Patch the Distribution Mismatch: RL Rewriting Agent for Stable Off-Policy SFT

Anonymous ACL submission

Abstract

Large language models (LLMs) have made rapid progress, yet adapting them to downstream scenarios still commonly relies on supervised fine-tuning (SFT). When downstream data exhibit a substantial distribution shift from the model's prior training distribution, SFT can induce catastrophic forgetting. To narrow this gap, data rewriting has been proposed as a data-centric approach that rewrites downstream training data prior to SFT. However, existing methods typically sample rewrites from a prompt-induced conditional distribution, so the resulting targets are not necessarily aligned with the model's natural QA-style generation distribution. Moreover, reliance on fixed templates can lead to diversity collapse. To address these issues, we cast data rewriting as a policy learning problem and learn a rewriting policy that better matches the backbone's QA-style generation distribution while preserving diversity. Since distributional alignment, diversity and task consistency are automatically evaluable but difficult to optimize end-to-end with differentiable objectives, we leverage reinforcement learning to optimize the rewrite distribution under reward feedback and propose an RL-based data-rewriting agent. The agent jointly optimizes QA-style distributional alignment and diversity under a hard task-consistency gate, thereby constructing a higher-quality rewritten dataset for downstream SFT. Extensive experiments show that our method achieves downstream gains comparable to standard SFT while reducing forgetting on non-downstream benchmarks by 12.34% on average. Our code is available at https://anonymous.4open.science/r/Patch-the-Prompt-Gap-4112.

1 Introduction

In recent years, Large Language Models (LLMs) have advanced rapidly (OpenAI et al., 2024) and demonstrated strong capabilities in tasks such as general dialogue (Touvron et al., 2023), information retrieval (Sun et al., 2024), and reasoning (Wei et al., 2023). However, when deployed in domain-specific or downstream scenarios, Supervised Fine-Tuning (SFT) on downstream data is still commonly required to improve performance on target tasks (Ouyang et al., 2022; Dong et al., 2024). Despite its effectiveness, downstream SFT can inadvertently erode previously acquired general capabilities (Luo et al., 2025; Huang et al., 2024a) ($i.e.$ catastrophic forgetting) (Li et al., 2024), especially under a substantial distribution shift between the downstream data and the model's prior training data distribution (Huang et al., 2025). In such cases, fine-tuning may bias the model toward the downstream distribution at the expense of other capabilities (Kotha et al., 2024). A prevalent perspective attributes this issue to the off-policy nature of downstream SFT (Zhang et al., 2025c; Chen et al., 2025), where demonstrations may be low-likelihood under the updated policy, inducing training instability.

To reduce this distribution shift, a data-centric approach intervenes at the data source by rewriting the downstream training data before SFT (Singh et al., 2024; Zhang et al., 2025a). The typical rewriting framework prompts the instruction-tuned base model π_0 with an input x, a reference solution $y^\star$, and a rewriting prompt x_{prompt} to sample a rewrite $\tilde{y}$, where $\tilde{y}$ is task-consistent with $y^\star$ yet more in-distribution under π_0 (Yang et al., 2024; Zhao et al., 2025). For example, SDFT (Yang et al., 2024) follows a standard data-rewriting pipeline and trains on the rewritten data, whereas Mind the Gap (Zhao et al., 2025) first lets the model attempt to solve each instance and rewrites expert demonstrations only for those it fails to solve.

Although data rewriting has shown promise in mitigating catastrophic forgetting, it still suffers from a key deficiency in distributional alignment. Standard rewrites are typically sampled from a constrained, prompt-induced conditional distribu-

tion, *i.e.* $\tilde{y} \sim \pi_0(\cdot \mid x, y^\star, x_{\text{prompt}})$, whereas downstream SFT more closely resembles QA-style completion conditioned only on x, corresponding to $\pi_0(\cdot \mid x)$. Therefore, the rewrites may be more in-distribution only under the rewriting-prompt–induced conditional distribution, rather than genuinely closer to the QA-style generation distribution, and thus can only partially narrow the effective distribution gap. This leads to two limitations: first, non-QA-style rewrites may introduce templated or unnatural phrasing, weakening the match between supervision and the model's natural completion mode and degrading the efficiency and stability of downstream learning; second, sampling $\tilde{y} \sim \pi_0(\cdot \mid x, y^\star, x_{\text{prompt}})$ does not guarantee reducing the key gap relevant to $\pi_0(\cdot \mid x)$ (or $\pi_\theta(\cdot \mid x)$), and therefore its ability to mitigate catastrophic forgetting is likewise constrained. Moreover, existing rewriting methods often rely on a small set of fixed prompt templates, which can further bias sampled rewrites toward templated and format-homogeneous patterns, inducing diversity collapse and under-representing many equally valid realizations (Yun et al., 2025).

Based on the above analysis, we model data rewriting as *policy learning*, aiming to learn a rewriting policy that is consistent with the QA-style generation distribution and yields higher-quality rewrites. In rewriting policy learning, task consistency, QA-style distributional alignment, and output diversity can all be formulated as automatically computable scalar feedback on sampled rewrites, making them naturally suitable as reward signals for policy optimization. Compared to maximum-likelihood fitting under a constrained rewriting prompt, reinforcement learning can directly maximize the expected quality of sampled rewrites under these criteria, while incorporating non-differentiable verification procedures via hard gating or auxiliary shaping rewards; therefore, training the rewriting policy with RL is a natural choice in this setting.

Motivated by this perspective, we propose an RL-based data-rewriting agent R_ϕ, which optimizes QA-style distributional alignment and diversity at the data source via automatically evaluable reward signals. To reduce reward noise and stabilize policy learning, we treat task consistency as a hard constraint: we compute and optimize alignment/-diversity rewards only for rewrites that pass a task-consistency verification gate. Meanwhile, we view prompt-induced rewriting as a local deviation from the base model's QA-style generation distribution and learn a lightweight low-rank residual "patch" via LoRA on top of a frozen base model π_0, enabling controllable local correction that suppresses policy drift and avoids over-correction (Hu et al., 2022).

Our contributions are summarized as follows:

- **Distribution Mismatch Insight:** We identify a distribution mismatch in existing data-rewriting pipelines and propose an RL-based rewriting framework that learns to generate rewrites closer to the backbone's QA-style generation distribution.

- **Unified Objective:** We introduce a unified objective that enforces task consistency while jointly optimizing distributional alignment and diversity, thereby constructing a higher-quality rewritten dataset with stronger distributional consistency.

- **Extensive Experiments:** Extensive experiments demonstrate that our method achieves downstream performance comparable to standard SFT, while substantially mitigating catastrophic forgetting on general-domain benchmarks.

2 Related Work

Catastrophic Forgetting and Optimization. Catastrophic forgetting is widely observed when adapting large language models (LLMs) to domain-specific or downstream data, where downstream gains often come at the expense of general capabilities (Luo et al., 2025; Kang et al., 2025). A traditional line of work mitigates forgetting via rehearsal-based continual learning and pseudo-rehearsal, replaying data from earlier stages to preserve prior abilities (Lopez-Paz and Ranzato, 2022; Wang et al., 2024a; Huang et al., 2024b; Du, 2025).

More recently, motivated by the off-policy nature of SFT, a growing line of work stabilizes post-training by introducing conservative-update mechanisms (e.g., DFT-style anchoring (Wu et al., 2025), proximal/trust-region constraints (Zhu et al., 2025a), and token-level clipping (Anonymous, 2025)) to bound per-step updates and suppress policy drift (Zhu et al., 2025b). However, while these methods improve stability, they can also restrict adaptation under distribution shift, reflecting a practical stability–plasticity tension (Dohare et al., 2024). In contrast, data rewriting offers a data-centric alternative: it proactively narrows the distribution gap between expert demonstrations and the current policy by rewriting supervision targets,

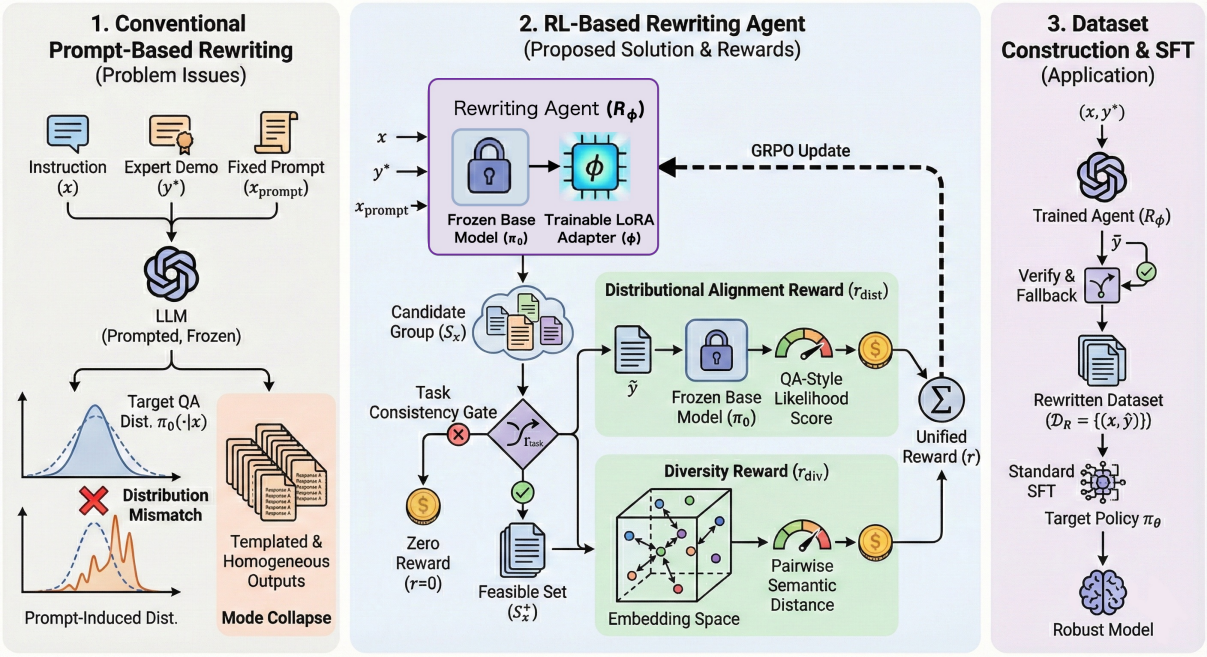

Figure 1: The framework of rewriting agent.

preserving expert knowledge while making downstream supervision more in-distribution, thereby stabilizing training and reducing forgetting (Yang et al., 2024; Zhao et al., 2025).

Diversity Collapse Induced by Formatting and Alignment. Recent studies show that structured templates and rigid formatting constraints can substantially suppress output diversity in instruction-tuned LLMs, leading to diversity/mode collapse (Yun et al., 2025; Xiao et al., 2025). Related analyses further attribute alignment-induced mode collapse to typicality bias in preference data, suggesting that alignment or template-like prompting can systematically contract the effective output space (Zhang et al., 2025b; Li et al., 2024).

3 Method

3.1 Problem Setup

Consider a downstream supervised dataset $\mathcal{D}_E = \{(x_i, y_i^\star)\}_{i=1}^N$ and a target policy π_θ initialized from π_0. Standard SFT on $\mathcal{D}_E$ can be optimization-unstable under distribution mismatch, exacerbating catastrophic forgetting; see Appendix B. We therefore construct an optimized training dataset that narrows this gap at the data level, stabilizing SFT updates and mitigating forgetting.

3.2 Framework

To mitigate the off-policy instability of downstream SFT, we adopt a data-centric intervention that rewrites supervision before fine-tuning. While prior data-rewriting pipelines (Yang et al., 2024; Zhao et al., 2025) largely rely on prompt-driven heuristics with a small set of fixed templates, we instead train a dedicated data-rewriting agent R_ϕ to perform rewriting. The goal is to produce higher-quality supervision that preserves expert knowledge, better matches the base model's natural QA-style distribution $\pi_0(\cdot \mid x)$, and explicitly avoids mode collapse by encouraging diverse yet acceptable realizations.

Stage I: RL training of the rewriting agent. We learn a parameter-efficient rewriting policy R_ϕ on top of a frozen π_0 via LoRA adapters (Hu et al., 2022). Given an input x, its expert demonstration $y^\star$, and a rewriting prompt x_{prompt}, the agent generates a rewrite

$$\tilde{y} \sim R_\phi(\cdot \mid x, y^\star, x_{\text{prompt}})\,. \tag{1}$$

We train R_ϕ with on-policy RL using automatically evaluable signals that jointly target (i) task consistency, (ii) distributional alignment to $\pi_0(\cdot \mid x)$ (conditioning on x only), and (iii) diversity as an anti-collapse regularizer; the unified reward is detailed in Sec. 3.4.

Stage II: dataset construction and downstream SFT. We apply the trained R_ϕ to construct a rewritten dataset $\mathcal{D}_R$ (with verification and fallback), and then perform standard SFT of π_θ (initialized from π_0) on $\mathcal{D}_R$ to obtain the final model. Details are given in Sec. 3.5.

3.3 Parameter-Efficient Rewriting Policy and GRPO Optimization

We employ the frozen instruction-tuned base model π_0 as the backbone and parameterize the rewriting policy R_ϕ with LoRA adapters, training only a small set of adapter parameters ϕ (Hu et al., 2022). This parameter-efficient design is well-suited for rewriting, which mainly requires lightweight calibration of existing generation behaviors rather than learning a new distribution from scratch. Moreover, freezing the backbone and restricting updates to low-rank adapters provides a capacity-limited "patch" over π_0, which helps bias learning toward targeted adjustments and empirically reduces unnecessary drift.

For optimization, we update R_ϕ using Group Relative Policy Optimization (GRPO) (Shao et al., 2024), a PPO-style on-policy policy-gradient method that estimates the baseline from groups of sampled outputs, eliminating the need for a separate value network. The overall objective is:

$$\max_{\phi} \mathbb{E}_{(x,y^\star)\sim\mathcal{D}_E,\ \tilde{y}\sim R_\phi(\cdot|x,y^\star,x_{\text{prompt}})}\big[r(x,y^\star,\tilde{y})\big]. \tag{2}$$

3.4 Unified Objective: Task Consistency, Distributional Alignment, and Diversity

To jointly optimize task consistency, distributional alignment, and diversity, we define a unified objective. For each expert sample $(x, y^\star)$, we sample a group of K candidate rewrites $\tilde{y}^{(k)} \sim R_\phi(\cdot \mid x, y^\star, x_{\text{prompt}})$ to form $\mathcal{S}_x = \{\tilde{y}^{(k)}\}_{k=1}^K$. For any candidate $\tilde{y}$, we define three reward components: $r_{\text{task}}(x, y^\star, \tilde{y})$, $r_{\text{dist}}(x, \tilde{y})$, and $r_{\text{div}}(x, \tilde{y}; \mathcal{S}_x^+)$ to measure task consistency, distributional alignment, and diversity, respectively, where $\mathcal{S}_x^+ = \{\tilde{y}^{(k)} \in \mathcal{S}_x : r_{\text{task}}(x, y^\star, \tilde{y}^{(k)}) = 1\}$. We treat $r_{\text{task}} \in \{0, 1\}$ as a hard feasibility gate: the distributional-alignment and diversity rewards are applied only if a rewrite passes the task-consistency check; otherwise, these auxiliary metrics are masked out. Accordingly, we use the following gated total reward:

$$r(x,y^\star,\tilde{y}) = r_{\text{task}}(x,y^\star,\tilde{y}) + r_{\text{task}}(x,y^\star,\tilde{y}) \cdot \big(\lambda_{\text{dist}}\, r_{\text{dist}}(x,\tilde{y}) + \lambda_{\text{div}}\, r_{\text{div}}(x,\tilde{y};\mathcal{S}_x^+)\big) \tag{3}$$

This gating ensures that invalid rewrites ($r_{\text{task}} = 0$) receive zero reward and that auxiliary rewards are skipped when infeasible, preventing misleading shaping signals while improving training stability and computational efficiency.

3.4.1 Task Consistency Reward

The task consistency reward $r_{\text{task}}(x, y^\star, \tilde{y})$ is a binary gate that checks whether a rewritten sample $\tilde{y}$ satisfies: (i) final-answer correctness and (ii) reasoning validity. We adopt a coarse-to-fine verification scheme: we first use a low-cost, rule-based verifier to check the final answer; only when the answer is deemed correct do we invoke a stronger *LLM-as-a-judge* to assess reasoning consistency and logical soundness, avoiding unnecessary evaluation on clearly incorrect samples and reducing cases where the answer is correct but the reasoning is unreliable (Zheng et al., 2023; Liu et al., 2023). Formally, we define

$$v_{\text{ans}}(x,y^\star,\tilde{y}) \in \{0,1\}, v_{\text{rea}}(x,y^\star,\tilde{y}) \in \{0,1\}, \tag{4}$$

where v_{ans} denotes the rule-based judgment of final-answer correctness, and v_{rea} denotes the LLM judge's assessment of reasoning validity (computed only when $v_{\text{ans}} = 1$). We define the task-consistency reward as

$$r_{\text{task}}(x,y^\star,\tilde{y}) = v_{\text{ans}}(x,y^\star,\tilde{y}) \cdot v_{\text{rea}}(x,y^\star,\tilde{y}) \in \{0,1\}. \tag{5}$$

3.4.2 Distributional Alignment Reward

To encourage rewrites that are in-distribution under standard QA-style generation, we score each rewrite using the frozen base model under the QA condition $\pi_0(\cdot \mid x)$ and reward higher generatability ($i.e.$ higher likelihood) under this distribution. Specifically, we compute the length-normalized negative log-likelihood (NLL) of $\tilde{y}$:

$$\ell_{\text{dist}}(x,\tilde{y}) = -\frac{1}{|\tilde{y}|}\sum_{t=1}^{|\tilde{y}|} \log \pi_0(\tilde{y}_t \mid x, \tilde{y}_{<t}). \tag{6}$$

We then map it to a bounded, monotonic reward via group-wise normalization. Specifically, for each input x, we compute the mean and standard deviation of ℓ_{dist} over the feasible candidate set $\mathcal{S}_x^+$, denoted as μ_x and σ_x, and define the normalized score

$$\hat{\ell}_{\text{dist}}(x,\tilde{y}) = \frac{\ell_{\text{dist}}(x,\tilde{y}) - \mu_x}{\sigma_x + \epsilon}, \tag{7}$$

where ϵ is a small constant for numerical stability. We then apply a bounded, monotonic mapping:

$$r_{\text{dist}}(x, \tilde{y}) = \frac{1}{1 + \exp\Big(\hat{\ell}_{\text{dist}}(x, \tilde{y})\Big)}. \quad (8)$$

This group-wise normalization improves numerical resolution by removing prompt-dependent scale in NLL values, while preserving the within-group ordering. It is also aligned with common stability practices in GRPO-style optimization, where reward whitening is used to stabilize learning dynamics. Consequently, the policy is encouraged to generate rewrites that are more in-distribution under the base model's QA-style generation $\pi_0(\cdot \mid x)$.

3.4.3 Diversity Reward

To mitigate mode collapse and template-induced homogeneity, we encourage *semantic diversity* among *feasible* (task-consistent) rewrites for the same input x, and compute diversity only on $\mathcal{S}_x^+$ to avoid noise from invalid samples. For brevity, when $(x, y^\star)$ is clear from context, we write $r_{\text{task}}(\tilde{y}) = r_{\text{task}}(x, y^\star, \tilde{y})$, and let $m = |\mathcal{S}_x^+|$. When $m < 2$, semantic diversity is ill-defined, so we set $r_{\text{div}} = 0$; when $m \geq 2$, we map each $\tilde{y}^{(k)} \in \mathcal{S}_x^+$ into a semantic space with an embedding function $f(\cdot)$ and normalize it as $e^{(k)} = \frac{f(\tilde{y}^{(k)})}{\|f(\tilde{y}^{(k)})\|}$. We define the pairwise semantic distance as

$$d\Big(e^{(i)}, e^{(j)}\Big) = \frac{1 - \cos\big(e^{(i)}, e^{(j)}\big)}{2} \in [0, 1]. \quad (9)$$

The set-level semantic diversity is

$$D(\mathcal{S}_x^+) = \frac{2}{m(m-1)} \sum_{1 \leq i < j \leq m} d\Big(e^{(i)}, e^{(j)}\Big). \quad (10)$$

To provide fine-grained credit assignment, we define a *marginal contribution* diversity reward for each feasible candidate. For $\tilde{y}^{(k)} \in \mathcal{S}_x^+$, let $\mathcal{S}_{x,-k}^+ = \mathcal{S}_x^+ \setminus \{\tilde{y}^{(k)}\}$, and define its marginal contribution as

$$\Delta^{(k)}(\mathcal{S}_x^+) = D(\mathcal{S}_x^+) - D(\mathcal{S}_{x,-k}^+). \quad (11)$$

Accordingly, the diversity reward is

$$r_{\text{div}}^{(k)} = \mathbb{1}\Big[r_{\text{task}}(\tilde{y}^{(k)}) = 1\Big] \cdot \big[\Delta^{(k)}(\mathcal{S}_x^+)\big]_+, \quad (12)$$

where $[z]_+ = \max(0, z)$. This marginal formulation encourages each feasible rewrite to contribute novel semantic content to the candidate set, directly discouraging template-induced homogeneity while avoiding spurious diversity signals from invalid rewrites.

3.5 Rewriting Dataset Construction and Downstream SFT

After training the rewriting policy R_ϕ, we construct the rewritten dataset via a GENERATE–VERIFY–FALLBACK pipeline. For each expert sample $(x, y^\star) \in \mathcal{D}_E$, we first sample a rewrite

$$\tilde{y} \sim R_\phi(\cdot \mid x, y^\star, x_{\text{prompt}}). \quad (13)$$

We then re-apply the same task-consistency check used during training, $r_{\text{task}}(x, y^\star, \tilde{y})$. If the check passes ($r_{\text{task}} = 1$), we adopt $\tilde{y}$ as the supervision target; otherwise ($r_{\text{task}} = 0$), we fall back to the original expert demonstration $y^\star$ to avoid losing supervision due to rewriting failures. Formally, we set

$$\hat{y} = \begin{cases} \tilde{y}, & r_{\text{task}}(x, y^\star, \tilde{y}) = 1, \\ y^\star, & \text{otherwise.} \end{cases} \quad (14)$$

Collecting $\{(x_i, \hat{y}_i)\}_{i=1}^N$ yields the rewritten training set $\mathcal{D}_R$. Finally, we perform standard supervised fine-tuning of the target policy π_θ (initialized from π_0) on $\mathcal{D}_R$ using maximum likelihood training, obtaining the final model.

4 Experiments

4.1 Experimental Setup

Following the Mind the Gap (Zhao et al., 2025) setting, we treat mathematical reasoning as the downstream capability to improve, and quantify catastrophic forgetting via performance changes on general-domain benchmarks.

Datasets. For training data, we use two widely adopted math corpora, `NuminaMath-CoT` (LI et al., 2024) and `OpenMathReasoning` (Moshkov et al., 2025). We randomly sample from the merged pool, filter out overlong examples (> 8192 tokens), and obtain 100K instances in total, split evenly into 50K for GRPO-based rewriting-agent training and 50K for downstream SFT. For downstream math evaluation, we report results on `Math500` (Lightman et al., 2023), `MinervaMath` (Lewkowycz et al., 2022), `AMC23` (Yao et al., 2025), `AGIEval-Math` (Zhong et al., 2023), and `IMO-Bench` (Luong et al., 2025). To assess out-of-domain generalization and quantify catastrophic forgetting, we further evaluate on `MMLU` (Hendrycks et al., 2021b,a), `MMLU-Pro` (Wang et al., 2024b), and `AGIEval` (Zhong et al., 2023) with math-related subsets removed. Detailed dataset descriptions are provided in Appendix C.

Model	MathAvg	Math↑ (%)	GeneralAvg	Gen↓ (%)	OverallAvg
Llama-3.2-1B-Instruct	12.42	–	27.34	–	19.88
+ SFT	15.40	↑**23.99**	22.60	↓17.34	19.00
+ SDFT	13.80	↑11.11	23.80	↓12.95	18.80
+ Mind the GAP	12.92	↑4.03	24.41	↓10.72	18.67
+ Rewriting-agent	15.10	↑21.58	25.92	↓**5.19**	**20.51**
Llama-3.2-3B-Instruct	18.90	–	45.21	–	32.06
+ SFT	21.81	↑**15.40**	37.26	↓17.58	29.54
+ SDFT	19.84	↑4.97	40.32	↓10.82	30.08
+ Mind the GAP	20.30	↑7.41	40.44	↓10.55	30.37
+ Rewriting-agent	21.12	↑11.75	43.17	↓**4.51**	**32.15**
Mistral-7B-Instruct-v0.3	10.61	–	40.53	–	25.57
+ SFT	16.36	↑54.19	32.76	↓19.17	24.56
+ SDFT	13.77	↑29.78	35.36	↓12.76	24.57
+ Mind the GAP	14.60	↑37.61	34.43	↓15.05	24.52
+ Rewriting-agent	16.47	↑**55.23**	37.55	↓**7.35**	**27.01**

Table 1: Math↑ denotes the relative MathAvg improvement over the instruct-tuned base within the same block: $(M - M_{\text{base}})/M_{\text{base}} \times 100$. Gen↓ denotes the relative GeneralAvg drop: $(G_{\text{base}} - G)/G_{\text{base}} \times 100$. OverallAvg $= (\text{MathAvg} + \text{GeneralAvg})/2$.

Backbone Models. We experiment with instruction-tuned backbones of different scales and training recipes: `Llama-3.2-1B-Instruct`, `Llama-3.2-3B-Instruct` (Grattafiori et al., 2024) and `Mistral-7B-Instruct-v0.3` (Jiang et al., 2023). This allows us to test whether the proposed rewriting-agent framework is robust across model sizes and families.

Baselines. Under the same backbone π_0, the same task-consistency verifier, and identical downstream SFT hyperparameters and data split, we compare: (1)`Vanilla SFT` (SFT on the original corpus), (2)`SDFT` (Yang et al., 2024) (template-based rewrite + verify + fallback), (3)`Mind the Gap` (Zhao et al., 2025) (self-solve, then rewrite failed cases + verify + fallback), and (4)`Rewriting-agent (Ours)` (learned policy rewriting + the same verify/fallback rule). Detailed descriptions are provided in Appendix D.

Training Details. For downstream SFT, we use `LLaMA-Factory` (Zheng et al., 2024) with the AdamW optimizer, a global batch size of 128, and fine-tune for 2 epochs. We use a learning rate of 5×10^{-6} for `Mistral-7B-Instruct-v0.3`, and 7×10^{-6} for both `Llama-3.2-1B-Instruct` and `Llama-3.2-3B-Instruct`. For training the rewriting agent, we implement GRPO optimization with `verl` (Sheng et al., 2024). We freeze the backbone π_0 and update only the LoRA adapter parameters. We use a global batch size of 512 prompts and sample K=10 candidate rewrites per prompt. Further details are provided in Appendix E.

4.2 Main Results

Table 1 summarizes the trade-off between downstream math gains and out-of-distribution generalization retention across three instruction-tuned backbones. Overall, `Vanilla SFT` yields the largest math gains, but it also incurs the most severe degradation on general-domain benchmarks, indicating pronounced catastrophic forgetting. Fixed-prompt self-rewriting baselines (`SDFT` and `Mind the Gap`) partially mitigate this drop, but achieve smaller math improvements, suggesting that heuristic prompting alone is insufficient to jointly optimize task improvement and distributional alignment.

In contrast, our `Rewriting-agent` achieves the best OverallAvg across all three backbones and consistently improves the gain–forgetting trade-off: it attains math gains broadly comparable to `Vanilla SFT` while exhibiting substantially smaller generalization drops. For example, on `Mistral-7B-Instruct-v0.3`, `Rewriting-agent` matches or slightly exceeds SFT in math gains ($+55.23\%$ vs. $+54.19\%$) while reducing the generalization drop from 19.17% to 7.35%. Overall, these results demonstrate the effectiveness of our method in improving the gain–forgetting trade-off: learning to rewrite supervision under explicit objectives of task consistency, QA-style generatabil-

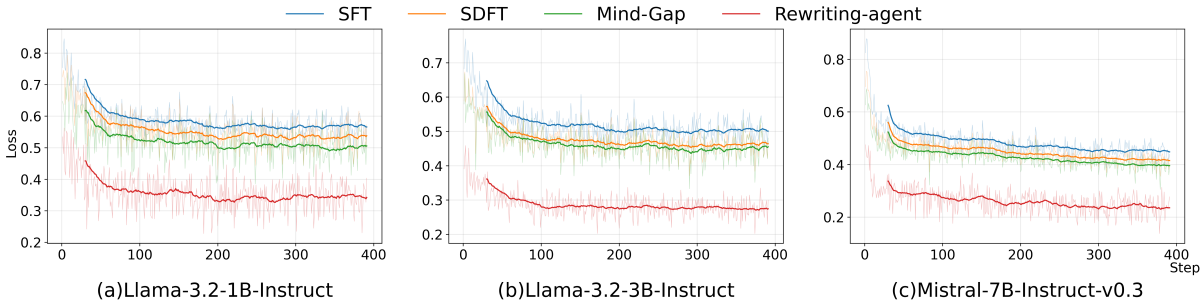

Figure 2: Downstream SFT training loss over training steps for three backbones and four methods.

Method	Llama-1B	Llama-3B	Mistral-7B
`SDFT`	53.39	66.33	69.28
`Mind the Gap`	53.47	67.42	69.74
`Rewriting-agent`	60.19	74.96	77.43

Table 2: Task-consistency yield (%): fraction of rewrites passing r_{task} for each method and backbone.

Variant	MathAvg	GeneralAvg	TC-Yield
Full (ours)	21.12	43.14	74.96
w/o r_{dist}	20.73	39.37	74.58
w/o r_{div}	20.18	40.93	75.34
task-only	20.07	39.23	78.25

Table 3: Reward-component ablations of the Stage I GRPO objective (TC-Yield in %).

ity, and diversity mitigates over-shifting and catastrophic forgetting with little to no sacrifice in downstream gains. Detailed per-benchmark results are reported in Appendix G.1.

4.3 Analysis

Why does the Rewriting-agent reduce forgetting? We attribute the improved retention to the stronger alignment between our rewritten data and the backbone's QA-style generation distribution. Specifically, Figure 2 shows that, across backbones, downstream SFT on `Rewriting-agent` targets starts from a noticeably lower training loss and converges to a lower and more stable plateau, outperforming `SFT` and fixed-prompt self-rewriting baselines (`SDFT` and `Mind the Gap`). This suggests that the rewritten data are easier to fit under the standard QA-style maximum-likelihood objective, which is consistent with more stable optimization and reduced reliance on abrupt, large-magnitude updates; such updates can over-shift the policy toward the downstream distribution and exacerbate catastrophic forgetting.

Task-consistency yield. We report the pass rate of the task-consistency gate r_{task} during rewriting in Table 2. Compared to fixed-prompt rewriting, the learned rewriting policy substantially increases the fraction of feasible rewrites, thereby reducing reliance on fallback to the original expert demonstrations. As a result, the constructed rewritten dataset $\mathcal{D}_R$ exhibits higher supervision quality and stronger distributional consistency, which in turn improves generalization retention and leads to less catastrophic forgetting.

4.4 Ablation Study

We conduct an ablation study on `Llama-3.2-3B-Instruct` to quantify the contribution of each key component in our rewriting-agent framework. Detailed per-benchmark ablation results for all variants are reported in Appendix G.2 (Table 7).

Effect of reward components. We ablate reward components in the Stage I GRPO objective while keeping the backbone π_0, task-consistency verifier, data split, and downstream SFT hyperparameters fixed (Table 3). Removing the distributional-alignment reward r_{dist} causes the largest drop in generalization retention, indicating that encouraging QA-style generatability under $\pi_0(\cdot \mid x)$ is crucial for mitigating forgetting. Removing the diversity regularizer r_{div} also degrades performance, with a more pronounced impact on in-domain math results, suggesting complementary regularization beyond feasibility and alignment. Notably, `task-only` attains the highest pass rate yet performs worst overall, showing that feasibility alone is insufficient without additional shaping toward in-distribution targets.

Success-only supervision quality. We conduct downstream SFT on the *success-only* subset

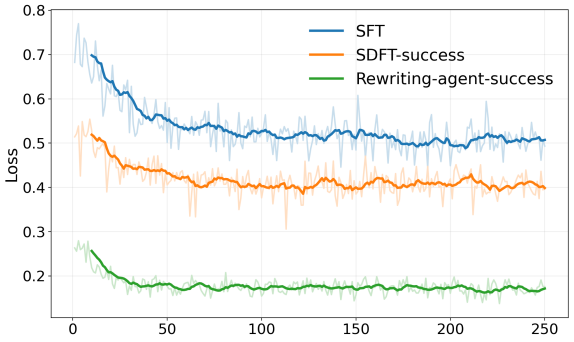

Figure 3: Downstream SFT training loss on the success-only subset for `Llama-3.2-3B-Instruct`.

Construction	Nums	MathAvg	GeneralAvg
Fallback (default)	50000	21.12	43.17
Success-only	37483	20.30	43.70

Table 4: Ablation of fallback in dataset construction on `Llama-3.2-3B-Instruct`. Num. denotes the number of training instances used for downstream SFT.

Variant	MathAvg	GeneralAvg	TC-Yield
Hard gate (ours)	21.12	43.14	74.96
Soft shaping	19.52	42.96	71.58

Table 5: Hard gating vs. soft shaping on `Llama-3.2-3B-Instruct`.

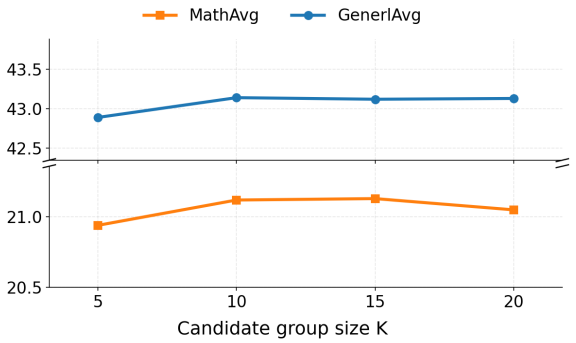

Figure 4: Effect of candidate group size K in GRPO training of the rewriting agent on `Llama-3.2-3B-Instruct`.

and report the training loss in Figure 3, enabling a controlled comparison between rewrites that are successfully generated by our rewriting agent and those produced via successful self-rewriting (i.e., `SDFT`). Figure 3 shows that `Rewriting-agent-success` converges to a lower and smoother loss than `SDFT-success`, suggesting that our learned rewrites are more in-distribution and thus easier to fit under the QA-style maximum-likelihood objective. We provide qualitative case studies in Appendix F, showing that although the model's direct rewrites (`SDFT`) are also task-correct, our learned rewrites better match the model's natural QA-style completion patterns, thereby yielding a more stable and effective training signal.

Generate–Verify–Fallback. We ablate the fallback step by training on *success-only* rewrites and compare it with the default GENERATE–VERIFY–FALLBACK construction. As shown in Table 4, *success-only* preserves general performance but limits downstream math gains due to reduced supervision coverage.

Hard gate vs. soft shaping. We compare the proposed hard gating with a soft variant that always applies auxiliary rewards (Table 5). Soft shaping reduces PassGate (74.96→71.58) and substantially hurts MathAvg (21.12→19.52), suggesting that applying auxiliary rewards to incorrect candidates can introduce noisy shaping signals. Hard gating yields a better overall trade-off by restricting auxiliary rewards to feasible rewrites.

Candidate group size K. We sweep $K \in \{5, 10, 15, 20\}$ in GRPO training (Figure 4). Larger K generally improves performance but shows diminishing returns; we use K=10 by default as a practical trade-off.

5 Conclusion

Downstream supervised fine-tuning (SFT) is an effective and practical approach for adapting instruction-tuned large language models to target tasks, yet under distribution shift, a mismatch between expert demonstrations and the model's natural generation distribution can induce optimization instability and catastrophic forgetting. To address this, we propose and learn an RL-based supervision-rewriting policy, implemented as lightweight LoRA patches on a frozen backbone and trained with GRPO using hard-gated, automatically evaluable reward signals, to produce supervision targets that are task-consistent, closer to the backbone's QA-style distribution, and diverse. Across multiple backbones, our method achieves downstream performance broadly comparable to standard SFT while substantially reducing degradation on general-domain benchmarks, improving the gain–retention trade-off and effectively mitigating catastrophic forgetting.

Limitations

Our study has several limitations. First, due to computational constraints, we evaluate the proposed rewriting-agent on small to mid-scale instruction-tuned backbones (up to 7B parameters). Extending the framework to larger models and longer-context settings may require additional engineering and compute. Second, our evaluation focuses on mathematical reasoning as the downstream domain and measures generalization retention primarily via math-removed general benchmarks. While these settings follow prior work, the conclusions may not fully generalize to other downstream domains (e.g., code, dialogue, or safety-critical applications) or to alternative retention metrics. Third, our reward design relies on automatically evaluable signals and a task-consistency verifier. Although this enables scalable training, the verifier and reward heuristics may be imperfect and could bias the rewriting behavior toward what is easiest to verify. Finally, we adopt a specific dataset-construction pipeline (verification with fallback). Other construction choices (e.g., different selection strategies or multi-rewrite retention) may further affect the gain–forgetting trade-off and warrant future investigation.

References

Anonymous. 2025. Off-policy token clipped supervised fine-tuning yields a robust cold-start. OpenReview. OpenReview submission.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. 2025. Retaining by doing: The role of on-policy data in mitigating forgetting. *Preprint*, arXiv:2510.18874.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. 2024. Loss of plasticity in deep continual learning. *Nature*, 632:768–774.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. How abilities in large language models are affected by supervised fine-tuning data composition. *Preprint*, arXiv:2310.05492.

Jun Wang · Liang Ding · Shuai Wang · Hongyu Li · Yong Luo · Huangxuan Zhao · Han Hu · Bo Du. 2025. Self-evolving pseudo-rehearsal for catastrophic forgetting with task similarity in llms. NeurIPS 2025 Virtual Poster.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. 2024a. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. *Preprint*, arXiv:2403.01244.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. 2024b. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1416–1428, Bangkok, Thailand. Association for Computational Linguistics.

Yuqing Huang, Rongyang Zhang, Qimeng Wang, Chengqiang Lu, Yan Gao, Yiwu, Yao Hu, Xuyang Zhi, Guiquan Liu, Xin Li, Hao Wang, and Enhong Chen. 2025. SelfAug: Mitigating catastrophic forgetting in retrieval-augmented generation via distribution self-alignment. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 14175–14190, Suzhou, China. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Jiazheng Kang, Le Huang, Cheng Hou, Zhe Zhao, Zhenxiang Yan, and Ting Bai. 2025. Self-evolving llms via continual instruction tuning. *Preprint*, arXiv:2509.18133.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. 2024. Understanding catastrophic forgetting in language models via implicit inference. *Preprint*, arXiv:2309.10105.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *Preprint*, arXiv:2309.06180.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models.

Hongyu Li, Liang Ding, Meng Fang, and Dacheng Tao. 2024. Revisiting catastrophic forgetting in large language model tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4297–4308, Miami, Florida, USA. Association for Computational Linguistics.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf).

Ziniu Li, Congliang Chen, Tian Xu, Zeyu Qin, Jiancong Xiao, Zhi-Quan Luo, and Ruoyu Sun. 2024. Preserving diversity in supervised fine-tuning of large language models. *Preprint*, arXiv:2408.16673. Accepted by ICLR 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. *Preprint*, arXiv:2303.16634.

David Lopez-Paz and Marc'Aurelio Ranzato. 2022. Gradient episodic memory for continual learning. *Preprint*, arXiv:1706.08840.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2025. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *Preprint*, arXiv:2308.08747.

Thang Luong, Dawsen Hwang, Hoang H. Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Clara Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu H. Trinh, Quoc V. Le, and Junehyuk Jung. 2025. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. 2025. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. *arXiv preprint arXiv:2504.16891*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

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

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

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, and 22 others. 2024. Beyond human data: Scaling self-training for problem-solving with language models. *Preprint*, arXiv:2312.06585.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2024. Is chatgpt good at search? investigating large language models as re-ranking agents. *Preprint*, arXiv:2304.09542.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Yifan Wang, Yafei Liu, Chufan Shi, Haoling Li, Chen Chen, Haonan Lu, and Yujiu Yang. 2024a. InsCL:

A data-efficient continual learning paradigm for fine-tuning large language models with instructions. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 663–677, Mexico City, Mexico. Association for Computational Linguistics.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. 2025. On the generalization of sft: A reinforcement learning perspective with reward rectification. *Preprint*, arXiv:2508.05629.

Jiancong Xiao, Ziniu Li, Xingyu Xie, Emily Getzen, Cong Fang, Qi Long, and Weijie Su. 2025. On the algorithmic bias of aligning large language models with rlhf: Preference collapse and matching regularization. *Journal of the American Statistical Association*, 120(552):2154–2164.

Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. 2024. Self-distillation bridges distribution gap in language model fine-tuning. *Preprint*, arXiv:2402.13669.

Jiarui Yao, Ruida Wang, and Tong Zhang. 2025. Fans – formal answer selection for natural language math reasoning using lean4. *Preprint*, arXiv:2503.03238.

Longfei Yun, Chenyang An, Zilong Wang, Letian Peng, and Jingbo Shang. 2025. The price of format: Diversity collapse in LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 15454–15468, Suzhou, China. Association for Computational Linguistics.

Dylan Zhang, Qirun Dai, and Hao Peng. 2025a. The best instruction-tuning data are those that fit. *Preprint*, arXiv:2502.04194.

Jiayi Zhang, Simon Yu, Derek Chong, Anthony Sicilia, Michael R. Tomz, Christopher D. Manning, and Weiyan Shi. 2025b. Verbalized sampling: How to mitigate mode collapse and unlock llm diversity. *Preprint*, arXiv:2510.01171.

Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2025c. On-policy rl meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting. *Preprint*, arXiv:2508.11408.

Shiwan Zhao, Xuyang Zhao, Jiaming Zhou, Aobo Kong, Qicheng Li, and Yong Qin. 2025. Mind the gap: Data rewriting for stable off-policy supervised fine-tuning. *Preprint*, arXiv:2509.15157.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. 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. Association for Computational Linguistics.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *Preprint*, arXiv:2304.06364.

He Zhu, Junyou Su, Peng Lai, Ren Ma, Wenjia Zhang, Linyi Yang, and Guanhua Chen. 2025a. Anchored supervised fine-tuning. *Preprint*, arXiv:2509.23753.

Wenhong Zhu, Ruobing Xie, Rui Wang, Xingwu Sun, Di Wang, and Pengfei Liu. 2025b. Proximal supervised fine-tuning. *Preprint*, arXiv:2508.17784.

A The use of Large Language Models(LLMs)

In the preparation of this work, we used LLMs as auxiliary tools in a limited capacity. Specifically, LLMs assisted in drafting portions of the code and in refining the wording of certain sentences for clarity and readability. All technical content, including the design of algorithms, experimental methodology, analysis, and interpretations, was independently developed by the authors. The use of LLMs was confined to language refinement and coding suggestions, and did not influence the scientific contributions or results reported in this paper.

B Why Standard SFT Can Be Unstable

Off-policy view and an importance-weighted form. Standard SFT minimizes the sequence-level negative log-likelihood on an expert dataset $\mathcal{D}_E = \{(x, y^\star)\}$:

$$\mathcal{L}_{\text{SFT}}(\theta) = \mathbb{E}_{(x,y^\star)\sim\mathcal{D}_E}[-\log \pi_\theta(y^\star \mid x)], \quad (15)$$

whose gradient is

$$\nabla_\theta \mathcal{L}_{\text{SFT}}(\theta) = \mathbb{E}_{(x,y^\star)\sim\mathcal{D}_E}[-\nabla_\theta \log \pi_\theta(y^\star \mid x)]\,. \tag{16}$$

This expectation is taken under a fixed expert behavior distribution rather than under on-policy sampling from the current model $\pi_\theta(\cdot \mid x)$, hence SFT can be viewed as an off-policy learning signal.

Following prior work (Wu et al., 2025), the same gradient can be written as an expectation under the current policy via an importance-weighted "resample + reweight" form. Let $\mathcal{D}_x$ denote the empirical marginal over inputs induced by $\mathcal{D}_E$. For each x, sample $y \sim \pi_\theta(\cdot \mid x)$; then

$$\mathbb{E}_{(x,y^\star)\sim\mathcal{D}_E}[-\nabla_\theta \log \pi_\theta(y^\star \mid x)] = \mathbb{E}_{x\sim\mathcal{D}_x,\, y\sim\pi_\theta(\cdot|x)}\left[\frac{\mathbb{1}[y = y^\star]}{\pi_\theta(y \mid x)}\left(-\nabla_\theta \log \pi_\theta(y \mid x)\right)\right]. \tag{17}$$

Intuitively, the indicator $\mathbb{1}[y = y^\star]$ keeps only the exact expert trajectory among on-policy samples, while the inverse-probability factor $1/\pi_\theta(y \mid x)$ reweights the remaining samples to match the expert expectation (closely related to inverse propensity scoring in off-policy learning).

Key issue: inverse-probability amplification and high variance. The crucial implication of the above form is the implicit inverse-probability factor $1/\pi_\theta(y^\star \mid x)$. When π_θ assigns extremely low probability to the expert demonstration $y^\star$, this factor becomes large, excessively amplifying the corresponding update and yielding high-variance, potentially unstable optimization dynamics. This is a well-known failure mode of importance weighting/off-policy estimation: small denominator probabilities lead to heavy-tailed weights and unstable learning signals (e.g., inverse propensity weighting) (**??**).

Connection to policy drift and catastrophic forgetting (token-level intuition). Expanding the log-likelihood in an autoregressive, token-level form,

$$\log \pi_\theta(y^\star \mid x) = \sum_{t=1}^{|y^\star|} \log \pi_\theta(y_t^\star \mid x, y_{<t}^\star)\,, \tag{18}$$

reveals that an SFT step aggregates gradients over all positions. If the expert sequence contains tokens that are very low-likelihood under the current model (especially early in fine-tuning under substantial distribution shift), those positions can dominate the gradient, producing optimization instability (e.g., more frequent or larger loss spikes / gradient fluctuations) and inducing stronger policy drift toward the downstream distribution. Such over-shifting can come at the expense of previously acquired general capabilities, thereby exacerbating catastrophic forgetting (Zhang et al., 2025c; Zhu et al., 2025a).

Relevance to our approach. Motivated by this off-policy perspective, our method intervenes at the data source: we learn a rewriting policy that adjusts supervision targets to be (i) task-consistent and (ii) more generatable under the backbone's QA-style distribution $\pi_0(\cdot \mid x)$. By reducing low-likelihood supervision targets under QA-style generation, downstream SFT is less exposed to implicitly amplified, high-variance updates, which helps stabilize optimization and mitigates forgetting.

C Dataset Details

C.1 Training Corpora

NuminaMath-CoT. `NuminaMath-CoT` is a large-scale math reasoning corpus consisting of diverse competition- and school-level problems paired with chain-of-thought (CoT) solutions. Its sources span Chinese K-12 exercises, AMC/AIME-style contests, and international Olympiad problems, collected primarily from online exam PDFs and mathematics discussion forums. The dataset undergoes OCR, segmentation into problem–solution pairs, translation into English, and post-processing to align solutions into a CoT format with standardized final answers. Each instance provides the problem statement and an English CoT solution (with a final answer), together with a coarse-grained source tag. In our experiments, we sample from this corpus and apply a length filter to exclude overlong instances.

OpenMathReasoning. `OpenMathReasoning` is a large-scale math reasoning dataset built primarily from AoPS problems. It contains multiple supervision modalities, including long CoT solutions and tool-integrated reasoning (TIR) solutions, as well as a selection set that chooses the most promising solution among candidates. Problem statements are preprocessed for quality, and solutions are synthesized by strong open models. We treat `OpenMathReasoning` as complementary training data to `NuminaMath-CoT` due to its high coverage of contest-style problems and its diverse solution

traces. In our setup, we merge the two corpora, randomly sample instances, and filter out examples whose tokenized length exceeds the model context limit.

C.2 Math Evaluation Benchmarks

Math500. `Math500` is a curated 500-problem subset of the `MATH` benchmark. It features competition-style questions across multiple subjects (e.g., algebra, number theory, geometry, precalculus) and difficulty levels. Each example provides a problem statement and a reference solution with a final answer string. We evaluate by extracting the model's final answer and matching it against the reference answer under our normalization rules.

MinervaMath. `MinervaMath` targets advanced quantitative reasoning problems at the undergraduate level, covering broad STEM topics (e.g., physics/astronomy, basic engineering-style calculations, and mathematical modeling). Problems typically require multi-step derivations and yield a short verifiable final answer (number or expression). We use the official test split and report answer accuracy based on final-answer extraction.

AMC23. `AMC23` consists of problems from the 2023 AMC (American Mathematics Competitions), which are short, competition-style questions with concise numeric answers. Compared with larger math benchmarks, AMC problems emphasize algebraic manipulation, counting/probability, and geometry in a compact format. We report accuracy using final-answer matching.

AGIEval-Math. `AGIEval` is a suite of standardized-exam questions spanning multiple subjects and languages. We define `AGIEval-Math` as the math-related subsets in `AGIEval`, including SAT-style math, AQuA-RAT quantitative reasoning, and Gaokao math in both QA and cloze-style formats, as well as the `MATH` subset included by AGIEval. This benchmark mixes multiple-choice and open-form (cloze) questions; we follow the standard evaluation protocol for each subset and compute overall accuracy.

IMO-Bench (Answer subset). `IMO-Bench` targets Olympiad-level reasoning. Since proof grading is expensive and often requires expert assessment, we focus on its answer-verifiable component (often referred to as *IMO-AnswerBench*), which contains Olympiad problems with short, checkable final answers. The problems are curated to cover diverse topics (algebra, combinatorics, geometry, number theory) and to reduce memorization via expert editing/robustification. We report final-answer accuracy under the same extraction and normalization pipeline as other open-answer math benchmarks.

C.3 General-Domain Benchmarks for Forgetting

MMLU (math removed). `MMLU` is a general-domain multiple-choice benchmark covering a broad set of academic subjects. To quantify catastrophic forgetting outside the math domain, we remove the math-related subjects (e.g., abstract algebra, college mathematics, elementary mathematics, high school mathematics, and high school statistics) and evaluate on the remaining subjects using the standard multiple-choice protocol.

MMLU-Pro (math removed). `MMLU-Pro` is a more challenging and carefully curated extension of `MMLU`, designed to reduce shortcut artifacts and increase difficulty. We exclude the mathematics domain (and any explicitly math-labeled subsets) and report accuracy on the remaining domains using the standard multiple-choice protocol.

AGIEval (math removed). To measure general-domain transfer beyond math within the `AGIEval` suite, we exclude the math-related subsets used to form `AGIEval-Math` and evaluate on the remaining exam tasks (primarily multiple-choice) following the official evaluation procedure.

D Baseline Details

Common setup. All baselines use the same backbone model π_0, the same task-consistency verifier, the same training/evaluation split, and identical downstream SFT hyperparameters. Let each training instance be $(x, y^\star)$, where x is the instruction/input and $y^\star$ is the expert demonstration. For methods involving rewriting, we construct a training target y via a unified *verify–fallback* rule:

$$y = \begin{cases} \tilde{y}, & \text{if } \mathrm{Verify}(x, y^\star, \tilde{y}) = 1, \\ y^\star, & \text{otherwise,} \end{cases}$$

where $\tilde{y}$ denotes the generated rewrite/candidate output.

`Vanilla SFT`. We directly perform supervised fine-tuning on the original corpus, i.e., $y = y^\star$ for every instance. No rewriting is applied.

SDFT (Yang et al., 2024). For each instance, we prompt the frozen backbone π_0 with a fixed rewriting template to produce a candidate rewrite $\tilde{y}$ from $(x, y^\star)$. We then apply the same task-consistency verifier. Verified rewrites are used as SFT targets; otherwise we fall back to the original expert demonstration $y^\star$.

Mind the Gap (Zhao et al., 2025). We first prompt π_0 to *self-solve* the input x, producing a solution $\hat{y}$. If $\hat{y}$ passes verification, we keep it as the SFT target. Otherwise, we rewrite the expert demonstration using a fixed rewriting template to obtain $\tilde{y}$, verify it, and fall back to $y^\star$ if verification still fails. This procedure rewrites supervision only for the instances where self-solving fails.

Rewriting-agent (Ours). We replace template-based rewriting with a learnable rewriting policy R_ϕ that generates a candidate rewrite $\tilde{y} = R_\phi(x, y^\star)$ (with π_0 as the backbone). We then construct the rewriting dataset using the same verify–fallback rule above and perform downstream SFT on the resulting targets. Compared to template-based baselines, this keeps the verification and fallback mechanism fixed while changing only the rewriting policy.

E Training Details

E.1 Downstream SFT

We perform downstream supervised fine-tuning (SFT) with `LLaMA-Factory` (Zheng et al., 2024) using the AdamW optimizer and bfloat16 mixed precision. Unless otherwise specified, we use the default AdamW hyperparameters (i.e., β_1=0.9, β_2=0.999, ϵ=10^{-8}), apply gradient clipping with max norm 1.0, and set weight decay to 0.0. For all backbones, we fine-tune for 2 epochs with a per-device batch size of 4 and gradient accumulation of 8, resulting in a global batch size of 128 when using 4 GPUs. We adopt a cosine learning-rate schedule with a warmup ratio of 0.1. We use a learning rate of 5×10^{-6} for `Mistral-7B-Instruct-v0.3` and 7×10^{-6} for both `Llama-3.2-1B-Instruct` and `Llama-3.2-3B-Instruct`. To support full-parameter fine-tuning under limited GPU memory, we enable DeepSpeed ZeRO-3 with the configuration file `ds_z3_config.json` (provided in the `LLaMA-Factory` examples), which shards optimizer states, gradients, and model parameters across GPUs. We run ZeRO-3 without CPU offloading. Training is executed with distributed data parallelism; we set a sufficiently large DDP timeout to avoid premature termination in long-sequence runs.

E.2 Rewriting-agent training (GRPO + LoRA)

We train the rewriting agent with `verl`, an RL training framework for LLMs. The instruction-tuned backbone π_0 is frozen, and we parameterize the rewriting policy R_ϕ using LoRA patch adapters applied to all linear layers, with rank r=64 and α=32. We optimize only the LoRA parameters using GRPO-style policy optimization. For GRPO training, we use a global batch of 512 prompts and sample K candidate rewrites per prompt for group-relative updates (default K=10, unless explicitly swept). We cap the maximum prompt length and response length to 2048 and 4096 tokens, respectively, and filter or truncate overlong prompts following the `verl` data pipeline.

To accelerate rollouts and log-probability computation, we use a `vLLM`-based rollout engine during GRPO training, with conservative GPU memory utilization to stabilize long-sequence decoding. Reward computation uses our custom automatically evaluable signals (task-consistency gate, QA-style generatability under $\pi_0(\cdot \mid x)$, and within-group diversity) as described in Sec. 3.4. For embedding-based similarity computations in the reward (e.g., the diversity term), we use the `Qwen-Embedding-0.6B` model.

E.3 Rewriting inference with vLLM

To generate rewritten datasets from the trained agent, we perform batched decoding with `vLLM` (Kwon et al., 2023) using tensor parallelism over 4 GPUs. We use greedy decoding for determinism (temperature $=$ 0.0, top_p $=$ 1.0) and allow up to 8192 new tokens per output, without specifying an explicit stop sequence. We use a batch size of 36 for generation and do not apply retries for failed rewrites (failed cases are handled by verification and fallback during dataset construction).

E.4 Prompt Templates

We use those prompt templates for rewriting, math evaluation, and general multiple-choice evaluation.

Rewriting Prompt

```
You are an expert in math word problems.
    Below is a math word problem and an
    existing step-by-step solution.
Please rewrite the solution in your own
    words while keeping all reasoning steps
    correct and keeping the final answer the
     same.
Follow these rules:
- Keep the explanation clear and step by
    step.
- Do NOT mention that you are rewriting
    another solution.
- At the end, on a separate line, output
    ONLY the final answer in LaTeX boxed
    format, exactly like:
$\boxed{56}$
Use dollar signs and \boxed{} exactly as
    shown.

Problem:
{question}

Existing solution (for reference, do NOT
    copy it verbatim):
{original_solution}
```

Math Evaluation Prompt

```
You are an expert in math word problems.
Read the following problem and solve it step
     by step.
At the end, on a separate line, output ONLY
    the final answer in LaTeX boxed format,
    exactly like:
$\boxed{56}$
Use dollar signs and \boxed{} exactly as
    shown.
Problem: {question}
```

General Evaluation Prompt

```
You are an expert problem solver.
Read the following multiple-choice question
    and solve it step by step.
Choose the best option.
At the end, on a separate line, output ONLY
    the final option letter in LaTeX boxed
    format, exactly like:
$\boxed{A}$
Use dollar signs and \boxed{} exactly as
    shown.
Question: {question}
Options:
{options}
```

F Case Study

Example Problem

```
What is the greatest 3-digit base 8 positive
     integer that is divisible by 7?
(Express your answer in base 8.)
```

Original Demonstration (loss ≈ 0.74)

```
1. The greatest 3-digit number in base 8 is
    $777_8$.
2. Convert $777_8$ to decimal:
   \[
   7 \cdot 8^2 + 7 \cdot 8^1 + 7 \cdot 8^0 =
     7 \cdot 64 + 7 \cdot 8 + 7 = 448 + 56 +
     7 = 511
   \]
3. Find the largest number less than or
    equal to 511 that is divisible by 7. The
     closest number is:
   \[
   511 \div 7 = 73 \text{ R } 0
   \]
   Therefore, 511 itself is divisible by 7.
4. Convert 511 back to base 8:
   - $511 \div 8 = 63$ remainder 7
   - $63 \div 8 = 7$ remainder 7
   - $7 \div 8 = 0$ remainder 7
   Thus, $511_{10} = 777_8$.

Conclusion:
The greatest 3-digit base 8 number divisible
     by 7 is $\boxed{777_8}$.
```

Direct Rewrite (loss ≈ 0.63)

```
To find the greatest 3-digit base 8 positive
     integer that is divisible by 7, we
    first identify the largest 3-digit
    number in base 8, which is $777_8$.

We then convert $777_8$ to decimal to check
    for divisibility by 7. The conversion
    involves calculating $7 \cdot 8^2 + 7 \
    cdot 8^1 + 7 \cdot 8^0$, which
    simplifies to $7 \cdot 64 + 7 \cdot 8 +
    7 = 448 + 56 + 7 = 511$.

Since $511$ is divisible by 7 with no
    remainder, it is the number we are
    looking for. To express $511$ in base 8,
     we perform successive divisions by 8
    and record the remainders: $511 \div 8 =
     63$ remainder 7, $63 \div 8 = 7$
    remainder 7, and $7 \div 8 = 0$
    remainder 7. Therefore, $511_{10} = 777
    _8$.

The greatest 3-digit base 8 number divisible
     by 7 is $\boxed{777_8}$.
```

Ours (loss ≈ 0.44)

```
Step 1: Understand the problem
We are asked to find the greatest 3-digit
    base 8 positive integer that is
    divisible by 7.

Step 2: Find the greatest 3-digit base 8
    number
The greatest 3-digit number in base 8 is
    $777_8$.

Step 3: Convert $777_8$ to decimal
To convert $777_8$ to decimal, we need to
    multiply each digit by the corresponding
     power of 8 and add them together.
$7 \cdot 8^2 + 7 \cdot 8^1 + 7 \cdot 8^0 = 7
     \cdot 64 + 7 \cdot 8 + 7 = 448 + 56 + 7
     = 511$

Step 4: Find the largest number less than or
     equal to 511 that is divisible by 7
To find the largest number less than or
    equal to 511 that is divisible by 7, we
    can divide 511 by 7.
$511 \div 7 = 73$ with a remainder of 0, so
    511 is divisible by 7.

Step 5: Convert 511 back to base 8
To convert 511 back to base 8, we can divide
     it by 8 and keep track of the
    remainders.
$511 \div 8 = 63$ with a remainder of 7
$63 \div 8 = 7$ with a remainder of 7
$7 \div 8 = 0$ with a remainder of 7
So, $511_{10} = 777_8$

Step 6: Determine the greatest 3-digit base
    8 number divisible by 7
Since 511 is the greatest 3-digit base 8
    number that is divisible by 7, the
    greatest 3-digit base 8 number divisible
     by 7 is $777_8$.

$\boxed{777_8}$
```

Brief Analysis. This example is task-correct across all versions, but the supervision targets differ in how learnable and distribution-aligned they are under SFT.

- **Original (0.74)** is correct but includes more verbose narration and repeated conversion scaffolding, which increases sequence length and token-level uncertainty.
- **Direct rewrite (0.63)** reduces redundancy, improving conciseness and slightly lowering loss, but still follows a generic narrative template.
- **Ours (0.44)** yields the lowest loss: the rewritten target retains only the essential dependencies needed to justify the answer, while matching the model's preferred structured problem-solving format, resulting in a sharper training signal.

G Details Results

G.1 Per-benchmark Breakdown of Main Results

This appendix reports the full per-benchmark accuracies (%) corresponding to the aggregated scores in Table 1. We evaluate five in-domain math benchmarks (AMC23, AGI-Math, IMO, Math500, MinervaMath) and three out-of-domain generalization benchmarks (MMLU, MMLU-Pro, AGI), where math-related subsets are removed for the generalization suites as described in §4.1. All results are reported as accuracy (%), and higher is better.

G.2 Per-benchmark Ablation Results

Table 7 provides a per-benchmark breakdown of all ablation variants on `Llama-3.2-3B-Instruct`. We report accuracy (%) on downstream math benchmarks and on general-domain benchmarks with math-related subsets removed. Unless otherwise specified, variants use the same verification protocol and dataset construction procedure as in the main setting; `Success-only` trains downstream SFT using only rewrites that pass the task-consistency gate.

Model	Math Benchmarks					Generalization		
	AMC23	AGI-Math	IMO	Math500	MinervaMath	MMLU	MMLU-Pro	AGI
Llama-3.2-1B-Instruct	15.00	19.55	2.63	22.00	2.94	38.08	17.55	26.39
+ SFT	17.50	26.82	5.26	23.40	4.04	33.92	12.51	21.37
+ SDFT	17.50	21.36	3.51	23.20	3.44	34.02	14.21	23.17
+ Mind the Gap	15.00	20.18	3.51	22.60	3.31	34.83	15.09	23.32
+ Rewriting-agent	17.50	25.45	4.39	24.20	3.94	36.68	15.94	25.14
Llama-3.2-3B-Instruct	17.50	30.36	5.70	33.20	7.72	61.07	35.11	39.45
+ SFT	25.00	32.72	5.70	36.80	8.82	48.26	31.06	32.45
+ SDFT	20.00	32.27	5.82	33.00	8.09	55.19	31.02	34.75
+ Mind the Gap	22.50	31.09	6.41	33.40	8.09	54.12	31.53	35.68
+ Rewriting-agent	22.50	33.18	6.65	35.00	8.25	58.51	33.44	38.75
Mistral-7B-Instruct-v0.3	10.00	22.73	3.95	11.20	5.15	52.14	29.52	39.95
+ SFT	10.00	31.82	5.26	22.40	7.35	38.83	23.52	35.94
+ SDFT	12.50	24.55	5.14	20.40	6.25	44.77	24.10	37.21
+ Mind the Gap	15.00	25.00	5.26	20.20	7.56	43.71	23.18	36.41
+ Rewriting-agent	12.50	32.45	7.45	21.40	8.56	47.68	26.86	38.12

Table 6: Per-benchmark main results (accuracy, %). Math benchmarks evaluate in-domain mathematical reasoning. Generalization benchmarks are reported on math-removed subsets (higher is better).

Variant	Math Benchmarks					Generalization (math-removed)		
	AMC23	AGI-Math	IMO	Math500	MinervaMath	MMLU	MMLU-Pro	AGI
Rewriting-agent	22.50	33.18	6.65	35.00	8.25	58.51	33.44	38.75
w/o r_{dist}	22.50	32.86	6.15	34.20	7.93	53.14	31.20	33.76
w/o r_{div}	20.00	32.17	6.43	34.00	8.31	55.13	32.13	35.54
task-only	20.00	32.46	6.12	33.60	7.84	53.10	30.42	34.19
Soft shaping	17.25	32.14	6.01	34.24	7.94	58.34	32.81	37.72
Success-only	22.50	31.09	6.41	33.40	8.08	58.82	33.53	38.75

Table 7: Per-benchmark ablation results on `Llama-3.2-3B-Instruct` (accuracy, %). Math benchmarks evaluate downstream mathematical reasoning; generalization benchmarks are evaluated on math-removed subsets. Higher is better.