Scaling Reward Modeling without Human Supervision

Anonymous Author(s)
Affiliation
Address
email

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. Furthermore, using an RBS checkpoint as a mid-training initialization substantially amplifies gains from subsequent supervised fine-tuning. Overall, we demonstrate the feasibility and promise of training reward models without costly and potentially unreliable human annotations.

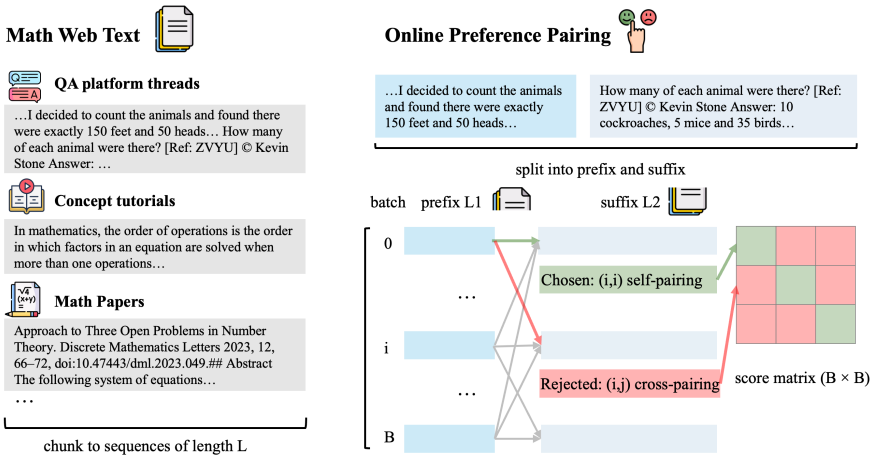


Figure 1: Schematic overview of our reward model training workflow from web math text. Raw documents are split into prefix–suffix pairs, where the true continuation is treated as the chosen response and other in-batch continuations serve as implicit negatives. The reward model is trained with a Bradley–Terry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

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

1 Introduction

Reinforcement learning from human feedback (RLHF) [9] has been the workhorse for building helpful and harmless language models [44, 3]. 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 [44, 66]. However, curating and annotating preference datasets can be resource-intensive [12, 64, 3, 29] (Section D.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 [9, 44, 7, 67, 21]. Such systematic noise can significantly misguide reward models, making data cleaning as critical as model design [72]: evidence in [52] shows that in RLHF, data quality often matters more than data quantity. Worsely, performing RLHF with the misguided reward model can naturally generalize into more serious misalignment like deception, alignment faking, and even sabotage, revealing reward hacking [55] as a potential root of unintended harmful behavior [31, 40]. Furthermore, as model capabilities scale, we face aligning systems whose relevant behaviors and failure modes exceed what humans (and comparably weak models) can reliably judge, making human-labeled reward signals fundamentally limited [5, 17].

These problems motivate us to investigate how much typical human supervision 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, Section E.1). By treating natural continuations as "chosen" responses and mismatched continuations as in-batch negatives [24, 8], 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 [41], 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 [44] or policy optimization [30, 50], our reward models substantially boost GSM8K [11] and MATH [27] accuracy and perform competitively with strong supervised reward model baselines of comparable size. We further show that our unsupervised RM checkpoint serves as an effective mid-training initialization, substantially amplifying gains from subsequent curated preference fine-tuning. 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: each fixed-length token block drawn from the web corpus is split at a mid-sequence breakpoint into a prefix p of L_1 tokens and a suffix r of L_2 tokens, directly mimicking the prompt–response structure of standard preference pairs. 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 [6] 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

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

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 add score-centering regularization to the Bradley–Terry objective [16]. 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* [1] dataset, which contain mathematical content filtered from CommonCrawl (Section E.3). 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, Section E). 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` [42] and instruction-tuned models `Qwen2.5-3B-Instruct` and `Qwen2.5-7B-Instruct` [48]. 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 [33] and RewardBench v2 [41], two general preference-alignment benchmarks. RewardBench covers chat and instruction following (`Chat`, `Chat Hard`), safety/refusal (`Safety`), and math/coding reasoning (`Reasoning`), while RewardBench v2 extends this to a broader multi-skill suite including `Math`, `Factuality`, `Precise IF`, `Focus`, `Ties`, and `Safety`. Since our RMs are trained on math-related web text, we report four metrics: overall score, *in-domain* (ID) performance on math-related subsets (`Reasoning` is ID for RewardBench v1 and `Math` as ID for RewardBench v2), *OOD-Safety*

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

performance on safety/refusal subsets (`Safety` is OOD-Safety for both benchmarks), and *OOD-Others* performance on all remaining subsets. We separate `Safety` because prior work suggests that optimizing for reasoning can degrade safety alignment [35, 28].

Using RewardBench v2, we also compare against training the same model under a matched 11M-token budget on two curated preference datasets: `Skywork-Reward-Preference-80K-v0.2` [37], a decontaminated 77K-pair dataset primarily built from Magpie-series synthetic preferences ranked by ArmoRM, and `RLHFMix1` proposed by [13], a uniformly sampled mixture of UltraFeedback [12], HH-RLHF [3], SHP [18], and Summarize-from-Feedback [57]. Section F.1 provides dataset composition, token statistics, and the corresponding number of pairs under the 11M-token budget.

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 [36] and GSM8K [11], and two safety focused OOD tasks, Toxigen [25] and IFEval [71]. For each prompt we sample $N \in 1, 2, 4, 8, 16, 32$ candidates and select the top-scoring one under the RM (Section G). 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 [38]. 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) [51] 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 as follows:

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

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 [50], and the second term is practically implemented using an MSE-style KL regularizer [49, 58] 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 [73, 44].

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 (Section H 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 Section I.

Mid-Training for Reward Modeling We hypothesize that our method can be viewed as an RM analogue of the mid-training stage commonly used in language model training: a continued pretraining

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

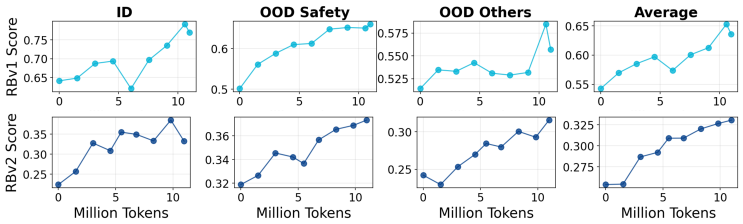


Figure 2: Scalability of our method with respect to data size. Reward models trained from scratch on Llama-3.2-3B improve steadily on RewardBench v1/v2 all subsets as token budget increases to 11M.

phase that can improve the effectiveness of later fine-tuning on curated datasets. To test whether our unsupervised web-text RM training provides a beneficial initialization for curated preference training, we take a representative RM checkpoint trained on a fixed training budget of 11M tokens and then continue training on `RLHFMix1` (introduced in Section 2.2). We compare three conditions: (1) training `Llama-3.2-3B` directly on the curated dataset mixture, (2) continuing training on the same curated mixture from our RM checkpoint, and (3) training `Llama-3.2-3B-Instruct` directly on the same curated mixture `RLHFMix1`. We evaluate all resulting RMs using RewardBench v2 under the same ID/OOD split protocol described above, reporting overall, ID, OOD-Safety, and OOD-Others performance to measure whether unsupervised initialization increases the performance gains attainable from curated preference supervision.

3 Experiments

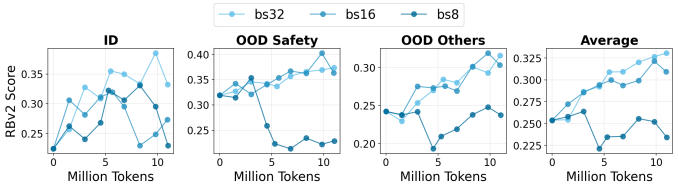
3.1 RM Training and Ablation

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 with Algorithm 1 on `Llama-3.2-3B` using *FineMath-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; Section H.2). Notably, at this token budget our method substantially outperforms training on token-matched portions of two widely used preference datasets, while remaining competitive with those datasets used in full (Section F.2). To identify what drives these gains, we ablate key pipeline choices (reported as improvements over the initialized seed): number of in-batch negatives, and data-splitting format and reward-centering loss. We report only RewardBench v2 accuracy for ablations, as it is newer and more challenging.

Batch Size Increasing batch size in our 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.

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



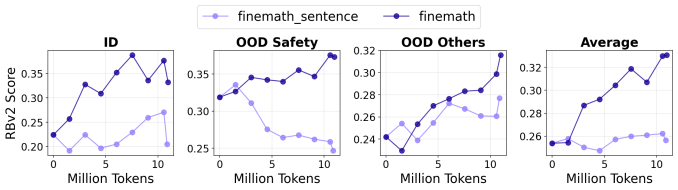
(b) RewardBench v2 learning curves across 11M token budget vs. batch size

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

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

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.

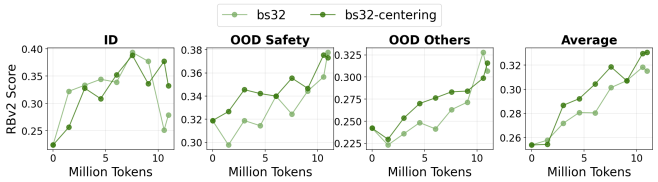


(b) RewardBench v2 learning curves across 11M token budget vs. data splitting design.

Figure 4: 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 5: Effect of centering loss on RewardBench v2 peak gains, learning trajectory and BoN MAP.

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 ID/OOD subsets (Fig. 4 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. Section E 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 its 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. 5 a,b).

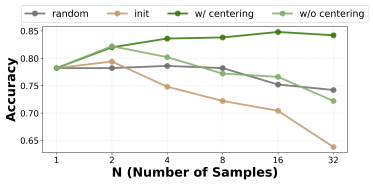


Figure 6: BoN accuracy curve on `Llama-3.1-8B-Instruct` GSM8K rollouts for RMs trained with or without centering loss (`init`–initialized seed).

We also assess downstream utility via BoN selection. Under weak, noisy Bradley–Terry supervision, reward scale can drift and margins can become overconfident, producing heavy-tailed scores that BoN is especially sensitive to. Score-centering constrains reward magnitudes and limits this drift (Section H.1), reducing spurious high-score outliers. On GSM8K with a `Llama-3.1-8B-Instruct` actor, centering substantially boosts BoN selection (+5.2 vs. +2.8, Fig. 5 a). The benefit also grows with selection strength: the centered RM's accuracy increases monotonically with N before saturating, while the uncentered RM peaks at small N and eventually falls below the random baseline (Fig. 6).

3.2 Generalization Across Initialization Backbones

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

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-3.2-1B	60.0	3	33.2	3
Llama-3.2-3B	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

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, Section H.3). These results indicate that our proposed RM training recipe transfers robustly across initialization choices.

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).

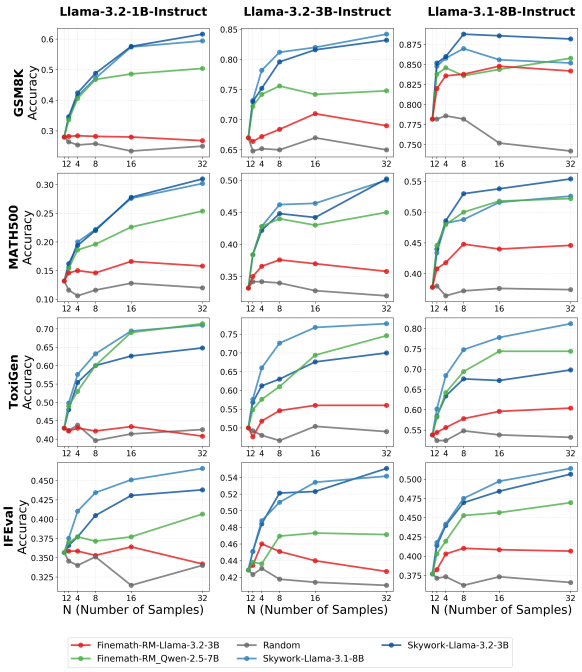


Figure 7: BoN scaling curves of RMs across two ID math tasks (GSM8K, MATH500) and two OOD safety/instruction-following tasks (Toxigen, IFEval).

Across ID and OOD tasks, our trained RMs show increasingly consistent BoN scaling as actor size grows. Accuracy generally improves with N: gains are modest or negligible for `FineMath-RM-Llama-3.2-3B` with the smallest `Llama-3.2-1B-Instruct` actor, but become clear for larger actors. `FineMath-RM-Qwen-2.5-7B` yields larger, often monotonic gains, especially with stronger actors. These results suggest that our RMs reliably rank and select better actor-generated candidates for both ID math reasoning and OOD safety-focused tasks. Actor-size scaling benefits `FineMath-RM-Llama-3.2-3B` more on ID than OOD tasks, while `FineMath-RM-Qwen-2.5-7B` shows similar trends across tasks (Fig. 7).

We further compare against two strong baselines, `Skywork-Reward-V2-Llama-3.2-3B` and `Skywork-Reward-V2-Llama-3.1-8B`, trained on 26M curated preference pairs [38]. Although Skywork achieves higher absolute MAP in most settings, the gap narrows as actor size increases. Notably, `FineMath-RM-Qwen-2.5-7B` is competitive with, and sometimes exceeds, at least one Skywork baseline on the ID MATH500 task and the OOD Toxigen task (Fig. 7). This is notable because our RMs use far less curated supervision—11M tokens of noisy web data, without human or LLM-based labels, versus 26M preference pairs. Overall, our web-trained RMs provide effective selection signals whose utility improves with both RM and actor size.

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

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 measured using both peak or final-epoch accuracy (Table 2, Table 18). It achieves the best result on GSM8K with the 8B actor and remains consistently competitive elsewhere (Table 2, Section I).

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**	0.447	0.882
Skywork-Reward-V2-Llama-3.2-3B	**0.439**	0.819	**0.465**	0.884
FineMath-RM-Qwen-2.5-7B	0.420	0.823	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.

3.5 Mid-Training for Reward Modeling

We study whether unsupervised RM training can serve as a mid-training stage that improves the effectiveness of subsequent curated preference learning like the mid-training in language models [47, 60]. We continue training from three initializations—`Llama-3.2-3B` base, our `FineMath-RM-Llama-3.2-3B` 11M token checkpoint, and `Llama-3.2-3B-Instruct`—all on the same curated mixture `RLHFMix1`, and evaluate on RewardBench v2.

Table 3: RewardBench v2 gains from training on curated preference dataset `RLHFMix1` under different initializations.

Checkpoint	**ID**	**OOD Safety**	**OOD others**	**RBv2**
Llama-3.2-3B	+14.5	+22.3	+1.6	+8.3
Llama-3.2-3B-Instruct	+19.4	+22.6	**+7.8**	**+13.1**
Finemath-RM-Llama-3.2-3B	**+21.4**	**+24.4**	+2.0	+10.4

As expected, instruct initialization outperforms base initialization, consistent with standard practice of starting RM training from an SFT model [44, 3]. Crucially, our RM checkpoint recovers much of this benefit: it achieves +10.3 average gain on RewardBench v2, matching or exceeding instruct on key subcategories—OOD-Safety (+24.4 vs. +22.6) and ID (+21.4 vs. +19.4)—while substantially outpacing the base model on ID (+14.5) (Table 3 and Fig. 11). This indicates that our unsupervised RM training provides a particularly strong foundation for learning in-distribution preferences from curated data. Gains on OOD Others are modest, likely due to forgetting induced by distribution shift from web text to curated data; in OOD Safety, a similar initial drop is recovered and ultimately surpassed (Fig. 11).

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 [9, 34]. Reinforcement learning from human feedback (RLHF) substantially outperforms supervised fine-tuning across summarization and dialogue tasks [56, 2, 23, 44, 69, 59]. To reduce annotation costs, subsequent work explored synthetic feedback, including model-generated critiques for scaling reward model training [4, 39]. Optimizing imperfect proxy objectives can lead to specification gaming or reward hacking, where agents exploit reward loopholes instead of achieving

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

intended goals [32]. Such failures appear across domains, including traffic control, pandemic response, and medical treatment [45]. In language models, reward overoptimization degrades output quality [56] and can induce behaviors such as sycophancy when optimizing for helpfulness or harmlessness [46]. Most closely related, [19] characterizes scaling laws of proxy reward exploitation in language models under best-of-N sampling and RLHF, but do not study robustness interventions. Theoretically, proxy rewards are provably vulnerable to exploitation under broad conditions [55]. 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 [40].

LLM-as-a-Judge Large language models are increasingly used as automated evaluators for model comparison and preference elicitation, enabling scalable benchmarks [68, 15]. However, extensive work has documented systematic limitations of LLM-as-a-Judge, including position bias [54, 61], non-transitive and inconsistent preferences [62], and sensitivity to prompt framing and candidate ordering [22]. Judges also exhibit social, demographic, and stylistic biases, raising concerns about fairness and validity [63]. 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 [14]. 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. Beyond reliability concerns, LLM-based annotation still requires running capable frontier models over large candidate pools, incurring substantial computational cost even in the absence of human labor (Section D.1). We sidestep this entirely by constructing preference pairs directly from raw web text without any candidate generation or annotation step.

Self-supervised reward modeling Prior self-supervised methods explore iteratively self-labeling of unlabeled data and fine-tunes on high-confidence pairs [26], self-rewarding loops where the model judges its own outputs [65], and large-scale AI-feedback datasets [12]. These approaches reduce annotation cost but still require explicit preference data: collected prompts, candidate responses, and chosen/rejected labels from humans, models or bootstrapped RMs (Section D.1). In contrast, our method derives pairwise supervision from naturally occurring document structure. This makes reward modeling closer to self-supervised pretraining: the training signal comes from raw web text itself, without relying on the pipeline for preference data curation.

5 Concluding Remarks

We study whether strong reward models can be learned without explicit human supervision, using only large-scale raw web text. We introduce reward-based scaling (RBS), which treats next-token continuation as an implicit preference signal and converts uncurated text into dense pairwise training data at essentially zero annotation cost. Empirically, unsupervised reward models trained on just 11M tokens of math-focused web text yield consistent gains across in-domain reasoning, out-of-domain safety, and general preference evaluations. Applied to best-of-N selection and policy optimization, they substantially improve downstream math performance and approach or match supervised reward model baselines of comparable size. Ablations show that batch-scale contrastive supervision, hard negative construction, and reward centering are key to stable weakly supervised training. Moreover, mid-training with RBS before supervised fine-tuning produces larger preference gains than starting from a base model, suggesting that web-scale unsupervised training helps models use curated data more effectively. More broadly, our results suggest that much of the supervision attributed to curated human preferences may already be latent in large text corpora. This opens a complementary path toward more scalable, reproducible, and potentially less biased reward modeling, while raising questions about the limits and failure modes of implicit preference signals. Future work may extend RBS beyond math-heavy domains, combine unsupervised and human supervision, and study robustness under stronger downstream optimization. Overall, we view reward modeling without human supervision not as a replacement for human feedback, but as a foundation for reducing its cost and expanding its reach.

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

References

[1] 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. URL https://arxiv.org/abs/2502.02737.

[2] 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.

[3] 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.

[4] 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.

[5] Samuel R. Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Christopher Olah, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Jackson Kernion, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Liane Lovitt, Nelson Elhage, Nicholas Schiefer, Nicholas Joseph, Noemí Mercado, Nova DasSarma, Robin Larson, Sam McCandlish, Sandipan Kundu, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Ben Mann, and Jared Kaplan. Measuring progress on scalable oversight for large language models, 2022. URL https://arxiv.org/abs/2211.03540.

[6] 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. ISSN 00063444, 14643510. URL http://www.jstor.org/stable/2334029.

[7] 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.

[8] 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.

[9] 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.

[10] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators, 2020. URL https://arxiv.org/abs/2003.10555.

[11] 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.

[12] 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. URL https://arxiv.org/abs/2310.01377.

[13] 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. ISSN 2835-8856. URL https://openreview.net/forum?id=a13aYUU9eU.

[14] 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.

[15] 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.

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

[16] 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. URL https://arxiv.org/abs/2312.09244.

[17] Joshua Engels, David D. Baek, Subhash Kantamneni, and Max Tegmark. Scaling laws for scalable oversight. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=u1j6RqH8nM.

[18] Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. Understanding dataset difficulty with $\mathcal{V}$-usable information. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5988–6008. PMLR, 17–23 Jul 2022.

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

[20] 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. URL https://zenodo.org/records/12608602.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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. URL https://arxiv.org/abs/2203.09509.

[26] Yifei He, Haoxiang Wang, Ziyan Jiang, Alexandros Papangelis, and Han Zhao. Semi-supervised reward modeling via iterative self-training. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7365–7377, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.434. URL https://aclanthology.org/2024.findings-emnlp.434/.

[27] 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)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

[28] 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. URL https://arxiv.org/abs/2503.00555.

[29] Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Juntao Dai, Boren Zheng, Tianyi Qiu, Jiayi Zhou, Kaile Wang, Boxuan Li, Sirui Han, Yike Guo, and Yaodong Yang. Pku-saferlhf: Towards multi-level safety alignment for llms with human preference, 2025. URL https://arxiv.org/abs/2406.15513.

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

[31] 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.

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

[32] 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.

[33] 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. URL https://arxiv.org/abs/2403.13787.

[34] 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.

[35] 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. URL https://arxiv.org/abs/2505.11423.

[36] 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.

[37] Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms, 2024. URL https://arxiv.org/abs/2410.18451.

[38] 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. URL https://arxiv.org/abs/2507.01352.

[39] Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-time scaling for generalist reward modeling, 2025. URL https://arxiv.org/abs/2504.02495.

[40] 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.

[41] Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. Rewardbench 2: Advancing reward model evaluation, 2025. URL https://arxiv.org/abs/2506.01937.

[42] Meta. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

[43] 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

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

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. URL https://arxiv.org/abs/2410.21276.

[44] 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.

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

[46] 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.

[47] 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.

[48] 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. URL https://arxiv.org/abs/2412.15115.

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

[50] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[51] 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.

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

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

[53] 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.

[54] 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.

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

[56] 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.

[57] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, 2022. URL https://arxiv.org/abs/2009.01325.

[58] Yunhao Tang and Rémi Munos. On a few pitfalls in kl divergence gradient estimation for rl, 2025. URL https://arxiv.org/abs/2506.09477.

[59] Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin, Enyu Zhou, Chenyu Shi, et al. Secrets of rlhf in large language models part ii: Reward modeling. *arXiv preprint arXiv:2401.06080*, 2024.

[60] Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling, 2025. URL https://arxiv.org/abs/2506.20512.

[61] 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. URL https://openreview.net/forum?id=fvkElsJOsN.

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

[63] 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.

[64] 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. URL https://arxiv.org/abs/2404.02078.

[65] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models, 2025. URL https://arxiv.org/abs/2401.10020.

[66] Hanlin Zhang, Benjamin L Edelman, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, and Boaz Barak. Watermarks in the sand: Impossibility of strong watermarking for generative models. *arXiv preprint arXiv:2311.04378*, 2023.

[67] 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.

[68] 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.

[69] Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, et al. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint arXiv:2307.04964*, 2023.

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

[70] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL http://arxiv.org/abs/2403.13372.

[71] 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. URL https://arxiv.org/abs/2311.07911.

[72] 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.

[73] 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. URL https://arxiv.org/abs/1909.08593.

Appendices

Table of Contents

A Limitations and Broader Impacts **17**

B Reproducibility: Dataset and Code **17**

C LLM Usage **17**

D Extended Related Works **17**
D.1 Cost Estimate of Current RLHF Pipeline 17
D.2 RBS Algorithm Motivation 19

E Extended Details on RM training **19**
E.1 Intuition on Choosing Math-Domain Web Text 19
E.2 Dataset Parameters and Examples 19
E.3 Contamination Analysis 19
E.4 Intuition on In-batch Cross-pairs are Reasonable Negatives 20
E.5 Proxy-Judge Analysis of Training Signal Quality 21
E.6 Sentence-aware Splitting 21
E.7 RM Training Hyperparameters 21

F Curated Dataset Details for Token-Matched Comparison **22**
F.1 Dataset Statistics and Token-Equivalent Pair Counts 22
F.2 RewardBench v2 Results: Token-Matched vs. Full-Dataset Comparison 23

G Extended Details on BoN **23**
G.1 BoN Generation and Evaluation Hyperparameters 23

H Extended Results for RM Training **24**
H.1 Validation Reward Curve 24
H.2 Multi-Seed Robustness 24

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

H.3 Size-matched Reward Model Ranking 25

I Extended Results for Actor Training **28**

J Extended Results for Continuing Training on Curated Preference Dataset **29**

K Licenses for Existing Assets **29**

L Compute Resources **30**

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

A Limitations and Broader Impacts

Our empirical evaluation focuses primarily on math-heavy data and benchmarks. This is a deliberate first setting for RBS: prior observations indicate that reward-model scores for math problem responses are often correlated with answer correctness [47]. In this domain, continuation coherence can capture more than surface fluency; it can also reflect logical consistency, preservation of intermediate dependencies, and valid reasoning structure. This makes math a suitable testbed for studying whether prefix–suffix coherence can serve as an implicit preference signal for reward model training. The current results should therefore be interpreted as strongest evidence for logic-driven domains like math. Broader evaluation using text corpora on other domains like knowledge-heavy articles or creative writings remains an important direction for future work.

The broader impact of this work is to lower the cost and data-construction burden of reward model training, thus making the alignment process more scalable. Existing pipelines typically require prompt collection, candidate generation, and human or LLM-based preference annotation. LLM-as-a-judge and semi-supervised reward modeling reduce parts of this burden on human, but still rely on generated candidates, curated preference triples, or model-based annotation, which are still costly (Section D.1). RBS instead constructs preference pairs directly from raw web text, removing both candidate generation and explicit annotation. To our knowledge, this is the first unsupervised data curation and annotation approach that trains reward models from raw text alone with empirical performance comparable to supervised methods. We do not see this method as a full replacement of supervised methods especially when high-quality preference data is already available. Rather, we see two complementary roles for our method. First, as noted above, it opens a new route for settings where preference data is scarce or expensive to collect. Second, our mid-training experiments (Section 3.5) show that unsupervised pretraining on raw text serves as an effective initialization: reward models trained this way reach a regime where subsequent fine-tuning on small, curated preference datasets yields substantially larger performance gains than fine-tuning from a cold start. This positions our approach not as a replacement for supervised training, but as a scalable precursor to it. This opens a path toward scaling reward model training to large web corpora, analogous to language model pretraining, and may make reward modeling more accessible in settings where preference data is expensive or unavailable.

B Reproducibility: Dataset and Code

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

C LLM Usage

LLMs were used in a supporting role during the preparation of this paper. Specifically, we used LLM assistance for: (1) checking grammar errors and editing text for clarity and readability; and (2) helping write code to visualize experimental results (e.g., plotting scripts for figures). LLM assistance was not used for any part of the core research methodology and algorithm design.

D Extended Related Works

D.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 [13], 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 [13] RLHF workflow.

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

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
[13]	–	–	–	–	–	–	–	–	3700.22	–

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

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` [43]), 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 4, we report candidate generation cost estimates based on the above calculations Eq. (7) for the representative preference dataset included in [13]. 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 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_{in}^{judge}, P_{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 4, Table 5 reports candidate annotation cost estimates based on calculation of Eq. (10) for the representative preference dataset included in [13].

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

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
[13]	–	–	–	–	–	–	–	–	1979.10	–

Table 5: 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.

D.2 RBS Algorithm Motivation

Prior self-supervised pretraining work [10] 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.

E Extended Details on RM training

E.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.

E.2 Dataset Parameters and Examples

For experiments in this paper, we use $L = 1536$, $L_1 = 512$ and $L_2 = 1024$. These values are fixed across all main experiments. We also tested two alternative L_1/L_2 ratios under the same total length $L = 1536$: $(1024, 512)$ and $(768, 768)$. All three configurations yield positive RewardBench v2 gains with an average of $+7.87 \pm 0.96$, and we do not observe a clear pattern relating the ratio to performance. We therefore do not treat the L_1/L_2 split as a critical tunable hyperparameter; the main experiments use $(512, 1024)$ as a simple default.

For *FineMath*, we use the *FineMath-4plus* subset released by HuggingFace [1]. *4plus* indicates that they are the selective higher quality entries in the dataset (as opposed to *3plus*). A representative data example is shown in 8.

E.3 Contamination Analysis

We use *FineMath-4plus* as one of the main RBS training corpora. This corpus was released as part of the SmolLM2 data pipeline [1], which reports 13-gram decontamination against several standard math and reasoning benchmarks, including GSM8K, MATH, MMLU, ARC-E, and ARC-C. Since

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

 # Period Of Function

by -2 views

Period of sinusoidal functions from equation. You can figure this out without looking at a graph by dividing with the frequency which in this case is 2.

Periodic functions are used throughout science to describe oscillations, waves and other phenomena that exhibit periodicity.

This is why this function family is also called the periodic function family. Therefore in the case of the basic cosine function $f(x)$, when this occurs we call the horizontal shift the period of the function. If a function has a repeating pattern like sine or cosine it is called a periodic function. Amplitude a: let b be a real number. Horizontal stretch is measured for sinusoidal functions as their periods.

For basic sine and cosine functions the period is 2π. Any function that is not periodic is called aperiodic. The period is defined as the length of one wave of the function. The period is the length on the x-axis in one cycle. The period of a sinusoid is the length of a complete cycle.

More formally, we say that this type of function has a positive constant k where any input x:

$$f(x + k) = f(x)$$

The period is defined as the length of a function's cycle. According to the periodic function definition, the period of a function is represented as $f(x) = f(x + p)$, where p is the real number equal to the period of the given function $f(x)$.

Where $f(x+P) = f(x)$ for all values of x. The period of a periodic function is the interval of x-values on which the cycle of the graph that is repeated in both directions lies. In this case, one full wave is 180 degrees or π radians. So the period of $\sin$ or $\cos$ is 2π.

The period of the function can be calculated using:

$$P = \frac{2\pi}{|b|}$$

In the problems below we will use the formula for the period P of trigonometric functions of the form $y = a\sin(bx + c) + d$ or $y = a\cos(bx + c) + d$.

The amplitude of $y = a\sin x$ and $y = a\cos x$ represents half the distance between the maximum and minimum values of the function. The period is the length of the smallest interval that contains exactly one copy of the repeating pattern. A periodic function repeats its values at set intervals called periods.

Figure 8: *FineMath-4plus* dataset example.

GSM8K and MATH/MATH500 are among our main downstream math evaluations, this provides an existing contamination control for the core math benchmarks used in our study.

We further perform our own 13-gram contamination audit between the exact RBS training data used in our experiments and benchmarks not explicitly covered by the *FineMath* release, including RewardBench, IFEval, and ToxiGen. We find no 13-gram overlap with these additional evaluation sets. These results suggest that the reported gains are unlikely to be explained by direct benchmark leakage under this standard n-gram overlap criterion.

We will release the contamination-checking logs together with the code and training data processing scripts.

E.4 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

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

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

E.5 Proxy-Judge Analysis of Training Signal Quality

The preceding subsection argues that in-batch cross-pairs serve as reasonable hard negatives on average. However, the difficulty of a batch's implicit preference signal is not uniform: it depends on how contextually similar the in-batch negatives happen to be to each specific prefix. When negatives are drawn from passages that are topically or stylistically distant from the target prefix, the true continuation may be trivially distinguishable, providing little useful ranking supervision. Conversely, when negatives come from passages sharing close context—as encouraged by our non-sentence-aware splitting and larger batch sizes—the comparison becomes genuinely hard. This batch-level variation in signal difficulty can contribute to the noisy or non-monotonic gains observed during training, and provides an additional motivation for the design choices we ablate in Section 3.1: larger batches increase the proportion of contextually similar in-batch negatives quadratically, and non-sentence-aware splitting retains more contiguous chunks from the same underlying passage, both of which raise the average difficulty of the training signal.

To quantify this effect, we conduct a diagnostic experiment using an external LLM as a proxy judge. We sample 100 training batches from the RBS training corpus, each with batch size 32. For each example in a batch, the prefix is paired with its true suffix as the positive continuation and with mismatched suffixes from other examples in the same batch as negative continuations. We then ask a `Llama-3.2-3B-Instruct` model to identify which candidate suffix is the coherent continuation of the given prefix. The proxy-judge accuracy for a batch is computed as the fraction of examples for which the model selects the true suffix over the in-batch negatives.

The resulting batch-level accuracies vary substantially, ranging from approximately 0.3 to 0.8 across the 100 sampled batches. This indicates that the implicit preference signal is heterogeneous: in some batches the true continuation is relatively easy to distinguish from mismatched suffixes, while in others the comparison is much more challenging. Critically, this heterogeneity also confirms that in-batch negatives are not uniformly trivial—many batches contain difficult comparisons where a small instruction-tuned model cannot reliably identify the true continuation, supporting our claim that the cross-pair construction provides a non-degenerate learning signal. Together, these results suggest that strategies which increase the proportion of hard batches—larger batch sizes and non-sentence-aware splitting—are directly beneficial for training signal quality, consistent with the ablation results in Section 3.1.

E.6 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-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).

E.7 RM Training Hyperparameters

RM models of different backbones reported in Section 3.2 are trained using open source framework `verl` [53] with our proposed Algorithm 1 and hyperparameters reported in Table 6. 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 [38], 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

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

Table 6: RM training hyperparameters.

Backbone	Seed	Optimizer	Learning rate	Batch size	c (center)
Llama-3.2-1B	2025	Adam betas 0.9, 0.95	10^{-6}, constant 0.05 warmup ratio	32	0.01
Llama-3.2-3B	2025	Adam betas 0.9, 0.95	10^{-6}, constant 0.05 warmup ratio	32	0.01
Qwen2.5-3B-Inst.	2028	Adam betas 0.9, 0.95	10^{-6}, constant 0.05 warmup ratio	32	0.01
Qwen2.5-7B-Inst.	2025	Adam betas 0.9, 0.95	10^{-6}, constant 0.05 warmup ratio	8	0.01

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.

F Curated Dataset Details for Token-Matched Comparison

F.1 Dataset Statistics and Token-Equivalent Pair Counts

Section 2.2 compares our method against training on popular curated preference datasets under a matched 11M-token budget. Here we provide details on the two example datasets and the corresponding equivalent number of preference pairs.

Skywork-Reward-Preference-80K-v0.2. This dataset [37] contains 77,016 preference pairs drawn primarily from the Magpie LLM-synthesis series (Magpie-Ultra, Magpie-Pro-Llama-3.1, Magpie-Pro, Magpie-Air), OffsetBias, WildGuard adversarial data, and HelpSteer2. The vast majority of preference labels are produced automatically via ArmoRM scoring of LLM-generated responses, making it representative of the modern LLM-annotated preference dataset regime. The dataset was subsequently decontaminated (v0.2) by removing 4,957 pairs with n-gram overlap with RewardBench evaluation prompts.

Based on dataset statistics, mean per-entry lengths are approximately: instruction ≈ 460 characters (≈ 115 tokens), chosen response $\approx 2{,}057$ characters (≈ 514 tokens), rejected response $\approx 2{,}337$ characters (≈ 584 tokens), yielding a mean of approximately $1{,}213$ tokens per preference pair (using a standard $\sim$4 characters per token approximation). Under our 11M-token budget, this corresponds to training on approximately $11{,}000{,}000/1{,}213 \approx 9{,}070$ pairs (≈9K pairs).

RLHFMix1. This curated mixture [13] combines UltraFeedback [12], HH-RLHF [3], SHP [18], and Summarize-from-Feedback [57], uniformly sampled to form a balanced stream of 244,216 pairs across four sources. UltraFeedback contributes LLM-annotated (GPT-4 judged) preferences, while HH-RLHF, SHP, and Summarize-from-Feedback are human-annotated, making this a representative mixed-annotation dataset.

Based on dataset statistics, mean per-entry lengths are approximately: instruction ≈ 896 characters (≈ 224 tokens), chosen response ≈ 594 characters (≈ 149 tokens), rejected response ≈ 481 characters (≈ 120 tokens), yielding a mean of approximately 493 tokens per preference pair. Under our 11M-token budget, this corresponds to training on approximately $11{,}000{,}000/493 \approx 22{,}310$ pairs (≈22K pairs).

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

Table 7: Per-entry token statistics for curated comparison datasets and the equivalent number of preference pairs under an 11M-token training budget. Token counts are estimated at ~ 4 characters per token.

Dataset	#Pairs	Instruction (tokens)	Chosen (tokens)	Rejected (tokens)	Tokens/ pair	Equiv. pairs @ 11M tok.
Skywork-80K-v0.2	77,016	≈115	≈514	≈584	≈1,213	≈9,070
RLHFMix1	244,216	≈224	≈149	≈120	≈493	≈22,310

Table 8: RewardBench v2 gains (Δ over initialized seed) for our method and for training on two curated preference datasets at token-matched and full-dataset budgets. **Bold** marks the best result per column among all token-matched conditions (including our method).

Method	Budget	ID	OOD-Safety	OOD-Others	Overall
Ours (FineMath web text)	11M tokens	+16.1	+5.4	+7.4	+7.7
Skywork-80K-v0.2	≈9K pairs (matched)	-4.4	$+1.8$	$+3.9$	$+1.8$
Skywork-80K-v0.2	77K pairs (full)	$+4.6$	$+19.6$	$+8.3$	$+9.8$
RLHFMix1	≈22K pairs (matched)	$+0.8$	$+1.9$	$+0.4$	$+0.8$
RLHFMix1	244K pairs (full)	$+14.5$	$+22.3$	$+1.6$	$+8.3$

These token-per-pair estimates indicate that our 11M-token budget is equivalent to training on roughly 9K pairs from Skywork-80K and 22K pairs from RLHFMix1—both a small fraction of each dataset's full size—providing a strictly controlled comparison of data-efficiency across supervision regimes.

F.2 RewardBench v2 Results: Token-Matched vs. Full-Dataset Comparison

Table 8 reports RewardBench v2 gains (Δ over the initialized seed) for our method and for training the same backbone on the two curated datasets, under both the token-matched budget (≈9K pairs for Skywork-80K and ≈22K pairs for RLHFMix1) and their full dataset sizes (77K and 244K pairs respectively).

At the token-matched budget, both curated datasets yield substantially smaller gains than our method across all subsets. Skywork-80K trained at budget even produces a regression on ID (-4.37), while RLHFMix1 at budget yields only marginal improvements overall ($+0.76$). Our method, trained on the same 11M-token budget of raw web text without any human or LLM annotation, achieves substantially larger gains: $+16.1$ on ID, $+5.4$ on OOD-Safety, $+7.4$ on OOD-Others, and $+7.7$ overall. When the curated datasets are used in full (8–22$\times$ more pairs and correspondingly more tokens), their gains increase substantially and in some cases exceed ours—most notably Skywork-80K full achieves $+19.6$ on OOD-Safety and RLHFMix1 full achieves $+22.3$. This is expected given the much larger data budget. Under a matched token budget, our unsupervised method competes strongly with—and clearly outperforms—training on popular LLM-annotated preference datasets, demonstrating that raw web text provides a more data-efficient training signal per token than curated preference pairs at this scale.

G Extended Details on BoN

G.1 BoN Generation and Evaluation Hyperparameters

Response candidates generations for all BoN tasks are adapted from open source framework `lm-evaluation-harness` [20]. Specifically, we use the datasets, prompts, actor generation configs and evaluation metrics specified in Table 9.

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

Table 9: 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: `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: `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

H Extended Results for RM Training

H.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 9 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 on average differently but with a difference margin of also around 4 which resembles the margin we see in the centering loss training case.

H.2 Multi-Seed Robustness

To verify that the reported gains are not an artifact of a particular random seed, we run the default training setup (`Llama-3.2-3B`, *FineMath-4plus*, 11M-token budget) under three independently sampled seeds and report RewardBench v2 results in Table 10. Across all three seeds the overall average gain lies between $+7.2$ and $+8.4$, with a mean of $+7.7 \pm 0.35$, indicating that the improvement

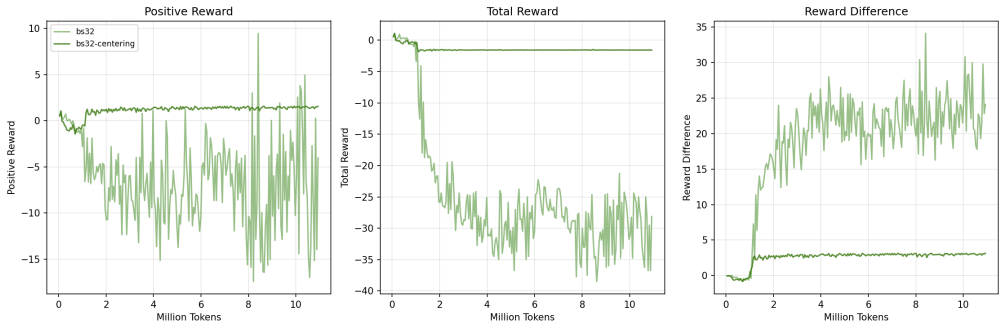


Figure 9: Validation reward curve with or without centering loss.

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

Table 10: RewardBench v2 gains (Δ over initialized seed) for three independent runs of the default `Llama-3.2-3B` setup. Mean and standard deviation are computed across seeds.

Seed	ID	OOD Safety	OOD Others	RBv2 Avg
2025	+16.1	+5.4	+7.4	+7.7
2027	+17.5	+10.5	+4.7	+8.4
2028	+19.1	+16.1	+0.2	+7.2
Mean	+17.6	+10.7	+4.1	$+7.7 \pm 0.35$

Table 11: 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

is stable and not sensitive to seed choice. The seed used for the main paper results (2025) falls within this range.

H.3 Size-matched Reward Model Ranking

In Section 3.2, we report trained RMs' RewardBench v1 and v2 size-matched rankings. Here in Tables 11 to 16 we provide the exact size-matched rankings with our trained RMs inserted.

Table 15: 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
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

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

Model	Params (B)	AvgAcc	Type
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
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

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

Table 12: 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

Table 13: 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 14: 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 16: 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

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

I Extended Results for Actor Training

Unless otherwise noted, all actors are trained using the open source framework `verl` [53] following the hyperparameters in Table 17. In Figure 10, 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.

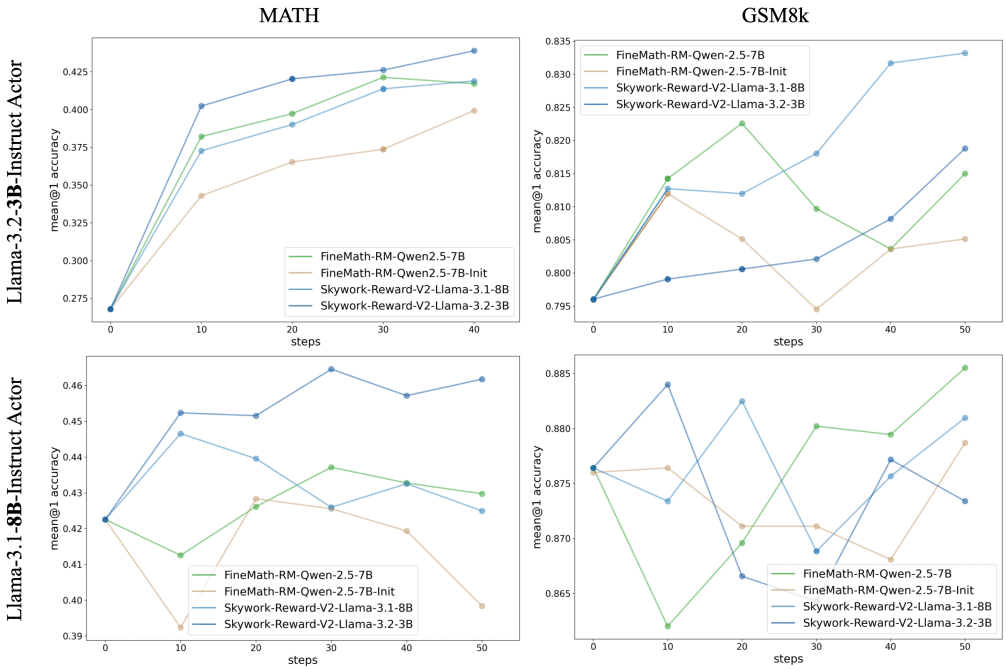


Figure 10: Actor evaluation mean@1 accuracy curve across fixed 5 epochs training budget.

Table 17: 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	Ep.	Optimizer	LR	λ KL	Rollout K	Max len.	Batch size	Eval metric
GSM8K	5	Adam betas 0.9, 0.95	10^{-6}	0.1	8	512	512	mean@1 test accuracy using `math-verify`
MATH	5	Adam betas 0.9, 0.95	10^{-6}	0.1	8	512	512	mean@1 test accuracy using `math-verify`

Table 18: Final-epoch mean@1 accuracy for actor training experiments. **Bold** indicates best result per column; underlined indicates second best. Results are consistent with within-budget peak results in Table 2, confirming that the claim holds under both measurement protocols.

Reward Model	`Llama-3.2-3B`		`Llama-3.1-8B`	
	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**	0.425	0.880
Skywork-Reward-V2-Llama-3.2-3B	**0.439**	0.819	**0.462**	0.873
FineMath-RM-Qwen-2.5-7B	0.420	0.815	0.430	**0.886**
FineMath-RM-Qwen-2.5-7B-Init	0.399	0.805	0.398	0.878

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

J Extended Results for Continuing Training on Curated Preference Dataset

Unless otherwise noted, all RMs are trained using the open source framework `Llamafactory` [70] following the hyperparameters in Table 19. In Figure 11, we plot for different initialization the RewardBench v2 score of the trained RM over the 1 epoch training course on the curated preference dataset `RLHFMix1`. Results reported in Table 3 are the RewardBench v2 performance gain achieved at the end of the training.

Table 19: Hyperparameters for RM training with curated preference dataset `RLHFMix1`. Cutoff length indicates the token number limits for prompt and response concatenated sequence.

Dataset	Epochs	Optimizer	LR	Warmup ratio	Cutoff len.	Batch size
`RLHFMix1`	1	Adam betas 0.9, 0.95	$2e{-6}$	0.03	2048	512

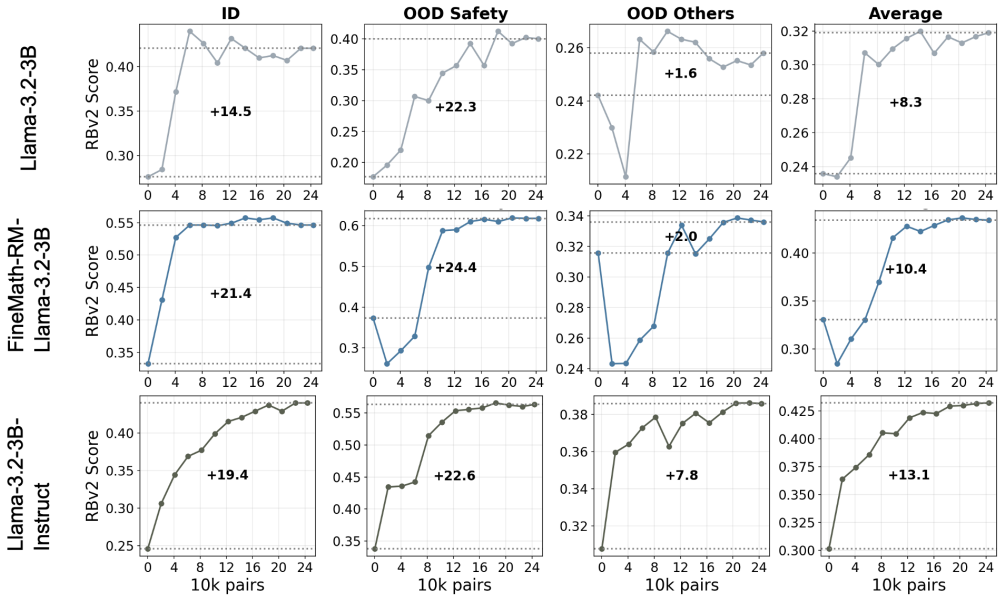


Figure 11: RewardBench v2 curve across the 1 epoch training course on the curated preference dataset `RLHFMix1` for 3 representative initializations `Llama-3.2-3B` base model, `FineMath-RM-Llama-3.2-3B` trained from our method, and `Llama-3.2-3B-Instruct`.

K Licenses for Existing Assets

Datasets.

- **FineMath-4plus** [1] — released by HuggingFaceFW under the ODC-By 1.0 license.
- **RLHFMix1** — a mixture of UltraFeedback [12] (MIT), HH-RLHF [3] (MIT), SHP [18] (CC BY 4.0), and Summarize from Feedback [57] (MIT).
- **Skywork-Reward-Preference-80K-v0.2** [37] — released under CC BY 4.0.
- **GSM8K** [11] — released by OpenAI under MIT.
- **MATH** [27] — released under MIT.
- **ToxiGen** [25] — released under MIT.
- **IFEval** [71] — released under Apache 2.0.
- **RewardBench v1** [33] and **RewardBench v2** [41] — released under Apache 2.0.

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

Models.

- **Llama 3 family** (`Llama-3.2-1B`, `Llama-3.2-3B`, `Llama-3.1-8B` and Instruct variants) [42] — released under the Llama 3 Community License.
- **Qwen2.5 family** (`Qwen2.5-3B-Instruct`, `Qwen2.5-7B-Instruct`) [48] — released under Apache 2.0.
- **Skywork-Reward-V2** models [38] — released under Apache 2.0.

Frameworks and Tools.

- **verl** [53] — released under Apache 2.0.
- **LlamaFactory** [70] — released under Apache 2.0.
- **lm-evaluation-harness** [20] — released under MIT.
- **math-verify** — released under Apache 2.0.

L Compute Resources

All experiments are run on NVIDIA A100 or H100 80GB GPUs. The approximate compute cost per experimental type is summarized below.

- **RM training.** Each run uses 1, 2, or 4 H100 GPUs and takes 1–2 days depending on the backbone size and training budget.
- **Best-of-N (BoN) evaluation.** Each evaluation run uses 1 H100 GPU and takes a few hours.
- **Actor training.** Each run uses 2 H100 GPUs and takes approximately half a day.
- **RM mid-training (curated preference dataset).** Each run uses 1 H100 GPU and takes a few hours.

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction describe the proposed continuation-based reward-modeling method, its intended scope, and the reported RewardBench, best-of-N, actor-training and RM mid-training results. The claims are supported by the methodology and experiments in Sections 2 and 3.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: The paper discusses limitations in Sections A and 5, including domain scope to math and noisy continuation-derived supervision.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A].

Justification: The paper proposes an algorithm and empirical study but does not state formal theoretical results or proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: The paper gives the algorithm, datasets, backbone models, evaluation protocol, and hyperparameters in Sections 2, E, G, I and J. Code and dataset release timing is discussed in Section B.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: The dataset and codebase will be released in an online repository after double-blind period ends (Section B). Detailed experimental instructions are already covered in Sections 2, E, G, I and J.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The main text describes the datasets, models, evaluation splits, and metrics in Sections 2.2 and 3, and the appendix provides training hyperparameters, optimizer choices, learning-rate sweeps, BoN evaluation details, and actor-training settings in Sections E, G, I and J.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: The appendix reports error bars and confidence intervals in at least three places: (1) Section E.1 reports mean $\pm$ standard deviation across dataset-parameter configurations; (2) the RM training hyperparameters section reports a 95% confidence interval over five random seeds for the initialized RewardBench v2 accuracy; and (3) Section H.2 reports mean

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

± standard deviation of RewardBench v2 gains across three independent seeds, confirming the stability of the main results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: Section L reports the GPU types (NVIDIA A100/H100 80GB) and approximate wall-clock time for each experimental category: RM training (1–4 H100s, 1–2 days), BoN evaluation (1 H100, a few hours), actor training (2 H100s, ∼half a day), and RM mid-training (1 H100, a few hours).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work uses public datasets and open-source evaluation/training frameworks, does not involve new human-subject data collection, and includes safety-oriented evaluation.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: Sections 1, A and 5 discuss potential benefits from reducing dependence on costly human annotation and risks from reward hacking, benchmark overfitting, and imperfect reward-model deployment.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The current submission does not release a high-risk generative model or scraped dataset.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

Answer: [Yes].

Justification: Section K lists the license for each dataset (FineMath-4plus: ODC-By 1.0; RLHFMix1 components: MIT/CC BY 4.0; Skywork-80K: CC BY 4.0; GSM8K/MATH/ToxiGen: MIT; IFEval/RewardBench: Apache 2.0), model (Llama 3: Llama Community License; Qwen2.5/Skywork-Reward-V2: Apache 2.0), and framework (verl/LlamaFactory/math-verify: Apache 2.0; lm-evaluation-harness: MIT) used in this work.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A].

Justification: The paper does not introduce new data assets; the training and evaluation data are all existing publicly available datasets. The code release (planned post-review) is a re-implementation of existing algorithms and does not constitute a new standalone asset.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A].

Justification: The paper does not involve new crowdsourcing experiments or new research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The paper does not conduct new human-subject studies, so IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes].

    Justification: LLMs were used for checking grammar errors, editing text for clarity, and helping write code to visualize experimental results. Details are provided in Section C.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.

Confidential reviewer copy. This manuscript is submitted to the 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Unauthorized sharing, redistribution, or disclosure is strictly prohibited.