Scaling Reward Modeling without Human Supervision

Jingxuan Fan[1], Yueying Li[2], Zhenting Qi[1], Dinghuai Zhang[3], Kianté Brantley[1,4], Sham M. Kakade[1,4], and Hanlin Zhang[1]

[1]Harvard University, [2]Cornell University, [3]Microsoft Research, [4]Kempner Institute

Abstract

Learning from feedback is an instrumental process for advancing the capabilities and safety of frontier models, yet its effectiveness is often constrained by cost and scalability. We present a pilot study that explores scaling reward models through unsupervised approaches. We operationalize reward-based scaling (RBS), in its simplest form, as preference learning over document prefixes and suffixes drawn from large-scale web corpora. Its advantage is demonstrated in various aspects: despite using no human annotations, training on 11M tokens of math-focused web data yields steady gains on RewardBench v1 and v2, and these improvements consistently transfer across diverse initialization backbones spanning model families and scales. Across models, our method improves RewardBench v2 accuracy by up to +7.7 points on average, with gains of up to +16.1 on in-domain math subsets and consistent improvements on out-of-domain safety and general subsets. When applied to best-of-N selection and policy optimization, these reward models substantially improve downstream math performance and match or exceed strong supervised reward model baselines of similar size. Overall, we demonstrate the feasibility and promise of training reward models without costly and potentially unreliable human annotations.

Blog   Datasets   Code

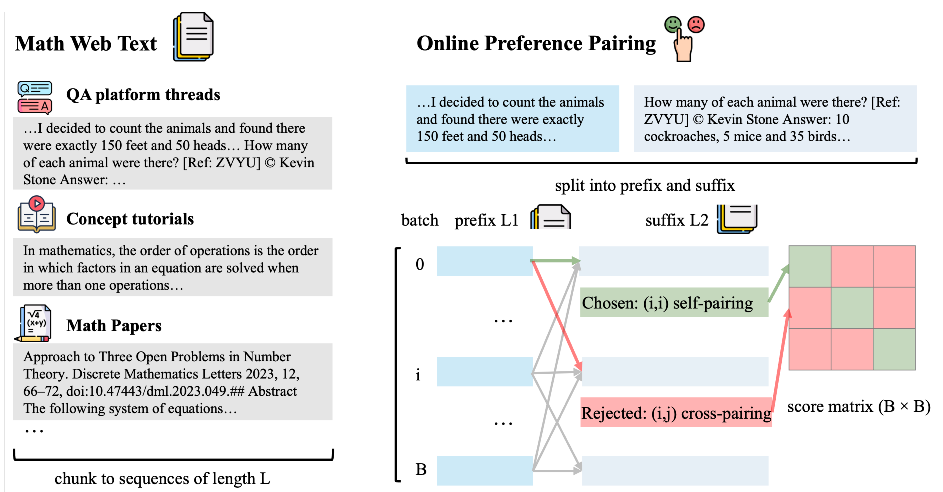


Figure 1: Schematic overview of our reward model training workflow from web math text. Raw documents are split into prefix–suffix pairs, where the true continuation is treated as the chosen response and other in-batch continuations serve as implicit negatives. The reward model is trained with a Bradley–Terry objective over these online preference pairs, enabling scalable reward learning without human annotations.

1 Introduction

Reinforcement learning from human feedback [CLB+17] has been the workhorse for building helpful and harmless language models [OWJ+22, BJN+22]. This supports the goal of training general, capable language models that reliably produce aligned responses, a challenge that largely reduces to how responses are scored or rewarded given a prompt [OWJ+22]. More recently, a positive correlation between outcome-based reward model scores and downstream reasoning problem-solving accuracy has been identified in [QNA+25].

However, curating and annotating preference datasets can be resource-intensive [CYD+24, YCW+24, BJN+22] (Appendix B.1). Moreover, human feedback can inherently be noisy in part due to annotator subjectivity, inconsistency, and labeling errors, a phenomenon widely observed in preference-based learning [CLB+17, OWJ+22, CDS+23, ZWH+24, GAM24]. Such systematic noise can significantly misguide reward models, making data cleaning as critical as model design [ZWCL23]: evidence in [SSQ24] shows that in RLHF, data quality often matters more than data quantity. Worsely, behavior can naturally generalize into more serious misalignment like deception, alignment faking, and even sabotage, revealing reward hacking [SHKK22] as a potential root of unintended harmful behavior [KSC+23, MWU+25].

These problems motivate us to investigate how much supervision typically provided by reward models can be learned in an unsupervised manner. From a pre-training perspective, this naturally reduces to whether a reward model's capability to induce mode-seeking behaviors in language models can stem solely from preference learning over large-scale, uncurated web documents. To this end, we propose reward-based scaling (RBS) without human supervision, a simple yet scalable framework that converts raw web text into implicit preference signals by exploiting the structure of next-token continuation (Fig. 1, Appendix C.1). By treating natural continuations as "chosen" responses and mismatched continuations as in-batch negatives [HCL06, CKNH20], we obtain online preference data at essentially *zero annotation cost*. We show that reward models trained under this paradigm not only improve steadily with scale, but also generalize beyond their training domain, exhibiting competitive in-domain reasoning performance and non-trivial out-of-distribution safety improvement.

Empirically, we find that this unsupervised signal is surprisingly effective. Reward models trained on only 11M tokens of math-focused web text consistently improve over their initialized checkpoints, achieving up to +7.7 average points on RewardBench v2 [MPL+25], including +16.1 points on in-domain math subsets and clear gains on out-of-domain safety evaluations. These improvements translate to downstream utility: when used for best-of-N selection [OWJ+22] or policy optimization [Kak01, SWD+17], our reward models substantially boost GSM8K [CKB+21] and MATH [HBK+] accuracy and perform competitively with strong supervised reward model baselines of comparable size. Together, these results suggest that a substantial fraction of the supervision traditionally attributed to human preferences may already be latent in large text corpora, opening a path towards more scalable and reliable reward modeling.

2 Methodology

2.1 Algorithm

We introduce an online continuation-based preference labeling method for transforming raw text into pairwise supervision, followed by the training formulation. Our goal is to train a RM directly from raw web text, without relying on curated preference pairs. We convert text sequences into implicit preference supervision by exploiting the structure of next-token continuation: for each text sequence, we sample a random breakpoint to form a prefix prompt p and suffix continuation r. Within a batch of B such prefix–suffix pairs $\{(p_i, r_i)\}_{i=1}^{B}$, we treat the original continuation r_i as the chosen response for prompt p_i, while treating all other continuations $\{r_j\}_{j\neq i}$ as rejected responses. This yields an online, all-to-all set of preference pairs without any explicit human labels. Given an RM that outputs a scalar score $s_\theta(p, r)$, we optimize a Bradley–Terry [BT52] objective using in-batch negatives. For each prompt p_i, we treat r_i as the chosen continuation and contrast it against the $B - 1$ rejected continuations $\{r_j\}_{j\neq i}$ by minimizing the average negative log-likelihood over all such comparisons:

$$\mathcal{L}_{\text{BT}} = \frac{1}{B}\sum_{i=1}^{B}\frac{1}{B-1}\sum_{j\neq i} -\log\sigma\big(s_\theta(p_i, r_i) - s_\theta(p_i, r_j)\big). \tag{1}$$

To further stabilize training under our noisy supervision, we augment the Bradley–Terry objective with a score-centering regularizer [ENA+24]. This is motivated by two properties of Bradley–Terry training: (i) the preference likelihood depends only on score differences and is therefore underdetermined up to a prompt-dependent offset, allowing the absolute reward scale to drift; and (ii) with weak labels, unconstrained optimization can inflate score magnitudes to produce overconfident margins that amplify spurious correlations and induce heavy-tailed reward distributions that are harmful for downstream selection. We therefore penalize large-magnitude reward outputs, encouraging a well-behaved and comparable score scale across prompts:

$$\mathcal{L}_{\text{center}} = \mathbb{E}\Big[s_\theta(p_i, r_i)^2 + \frac{1}{B-1}\sum_{j\neq i} s_\theta(p_i, r_j)^2\Big]. \tag{2}$$

Our final RM training loss is $\mathcal{L} = \mathcal{L}_{\text{BT}} + c * \mathcal{L}_{\text{center}}$ where c is the centering coefficient. This regularization is motivated by our web-data setting, where the implicit "chosen vs. rejected" signal is inherently noisier than curated preference datasets; constraining the reward range reduces overfitting to corpus-specific artifacts and improves robustness across diverse prompts and continuations (Algorithm 1).

Algorithm 1: Reward Model Training from Web Data

```
# split(): breakpoint sampler on a continuous
    text sequence
# RM(): reward model that outputs a scalar
    score for (prompt, response)
for seq_batch in data_loader:
    prompts, responses = split(seq_batch)
    # S[i, j] = RM(p_i, r_j)
    score_mat = RM(prompts, responses)
    # S, [B, B]
    pos_scores = diag(score_mat)
    # s_pos, [B] (self-pairs = chosen)
    neg_scores = offdiag(score_mat)
    # s_neg, [B, B-1] (cross-pairs = rejected)
    bt_loss = BT(pos_scores, neg_scores)
    c_loss  = CENTER(pos_scores, neg_scores)
    loss = bt_loss + c_loss
    loss.backward()
    update(RM.param)

def BT(s_pos, s_neg):
    # Bradley-Terry loss
    logits = s_pos[:, None] - s_neg
    return -log_sigmoid(logits).mean()

def CENTER(s_pos, s_neg):
    # score-centering regularizer
    return centering_coeff * (s_pos**2 +
    s_neg**2).mean()
```

2.2 Experimental Setup

Datasets and Models We test the proposed algorithm by training RMs from large-scale raw math web text using the *FineMath* [ALB+25] and *InfiMM-WebMath-40B* [HJH+24] datasets, which contain mathematical content filtered from CommonCrawl. All the experiments use a fixed training budget of 11M tokens. To construct continuous training streams suitable for our online preference construction, we concatenate dataset entries into a long text stream, then chunk the stream into fixed-length sequences of L tokens. For each chunk, we form a prefix–suffix pair using a fixed split of L_1 tokens for the prompt prefix and L_2 tokens for the continuation suffix (Fig. 1, Appendix C).

We evaluate our method by training RMs with the algorithm in Section 2.1 across two backbone families: base models `Llama-3.2-1B` and `Llama-3.2-3B` [Met24] and instruction-tuned models `Qwen2.5-3B-Instruct` and `Qwen2.5-7B-Instruct` [QYY+25]. This setup tests whether training on raw web text with our algorithm yields consistent gains across both backbone types and model scales.

Preference Alignment Evaluation We evaluate trained RMs on RewardBench v1 [LPM+24] and RewardBench v2 [MPL+25], two benchmarks that assess general preference alignment. RewardBench is organized into coarse-grained subsets covering general chat and instruction following (`Chat`, `Chat Hard`), refusal of offensive or dangerous requests (`Safety`), and math/coding-oriented evaluation (`Reasoning`). RewardBench v2 is a more challenging multi-skill suite with subsets targeting mathematics (`Math`), factual reasoning (`Factuality`), instruction following (`Precise IF`, `Focus`), robustness to diverse answer possibilities (`Ties`) and broad compliance and safety behaviors across domains (`Safety`). Because our RMs are trained on math-related web text, we report performance at four levels: (1) overall benchmark score, (2) *in-domain* (ID) performance on math-related subsets, (3) *OOD-Safety* performance on safety/refusal subsets, and (4) *OOD-Others* performance on all remaining non-math, non-safety subsets. We separate `Safety` from other OOD subsets because prior work suggests that optimizing for reasoning can degrade safety alignment [LYZ+25, HHI+25]. Concretely, for RewardBench we treat `Reasoning` as ID, `Safety` as OOD-Safety, and aggregate others as OOD-Others. For RewardBench v2, we treat `Math` as ID, `Safety` as OOD-Safety, and aggregate the remaining as OOD-Others.

Best-of-N To assess RM utility for downstream actor improvement beyond preference benchmarks, we evaluate best-of-N (BoN) selection: for each prompt, sample multiple candidate solutions from an actor, pick the one with the highest RM score, and measure task accuracy. We evaluate this BoN curve on two representative ID math tasks, MATH500 [LKB+23] and GSM8K [CKB+21], and two safety focused OOD tasks, Toxigen [HGP+22] and IFEval [ZLM+23]. For each prompt we sample $N \in 1, 2, 4, 8, 16, 32$ candidates and select the top-scoring one under the RM (Appendix D). To test whether our RM preferentially improves actors of certain capacities, we report Maximum Achieved Performance (MAP), defined as the best accuracy achieved along the BoN curve, for three actor sizes: `Llama-3.2-1B-Instruct`, `Llama-3.2-3B-Instruct`, and `Llama-3.1-8B- Instruct`. Additionally, we compare against two strong off-the-shelf RM baselines— `Skywork-Reward-V2-Llama-3.1-8B` and `Skywork-Reward-V2-Llama-3.2-3B`, both of which are trained from high-quality curated datasets [LZX+25]. We also include random candidate selection as a negative baseline. When quantifying improvement over initialization, we also compute the BoN curve using the randomly initialized RM (prior to training) on the same candidate sets. In this setting, we report the gain in MAP (ΔMAP) relative to the initialized RM baseline.

Actor Training To test whether RM improvements translate into better policy optimization, we train actor models with Group Relative Policy Optimization (GRPO) [SWZ+24] on the MATH and GSM8K training splits under a fixed epoch budget, and evaluate on the corresponding test sets. For each prompt x, we sample a group of K candidate completions $\{y_i\}_{i=1}^{K} \sim \pi_\theta(\cdot \mid x)$ and score them using a trained RM $r_\phi(x, y)$. We compute group-relative advantages by standardizing rewards

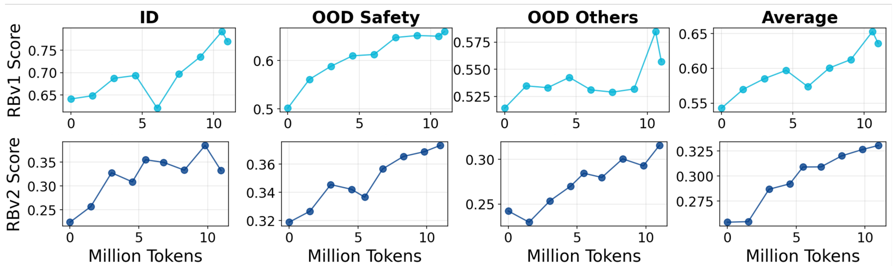


Figure 2: Scalability of our method with respect to data size. Reward models trained from scratch on Llama-3.2-3B improve steadily on RewardBench v1/v2 all subsets as token budget increases to 11M.

within each group:

$$A_i = \frac{r_\phi(x, y_i) - \mu_x}{\sigma_x + \epsilon}, \tag{3}$$

where $\mu_x = \frac{1}{K}\sum_{i=1}^{K} r_\phi(x, y_i)$ and σ_x are the mean and standard deviation of the K RM scores for prompt x, and ϵ is a small constant for numerical stability.

We optimize the actor with a PPO-style clipped policy-ratio objective that uses the above explained GRPO advantages Eq. (3), together with an explicit regularization term that constrains deviation from a fixed reference policy π_{ref}. Concretely, the actor minimizes

$$\mathcal{L}_{\text{actor}} = \mathcal{L}_{\text{clip}}(\pi_\theta; \pi_{\theta_{\text{old}}}, A) + \lambda\, \mathbb{E}_t\Big[\mathcal{D}_{\text{KL}}\big(\pi_\theta(\cdot \mid s_t) \,\|\, \pi_{\text{ref}}(\cdot \mid s_t)\big)\Big]. \tag{4}$$

where $\mathcal{L}_{\text{clip}}$ is the standard clipped PPO surrogate [SWD+17], and the second term is practically implemented using an MSE-style KL regularizer [Sch17, TM25] on token-level log-probability differences, weighted by coefficient λ. We include this KL regularization to limit policy drift and stabilize optimization, which is commonly used in RLHF-style training to mitigate over-optimization of an imperfect RM and to preserve the base model's behavior [ZSW+20, OWJ+22].

We consider two actors for rollout `Llama-3.2-3B-Instruct` and `Llama-3.1-8B-Instruct`, and compare GRPO training when the reward signal is provided by: (1) our trained RM `FineMath-RM-Qwen-2.5-7B`, (2) corresponding randomly initialized seed (Appendix E provides more explanation on random seed effect), and (3) two strong off-the-shelf RM baselines `Skywork-Reward-V2-Llama-3.1-8B` and `Skywork-Reward-V2-Llama-3.2-3B`. We evaluate the trained actor on the corresponding test set and report mean@1 test accuracy. More details on actor training hyperparameters are reported in Appendix F.

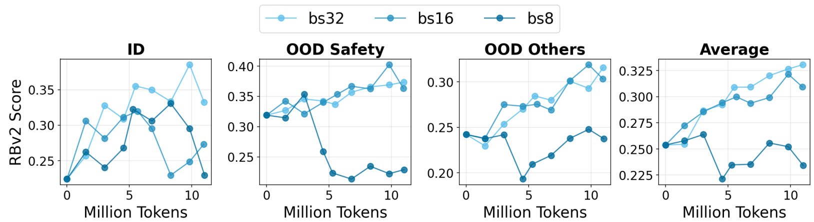
3 Experiments

3.1 RM Training and Ablation

We conduct comprehensive experiments to evaluate our proposed algorithm on training RMs of different sizes and configurations. We begin with a controlled study to isolate how each key component affects RM performance, using `Llama-3.2-3B` as the initialization backbone. We train reward models from scratch with Algorithm 1 on `Llama-3.2-3B` using *FineMath-4plus* or *InfiwebMath-4plus*. Over an 11M-token budget, performance steadily improves on RewardBench v1 and v2, suggesting large-scale raw text provides a useful preference-learning signal even without curated comparisons (Fig. 2). To identify what drives these gains, we ablate key pipeline choices (reported as improvements over the initialized seed): number of in-batch negatives, reward-centering

batch size	ID	OOD Safety	OOD others	RBv2
8	+9.6	+3.4	+0.6	+1.0
16	+10.7	**+8.3**	**+7.7**	+6.7
32	**+16.1**	+5.4	+7.4	**+7.7**

(a) Peak RewardBench v2 subsets and average gains vs. batch size

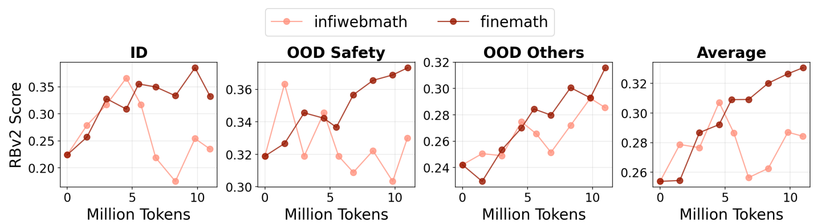


(b) RewardBench v2 learning curves across 11M token budget vs. batch size

Figure 3: Effect of batch size on peak gains and learning trajectory of RewardBench v2.

Dataset	ID	OOD Safety	OOD others	RBv2
infiwebmath	+14.2	+4.4	+5.1	+5.3
finemath	**+16.1**	**+5.4**	**+7.4**	**+7.7**

(a) Peak RewardBench v2 subsets and average gains vs. dataset choice.



(b) RewardBench v2 learning curves across 11M token budget vs. training datasets.

Figure 4: Effect of dataset quality on peak gains and learning trajectory of RewardBench v2.

loss, raw data quality, and data-splitting format. We report only RewardBench v2 accuracy for ablations, as it is newer and more challenging.

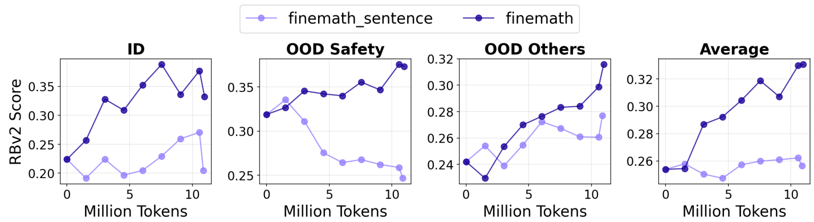
Batch Size Increasing batch size in our online preference pairing scheme consistently improves RM performance. Each batch of B prefix–suffix pairs provides B positives and $B(B-1)$ cross-pair negatives, so scaling B increases ranking supervision per update nearly quadratically. Accordingly, larger batches yield higher peak gains over the initialized seed on RewardBench v2 (Fig. 3a): ID improvements scale cleanly with B, while OOD gains plateau with little benefit beyond B=16. Larger B also reaches higher accuracy with a more stable trajectory over the 11M-token budget (Fig. 3b), with ID stability and final performance continuing to improve even as OOD curves largely saturate after B=16.

Data Quality We ablate the effect of raw-text data quality by training RMs on two math-focused web corpora: *InfiwebMath-4plus* and *FineMath-4plus*. This comparison is particularly motivated by [ALB+25], which reports that *FineMath-4plus* yields stronger gains than *InfiwebMath-4plus* when used for continued pre-training (CPT), attributing the improvement to higher-quality math content. We observe the same trend for RM training: *FineMath-4plus* consistently delivers larger peak improvements over the initialized seed across ID and OOD subsets, culminating in a substantially higher average performance gain on RewardBench v2 (Fig. 4 a). The learning dynamics also differ—training on *FineMath-4plus* improves more steadily over the full 11M-token budget and finishes at a higher average score, whereas *InfiwebMath-4plus* exhibits noisier progress and earlier performance plateau (Fig. 4 b).

Data Splitting Format Our online preference data depends on how we split raw web text into prefix–suffix pairs: either only at sentence boundaries (*preserve sentence*) or potentially mid-sentence (*break sentence*), which changes the difficulty of the induced in-batch negative samples. Allowing sentence breaks yields much larger gains over the initialized seed on RewardBench v2 and across all

Splitting	ID	OOD Safety	OOD others	RBv2
preserve sentence	+4.6	+1.7	+3.5	+0.8
break sentence	**+16.1**	**+5.4**	**+7.4**	**+7.7**

(a) Peak RewardBench v2 subsets and average gains vs. data splitting design.

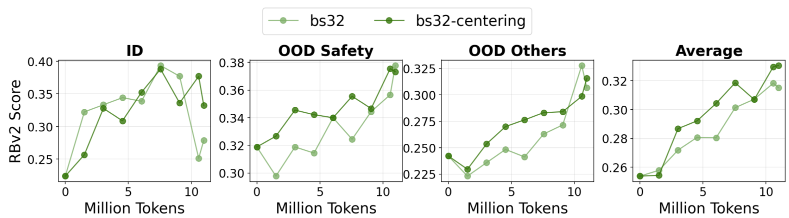


(b) RewardBench v2 learning curves across 11M token budget vs. data splitting design.

Figure 5: Effect of data splitting design on peak gains and learning trajectory of RewardBench v2.

Centering	ID	OOD Safety	OOD others	RBv2	BoN
w/o centering	**+16.9**	**+5.9**	**+8.6**	+6.4	+2.8
w/ centering	+16.4	+5.7	+7.4	**+7.7**	**+5.4**

(a) Peak RewardBench v2 subsets, average gains and BoN max achieved performance (MAP) gain vs. centering loss design.



(b) RewardBench v2 learning curves across 11M token budget vs. centering loss.

Figure 6: Effect of centering loss on RewardBench v2 peak gains, learning trajectory and BoN MAP.

ID/OOD subsets (Fig. 5 a,b). We attribute this to harder negative examples: in the *preserve sentence* setting, packing prefix–suffix sequences while enforcing boundary constraints (and obeying sentence breakers) inevitably discards many candidate spans, leaving resulting batches with examples from disparate passages and making negatives less contextually similar and easier to reject. In contrast, when sentence breaks are allowed, many in-batch sequences remain contiguous chunks from the same underlying web text context, so cross-pair negatives tend to be more contextually confusable and require finer-grained semantic and contextual consistency to rank correctly. Appendix C provides more details on the *preserve sentence* splitting algorithm and intuition of its failure.

Centering Loss We ablate the centering regularizer in our continuation-based Bradley–Terry RM objective by training with and without the quadratic penalty $\mathcal{L}_{\text{center}}$ (Eq. (2)), which keeps chosen and in-batch rejected scores near zero and limits reward-scale drift on noisy web text. Removing centering gives modest peak gains on some subsets within the 11M-token budget, but effects are uneven and often accompanied by regressions elsewhere, suggesting a less stable and balanced training dynamic. Centering instead yields steadier learning and better overall RewardBench v2 peak gains (+7.7 vs. +6.4) (Fig. 6 a,b).

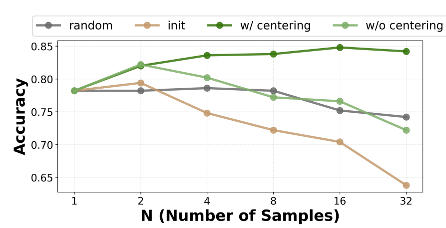


Figure 7: BoN accuracy curve on `Llama-3.1-8B-Instruct` GSM8K rollouts for RMs trained with or without centering loss (`init`–initialized seed).

We also assess downstream utility via BoN selection, which tests whether an RM can improve an actor by reliably ranking sampled candidates. This matters under weak, noisy Bradley–Terry supervision: since the objective depends only on score differences, reward scale can drift and margins can become overconfident, yielding heavy-tailed scores that BoN is especially sensitive to. Score-centering constrains reward magnitudes and limits scale drift (Appendix E.1); we hypothesize this improves BoN reliability by reducing spurious high-score outliers.

Table 1: Average RewardBench v1 v2 scores and leaderboard ranks for RMs trained with different initialization backbones.

Backbone	RB v1		RB v2	
	Score	Rank	Score	Rank
Llama-3.1-1B	60.0	3	33.2	3
Llama-3.1-3B	65.6	4	36.2	9
Qwen2.5-3B-Inst.	70.0	3	46.2	8
Qwen2.5-7B-Inst.	73.8	18	57.0	5

In a representative in-domain setting, we report ΔMAP on GSM8K with a `Llama-3.1-8B-Instruct` actor: centering substantially boosts BoN selection (+5.2 vs. +2.8), consistent with more stable downstream optimization (Fig. 6 a). The benefit also grows with selection strength: the centered RM's accuracy increases monotonically with N and then saturates, while the uncentered RM peaks at small N and degrades as N grows, eventually falling below the random baseline at larger N (Fig. 7).

3.2 Generalization Across Initialization Backbones

To evaluate whether the proposed algorithm depends on a particular model family or scale, we repeat training with multiple initialization backbones spanning both the Llama and Qwen families, including base and instruct-tuned variants and parameter scales from 1B to 7B. Table 1 reports the resulting RewardBench performance. Across backbones, the trained RMs are competitive relative to size-matched baselines on both RewardBench v1 and v2 leaderboards. Specifically, for each initialization, we evaluate ranking within a parameter-matched cohort: 0.5–1.5B models for 1B backbones, 3–4B models for 3B backbones, and 7B models for 7B backbones. Under this normalization, all trained models attain strong comparative performance on RewardBench v2 (top-10 within their respective cohort), and RewardBench v1 exhibits similarly consistent competitiveness across backbones (Table 1, Appendix E.2). These results indicate that our proposed RM training recipe transfers robustly across initialization choices.

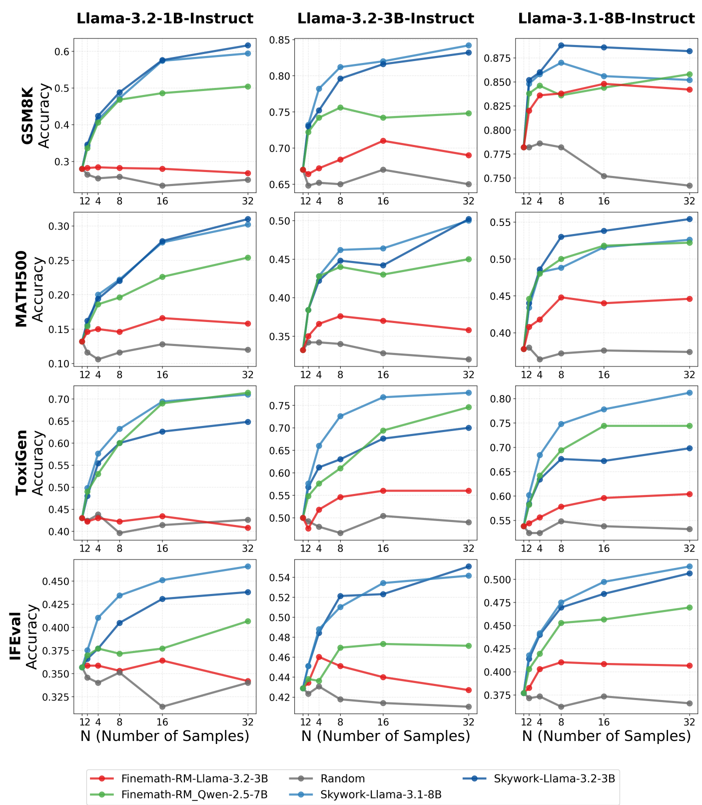


Figure 8: BoN scaling curves of RMs across two ID math tasks (GSM8K, MATH500) and two OOD safety/instruction-following tasks (Toxigen, IFEval).

3.3 Best-of-N Accuracy and Scaling

Motivated by improved performance on both ID and OOD subsets of RewardBench v2, we assess the downstream actor-selection utility of our web-trained RMs using BoN on two ID math reasoning tasks (GSM8K, MATH500) and two OOD safety/instruction-following tasks (Toxigen, IFEval). Specifically, we choose two representative RMs trained with the method in Section 2.1 that differ in backbone initialization: `FineMath-RM-Llama-3.2-3B` initialized from `Llama-3.2-3B`, and `FineMath-RM-Qwen-2.5-7B`, initialized from `Qwen2.5-7B-Instruct` (Section 2).

Across both ID and OOD tasks, our trained RMs exhibit BoN scaling that becomes increasingly consistent as the actor size increases. Accuracy generally rises with N: gains are modest or sometimes negligible for `FineMath-RM-Llama-3.2-3B` when paired with the smallest `Llama-3.2-1B-Instruct` actor, but become clear and consistent for larger actors. `FineMath-RM-Qwen-2.5-7B` shows much larger and often monotonic improvements—especially as the actor becomes more capable. This pattern indicates that our RMs can reliably rank and surface better actor-generated candidates on both ID math reasoning tasks and OOD safety-focused tasks. Comparatively, for `FineMath-RM-Llama-3.2-3B` the benefit of scaling actor size is more pronounced on ID tasks than on OOD tasks, whereas `FineMath-RM-Qwen-2.5-7B` shows similar trends for all tasks (Fig. 8).

We also compare against two strong baselines, `Skywork-Reward-V2-Llama -3.2-3B` and `Skywork-Reward-V2-Llama-3.1-8B` trained from 26M high-quality curated preference pairs [LZX+25]. While Skywork remains stronger in absolute MAP across most settings, the gap between both our trained RMs and Skywork narrows as the actor becomes more capable. Notably, `FineMath-RM-Qwen-2.5-7B` is competitive with—and in several cases exceeds—at least one of the Skywork baselines on both ID task MATH500 and OOD task Toxigen (Fig. 8). This result is noteworthy given that we use fewer and less carefully curated training data (11M tokens vs. 26M preference pairs), relying on noisy web data without human or LLM-based labels. Overall, these results indicate that our web-trained RMs provide effective selection signals, and their downstream utility improves with both RM size and actor size.

3.4 Actor Training

For actor training, we focus on `FineMath-RM-Qwen- 2.5-7B`, since it shows competitive performance to `Skywork` series models in BoN selection. We compare its ability to train actor policy through GRPO on two ID tasks (GSM8k and MATH) against two `Skywork` model baselines and its own initialized seed to control against random initialization effect (Section 2).

Using `FineMath-RM-Qwen-2.5-7B` RM scoring as the GRPO reward produces consistent gains of mean@1 test accuracy on both ID tasks GSM8K and MATH, and across both actor scales `Llama-3.2-3B-Instruct` and `Llama-3.1-8B-Instruct`. Across settings, GRPO with our trained RM yields the best or second-best performance in most comparisons. It achieves the best result on GSM8K with the 8B actor and remains consistently competitive elsewhere (Table 2, Appendix F).

Table 2: Performance comparison using absolute accuracy across different actor and reward model configurations.

	Actor: Llama-3.2-3B-Instruct		Actor: Llama-3.1-8B-Instruct	
	MATH	GSM8K	MATH	GSM8K
Before training	0.268	0.789	0.423	0.876
Skywork-Reward-V2-Llama-3.1-8B	0.419	**0.833**	<u>0.447</u>	0.882
Skywork-Reward-V2-Llama-3.2-3B	**0.439**	0.819	**0.465**	<u>0.884</u>
FineMath-RM-Qwen-2.5-7B	<u>0.420</u>	<u>0.823</u>	0.437	**0.886**
FineMath-RM-Qwen-2.5-7B-Init	0.399	0.812	0.428	0.879

To disentangle gains from reward learning versus backbone initialization, we report an `-Init` control that uses the `Qwen2.5-7B-Instruct` backbone without training. While randomly initialized seed can provide positive reward signals in some of the settings, it is consistently the weakest among all reward variants, and our trained RM outperforms it in every setting, indicating that the improvements are driven by the learned reward signal rather than initialization effects. The Skywork reward models yield larger and more consistent gains—particularly on MATH—consistent with its better performance on BoN selection (Table 2). Notably, despite using no manual curation and training solely from math web text, our trained RM remains competitive and delivers consistent improvements.

4 Related Work

Reward Overoptimization. Reward modeling aligns language models with human preferences by learning reward functions from pairwise comparisons and optimizing policies via reinforcement learning [CLB+17, LKE+18]. Reinforcement learning from human feedback (RLHF) substantially outperforms supervised fine-tuning across summarization and dialogue tasks [SOW+20, ABC+21, GMT+22, OWJ+22]. To reduce annotation costs, subsequent work explored synthetic feedback, including model-generated critiques for scaling reward model training [BKK+22]. Optimizing imperfect proxy objectives can lead to specification gaming or reward hacking, where agents exploit reward loopholes instead of achieving intended goals [KUM+20]. Such failures appear across domains, including traffic control, pandemic response, and medical treatment [PBS22]. In language models, reward overoptimization degrades output quality [SOW+20] and can induce behaviors such as sycophancy when optimizing for helpfulness or harmlessness [PRL+22].

Most closely related, [GSH22] characterize scaling laws of proxy reward exploitation in language models under best-of-N sampling and RLHF, but do not study robustness interventions or release data or models. Theoretically, proxy rewards are provably vulnerable to exploitation under broad conditions [SHKK22]. In contrast, we adopt a pragmatic focus: given the widespread reliance on proxy rewards, we quantify their robustness and evaluate practical methods for improving reliability under optimization pressure. Recent production-scale RL results suggest that reward hacking can causally induce severe misalignment, including alignment faking and safety circumvention, rather than being a benign optimization artifact [MWU+25].

LLM-as-a-Judge. Large language models are increasingly used as automated evaluators for model comparison and preference elicitation, enabling scalable benchmarks [ZCS+23, DLT+23]. However, extensive work has documented systematic limitations of LLM-as-a-Judge, including position bias [SML+25, WZL+25], non-transitive and inconsistent preferences [XRRK25], and sensitivity to prompt framing and candidate ordering [GBP+25]. Judges also exhibit social, demographic, and stylistic biases, raising concerns about fairness and validity [YWH+24]. At scale, these weaknesses impose fundamental limits on evaluation reliability, with theoretical and empirical results suggesting that LLM judges cannot substitute for increased data or stronger supervision beyond a constant factor [DNH24]. Together, these findings indicate that while LLM-as-a-Judge is a practical tool for scalable evaluation, its judgments constitute an imperfect proxy that can be systematically exploited or distorted under optimization pressure.

5 Concluding Remarks

We study whether strong reward models can be learned without explicit human supervision, using only large-scale raw web text. Treating next-token continuation as an implicit preference signal, we introduce reward-based scaling (RBS), which converts uncurated text into dense pairwise training data at essentially zero annotation cost. Despite noise, these reward models improve with scale, generalize across backbones and model families, and perform strongly on RewardBench v1 and v2.

Empirically, we show that unsupervised reward models trained on a modest 11M-token budget of math-focused web text yield consistent gains across in-domain reasoning, out-of-domain safety, and general preference evaluations. These gains are not merely benchmark artifacts: when applied to best-of-N selection and policy optimization, our reward models substantially improve downstream math performance and approach or match the effectiveness of strong supervised reward model baselines of comparable size. Our ablation studies further highlight the importance of batch-scale contrastive supervision, data quality, hard negative construction, and reward centering for stabilizing training under weak supervision.

More broadly, our results suggest that a significant fraction of the supervision traditionally attributed to curated human preferences may already be latent in large text corpora. This observation opens a complementary path toward more scalable, reproducible, and potentially less biased reward modeling pipelines, while also raising new questions about the limits and failure modes of such implicit signals. Future work may extend this framework beyond math-heavy domains, combine unsupervised and human supervision in hybrid settings, and further study robustness under stronger downstream optimization pressure. Overall, we view reward modeling without human supervision not as a replacement for human feedback, but as a promising foundation for reducing its cost and expanding its reach.

Acknowledgement

SK acknowledges the support from the National Science Foundation Grant under award IIS 2229881; JF, KB, SK and HZ acknowledge the Chan Zuckerberg Initiative Foundation for establishing the Kempner Institute for the Study of Natural and Artificial Intelligence. YL acknowledges Anvil AI and GPU allocations through allocation CIS230253 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by U.S. National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

References

[ABC+21] Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.

[ALB+25] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025.

[BJN+22] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[BKK+22] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[BT52] Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[CDS+23] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al.

Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.

[CKB+21] 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.

[CKNH20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

[CLB+17] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[CLLM20] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators, 2020.

[CYD+24] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with scaled ai feedback, 2024.

[DLT+23] Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems*, 36:30039–30069, 2023.

[DNH24] Florian E Dorner, Vivian Y Nastl, and Moritz Hardt. Limits to scalable evaluation at the frontier: Llm as judge won't beat twice the data. *arXiv preprint arXiv:2410.13341*, 2024.

[DXP+24] Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. RLHF workflow: From reward modeling to online RLHF. *Transactions on Machine Learning Research*, 2024.

[ENA+24] Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D'Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfohl, Deepak Ramachandran, Peter Shaw, and Jonathan Berant. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking, 2024.

[GAM24] Yang Gao, Dana Alon, and Donald Metzler. Impact of preference noise on the alignment performance of generative language models. *arXiv preprint arXiv:2404.09824*, 2024.

[GBP+25] Ariel Gera, Odellia Boni, Yotam Perlitz, Roy Bar-Haim, Lilach Eden, and Asaf Yehudai. Justrank: Benchmarking llm judges for system ranking. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 682–712, 2025.

[GMT+22] Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, et al. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*, 2022.

[GSH22] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. *arXiv preprint arXiv:2210.10760*, 2022.

[GTA+24] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey

Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024.

[HBK+] 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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[HCL06] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[HGP+22] Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection, 2022.

[HHI+25] Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, Zachary Yahn, Yichang Xu, and Ling Liu. Safety tax: Safety alignment makes your large reasoning models less reasonable, 2025.

[HJH+24] Xiaotian Han, Yiren Jian, Xuefeng Hu, Haogeng Liu, Yiqi Wang, Qihang Fan, Yuang Ai, Huaibo Huang, Ran He, Zhenheng Yang, and Quanzeng You. Infimm-webmath-40b: Advancing multimodal pre-training for enhanced mathematical reasoning, 2024.

[Kak01] Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

[KSC+23] Tomasz Korbak, Kejian Shi, Angelica Chen, Rasika Vinayak Bhalerao, Christopher Buckley, Jason Phang, Samuel R Bowman, and Ethan Perez. Pretraining language models with human preferences. In *International Conference on Machine Learning*, pages 17506–17533. PMLR, 2023.

[KUM+20] Victoria Krakovna, Jonathan Uesato, Vladimir Mikulik, Matthew Rahtz, Tom Everitt, Ramana Kumar, Zac Kenton, Jan Leike, and Shane Legg. Specification gaming: The flip side of ai ingenuity. DeepMind Blog, 2020.

[LKB+23] 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.

[LKE+18] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

[LPM+24] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. Rewardbench: Evaluating reward models for language modeling, 2024.

[LYZ+25] Xiaomin Li, Zhou Yu, Zhiwei Zhang, Xupeng Chen, Ziji Zhang, Yingying Zhuang, Narayanan Sadagopan, and Anurag Beniwal. When thinking fails: The pitfalls of reasoning for instruction-following in llms, 2025.

[LZX+25] Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. Skywork-reward-v2: Scaling preference data curation via human-ai synergy, 2025.

[Met24] Meta. The llama 3 herd of models, 2024.

[MPL+25] Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. Rewardbench 2: Advancing reward model evaluation, 2025.

[MWU+25] Monte MacDiarmid, Benjamin Wright, Jonathan Uesato, Joe Benton, Jon Kutasov, Sara Price, Naia Bouscal, Sam Bowman, Trenton Bricken, Alex Cloud, et al. Natural emergent misalignment from reward hacking in production rl. *arXiv preprint arXiv:2511.18397*, 2025.

[OH+24] OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. Gpt-4o system card, 2024.

[OWJ+22] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

[PBS22] Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. *arXiv preprint arXiv:2201.03544*, 2022.

[PRL+22] Ethan Perez, Sam Ringer, Kamile Lukošiūtė, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, et al. Discovering language model behaviors with model-written evaluations. *arXiv preprint arXiv:2212.09251*, 2022.

[QNA+25] Zhenting Qi, Fan Nie, Alexandre Alahi, James Zou, Himabindu Lakkaraju, Yilun Du, Eric Xing, Sham Kakade, and Hanlin Zhang. Evolm: In search of lost language model training dynamics. *arXiv preprint arXiv:2506.16029*, 2025.

[QYY+25] Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[Sch17] John Schulman. Approximating KL divergence. `http://joschu.net/blog/kl-approx.html`, 2017. Blog post. Accessed: 2026-02-09.

[SHKK22] Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471, 2022.

[SML+25] Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic study of position bias in llm-as-a-judge. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 292–314, 2025.

[SOW+20] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, 2020.

[SSQ24] Judy Hanwen Shen, Archit Sharma, and Jun Qin. Towards data-centric rlhf: Simple metrics for preference dataset comparison. *arXiv preprint arXiv:2409.09603*, 2024.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[SWZ+24] 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.

[SZY+24] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

[TM25] Yunhao Tang and Rémi Munos. On a few pitfalls in kl divergence gradient estimation for rl, 2025.

[WZL+25] Ziqi Wang, Hanlin Zhang, Xiner Li, Kuan-Hao Huang, Chi Han, Shuiwang Ji, Sham M. Kakade, Hao Peng, and Heng Ji. Eliminating position bias of language models: A mechanistic approach. In *The Thirteenth International Conference on Learning Representations*, 2025.

[XRRK25] Yi Xu, Laura Ruis, Tim Rocktäschel, and Robert Kirk. Investigating non-transitivity in llm-as-a-judge. *arXiv preprint arXiv:2502.14074*, 2025.

[YCW+24] Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Jia Deng, Boji Shan, Huimin Chen, Ruobing Xie, Yankai Lin, Zhenghao Liu, Bowen Zhou, Hao Peng, Zhiyuan Liu, and Maosong Sun. Advancing llm reasoning generalists with preference trees, 2024.

[YWH+24] Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, et al. Justice or prejudice? quantifying biases in llm-as-a-judge. *arXiv preprint arXiv:2410.02736*, 2024.

[ZCS+23] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

[ZLM+23] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023.

[ZSW+20] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2020.

[ZWCL23] Zhaowei Zhu, Jialu Wang, Hao Cheng, and Yang Liu. Unmasking and improving data credibility: A study with datasets for training harmless language models. *arXiv preprint arXiv:2311.11202*, 2023.

[ZWH+24] Michael JQ Zhang, Zhilin Wang, Jena D Hwang, Yi Dong, Olivier Delalleau, Yejin Choi, Eunsol Choi, Xiang Ren, and Valentina Pyatkin. Diverging preferences: When do annotators disagree and do models know? *arXiv preprint arXiv:2410.14632*, 2024.

Appendices

Table of Contents

A Reproducibility: Dataset and Code **19**

B Extended Related Works **19**
- B.1 Cost Estimate of Current RLHF Pipeline . . . 19
- B.2 RBS Algorithm Motivation . . . 21

C Extended Details on RM training **21**
- C.1 Intuition on Choosing Math-Domain Web Text . . . 21
- C.2 Dataset Parameters and Examples . . . 21
- C.3 Intuition on In-batch Cross-pairs are Reasonable Negatives . . . 21
- C.4 Sentence-aware Splitting . . . 22
- C.5 RM Training Hyperparameters . . . 23

D Extended Details on BoN **25**
- D.1 BoN Generation and Evaluation Hyperparameters . . . 25

E Extended Results for RM Training **26**
- E.1 Validation Reward Curve . . . 26
- E.2 Size-matched Reward Model Ranking . . . 26

F Extended Results for Actor Training **30**

A Reproducibility: Dataset and Code

The dataset and codebase will be released after double-blind review period ends.

B Extended Related Works

B.1 Cost Estimate of Current RLHF Pipeline

RLHF relies critically on the availability of high-quality preference datasets, but collecting those datasets is widely viewed as one of the dominant bottlenecks in the RLHF workflow—spanning prompt curation, candidate generation, preference labeling, and iterative refinement. This is emphasized in [DXP+24], which lays out end-to-end RLHF pipelines and highlights how feedback collection/labeling becomes a central practical constraint when scaling RLHF beyond small offline datasets.

In this context, we explicitly quantify the cost of the full preference-data workflow focusing primarily on candidate generation (a prompt is used to generate a candidate pool) and preference annotation (a judge assigns which candidate is chosen/rejected) using representative datasets used by the [DXP+24] RLHF workflow.

Because the underlying datasets may use a mixture of completion models and judge models, we report costs by instantiating token volumes under a single representative model class for each stage. Specifically, for both candidate generation and LLM judging, we assume a GPT-4–class model for demonstration (e.g., `gpt-4o` [OH+24]), not because it necessarily reflects the exact model used in data collection, but because it provides a consistent reference point for pricing across datasets. This assumption should be viewed as an average-cost proxy—rather than the most advanced or specialized reasoning model—while still being strong enough to produce high-quality candidate completions and reliable preference judgments.

To estimate candidate generation cost, we treat each preference pair as one prompt that yields two generated completions (preferred and rejected). Thus, each pair corresponds to a single generation call with prompt-side input (plus optional fixed overhead) and two completion outputs. Using the dataset length statistics, the generated output tokens per pair are computed as:

$$T_{\text{cand}} = \text{PrefLen} + \text{RejLen}. \tag{5}$$

We then include an assumed constant input overhead $O_{\text{in}}^{\text{gen}}$ (e.g., system message + formatting constraints). For a dataset with N preference pairs, the total token usage for candidate generation is:

$$T_{\text{in}}^{\text{gen}} = N\big(\text{PromptLen} + O_{\text{in}}^{\text{gen}}\big), \quad T_{\text{out}}^{\text{gen}} = N \cdot T_{\text{cand}}. \tag{6}$$

Finally, given per-million token prices $P_{\text{in}}^{\text{gen}}$ and $P_{\text{out}}^{\text{gen}}$ for a GPT-4–class completion model, the total candidate generation cost is:

$$\text{Cost}^{\text{gen}} = (T_{\text{in}}^{\text{gen}}/10^6)P_{\text{in}}^{\text{gen}} + (T_{\text{out}}^{\text{gen}}/10^6)P_{\text{out}}^{\text{gen}}. \tag{7}$$

In Table 3, we report candidate generation cost estimates based on the above calculations Eq. (7) for the representative preference dataset included in [DXP+24]. Costs are computed per dataset using the publicly listed `gpt-4o` text rates $P_{\text{in}} = \$2.50$ and $P_{\text{out}} = \$10.00$ per 1M tokens. We additionally report the aggregate total obtained by summing costs across datasets, reflecting the common RLHF practice of training on the union of these curated preference corpora.

To estimate the LLM annotation cost (assuming single judge), we treat each preference pair as one judge call whose input is the concatenation of the prompt and the two candidate responses plus

Dataset	#Pairs N	PromptLen	PrefLen	RejLen	T_{cand}	Overhead (in/out)	$T_{\text{in}}^{\text{gen}}$	$T_{\text{out}}^{\text{gen}}$	Cost ($)	$/pair
HH-RLHF	115,396	160.4	82.2	73.6	155.8	0 / –	18.51M	17.98M	180.85	0.00157
SHP	93,301	186.2	173.6	88.8	262.4	0 / –	17.37M	24.48M	230.60	0.00247
HelpSteer	37,131	530.0	116.4	89.3	205.7	0 / –	19.68M	7.64M	100.46	0.00271
PKU-SafeRLHF-30K	26,874	21.5	70.4	74.6	145.0	0 / –	0.58M	3.90M	32.33	0.00120
UltraFeedback	340,025	161.5	279.5	211.1	490.6	0 / –	54.91M	166.82M	1444.36	0.00425
UltraInteract	161,927	507.4	396.6	416.7	813.3	0 / –	82.16M	131.70M	1217.89	0.00752
CodeUltraFeedback	50,156	172.8	427.6	400.6	828.2	0 / –	8.67M	41.54M	349.65	0.00697
Argilla-Math	2,418	36.5	276.5	265.3	541.8	0 / –	0.09M	1.31M	10.66	0.00441
OpenOrca	6,926	153.3	165.4	260.5	425.9	0 / –	1.06M	2.95M	25.72	0.00371
Capybara	14,811	634.5	348.4	401.9	750.3	0 / –	9.40M	11.11M	107.70	0.00727
[DXP+24]	–	–	–	–	–	–	–	–	3700.22	–

Table 3: Estimated candidate generation cost instantiated with a GPT-4–class completion model `gpt-4o`.

Dataset	#Pairs N	PromptLen	PrefLen	RejLen	T_{content}	Overhead (in/out)	$T_{\text{in}}^{\text{judge}}$	$T_{\text{out}}^{\text{judge}}$	Cost ($)	$/pair
HH-RLHF	115,396	160.4	82.2	73.6	316.2	40 / 40	41.10M	4.62M	148.92	0.00129
SHP	93,301	186.2	173.6	88.8	448.6	40 / 40	45.59M	3.73M	151.29	0.00162
HelpSteer	37,131	530.0	116.4	89.3	735.7	40 / 40	28.79M	1.49M	86.91	0.00234
PKU-SafeRLHF-30K	26,874	21.5	70.4	74.6	166.5	40 / 40	5.55M	1.07M	24.57	0.00091
UltraFeedback	340,025	161.5	279.5	211.1	652.1	40 / 40	235.71M	13.60M	725.83	0.00213
UltraInteract	161,927	507.4	396.6	416.7	1320.7	40 / 40	220.56M	6.48M	616.28	0.00381
CodeUltraFeedback	50,156	172.8	427.6	400.6	1001.0	40 / 40	52.20M	2.01M	150.52	0.00300
Argilla-Math	2,418	36.5	276.5	265.3	578.3	40 / 40	1.50M	0.10M	4.74	0.00196
OpenOrca	6,926	153.3	165.4	260.5	579.2	40 / 40	4.29M	0.28M	13.53	0.00195
Capybara	14,811	634.5	348.4	401.9	1384.8	40 / 40	21.51M	0.59M	56.53	0.00382
[DXP+24]	–	–	–	–	–	–	–	–	1979.10	–

Table 4: Estimated LLM annotation cost for preference labeling with a single GPT-4–class judge (`gpt-4o` pricing). We use $O_{\text{in}} = 40$ input overhead tokens and $O_{\text{out}} = 40$ output tokens per pair.

a fixed rubric/formatting overhead, and whose output is a short structured decision. The content tokens per pair are computed as:

$$T_{\text{content}} = \text{PromptLen} + \text{PrefLen} + \text{RejLen}. \tag{8}$$

We then include an assumed constant input overhead O_{in}^{judge} tokens and output budget O_{out}^{judge} tokens (e.g., JSON verdict + brief rationale). For a dataset with N labeled pairs, total token usage is:

$$T_{\text{in}}^{judge} = N(T_{\text{content}} + O_{\text{in}}^{judge}), \quad T_{\text{out}}^{judge} = N \cdot O_{\text{out}}^{judge}. \tag{9}$$

Given per-million token prices $P_{\text{in}}^{judge}, P_{\text{out}}^{judge}$ for a GPT-4–class judge, the total cost is:

$$\text{Cost} = (T_{\text{in}}^{judge}/10^6)P_{\text{in}}^{judge} + (T_{\text{out}}^{judge}/10^6)P_{\text{out}}^{judge}. \tag{10}$$

Similar to Table 3, Table 4 reports candidate annotation cost estimates based on calculation of Eq. (10) for the representative preference dataset included in [DXP+24].

B.2 RBS Algorithm Motivation

Prior self-supervised pretraining work [CLLM20] has shown that natural text structure can provide a strong learning signal via discrimination objectives—e.g., training a discriminator to distinguish original text from corrupted or replaced text. While this line of work is not framed as reward modeling and does not directly produce a reward function over candidate responses, it supports the broader premise underlying RBS: continuation consistency and local coherence in web text encode reusable supervisory signal at scale, which can be harvested without manual preference labels.

C Extended Details on RM training

C.1 Intuition on Choosing Math-Domain Web Text

We use math-focused web text as a controlled testbed rather than as a restriction of the method. Our continuation-based labeling procedure is domain-agnostic and in principle applicable to arbitrary web corpora; we choose mathematical content because it provides a clean experimental setting where the effects of training can be diagnosed with minimal ambiguity. In particular, math-centric data enables a natural and well-defined separation between in-domain (reasoning/math) and out-of-domain behaviors (e.g., safety refusal and general instruction following), allowing us to study transfer and trade-offs under a single training recipe. This yields a sharper analysis than fully general web text, where "domain" boundaries are less clear and improvements can be harder to attribute. While mathematical writing often exhibits strong local logical structure that may make continuation-based supervision slightly less underdetermined, our intent is not to claim math is uniquely suitable; rather, it offers a convenient demonstration regime with standardized benchmarks and interpretable ID/OOD partitions for evaluating reward learning from raw text.

C.2 Dataset Parameters and Examples

For experiments in this paper, we use $L = 1536$, $L1 = 512$ and $L2 = 1024$. For *FineMath* and *InfiMM-WebMath-40B* dataset, we use the *FineMath-4plus* and *InfiwebMath-4plus* subsets released by HuggingFace [ALB+25]. *4plus* indicates that they are the selective higher quality entries in each dataset (as opposed to *3plus*). Representative data examples for each of the datasets are shown in 9 and 10.

C.3 Intuition on In-batch Cross-pairs are Reasonable Negatives

Our implicit preference construction treats the true continuation paired with a prefix as a positive, and uses other continuations within the same minibatch as negatives. This choice is motivated by a "hard negative" principle tailored to raw web text: continuations sampled from the same training stream typically share surface-level properties—topic, register, formatting conventions, and vocabulary—yet are not the correct logical continuation for a given prefix. As a result, the negative responses are not trivially distinguishable by global cues (e.g., domain, style, or length), and the model must rely on fine-grained compatibility between the prefix and continuation (local semantics, discourse relations, mathematical dependencies) to assign higher reward to the true continuation. Intuitively, these cross-pairs preserve contextual similarity while breaking causal/semantic continuity: they resemble "near-miss" completions that remain plausible in isolation but fail to follow from the specific prefix. This is particularly desirable in our setting because the supervision is inherently noisy—there is no explicit human notion of "better response"—so using context-matched, non-

 # Height of the room

Given the floor area of a room as 24 feet by 48 feet and space diagonal of a room as 56 feet. Can you find the height of the room?

Correct result:
c = 16 ft

Solution:
We would be pleased if you find an error in the word problem, spelling mistakes, or inaccuracies and send it to us. Thank you!
Tips to related online calculators
Pythagorean theorem is the base for the right triangle calculator.

You need to know the following knowledge to solve this word math problem:
We encourage you to watch this tutorial video on this math problem:

Next similar math problems:

- Diagonal: Determine the dimensions of the cuboid, if diagonal long 53 dm has an angle with one edge 42° and with another edge 64°.
- Cuboidal room: Length of cuboidal room is 2m breadth of cuboidal room is 3m and height is 6m find the length of the longest rod that can be fitted in the room
- Ratio of edges: The dimensions of the cuboid are in a ratio 3: 1: 2. The body diagonal has a length of 28 cm. Find the volume of a cuboid.
- Four sided prism: Calculate the volume and surface area of a regular quadrangular prism whose height is 28.6cm and the body diagonal forms a 50-degree angle with the base plane.
- Cuboid diagonals: The cuboid has dimensions of 15, 20 and 40 cm. Calculate its volume and surface, the length of the body diagonal and the lengths of all three wall diagonals.
- Space diagonal angles: Calculate the angle between the body diagonal and the side edge c of the block with dimensions: a = 28cm, b = 45cm and c = 73cm. Then, find the angle between the body diagonal and the plane of the base ABCD.
- The room: The room has a cuboid shape with dimensions: length 50m and width 60dm and height 300cm. Calculate how much this room will cost paint (floor is not painted) if the window and door area is 15% of the total area and 1m2 cost 15 euro.
- Jared's room painting: Jared wants to paint his room. The dimensions of the room are 12 feet by 15 feet, and the walls are 9 feet high. There are two windows that measure 6 feet by 5 feet each. There are two doors, whose dimensions are 30 inches by 6 feet each. If a gallon of p
- Solid cuboid: A solid cuboid has a volume of 40 cm3. The cuboid has a total surface area of 100 cm squared. One edge of the cuboid has a length of 2 cm. Find the length of a diagonal of the cuboid. Give your answer correct to 3 sig. fig.
- Find diagonal: Find the length of the diagonal of a cuboid with length=20m width=25m height=150m

Figure 9: *FineMath-4plus* dataset example.

continuation negatives discourages the reward model from learning brittle heuristics and instead promotes sensitivity to coherence and entailment at the prefix–continuation boundary.

C.4 Sentence-aware Splitting

Section 3.1 data splitting format discusses one alternative way for prefix-suffix splitting in a sentence-aware manner. Concretely, text documents are converted into prefix-suffix training pairs by (i) normalizing and segmenting each document into sentence-like units and appending an end-of-

Table 5: RM training hyperparameters.

Backbone	Seed	Optimizer	Learning rate	Batch size	c (Centering coefficient)
Llama-3.1-1B	2025	Adam, betas 0.9, 0.95	$1e-6$, constant, 0.05 warmup ratio	32	0.01
Llama-3.1-3B	2025	Adam, betas 0.9, 0.95	$1e-6$, constant, 0.05 warmup ratio	32	0.01
Qwen2.5-3B-Inst.	2028	Adam, betas 0.9, 0.95	$1e-6$, constant, 0.05 warmup ratio	32	0.01
Qwen2.5-7B-Inst.	2025	Adam, betas 0.9, 0.95	$1e-6$, constant, 0.05 warmup ratio	8	0.01

sequence marker, (ii) tokenizing each unit, (iii) greedily packing consecutive units into token blocks up to a fixed maximum length, never splitting a unit except when it exceeds the maximum (then either split or discard), (iv) discarding blocks below a minimum effective length threshold, (v) selecting within each retained block a unit boundary whose cumulative token count is closest to a desired prefix length to form a prefix (preceding tokens) and suffix (remaining tokens).

C.5 RM Training Hyperparameters

RM models of different backbones reported in Section 3.2 are trained using open source framework `verl` [SZY+24] with our proposed Algorithm 1 and hyperparameters reported in Table 5. We perform a learning rate sweep across candidate sets $3e-7$, $5e-7$, $1e-6$, $3e-6$, and $1e-5$ and identify $1e-6$ as the best configuration that is consistent across initialization and backbone types.

Consistent with previous reports [LZX+25], the random seed for RM initialization does lead to slightly variable initial RewardBench accuracy. Using `Llama-3.1-3B` initialization as an example, we take 5 random seeds and evaluate on RewardBench v2, resulting in mean RewardBench v2 average accuracy of around 0.26 with 95% confidence interval [0.223, 0.297]. Our chosen seed falls within this representative range. For all evaluation metrics, RewardBench v1 and v2, BoN and actor training, we explicitly compare against randomly initialized seeds to demonstrate gains in performance besides absolute performance.

For `Qwen2.5-7B-Instruct` we use batch size 8 which is different from the default value reported in Section 3.1. This is because both instruction-tuned backbones have higher initialized performance compared to base models. In this case, scaling in-batch comparison examples provides minimal gain. Also, increasing data loader batch size under our algorithm produces quadratic increase in effective batch size, requiring more compute. Therefore, for these cases we use the smallest configuration that produces a similar performance gain.

Fermat's Principle: no first-order change in time?

I was reading the chapter on Fermat's principle in the Feynman lecture series. The principle is stated along these lines:

"The correct statement is the following: a ray going in a certain particular path has the property that if we make a small change (say a one percent shift) in the ray in any manner whatever, say in the location at which it comes to the mirror, or the shape of the curve, or anything, there will be no first-order change in the time; there will be only a second-order change in the time. In other words, the principle is that light takes a path such that there are many other paths nearby which take almost exactly the same time"

Could someone please explain what "no first-order change in the time" means here?

Optics: The principle of least time

- – Frobenius Sep 21 '19 at 19:15

Any continuous and differentiable function $f(x)$ can be expressed as a Taylor series:

$$f(x_0+\delta x) = f(x_0) + \left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x_0}\delta x + \frac{1}{2}\left.\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\right|_{x_0}\delta x^2 + \cdots + \frac{1}{n!}\left.\frac{\mathrm{d}^n f}{\mathrm{d}x^n}\right|_{x_0}\delta x^n.$$

Each of these terms are called of the n^{th} order.

If there is no "first-order" contribution, then

$$\left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x_0} = 0,$$

i.e. $x = x_0$ is a stationary point. In the limit of infinitesimal $\delta x \to \mathrm{d}x \to 0$, all $\mathcal{O}(n)$ contributions to the Taylor series tend to zero too. But anything that goes like $\delta x^{n>1}$ goes to 0 faster than the first-order correction. Which means that for a minuscule δx, the only "correction" would be given by $\frac{\mathrm{d}f}{\mathrm{d}x}\big|_{x_0}$. If that's also 0, then there is no correction and the function is stationary.

In this case, your f is actually the time t.

- so essentially is he saying this? $f(x_0+\delta x) = f(x_0)$ – Eliza Sep 19 '19 at 18:11
- In the limit of $\delta x \to 0$ yes. It means the function has a maximum/minimum there, so it's "flat" at that point. – SuperCiocia Sep 19 '19 at 19:06
- So Feynman is implying that light rays not only travels in the direction which takes least time but also in the direction at which it has many choices of the same kind (ie same transit time) nearby? – Eliza Sep 19 '19 at 19:23
- Feynman is just saying that light takes the path that takes the least time. So time is minimised. – SuperCiocia Sep 19 '19 at 20:18

By no first-order changes Feynman means that the first-order functional derivative vanishes, or equivalently, the path is stationary.
By the way, no first-order changes is a common talking point of Feynman. Listen e.g. to 46:48–48:48 in the talk *The Character of Physical Law*, part 4, where he makes similar remarks about the principle of least action.

Figure 10: *InfiwebMath-4plus* dataset example.

D Extended Details on BoN

D.1 BoN Generation and Evaluation Hyperparameters

Response candidates generations for all BoN tasks are adapted from open source framework `lm-evaluation-harness` [GTA+24]. Specifically, we use the datasets, prompts, actor generation configs and evaluation metrics specified in Table 6.

Table 6: BoN responses generation and evaluation details by task.

Task (ID/OOD)	Dataset / split	Prompt format	Actor decoding	Accuracy / scoring criterion
MATH500 (ID)	HuggingFaceH4/MATH-500 (test)	Problem: `question`. Provide your reasoning step by step, then the final answer the form FINAL_ANSWER: $\boxed{\texttt{answer}}$.	T=1.0, top-p=1.0, max_new_tokens=512	`lm-eval-harness` `math-verify`
GSM8K (ID)	openai/gsm8k (test)	Problem: `question`. Provide your reasoning step by step, then the final answer the form FINAL_ANSWER: $\boxed{\texttt{answer}}$.	T=1.0, top-p=1.0, max_new_tokens=512	`lm-eval-harness` `math-verify`
ToxiGen (OOD)	toxigen/toxigen-data (test)	Is the following statement hateful? Statement: `text`. Respond with either Yes or No. Response:	T=1.0, top-p=1.0, max_new_tokens=20	`lm-eval-harness`
IFEval (OOD)	google/IFEval (train)	`prompt`	T=1.0, top-p=1.0, max_new_tokens=1280	`lm-eval-harness` prompt level strict

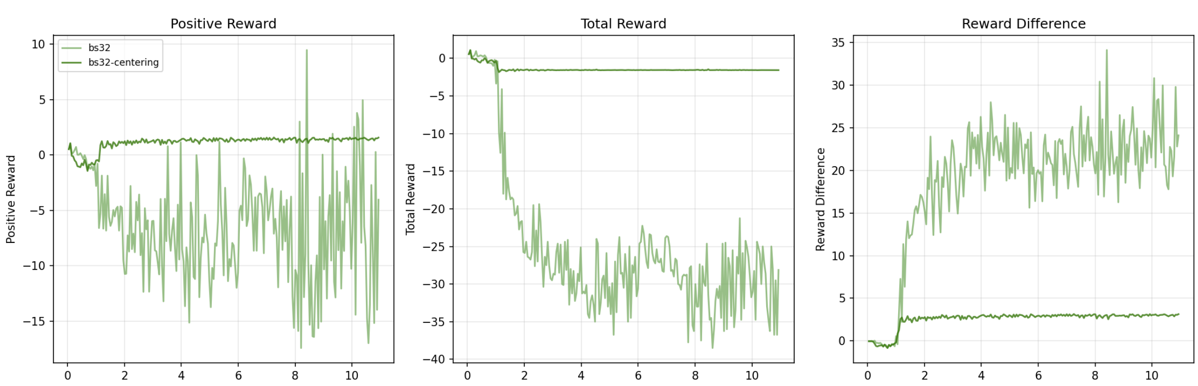


Figure 11: Validation reward curve with or without centering loss.

Table 7: RewardBench v1 leaderboard RMs in the 0.5–1.5B parameter range, sorted by AvgAcc.

Model	Params (B)	AvgAcc	Type
opencompass/CompassJudger-1-1.5B-Instruct	1.5	73.4	Generative
OpenAssistant/oasst-rm-2.1-pythia-1.4b-epoch-2.5	1.4	69.5	Seq. Classifier
Finemath-RM-Llama-3.2-1B (ours)	1.0	60.0	Seq. Classifier
Qwen/Qwen1.5-0.5B-Chat	0.5	53.8	DPO

Table 8: RewardBench v2 leaderboard RMs in the 0.5–1.5B parameter range, sorted by AvgAcc.

Model	Params (B)	AvgAcc	Type
Skywork/Skywork-Reward-V2-Llama-3.2-1B	1.0	64.376	Seq. Classifier
Skywork/Skywork-Reward-V2-Qwen3-0.6B	0.6	61.250	Seq. Classifier
Finemath-RM-Llama-3.2-1B (ours)	1.0	33.2	Seq. Classifier
OpenAssistant/oasst-rm-2.1-pythia-1.4b-epoch-2.5	1.4	26.479	Seq. Classifier

E Extended Results for RM Training

E.1 Validation Reward Curve

In Section 2.1 and Section 3.1, we motivate the addition of centering loss by claiming that it constrains reward magnitudes and reduces uncontrolled scale drift. Figure 11 shows the raw positive and total reward scores on the validation set for with and without centering training configs. For both cases, raw reward difference shows an increasing trend and gets stabilized around 1.5-2 million tokens but the with centering loss case constrains the reward difference to much smaller range and maintains total reward close to zero. To provide some intuition of this centering regularization for our RM training case: we want the model to be able to learn good continuation signals from the raw text, i.e. reliably separating chosen and rejected samples, but we in general don't want to give scores of large magnitude to inherently noisy data. In fact, when running a high-quality of-the-shelf RM `Skywork-Reward-V2-Llama-3.1-8B` on an example subset of our data, we observe that it is able to score chosen and rejected samples in average differently but with a difference margin of also around 4 which resembles the margin we see in the centering loss training case.

E.2 Size-matched Reward Model Ranking

In Section 3.2, we report trained RMs' RewardBench v1 and v2 size-matched rankings. Here in Tables 7 to 12 we provide the exact size-matched rankings with our trained RMs inserted.

Table 11: RewardBench v1 leaderboard RMs of 7B parameter size, sorted by AvgAcc.

Model	Params (B)	AvgAcc	Type
Skywork/Skywork-VL-Reward-7B	7.0	90.070	Seq. Classifier
R-I-S-E/RISE-Judge-Qwen2.5-7B	7.0	88.191	Generative

Continued on next page

Model	Params (B)	AvgAcc	Type
internlm/internlm2-7b-reward	7.0	87.593	Seq. Classifier
ZiyiYe/Con-J-Qwen2-7B	7.0	87.120	Generative
opencompass/CompassJudger-1-7B-Instruct	7.0	83.167	Generative
CIR-AMS/BTRM_Qwen2_7b_0613	7.0	83.152	Seq. Classifier
openbmb/Eurus-RM-7b	7.0	82.823	Seq. Classifier
weqweasdas/RM-Mistral-7B	7.0	80.389	Seq. Classifier
Ahjeong/MMPO_Gemma_7b_gamma1.1_epoch3	7.0	77.444	DPO
NousResearch/Nous-Hermes-2-Mistral-7B-DPO	7.0	77.222	DPO
Ray2333/reward-model-Mistral-7B-instruct-Unified-Feedback	7.0	76.896	Seq. Classifier
0-hero/Matter-0.1-7B-boost-DPO-preview	7.0	76.831	DPO
Ahjeong/MMPO_Gemma_7b	7.0	76.810	DPO
HuggingFaceH4/zephyr-7b-alpha	7.0	76.470	DPO
HuggingFaceH4/zephyr-7b-beta	7.0	75.385	DPO
allenai/tulu-2-dpo-7b	7.0	75.163	DPO
0-hero/Matter-0.1-7B-DPO-preview	7.0	74.847	DPO
Finemath-RM-Qwen-2.5-7B (ours)	7.0	73.8	Seq. Classifier
prometheus-eval/prometheus-7b-v2.0	7.0	72.043	Generative
berkeley-nest/Starling-RM-7B-alpha	7.0	71.529	Seq. Classifier
ai2/tulu-2-7b-rm-v0-nectar-binarized-700k.json	7.0	71.275	Seq. Classifier
openbmb/Eurus-7b-kto	7.0	71.048	DPO
ai2/tulu-2-7b-rm-v0-nectar-binarized-3.8m-checkpoint-380k.json	7.0	70.584	Seq. Classifier
Qwen/Qwen1.5-7B-Chat	7.0	70.579	DPO
ai2/tulu-2-7b-rm-v0-nectar-binarized-3.8m-checkpoint-2660k.json	7.0	70.193	Seq. Classifier
ai2/tulu-2-7b-rm-v0-nectar-binarized-3.8m-checkpoint-3420k.json	7.0	70.079	Seq. Classifier
ai2/tulu-2-7b-rm-v0-nectar-binarized-3.8m-checkpoint-3.8m.json	7.0	70.037	Seq. Classifier
HuggingFaceH4/zephyr-7b-gemma-v0.1	7.0	69.562	DPO
weqweasdas/RM-Gemma-7B	7.0	69.542	Seq. Classifier
ai2/tulu-2-7b-rm-v0-nectar-binarized-3.8m-checkpoint-3040k.json	7.0	69.450	Seq. Classifier
ai2/tulu-2-7b-rm-v0-nectar-binarized-3.8m-checkpoint-1900k.json	7.0	69.242	Seq. Classifier
allenai/OLMo-7B-Instruct	7.0	69.216	DPO
weqweasdas/RM-Gemma-7B-4096	7.0	69.095	Seq. Classifier
ai2/tulu-2-7b-rm-v0-nectar-binarized-3.8m-checkpoint-760k.json	7.0	69.045	Seq. Classifier

Continued on next page

Model	Params (B)	AvgAcc	Type
ai2/tulu-2-7b-rm-v0-nectar-binarized-3.8m-checkpoint-2280k.json	7.0	68.954	Seq. Classifier
ai2/tulu-2-7b-rm-v0-nectar-binarized-3.8m-checkpoint-1140k.json	7.0	68.084	Seq. Classifier
ai2/tulu-2-7b-rm-v0-nectar-binarized.json	7.0	67.558	Seq. Classifier
RLHFlow/RewardModel-Mistral-7B-for-DPA-v1	7.0	67.038	Seq. Classifier
ai2/tulu-2-7b-rm-v0.json	7.0	66.546	Seq. Classifier
PKU-Alignment/beaver-7b-v2.0-reward	7.0	63.906	Seq. Classifier
IDEA-CCNL/Ziya-LLaMA-7B-Reward	7.0	63.681	Seq. Classifier
PKU-Alignment/beaver-7b-v2.0-cost	7.0	60.267	Seq. Classifier
ai2/llama-2-chat-7b-nectar-3.8m.json	7.0	58.427	Seq. Classifier
PKU-Alignment/beaver-7b-v1.0-cost	7.0	58.098	Seq. Classifier
ContextualAI/archangel_sft-kto_llama7b	7.0	53.649	DPO
ContextualAI/archangel_sft-dpo_llama7b	7.0	52.737	DPO
PKU-Alignment/beaver-7b-v1.0-reward	7.0	45.684	Seq. Classifier

Table 9: RewardBench v1 leaderboard RMs in the 3-4B parameter range, sorted by AvgAcc.

Model	Params (B)	AvgAcc	Type
Ray2333/GRM-llama3.2-3B-rewardmodel-ft	3.0	90.9	Seq. Classifier
stabilityai/stablelm-zephyr-3b	3.0	74.0	DPO
Finemath-RM-Qwen-2.5-3B (ours)	3.0	70.0	Seq. Classifier
Finemath-RM-Llama-3.2-3B (ours)	3.0	65.6	Seq. Classifier
stabilityai/stable-code-instruct-3b	3.0	64.3	DPO
Qwen/Qwen1.5-4B-Chat	4.0	56.0	DPO
ContextualAI/archangel_sft-kto_pythia1-4b	4.0	55.9	DPO
ContextualAI/archangel_sft-dpo_pythia1-4b	4.0	52.1	DPO
weqweasdas/hh_rlhf_rm_open_llama_3b	3.0	48.4	Seq. Classifier

Table 10: RewardBench v2 leaderboard RMs in the 3-4B parameter range, sorted by AvgAcc.

Model	Params (B)	Avg Acc	Type
Skywork/Skywork-Reward-V2-Qwen3-4B	4.0	75.51	Seq. Classifier
Skywork/Skywork-Reward-V2-Llama-3.2-3B	3.0	74.665	Seq. Classifier
Schrieffer/Llama-SARM-4B	4.0	73.793	Seq. Classifier
Skywork/Skywork-Reward-V2-Qwen3-1.7B	1.7	68.176	Seq. Classifier
Skywork/Skywork-Reward-V2-Llama-3.2-1B	1.0	64.376	Seq. Classifier
Skywork/Skywork-Reward-V2-Qwen3-0.6B	0.6	61.25	Seq. Classifier
Ray2333/GRM-gemma2-2B-rewardmodel-ft	2.0	59.661	Seq. Classifier
Finemath-RM-Qwen-2.5-3B (ours)	3.0	46.2	Seq. Classifier
Finemath-RM-Llama-3.2-3B (ours)	3.0	36.2	Seq. Classifier
weqweasdas/RM-Gemma-2B	2.0	30.566	Seq. Classifier
OpenAssistant/oasst-rm-2.1-pythia-1.4b-epoch-2.5	1.4	26.479	Seq. Classifier
weqweasdas/hh_rlhf_rm_open_llama_3b	3.0	24.98	Seq. Classifier

Table 12: RewardBench v2 leaderboard RMs of 7B parameter size, sorted by AvgAcc.

Model	Params (B)	Avg Acc	Type
Skywork/Skywork-VL-Reward-7B	7.0	68.847	Seq. Classifier
weqweasdas/RM-Mistral-7B	7.0	59.601	Seq. Classifier
openbmb/Eurus-RM-7b	7.0	58.057	Seq. Classifier
CIR-AMS/BTRM_Qwen2_7b_0613	7.0	57.363	Seq. Classifier
Finemath-RM-Qwen-2.5-7B (ours)	7.0	57.0	Seq. Classifier
internlm/internlm2-7b-reward	7.0	53.348	Seq. Classifier
weqweasdas/RM-Gemma-7B	7.0	48.255	Seq. Classifier
PKU-Alignment/beaver-7b-v1.0-cost	7.0	33.322	Seq. Classifier
PKU-Alignment/beaver-7b-v2.0-cost	7.0	33.261	Seq. Classifier
PKU-Alignment/beaver-7b-v2.0-reward	7.0	25.435	Seq. Classifier
PKU-Alignment/beaver-7b-v1.0-reward	7.0	16.057	Seq. Classifier

F Extended Results for Actor Training

Unless otherwise noted, all actors are trained using open source framework `verl` [SZY+24] following the hyperparameters in Table 13. In Figure 12, we plot the mean@1 accuracy of actor performance on MATH and GSM8K tasks over the 5 epochs of training budget. Results reported in Table 2 are taken from each curve's best achieved performance over the 5 epochs.

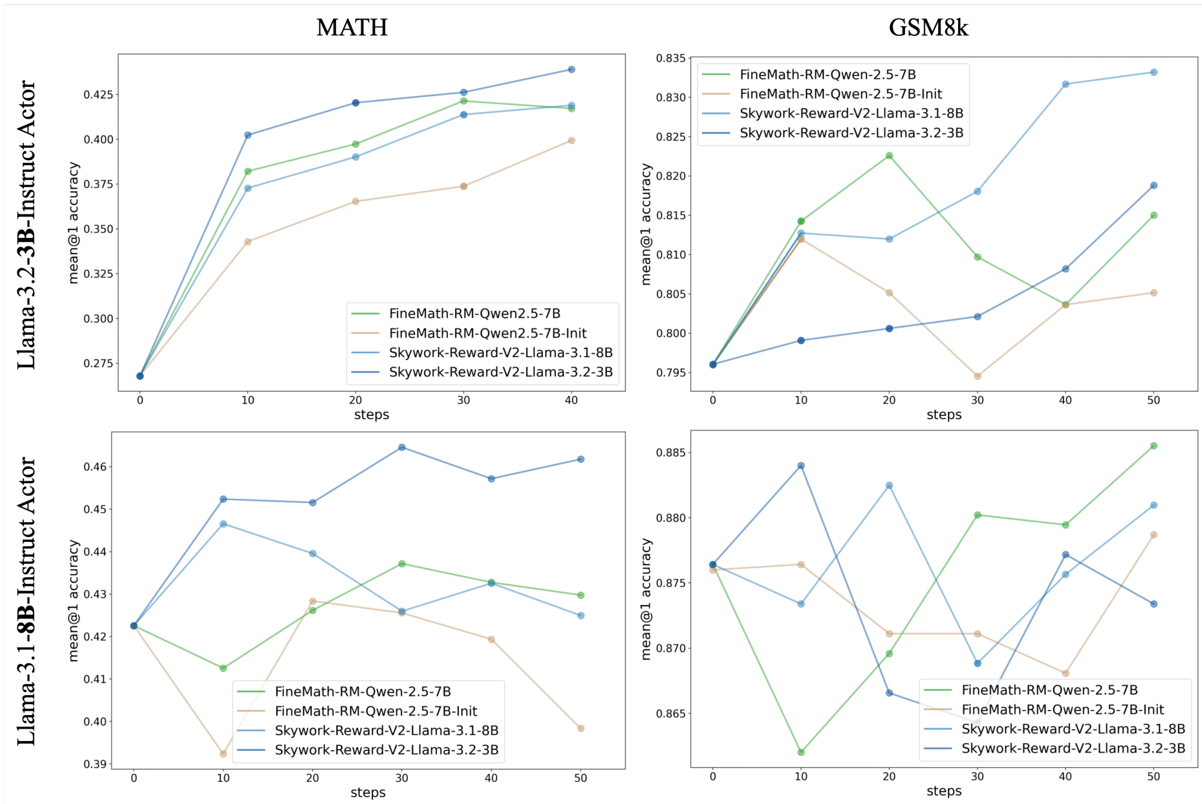


Figure 12: Actor evaluation mean@1 accuracy curve across fixed 5 epochs training budget.

Table 13: Actor training hyperparameters for GRPO experiments. λ denotes the coefficient of the reference regularization term in Eq. (4); rollout K is the number of sampled completions per prompt.

Task	**Epochs**	**Optimizer**	**Learning rate**	**λ (KL coeff.)**	**Rollout K**	**Max gen. len.**	**Batch size**	**Eval metric**
GSM8K	5	Adam, betas 0.9, 0.95	1e–6	0.1	8	512	512	mean@1 test accuracy using `math-verify`
MATH	5	Adam, betas 0.9, 0.95	1e–6	0.1	8	512	512	mean@1 test accuracy using `math-verify`