256
MATH-500										
DeltaPRM	sum	0.806	0.840	0.850	0.854	0.874	0.888	0.900	0.892	0.892
Qwen2.5-Math-PRM-7B	mean	0.806	**0.842**	**0.862**	**0.870**	**0.888**	**0.892**	**0.906**	**0.906**	**0.914**
Math-Shepherd	min	0.806	**0.842**	0.844	0.848	0.844	0.856	0.852	0.854	0.846
ORM	—	0.806	**0.842**	0.854	0.858	0.856	0.848	0.838	0.830	0.814
Self-consistency	—	0.806	0.824	0.850	0.874	0.886	0.884	0.894	0.890	0.892
Random pick	—	0.806	0.804	0.810	0.809	0.807	0.807	0.808	0.809	0.809
pass@N	—	0.806	0.864	0.894	0.922	0.942	0.954	0.962	0.972	0.976
MMLU-Pro										
DeltaPRM	last	0.457	**0.511**	**0.581**	**0.601**	**0.618**	**0.624**	**0.639**	**0.622**	**0.607**
VersaPRM	mean	0.457	0.472	0.522	0.530	0.532	0.545	0.567	0.554	0.547
ORM	—	0.457	0.496	0.551	0.562	0.582	0.582	0.575	0.560	0.556
Self-consistency	—	0.457	0.481	0.543	0.560	0.577	0.575	0.579	0.577	0.586
Random pick	—	0.457	0.449	0.457	0.452	0.451	0.453	0.453	0.453	0.452
pass@N	—	0.457	0.552	0.674	0.719	0.770	0.822	0.845	0.873	0.899

F Results on the other domains

This section collects results beyond the main text's domains: the PaperSearchQA checkpoint (agentic search over scientific abstracts), with its biomedical transfer battery (Table 14) and step-level verification (Table 15); and, for the MATH checkpoint of Section 5, the math transfer battery (Table 16) and generator transfer on MATH-500 (Table 17). All cells follow Appendix B; the compute behind each PRM is in Appendix I.

PaperSearchQA and its biomedical transfer battery. PaperSearchQA is a single-hop paper-search dataset with GPT-written questions over PubMed abstracts. Single-hop retrieval rewards finding the correct document—in effect an outcome reward—so PRMs are not typically trained on it; we train DeltaPRM on it as a sanity check that our labels capture outcome information on par with an actual ORM. DeltaPRM matches the ORM's MCTS accuracy on PaperSearchQA and on the transfer set PubMedQA, and exceeds it on BioASQ factoid (Table 14).

Table 14: Guided decoding for the PaperSearchQA checkpoint: in domain (500 questions) and on its biomedical transfer battery (PubMedQA* MIRAGE 500, MMLU-Med 1,089, BioASQ factoid 500), strict exact match, marked as Table 1.

	PaperSearchQA		**PubMedQA***		**MMLU-Med**		**BioASQ factoid**	
method	beam	MCTS	beam	MCTS	beam	MCTS	beam	MCTS
DeltaPRM	0.594†	0.610†	0.716	0.738†	0.783	0.764	**0.328**	**0.340**
Med-PRM	0.502	0.500	**0.750**	**0.756**	0.787	0.786	0.304†	0.294
RAG-Gym ReAct-MedQA-PRM	0.492	0.480	0.680	0.688	0.783	0.791	0.268	0.280
VersaPRM	0.486	0.512	0.728†	0.712	**0.817**	**0.815**	0.300†	0.302
ORM	**0.620**	**0.618**	0.744†	0.754†	0.758	0.764	0.312†	0.310

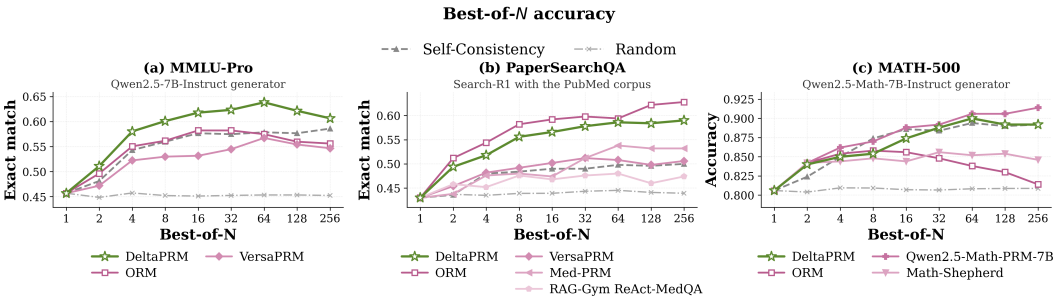


Figure 5: Best-of-N reranking on MMLU-Pro, PaperSearchQA and MATH-500: one fixed pool of 256 generations per question per panel (534 questions on MMLU-Pro, 500 on the other two), the same method swap as Figure 4. Numbers for MATH-500 and MMLU-Pro in Table 13.

Step-level verification with the PaperSearchQA checkpoint. Table 15 reads the PubMed-corpus methods on the same `searchR1_hotpotqa` subset of AgentProcessBench as Table 2, so every row is corpus transfer. Med-PRM takes the AUROC column, with the PaperSearchQA checkpoint and VersaPRM inseparable from it; the threshold columns separate no PRM (Appendix G).

Table 15: Step-level process quality of the PubMed-corpus methods on AgentProcessBench's `searchR1_hotpotqa` subset (corpus transfer for every row); metrics, thresholds and paired cluster bootstrap as Table 2.

	DeltaPRM	Med-PRM	RAG-Gym ReAct-MedQA	VersaPRM	ORM
AUROC	73.5†	**78.3**	38.3	70.1†	60.4
Step acc.	68.4†	**70.6**	68.0†	67.8†	68.4†
First-error acc.	59.6†	60.8†	58.4†	60.4†	**61.2**

Math transfer battery. Table 16 carries the MATH checkpoint, unchanged, to nine further math benchmarks under the same two decoders. It takes or ties the best value in 4 of 9 beam and 3 of 9 MCTS columns, and is not significantly worse than the best everywhere except Olympiad under MCTS: transfer is parity with Qwen2.5-Math-PRM-7B and the ORM rather than a win. Accuracy against each PRM's compute is drawn in Figure 6.

Table 16: Guided decoding across nine further math benchmarks, beside the in-domain MATH-500 row. *Ours* DeltaPRM, *Qwen* Qwen2.5-Math-PRM-7B, *Shep* Math-Shepherd, and the ORM; every cell uses the same frozen generator (Qwen2.5-Math-7B-Instruct) and the MATH-500 decoder settings (beam B=4; MCTS b=5, 16 simulations; 5 candidates per step; 50-step cap), boxed verifier. Question counts: 500; then 30, 30, 40, 134, 272, 674, 500, 500, 500 (Omni-MATH is a seeded subsample). Within a row of a decoder block, bold is the best (ties share it) and † every value McNemar cannot separate from it. The *transfer mean* macro-averages the nine transfer sets and carries no test.

	Beam B=4				**MCTS** b=5			
Set	Ours	Qwen	Shep	ORM	Ours	Qwen	Shep	ORM
MATH-500	0.882†	**0.890**	0.828	0.860	**0.894**	0.886†	0.850	0.880†
AIME'24	0.133†	**0.267**	0.167†	0.233†	0.200†	**0.233**	0.167†	**0.233**
AIME'25	0.100†	0.167†	0.133†	**0.233**	0.200†	**0.300**	0.167†	0.133†
AMC23	**0.625**	**0.625**	0.525†	**0.625**	0.600†	**0.650**	0.575†	0.625†
MATH L5	**0.716**	0.694†	0.642	0.634	0.731†	**0.739**	0.664	**0.739**
Minerva	0.360†	**0.364**	0.324†	0.342†	**0.379**	**0.379**	0.364†	0.371†
Olympiad	0.497†	**0.515**	0.423	0.475	0.512	**0.546**	0.481	0.475
Big-Math	**0.760**	0.758†	0.690	0.748†	**0.776**	0.768†	0.710	**0.776**
Numina	0.350†	0.312	0.266	**0.360**	**0.364**	0.360†	0.322	0.354†
Omni-MATH	**0.362**	0.334†	0.270	0.332†	0.372†	**0.390**	0.322	0.358†
transfer mean	*0.434*	*0.448*	*0.382*	*0.443*	*0.459*	*0.485*	*0.419*	*0.452*

Generator transfer on MATH-500. Table 17 repeats the generator-transfer protocol of Section 5 for the MATH checkpoint. The frozen checkpoint transfers to Llama-3.1-8B-Instruct and to Qwen2.5-Math-1.5B-Instruct. Re-deriving the pool and labels on Llama buys nothing over trans-

ferring the frozen PRM; swapping the backbone to Llama is what breaks it (gain +26.6 to +7.2), and the reverse direction with the Llama backbone does not transfer. Labels from a 14B model do not improve on the 7B labels.

Table 17: Generator transfer of the MATH checkpoint, MCTS b=5 on MATH-500, boxed verifier. Read as Table 11: the gain is guided minus each row's own random-pick control, in points, bold when separated from zero (exact paired McNemar; p in the last column). The recipe rows change the PRM itself: the trajectory and label model, and the backbone.

Setting	Generator scored	Guided	Random pick	Gain	p
The PRM is frozen; only the generator scored changes					
In domain	Qwen2.5-Math-7B-Instruct	0.894	0.848	**+4.6**	<0.001
Other model family	Llama-3.1-8B-Instruct	0.744	0.478	**+26.6**	<0.001
Other model family, scored in our own template	Llama-3.1-8B-Instruct	0.752	0.478	**+27.4**	<0.001
Smaller generator, same family	Qwen2.5-Math-1.5B-Instruct	0.852	0.738	**+11.4**	<0.001
The recipe is re-run on the other model; the PRM itself changes					
Its trajectories and labels (Qwen backbone)	Llama-3.1-8B-Instruct	0.744	0.478	**+26.6**	<0.001
Its trajectories, labels and backbone	Llama-3.1-8B-Instruct	0.550	0.478	**+7.2**	0.001
Reverse: its trajectories and labels (Qwen backbone)	Qwen2.5-Math-7B-Instruct	0.874	0.848	**+2.6**	0.047
Reverse: its trajectories, labels and backbone	Qwen2.5-Math-7B-Instruct	0.828	0.848	−2.0	0.203
Scale of the labelling model (Qwen2.5-Math-7B-Instruct scored)					
Labels from a 14B model, threshold refit on its pool	Qwen2.5-Math-7B-Instruct	0.880	0.848	**+3.2**	0.026
Labels from a 14B model	Qwen2.5-Math-7B-Instruct	0.884	0.848	**+3.6**	0.008

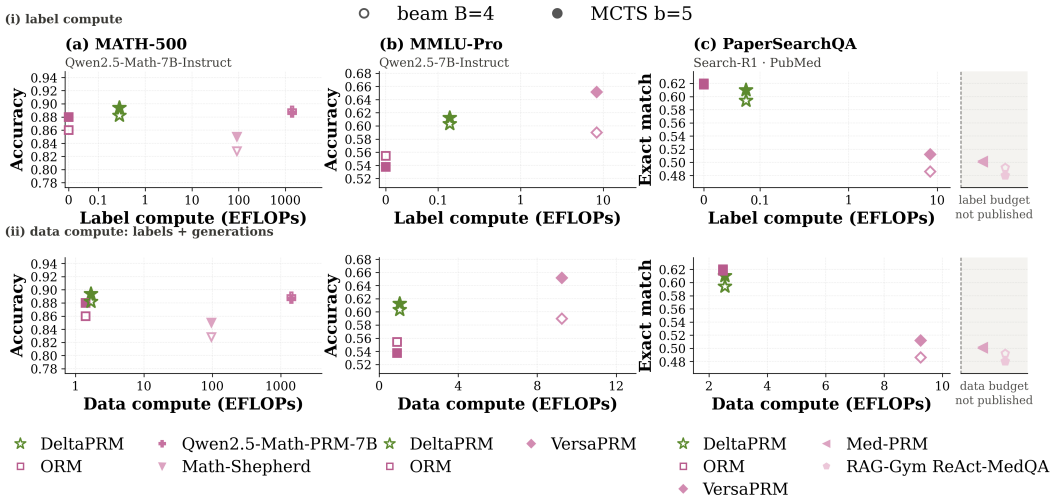


Figure 6: Accuracy against build compute on the three benchmarks off HotpotQA, read as Figure 1: row (i) charges only the labelling of the training data (Table 22), row (ii) the training-data budget, generation plus labels (Tables 21 and 22); the fine-tune is charged on neither. 500 questions per benchmark (534 on MMLU-Pro). The MMLU-Pro total axis is linear because the gap there is 8.9× rather than orders of magnitude.

G AGENTPROCESSBENCH PROTOCOL

Metrics. AUROC is computed over the pooled labeled steps with neutral (0) steps dropped, tie-corrected; its paired test is a cluster bootstrap over the query index, re-ranked inside each resample (2,000 resamples, index sets shared across arms). StepAcc is the share of labeled steps classified correctly at a threshold; FirstErrAcc is the share of trajectories whose first erroneous step the PRM points at (an error-free trajectory counts only if no step is flagged). Each row's threshold maximises that column's metric on the evaluated set, granted to every row alike. Under a held-out protocol (a cut fitted on four of five folds grouped by query index) every value drops two to three points and the HotpotQA StepAcc column reorders: ours reads 65.1 / 53.2 and RAG-Gym ReSearch takes the column at 68.0 (p=0.019 uncorrected, ∼0.11 Holm-corrected). The random control gains only +0.2 / +0.4 from the in-sample fit, bounding what the fitted cut buys from noise.

Positional control. The benchmark propagates a −1 forward, so later steps are worse by construction and "score = minus the step index" reaches 75.0 AUROC. Coverage differs between arms (our 7B checkpoints drop 34 steps beyond the 16,384-token context, VersaPRM 56 at 8,192, the Llama

arms none), so each row is read against the control on exactly the steps it scored. Rows that do not beat their own control: the ORM and $\lambda = 0$ ablation on HotpotQA (63.0 and 39.4 against 69.2 and 75.0), the three RAG-Gym HotpotQA adapters (44.2, 64.6, 34.1), the ORM of the PaperSearchQA checkpoint (60.4) and RAG-Gym ReAct-MedQA-PRM (38.3), all against 75.0. Trajectory-level bootstrap: the HotpotQA checkpoint beats position by $+9.2$ $[+1.5, +17.1]$, the PaperSearchQA checkpoint by $+4.3$ $[-3.8, +12.6]$ (not significant); their paired difference is $+4.9$ $[+1.7, +8.0]$.

Why the threshold columns do not separate. StepAcc's gain over "flag nothing" is $(n_{\text{err}} \cdot \text{TPR} - n_{\text{ok}} \cdot \text{FPR})/N$, so a cut pays only where the ROC curve is steeper than the prior odds $n_{\text{ok}}/n_{\text{err}} = 2.79$ (179 harmful against 555 other labeled steps). Most PRMs' optimal cut therefore flags almost nothing: the first error is caught on 0–33% of error-bearing trajectories while 74–100% of error-free ones are called clean. Two fixed handicaps sit under every row: the 56 neutral steps are unreachable under a binary cut (ceiling 92.4), and steps a PRM drops for context count as wrong under the benchmark's reference-denominator rule.

H GRPO DETAILS

Each rollout's advantage is $A_i = A_i^o + \lambda\, A_i^{\text{PRM}}$ with λ=0.3, where A_i^o is the group-normalized advantage of the exact match outcome and

$$A_i^{\text{PRM}} = \begin{cases} \dfrac{v_i - \frac{1}{|\mathcal{C}(i)|}\sum_{j\in\mathcal{C}(i)} v_j}{\text{std}_{\text{batch}}(v)} & \text{if } |\mathcal{C}(i)| \geq 2, \\ 0 & \text{otherwise,} \end{cases}$$

$$\mathcal{C}(i) = \left\{\, j \in g(i) : o_j = o_i,\ n_{\text{srch},j} = n_{\text{srch},i} \,\right\},$$

where the trajectory score $v_i = \sum_{t=1}^{T} G_t$ sums the discounted returns $G_t = \sum_{k=t}^{T} \gamma^{\,k-t}\,(V_k - V_{k-1})$ over the trajectory's T steps, with V_t the PRM's score after step t ($V_1 - V_0 \coloneqq 0$) and γ=0.8.

Table 19 gives the shared training configuration. The runs of this appendix start from Qwen2.5-7B-Instruct, a generator not already trained on HotpotQA with an outcome reward; the comparison of Table 3 in the main text instead starts from the Qwen2.5-7B *base* model, whose outcome-only baseline is accordingly weaker and seed-sensitive (45.2 and 58.1 strict EM on the two seeds, against the instruct run's 59.4). Each step draws 64 prompts from a fixed stream of 10,000 HotpotQA training questions (disjoint from the PRM's pool) and samples 10 rollouts per prompt at temperature 1, retrieving from the same wiki-18 E5 index the PRM was built on. Table 18 breaks the pooled main-text comparison (Table 3) down by training seed. The reward-side run uses $r = o + \lambda\bar{v}$, with o the strict-EM outcome, $\bar{v}$ the mean per-step $P(+)$ and λ=0.3. The rollout-side run keeps $r = o$; the last 5 of every 10 rollouts are produced by step-wise best-of-4 under the PRM (each candidate turn completed with its own retrieval before scoring), and each failed trajectory's outcome advantage is redistributed over its steps by a softmax over the PRM's value drops (temperature 0.5, half of each step's mass kept uniform), with the trajectory total unchanged.

Table 18: GRPO final checkpoints on 3,000 held-out HotpotQA questions, by training seed: the base-model runs pooled in Table 3. The reward is the conditioned bonus $r = o + \lambda\,[\,v - \mathbb{E}(v \mid o, n_{\text{search}})]$, λ=0.3, with o the exact-match outcome, v the summed PRM score and n_{search} the search count. $\pm$ is half the 95% CI width (paired bootstrap over questions, 10,000 resamples). Bold: the row's best arm; both baselines are significantly worse (exact paired McNemar, α=0.05).

Seed	DeltaPRM	Outcome	VersaPRM
42	**59.0** $\pm$ 1.8	45.2 $\pm$ 1.8	56.5 $\pm$ 1.8
43	**60.0** $\pm$ 1.8	58.1 $\pm$ 1.8	56.7 $\pm$ 1.8

I COMPUTE ACCOUNTING

We compare the cost of producing process labels with the two dominant annotation strategies, Monte-Carlo (MC) estimation and LLM-as-a-judge. Since MC estimation has not previously been applied to search agents, we instantiate it ourselves following the Math-Shepherd protocol (Wang et al., 2024a), sampling K=8 continuations from each step to a final answer. For LLM-as-a-judge, we cover the three designs in the literature: a judge that issues a verdict at every prefix (a 72B model in our instantiation) (Khalifa et al., 2025), one that labels a whole rollout in a single generation (Llama-3.1-70B for VersaPRM, the sparse 120B-total GPT-OSS-120B for Agent-RRM) (Zeng

Table 19: GRPO configuration, shared by the outcome-only run and the PRM runs. The main-text comparison (Table 3) differs only in the initial checkpoint (the Qwen2.5-7B base model) and in running two seeds (42 and 43, Table 18).

Generator	Qwen2.5-7B-Instruct, full fine-tune
Prompts per step / rollouts per prompt	64 / 10
Steps	156 (one pass over 10,000 prompts)
Learning rate / warmup	10^{-6} / 44 steps
KL coefficient	0.001 (resident reference policy)
Max turns / tokens per turn	8 / 1,024
Max retrieved tokens per turn	500 (top-3 passages, wiki-18, E5)
Context	16,384
Reward	strict exact match ($+\lambda\bar{v}$, λ=0.3, reward-side run only)
Guided rollouts (rollout-side run)	5 of 10, step-wise best-of-4 under the PRM
Credit (rollout-side run)	blame-weighted: softmax over PRM value drops, temperature 0.5, floor 0.5
Reward model	DeltaPRM, frozen; or VersaPRM, frozen
Evaluation	greedy strict EM; 200 q every 10 steps, 500 q and 3,000 q at the end
Seed	42

et al., 2025; Fan et al., 2026a), and GPT-4o ranking candidate next steps (Xiong et al., 2026). All methods label the same pool of 76,000 Search-R1 rollouts on HotpotQA (256,164 steps).

Pricing convention. We estimate compute with the standard approximation of $2P$ FLOPs per token forwarded by a P-parameter model (Kaplan et al., 2020), prefill and decode priced alike, attention terms dropped, and any prefix already resident in a KV cache free. A rule that forwards T tokens per rollout thus costs $2PnT$ FLOPs over a pool of n rollouts. Fine-tuning is charged at $6P$ FLOPs per token per epoch for a full fine-tune and $4P$ for LoRA. Every label rule is charged at its cheapest correct implementation: ours forwards the $S{+}1$ copies of the wrapped correct final answer (11–17 tokens) against the sampler's KV cache, fused into generation; an MC completion is charged only its decoded remainder; a judge is charged the solution it reads plus its decoded verdict, with the system prompt cached.

What is measured and what is assumed. Our rows are token censuses of the files we trained on; external rows apply the papers' published counts at our census's mean lengths, and a cell reads *undisclosed* where the source paper publishes no count. FLOPs price neither retrieval nor GPU time, and decoding is memory-bound, so the ratios over MC are lower bounds; the hosted judges' list-price API bill is the only dollar figure. Two heavier bookkeepings of our own rule are kept beside the fused one: a standalone rescore that reads each solution once (24–37$\times$ the fused figure on the two agent pools) and the as-run figure, a full forward per prefix with no reuse (70–89$\times$).

Tables. Table 20 prices one process label per rule on the HotpotQA pool; Table 21 gives the training data behind each PRM and the compute to sample it, Table 22 the label leg on every domain, and Table 23 the fine-tune.

J Ablation details

Arms. Every arm of Table 25 is trained on the HotpotQA pool of Table 7 with the paper checkpoint's flags and one change, asking three questions. *Which quantity should the label be?* The raw log-likelihood ℓ_t, its per-step gain δ_t (the return at γ=0), or the discounted return G_t. *How far should credit reach?* A sweep of $\gamma \in \{0, 0.3, 0.5, 0.9, 0.99, 1\}$, each soft arm at τ=std(G) for its own γ; the γ=0.3 row is the paper's checkpoint (τ=15.11). *How should the target be written?* The soft target $\sigma(G_t/\tau)$ against its hard discretisation (the same cross-entropy toward $\mathrm{sign}(G_t) \in \{0, 1\}$ at γ=0.3), and with the outcome auxiliary of the loss (Section 3.3) switched off (λ=0); the *log-likelihood* arm regresses the margin on the undifferenced ℓ_t by mean-squared error with no outcome term. The two outcome-only rows are the ORM and $\lambda = 0$ ablation of Section 5, where the step label is removed altogether. Each arm is read two ways on the validation questions: best-of-256 reranking of one fixed pool (400 questions, strict EM), which involves no decoding and so isolates the PRM (quoted below), and beam B=4 on 500 questions (Table 25). MCTS was decoded only for the paper's checkpoint and the ORM and $\lambda = 0$ ablation (Table 5); greedy is measured but not printed.

What the label is. Differencing turns the likelihood into a step label: the undifferenced ℓ_t says how likely the answer already is after a prefix—a quantity that rises across every step of a solved question whether or not the step did anything—while δ_t asks what the step changed and G_t what it changed for later steps. Under beam search the undifferenced arm reads 0.536 against 0.616 (gain)

Table 20: What a process label costs on our HotpotQA pool (n=76,000 rollouts, 256,164 steps; census means per rollout: $\bar{S}$=3.37 steps, R=1,422 response tokens, L=1,590 prompt+response tokens, A=15.7-token gold answer; judge verdicts floored at c=20 tokens, as-run ∼37). Each rule forwards T tokens per rollout, so label EFLOPs = $2PnT \times 10^{-18}$, with P the judge each method actually used; Agent-RRM's GPT-OSS-120B is a sparse MoE charged at total parameters (at its 5.1B active parameters per token: 1.2 EFLOPs, 16×); our row's formula reproduces the exact tally to 0.1%. MC's 1.6M extra retriever calls are unpriced by FLOPs, and RAG-Gym's GPT-4o ranker has undisclosed size, so both ratios are floors.

Method	**Label source (P)**	**Label calls**
MC, K=8 (Math-Shepherd rule)	generator (7.6B) + retriever	K per step
ThinkPRM	LLM-as-a-judge (72B)	1 per step
VersaPRM	LLM-as-a-judge (70B)	1 per rollout
Agent-RRM	LLM-as-a-judge (120B MoE)	1 per rollout
RAG-Gym ReAct	LLM-as-a-judge (GPT-4o, API)	1 per step
DeltaPRM	**generator (7.6B)**	**1 fwd per step**

Method	**Labeler tokens T / rollout (what is forwarded)**	**Tokens / step**	**Extra search calls / step**	**Label EFLOPs (pool, $2PnT$)**	**× ours**
MC, K=8 (Math-Shepherd rule)	$K\bar{S}R/2$: decoded remainders of K fresh completions per step	5,687	6.3	22.1	278×
ThinkPRM	$L + (\bar{S}{+}1)\,c$: one read, a verdict at every prefix	498	0	18.5	233×
VersaPRM	$L + c$: one read, one verdict	478	0	17.3	217×
Agent-RRM	$L + c$: one read, one verdict	478	0	28.6	359×
RAG-Gym ReAct	$10R$: 10 sampled candidate next steps (their n_{actions}=10), decoded (+ GPT-4o reads, unpriceable)	4,218	0	≳16 + API	≳207×
DeltaPRM	**$(\bar{S}{+}1)\,A$: gold-answer probes vs. the sampler's KV cache, nothing decoded**	**20**	**0**	**0.08**	**1×**

Table 21: Training data behind each PRM, and the compute to sample it. Agent-RRM's question count is a floor recovered from its released trajectories.

PRM	**Questions**	**Sols/q**	**Solutions**	**Steps**	**Gen tokens**	**Gen EF**	**× ours**
HotpotQA							
DeltaPRM	9,500	8	76,000	256,164	121M	1.8	1.0×
VersaPRM	5,750	16	84,098	487,380	57M	0.9	0.5×
Agent-RRM	≥ 29,486	*undisclosed*	118,000	397,728	188M	3.1	1.7×
RAG-Gym ReAct	1,000	*undisclosed*	*undisclosed*	*undisclosed*	*undisclosed*	*undisclosed*	*undisclosed*
$\lambda = 0$ ablation	9,500	8	76,000	256,164	121M	1.8	1.0×
ORM	9,500	8	76,000	76,000	121M	1.8	1.0×
PaperSearchQA							
DeltaPRM	9,500	8	76,000	191,206	163M	2.5	1.0×
VersaPRM	5,750	16	84,098	487,380	57M	0.9	0.4×
$\lambda = 0$ ablation	9,500	8	76,000	191,206	163M	2.5	1.0×
ORM	9,500	8	76,000	76,000	163M	2.5	1.0×
MMLU-Pro							
DeltaPRM	5,468	16	87,488	740,232	60M	0.9	1.0×
VersaPRM	5,750	16	84,098	487,380	57M	0.9	1.0×
$\lambda = 0$ ablation	5,468	16	87,488	740,232	60M	0.9	1.0×
ORM	5,468	16	87,488	87,488	60M	0.9	1.0×
MATH-500							
DeltaPRM	7,201	16	115,216	1.2M	93M	1.4	1.0×
Qwen2.5-Math-PRM-7B	500,000	7	3.5M	21M	2.8B	43	30×
Math-Shepherd	15,000	30	440,000	2.6M	353M	5.1	3.6×
$\lambda = 0$ ablation	7,201	16	115,216	1.2M	93M	1.4	1.0×
ORM	7,201	16	115,216	115,216	93M	1.4	1.0×

and 0.608 (return, γ=0.3) and is separated from the top; under best-of-256 the three read 0.515, 0.528, 0.535 and none is separated—the pattern of a method whose ranking is right but whose step-level resolution is not.

How far credit reaches. A retrieval step's value is usually realized at once: the passage either contains what the answer needs or it does not, and ℓ_t moves at that step, so a small γ suits the agent domains while a large one risks handing a late answer's credit to early steps. The sweep agrees: under beam every $\gamma \in \{0, 0.3, 0.5, 0.9\}$ is inseparable from the best (0.608–0.630), while γ=0.99

Table 22: What produces each PRM's step labels and what it costs, at the cheapest KV-cached implementation of every rule. Ours: $S{+}1$ probes of the correct final answer per solution (11–17 tokens per pool), fused into generation. MC completions are charged their decoded response remainders; Agent-RRM's GPT-OSS-120B judge is a sparse MoE charged at total parameters as in Table 4 (at its 5.1B active parameters per token: 2.6 EF, 33×). A checkpoint read off-domain is labeled by its native rule at its native cost.

PRM	Rule	Model calls	Tokens	Label EF	× ours	Judge API
HotpotQA						
DeltaPRM	gold-answer log-prob at every step	332,164	5.2M	0.08	1.0×	$0
VersaPRM	70B judge, one pass per CoT	84,098	59M	8.3	105×	$36
Agent-RRM	GPT-OSS-120B critique + score per trajectory	118,000	258M	60	759×	$82
RAG-Gym ReAct	GPT-4o ranks candidate actions per state	*undisclosed*	*undisclosed*	*undisclosed*	*undisclosed*	*undisclosed*
ORM and $\lambda = 0$ ablation	final-answer string match	0	0	0	0×	$0
PaperSearchQA						
DeltaPRM	gold-answer log-prob at every step	267,206	4.6M	0.07	1.0×	$0
VersaPRM	70B judge, one pass per CoT	84,098	59M	8.3	120×	$36
ORM and $\lambda = 0$ ablation	final-answer string match	0	0	0	0×	$0
MMLU-Pro						
DeltaPRM	gold-answer log-prob at every step	827,720	9.1M	0.1	1.0×	$0
VersaPRM	70B judge, one pass per CoT	84,098	59M	8.3	60×	$36
ORM and $\lambda = 0$ ablation	final-answer string match	0	0	0	0×	$0
MATH-500						
DeltaPRM	gold-answer log-prob at every step	1.3M	19M	0.3	1.0×	$0
Qwen2.5-Math-PRM-7B	8 MC completions per step + one 72B judge pass	172M	58B	1,388	4,926×	$2,494
Math-Shepherd	8 MC completions per step	21M	6.8B	92	325×	$0
ORM and $\lambda = 0$ ablation	final-answer string match	0	0	0	0×	$0

Table 23: The fine-tune: $6P\cdot$ tokens $\cdot$ epochs for a full fine-tune, $4P$ for LoRA. Agent-RRM adds a GRPO leg whose 76,800 generated critiques are charged at $8P$. For our recipe the fine-tune, not the labeling, is the largest of the three legs.

PRM	Params	Train tokens	Epochs	Mode	Train EF	× ours
HotpotQA						
DeltaPRM	7.6B	121M	1	full	5.5	1.0×
VersaPRM	8.0B	57M	1	LoRA	1.8	0.3×
Agent-RRM	8.2B	427M	1	SFT + GRPO	24	4.3×
RAG-Gym ReAct	8.0B	*undisclosed*	1	LoRA	*undisclosed*	*undisclosed*
ORM and $\lambda = 0$ ablation	7.6B	121M	1	full	5.5	1.0×
PaperSearchQA						
DeltaPRM	7.6B	163M	1	full	7.4	1.0×
VersaPRM	8.0B	57M	1	LoRA	1.8	0.2×
ORM and $\lambda = 0$ ablation	7.6B	163M	1	full	7.4	1.0×
MMLU-Pro						
DeltaPRM	7.6B	60M	1	full	2.7	1.0×
VersaPRM	8.0B	57M	1	LoRA	1.8	0.7×
ORM and $\lambda = 0$ ablation	7.6B	60M	1	full	2.7	1.0×
MATH-500						
DeltaPRM	7.6B	93M	1	full	4.2	1.0×
Qwen2.5-Math-PRM-7B	7.6B	1.2B	1	full	55	13×
Math-Shepherd	7.2B	353M	1	full	15	3.6×
ORM and $\lambda = 0$ ablation	7.6B	93M	1	full	4.2	1.0×

(0.596) and γ=1 (0.582) fall significantly below; under best-of-256 no soft γ is separated. On the chain-of-thought domains, where a step's contribution is often realised only when a later step uses it, the recipe uses γ=0.8 (Table 7).

How the target is written. The soft target keeps a gain's size, so a search that lost the answer from context is charged more than one that returned a duplicate; the hard target keeps only the sign and costs 3.6 points under beam (0.572 against 0.608) and 3.5 under best-of-256, both significant. Removing the outcome auxiliary (λ=0) is the largest single loss in the table: 0.486 under beam and 0.465 under best-of-256, below every other soft arm. Both loss terms act on one projection of the hidden state (Section 3.3): the auxiliary is a second constraint on the same number, not a second head.

Outcome supervision at the same readout (ORM and $\lambda = 0$ ablation). The $\lambda = 0$ ablation shares DeltaPRM's pool, backbone, delimiter positions, trainer and decoding, with the likelihood

target replaced by the outcome bit copied to every step. On HotpotQA it costs 10.0 and 10.6 points under beam and MCTS, 7.4 points of best-of-256 accuracy (Table 5) and 15.4 AUROC on human step labels; on 2WikiMultiHopQA, PopQA and StrategyQA the gap is 8.6–24.0 points (measured, not printed). The whole-trajectory ORM—the ORM baseline of every results table—is far stronger under guided search (0.596 against 0.526 at MCTS) and ahead of the per-step $\lambda = 0$ ablation on best-of-N, its native task (0.476 against 0.408 at N=256), but is still significantly behind the step label under both decoders (p=0.016 and 0.013). Together the ORM and $\lambda = 0$ ablation say two things: the outcome broadcast to every prefix is a noisy step target (the same bit placed once at the end does better), and the likelihood label at those positions is not (it beats both). On PaperSearchQA the same ablation is worth nothing (Table 24), and under best-of-N and AgentProcessBench it goes the other way: PaperSearchQA trajectories are shorter (2.5 steps against 3.4), so the broadcast outcome bit is closer to a step label there, while on multi-hop questions a correct trajectory routinely contains a wasted search and a wrong one a correct first hop—exactly the within-outcome credit the likelihood label carries (Appendix K). This is the boundary of the result: the likelihood label pays where outcome supervision is a poor per-step signal, and is neutral where it is not.

Table 24 carries the same ablation across the other three domains: on MMLU-Pro the HotpotQA pattern repeats in weaker form; on MATH-500 and PaperSearchQA the three arms are inseparable almost everywhere, with the ORM taking the best PaperSearchQA search columns and the best MATH-500 greedy column.

Table 24: The ORM-twin ablation on the other three domains: strict accuracy under PRM-guided greedy, beam (B=4) and MCTS (b=5) decoding, marked as Table 1 (bold the best in a column, † not significantly worse, exact paired McNemar, α=0.05).

	PaperSearchQA			**MATH-500**			**MMLU-Pro**		
	Greedy	Beam	MCTS	Greedy	Beam	MCTS	Greedy	Beam	MCTS
DeltaPRM	**0.566**	0.594†	0.610†	0.838†	**0.882**	0.894†	**0.558**	**0.603**	**0.612**
$\lambda = 0$ ablation	0.548†	0.602†	0.610†	0.832†	0.876†	**0.898**	0.554†	0.573†	0.567
ORM	0.538†	**0.620**	**0.618**	**0.840**	0.860†	0.880†	0.513	0.554	0.537

Table 25: Label target ablation and γ ablation. Our PRM training recipe on multi-hop agentic search with only the training target changed; soft targets use τ=std(G) at their own γ. Bold is the best value and † marks every value an exact paired McNemar test cannot separate from it (α=0.05). The γ=0.3 row is the paper's DeltaPRM checkpoint.

Target	**Form**	γ	**Beam** B=4
how far credit reaches			
gain δ_t (return at γ=0)	$\sigma(G_t/\tau)$	0	0.616†
discounted return G_t (paper)	$\sigma(G_t/\tau)$	0.3	0.608†
discounted return G_t	$\sigma(G_t/\tau)$	0.5	**0.630**
discounted return G_t	$\sigma(G_t/\tau)$	0.9	0.624†
discounted return G_t	$\sigma(G_t/\tau)$	0.99	0.596
undiscounted return	$\sigma(G_t/\tau)$	1.0	0.582
how the target is written			
log-likelihood ℓ_t (undifferenced)	regression	—	0.536
discounted return G_t	sign(G_t)	0.3	0.572
discounted return G_t, no outcome auxiliary (λ=0)	$\sigma(G_t/\tau)$	0.3	0.486
$\lambda = 0$ ablation	BCE on o	—	0.508
ORM	BCE on o	—	0.570

K WORKED EXAMPLES

Figure 7 shows the label on two real rollouts from the HotpotQA training pool, and Figure **??** contrasts what each family of step label estimates. The label is not the outcome in disguise: a correct rollout is still charged for the turn it wasted (left, −48.2 nats for a search that returned two other people of the same name), and a wrong rollout is charged at the search that lost the answer from context (right, −52.7 nats), not at the final step that merely states the wrong answer (−6.1).

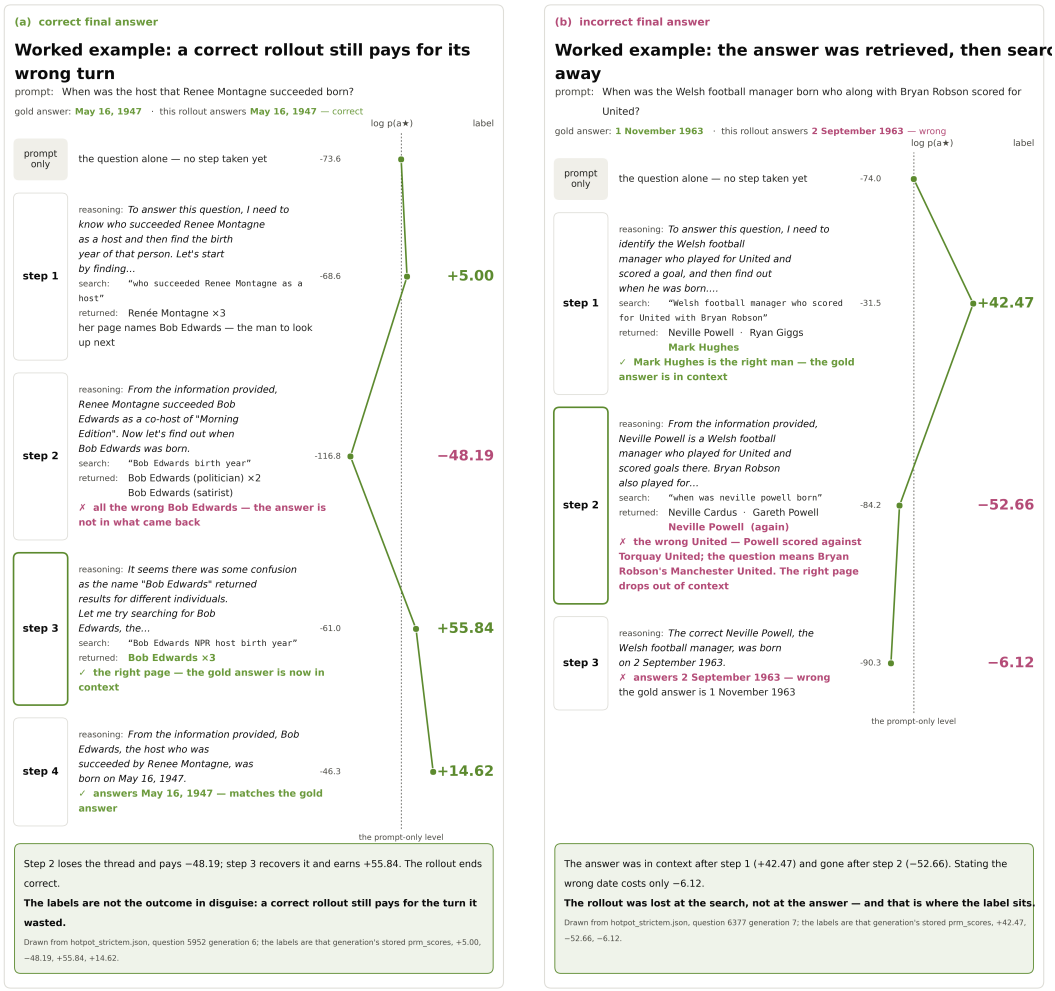


Figure 7: Two HotpotQA training rollouts with their per-step ℓ_t and labels δ_t in nats. Left: ends on the right answer but pays -48.2 for a wasted turn. Right: retrieved the answer's page at step 1 ($+42.5$) and searched it away at step 2 (-52.7); the wrong final answer costs only -6.1 because the answer was already lost.

L BEAM SEARCH AND MCTS DETAILS

Beam search and MCTS are test-time scaling methods that improve final-answer accuracy through additional sampling and PRM guidance, both at the granularity of a complete *step* rather than an individual token: a variable-length token sequence terminated by a pre-specified delimiter or EOS. In practice we use vLLM's `SamplingParams` to sample n continuations per prefix, with `stop` strings defining step boundaries and `max_tokens` capping each continuation's length; the search procedure scores the candidates with the PRM and selects which prefixes to extend.

Beam search and MCTS share the same generation and scoring setup: the sampling parameters, candidates per step n, per-step token budgets, and depth caps T on every domain are those of the decoding harness in Appendix B.

We define the *path value* of a partial or complete trajectory as its mean per-step PRM score, $\bar{V}(s_{1:t}) = \frac{1}{t}\sum_{j=1}^{t} V(x, s_{1:j})$, where $V(x, s_{1:j})$ evaluates step j conditioned on the prompt x and the full prefix.

Beam search (Algorithm 1). We use $B{=}4$. Each round samples $n{=}5$ candidate continuation steps from every live trajectory and the PRM scores each candidate given the prompt and the preceding trajectory, so a full beam generates and scores $Bn{=}20$ candidates. Trajectories are ranked by the mean of their per-step PRM scores and the highest-scoring B unfinished ones are retained; candidates that emit a final answer are stored separately and do not consume live slots. Search stops when

Algorithm 1 PRM-guided beam search with mean path scores

Require: prompt x, generator π, PRM V, beam width B, candidates per step n, maximum depth T

$s_0 \leftarrow$ empty trajectory
$\mathcal{L} \leftarrow \{s_0\}$ ▷ Live trajectories
$\mathcal{D} \leftarrow \emptyset$ ▷ Finished trajectories
for $t = 1, \ldots, T$ **do**
  $\mathcal{C} \leftarrow \emptyset$ ▷ Candidates
  for all $s \in \mathcal{L}$ **do**
    Sample $a_1, \ldots, a_n \sim \pi(\cdot \mid x, s)$
    Form s_j by extending s with a_j, for $j = 1, \ldots, n$
    Store $r_j \leftarrow V(x, s_j)$ with each s_j, for $j = 1, \ldots, n$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{s_j\}_{j=1}^{n}$
  end for
  Sort $\mathcal{C}$ by $\bar{V}(s)$, descending
  $\mathcal{L} \leftarrow \emptyset$
  for all $s \in \mathcal{C}$ in order **do**
    if $\mathrm{Done}(s)$ **then**
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{s\}$ ▷ Finished; does not consume a live slot
    else if $|\mathcal{L}| < B$ **then**
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{s\}$
    end if
  end for
  if $\mathcal{L} = \emptyset$ **then**
    break
  end if
end for
$\mathcal{P} \leftarrow \mathcal{D}$ if nonempty, otherwise $\mathcal{L}$
$s^* \leftarrow \arg\max_{s \in \mathcal{P}} \bar{V}(s)$
return s^*

no unfinished trajectories remain or depth T is reached, and returns the finished trajectory with the highest mean PRM score.

MCTS (Algorithm 2). We use k=16 simulations and c_{puct}=1.5. Each tree node holds a step prefix, a visit count N, and a total backed-up value W. Selection descends by PUCT,

$$u \;\leftarrow\; \operatorname*{arg\,max}_{c \,\in\, \mathrm{children}(u)} \; \tilde{Q}(c) \;+\; c_{\mathrm{puct}}\, P(c)\, \frac{\sqrt{N(u)+1}}{1+N(c)}.$$

For a visited child, $Q(c) = W(c)/N(c)$; an unvisited child inherits its parent's estimate via first-play urgency. We min-max normalize Q over the sibling set,

$$\widetilde{Q}_u(v) = \frac{Q(v) - \min_{w \in \mathrm{children}(u)} Q(w)}{\max\!\big\{\max_{w \in \mathrm{children}(u)} Q(w) - \min_{w \in \mathrm{children}(u)} Q(w),\; 10^{-2}\big\}},$$

so that exploitation and exploration are compared on a common scale regardless of the PRM's output range.

At each rollout phase, we sample n candidate continuation steps, score them with the PRM, and continue with the highest-scoring candidate. This process repeats until a final answer is produced or the depth cap T is reached. The n candidates generated at the first rollout step are also used to expand the selected leaf, becoming its children in the search tree.

The value backed up from a rollout is the path value of the *complete* trajectory through the leaf, including both the prefix scores from the root to the leaf and the rollout tail. Thus, every completed trajectory is evaluated on the same basis. If the selected leaf is already terminal or depth-capped, it is recorded and backed up directly, without an additional rollout. After all simulations, we return the highest-scored trajectory among those that emit a final answer.

Because the root is unexpanded at simulation 1, selection terminates at the root and the first rollout is exactly the greedy decode. This trajectory is replaced only if a later simulation produces a strictly higher path value. Consequently, under the PRM's own path-value measure, the trajectory returned by MCTS has value no lower than the greedy trajectory.

Algorithm 2 PRM-guided MCTS (PUCT)

Require: prompt x, generator π, PRM V, simulations k, candidates per step n, maximum depth T, exploration constant c_{puct}

$o \leftarrow$ root with an empty prefix
$\mathcal{C} \leftarrow \emptyset$
for $\mathrm{sim} = 1, \ldots, k$ **do**
  $u \leftarrow o$; $\tau \leftarrow [o]$ ▷ Selection
  while u has children and is nonterminal **do**
    $u \leftarrow \arg\max_{v \in \mathrm{children}(u)} \left[\widetilde{Q}(v) + c_{\mathrm{puct}} P(v) \frac{\sqrt{N(u)+1}}{1+N(v)} \right]$ ▷ Descend by PUCT
    Append u to τ
  end while
  $s \leftarrow s(u)$
  $F \leftarrow \emptyset$
  while s is unfinished and has fewer than T steps **do** ▷ Best-of-n rollout
    Sample $a_1, \ldots, a_n \sim \pi(\cdot \mid x, s)$
    Form s_j by extending s with a_j, for $j = 1, \ldots, n$
    $r_j \leftarrow V(x, s_j)$ for $j = 1, \ldots, n$
    if $F = \emptyset$ **then**
      $F \leftarrow \{(a_j, r_j)\}_{j=1}^{n}$
    end if
    $j^* \leftarrow \arg\max_j r_j$; $s \leftarrow s_{j^*}$
  end while
  if $F \neq \emptyset$ **then** ▷ Expand only the selected leaf
    for all $(a_j, r_j) \in F$ **do**
      Create child v_j by extending u's prefix with a_j
      $r(v_j) \leftarrow r_j$; $P(v_j) \leftarrow \exp(\tilde{r}_j) / \sum_i \exp(\tilde{r}_i)$, with $\tilde{r}$ the min–max normalisation of r over F (uniform when all r_i coincide)
    end for
  end if
  $z \leftarrow \bar{V}(s)$; $\mathcal{C} \leftarrow \mathcal{C} \cup \{(s, z)\}$ ▷ Full-path evaluation
  for all $m \in \tau$ **do** ▷ Backup
    $N(m) \leftarrow N(m) + 1$; $W(m) \leftarrow W(m) + z$
    $Q(m) \leftarrow W(m)/N(m)$
  end for
end for
$(s^*, z^*) \leftarrow \arg\max_{(s,z) \in \mathcal{C}:\, \mathrm{Done}(s)} z$
return s^*