Rethinking Groups in Critic-Free RLVR

Anonymous ACL submission

Abstract

Reinforcement learning (RL) has become a central paradigm for post-training large language models. Existing critic-free RL methods typically generate a group of rollouts for the same question to estimate value baselines for advantage computation. However, this design suffers from data inefficiency, group synchronization barriers, and inflexibility with structured rollouts. In this work, we revisit the role of the "group" and show that its underlying function is not merely to estimate baselines but to prevent false penalties on negative samples. Building on this insight, we propose *negative token filtering*, a simple and effective strategy that enables stable single-rollout training. We apply it to two batch-level advantage methods, achieving comparable performance on reasoning tasks and stronger performance on agentic tasks relative to group-based RL techniques.

1 Introduction

Reinforcement learning (RL) has become the *de facto* paradigm for post-training large language models (LLMs) to enhance their capabilities. To avoid the computational and memory overhead of a separate critic network, as in critic-based methods like PPO (Schulman et al., 2017), critic-free methods have been widely adopted. Most of these—such as GRPO (Shao et al., 2024), RLOO (Ahmadian et al., 2024), and ReMax (Li et al., 2023)—generate multiple rollouts per prompt and use the resulting group to estimate value baselines for advantage computation.

Although critic-free methods are more efficient than critic-based alternatives, their reliance on generating and grouping multiple rollouts still incurs a cost. For example, rollout grouping can introduce synchronization barriers (Xu and Ding, 2026), force discarding groups whose rollouts receive identical rewards (Yu et al., 2025), and prove inflexible for structured rollouts in agentic RL (Feng et al., 2026). Therefore, recent works have explored critic-free methods that avoid multiple rollouts per prompt and group-based advantage computation (Hu et al., 2025; Xu and Ding, 2026). REINFORCE++ (Hu et al., 2025) replaces group normalization with batch-level normalization, computing advantages from generation batch rather than group-level statistics (Fig. 1). However, for reasoning tasks it still typically uses a group size larger than one to improve training stability; with a single rollout, it suffers from severe training instability (Fig. 3). To fully enable single rollout generation, SPO (Xu and Ding, 2026) builds on batch-level normalization and introduce an additional tracker that estimates the value baseline via historical information. However, this tracker requires extra sampling before training, adding computational overhead.

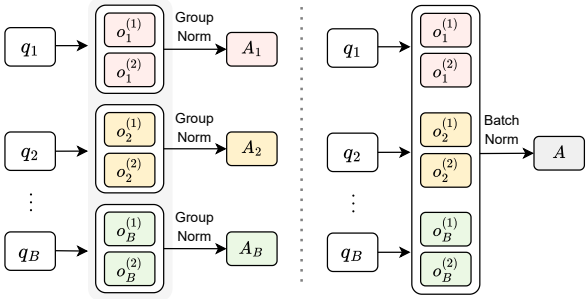

Figure 1: Advantage computation in GRPO vs. REINFORCE++. GRPO (left) computes advantages at the group level, whereas REINFORCE++ (right) computes them at the generation-batch level.

To uncover the role of grouping, we reverse-engineer the functional mechanism of rollout groups and use the resulting insight to develop an alternative single-rollout, critic-free policy optimization method. We make two observations: (1) the training instability of single-rollout RL originates from negative samples; and (2) using exactly one positive and one negative rollout per prompt restores stable training. We interpret these observations as follows. First, an incorrect reasoning trajectory is rarely entirely wrong—it still contains

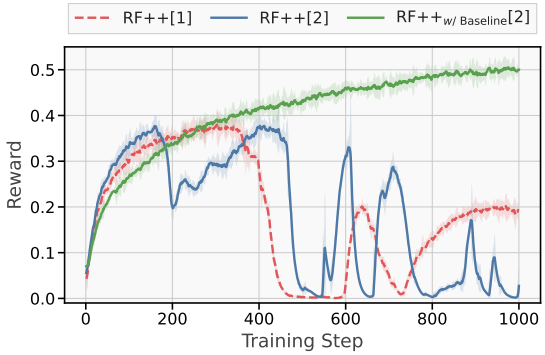

Figure 2: Training Curves of RF++[1], RF++[2] and RF++$_{w/\text{ Baseline}}$[2]. This indicates that multi-rollout generation cannot guarantee stable training. The grouping mechanism is more critical for training stability,.

many useful token patterns, such as formatting, intermediate reasoning steps, and tool-use cues. Penalizing all of them equally therefore leads to harmful updates. Second, this harm is strongly alleviated when a positive trajectory for the same prompt is present: since the positive and negative rollouts typically share these functional tokens, the negative gradient on the shared tokens is partially cancelled out. In other words, the group does not merely estimate a baseline; it *de facto* protects shared useful tokens from being over-penalized. We confirm this cancellation effect through (1) a statistical analysis of token overlap and (2) gradient projection onto the Top-K subspace of the weight matrices. Stemming from this insight, we propose a simple filtering strategy for negative trajectories that retains only the Top-10% lowest-probability tokens in the negative loss. We empirically verify this filtering on two batch-level advantage computation methods, and the resulting group-free methods outperform their group-based counterparts.

2 Analysis

Most critic-free RL methods for LLMs introduce two mechanisms: multi-rollout sampling per prompt, and group-based advantage computation. Note that a method with multi-rollout sampling might not introduce group-based advantage, as in PPO or REINFORCE++. On the other hand, the group-based advantage computation relies on multi-rollout sampling.

In this section, we disentangle these two mechanisms and study their effects in isolation.

2.1 The Effect of Grouping

We use REINFORCE++ (RF++) series (Hu et al., 2025) as the base methods, which enables us to isolate the effects of multi-rollout generation and grouping. Motivated by recent work (Wu et al., 2025), we focus on a group size G of 2 as the minimal multi-rollout setting, which allows a clean analysis while avoiding unnecessary experimental complexity.

Specifically, we compare three variants: RF++[1], RF++[2] and RF++$_{w/\text{ Baseline}}$[2], where $[n]$ denotes the group size of n. The RF++ performs reward normalization over the mini-batch of rollouts, rather than within each prompt-level group as in GRPO. The RF++$_{w/\text{ Baseline}}$ further subtracts the group-wise sample mean before batch-level normalization, thereby incorporating a GRPO-style grouping mechanism prior to mini-batch normalization – a group of rollouts, which are all correct or all incorrect, will yield zero advantages and thus contributes no gradient to the optimization.

As shown in Fig. 2, although RF++[2] is more stable than RF++[1], it still eventually collapses. By contrast, RF++$_{w/\text{ Baseline}}$[2] remains stable throughout training, with steadily increasing rewards and no visible sign of collapse. This suggests that multi-rollout sampling can partially mitigate instability, but does not guarantee stable training. Compared with multi-rollout sampling alone, the grouping mechanism is more critical for training stability – it ensures each effective group contains both positive and negative samples.

2.2 The Impact of the Negative

In the previous section, we reveal that the group-based advantage can effectively mitigate training instability. In this section, we next investigate, in the absence of the grouping, which component is primarily responsible for instability: positive or negative samples.

Trajectories with negative advantages induce updates that decrease the likelihood of the sampled tokens. However, especially in RLVR, a negative sample is labeled only by its incorrect final answer. As the old saying goes, *Bonum ex integra causa, malum ex quocumque defectu.* [1] An incorrect final answer does not imply that every token in the trajectory is erroneous or should be penalized. We therefore hypothesize that uniformly penalizing negative trajectories can falsely discourage useful reasoning patterns, grammatical tokens, and other functional components – we denote them as

[1] Literal translation: *Good arises from an integral cause; evil from any defect whatsoever.*

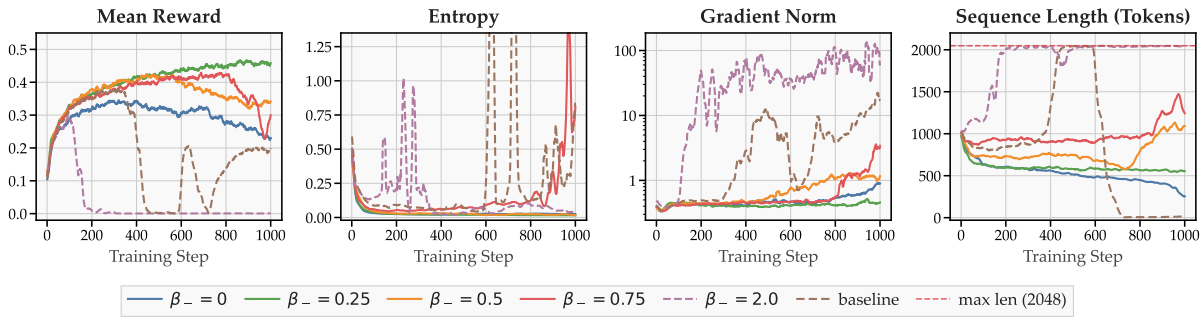

Figure 3: Learning dynamics under varying negative coefficients β_k. We train the Qwen2.5-Math-1.5B model on a 7.5k-example subset of the DAPO-MATH dataset for 1k steps with a learning rate of 1×10^{-3}. Each step contains 512 prompts and uses a mini-batch size of 32. We apply an in-reward KL penalty with a coefficient of 1×10^{-3}.

supporting tokens (an example is shown in Fig. 5) – thereby inducing unstable and potentially destructive updates.

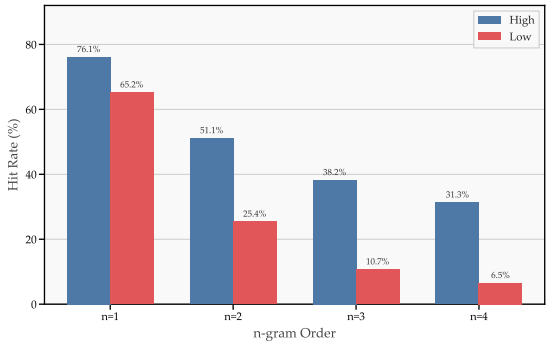

Figure 4: Hit rates in positive rollouts of high- and low-probability n-grams from negative rollouts. Results are computed with Qwen2.5-Math-1.5B on MATH500. For each prompt, we generate 10 responses and discard degenerated trajectories. Among 500 groups, 141 groups with all correct or all incorrect responses are excluded.

To verify our hypothesis, we conduct a controlled experiment on RF++[1] that varies the strength of the negative signal while keeping the positive signal fixed. Specifically, we scale the negative advantage by a coefficient β_- and examine its effect on training dynamics, monitoring reward, entropy, gradient norm, and sequence length throughout training. As shown in Fig. 3, collapse manifests as a sharp drop in reward, accompanied by simultaneous increases in entropy, gradient norm, and sequence length. Once the sequence length saturates at the preset maximum of 2048 tokens, training briefly stabilizes before collapsing again. During this transient phase, the KL penalty becomes the dominant term in the objective, driving the sequence length back down. By this point, the model has already suffered substantial degradation.

As we decrease β_- from 1 (the baseline) toward 0, training becomes progressively more stable, and the onset of collapse is delayed. [2] At $\beta_- = 0.25$, training remains stable throughout the full 1k-step training horizon: the reward increases steadily, with no abrupt drops, while entropy, gradient norm, and sequence length remain well-behaved. Since this horizon is substantially longer than the typical collapse time observed under larger negative coefficients, the stability is unlikely to be incidental. The opposite further supports this conclusion: when $\beta_- = 2$, corresponding to the strongest negative signal we test, collapse occurs earlier than in all other configurations. Together, these results indicate that training instability is primarily driven by negative samples rather than positive samples.

2.3 Supporting Tokens in Rollouts

Motivated by the previous findings – the instabilities stem from negative samples, whereas the positive rollouts from the same group can mitigate –, we hypothesize that: 1) in the absence of positive rollouts, negative samples lead to penalties on supporting tokens, inducing harmful updates; 2) when optimized jointly, positive samples can offset these harmful gradients through their shared supporting tokens (Cheng et al., 2026).

This hypothesis is empirically verified by measuring token-level overlap between positive and negative samples generated from the same prompt (shown in Fig. 4). Notably, we further categorize the tokens into two types: high-probability tokens as those whose probabilities fall within the top 90% in a sequence, and define the remaining tokens as low-probability tokens. As shown in Fig. 4, high-

[2] Note that collapse is stochastic: even under the same configuration, it may occur at different training steps. However, the overall trend is clear.

Math Question: A class of 30 students recently took a test. If 20 students scored 80, 8 students scored 90, and 2 students scored 100, then what was the class average (the mean) on this test?
Answer: **84**

Positive Rollout

To find the class average (mean) on the test, we need to calculate the total score of all the students and then divide it by the number of students.
Here's the step-by-step process:
1. Calculate the total score for the 20 students who scored 80.
…
5. Divide the overall total score by the total number of students (30) to get the class average.

Let's calculate this using Python:
```python
# Number of students and their respective scores
students_80 = 20  students_90 = 8  students_100 = 2
score_80 = 80  score_90 = 90  score_100 = 100

# Calculate the total score for each group of students
total_score_80 = students_80 * score_80
total_score_90 = students_90 * score_90
total_score_100 = students_100 * score_100

# Calculate the overall total score
overall_total_score = total_score_80 + total_score_90 + total_score_100

# Calculate the class average
total_students = students_80 + students_90 + students_100
class_average = overall_total_score / total_students
print(class_average)
```
```output 84.0 ```
The class average on the test is **84.0**.

Negative Rollout

To find the class average (mean) on the test, we need to calculate the total score of all students and then divide that by the number of students.

1. Calculate the total score for the 20 students who scored 80.
…
5. Divide the overall total score by the total number of students (30) to get the class average.

Let's do this step-by-step using Python code.
```python
# Number of students and their scores
students_80 = 20  students_90 = 8  students_100 = 2
score_80 = 80  score_90 = 90  score_100 = 100

# Total scores
total_score_80 = students_80 * score_80
total_score_90 = students_90 * score_90
total_score_100 = students_100 * score_100

# Overall total score for the class
total_score = total_score_80 + total_score_90 + total_score_100
# Total number of students
total_students = students_80 + students_90 + students_100
# Class average
class_average = total_score / total_students
print(class_average)
```
```output 84.66666666666667 ```
The class average (mean) on the test is **84.67**.

Figure 5: An example illustrating that positive and negative rollouts for the same prompt can share many *supporting tokens*, shown as uncolored text.

probability n-grams consistently achieve higher hit rates than low-probability n-grams across different values of n. This evidence supports two key findings: 1) positive samples within the same group protect training stability by offsetting harmful updates induced by negative samples on shared supporting tokens; 2) these supporting tokens, particularly at the n-gram level, are more likely to fall into the high-probability token category.

This observation aligns with a natural intuition: tokens assigned higher probabilities by the policy are more likely to be supporting tokens, as the pretrained base model already encodes strong language-modeling and reasoning priors. This insight also suggests a straightforward mitigation for single-rollout RL methods: suppress supporting tokens in negative samples with prediction confidence as a proxy, motivating our proposed technique – NTF – discussed in Section 3.2.

2.4 Top-K Subspace Alignment

In the previous section, we have statistically shown shared supporting tokens among the positive and negative rollouts within a group. The failure mode is not a gradual decline in reward but an abrupt collapse of the model's language and reasoning ability—the signature of catastrophic forgetting (Kirkpatrick et al., 2017). We hypothesize that a layer's pretrained competence is carried mainly by the dominant directions of its weight matrix, i.e., its top singular subspaces, so that an update is harmful to the extent that its gradient concentrates there. This motivates a direct question: for each type of update, how much of the gradient's energy falls within the top singular subspace of the weights? We answer it by projecting the gradient onto the top-k singular subspaces, as described below.

For a weight matrix $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, we compute its singular value decomposition:

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}, \tag{1}$$

where $\boldsymbol{U} \in \mathbb{R}^{d_{\text{out}} \times r}$ and $\boldsymbol{V} \in \mathbb{R}^{d_{\text{in}} \times r}$ have orthonormal columns, $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \ldots, \sigma_r)$ with $\sigma_1 \geq \cdots \geq \sigma_r \geq 0$, and $r = \min(d_{\text{out}}, d_{\text{in}})$. Let $\boldsymbol{U}_k$ and $\boldsymbol{V}_k$ contain the leading k left and right singular vectors, respectively; these vectors span the top-k left and right singular subspaces of $\boldsymbol{W}$.

Given a gradient $\boldsymbol{G} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ with the same shape as $\boldsymbol{W}$, we project it onto the top-k singular

subspaces of $\boldsymbol{W}$:

$$\boldsymbol{P}_k = \boldsymbol{U}_k^\intercal \boldsymbol{G} \boldsymbol{V}_k \in \mathbb{R}^{k \times k}. \tag{2}$$

This projection is the leading $k \times k$ block of $\boldsymbol{G}$ expressed in the singular-vector basis of $\boldsymbol{W}$. We then define the normalized block energy as

$$\rho_k = \frac{\|\boldsymbol{P}_k\|_F^2}{\|\boldsymbol{G}\|_F^2} \in [0, 1], \tag{3}$$

which measures the fraction of the gradient's Frobenius energy captured by the top-k singular subspaces. Since orthogonal transformations preserve the Frobenius norm, ρ_k is non-decreasing in k and reaches 1 at $k = r$. A large ρ_k for small k therefore indicates that the gradient is concentrated along the dominant directions of $\boldsymbol{W}$.

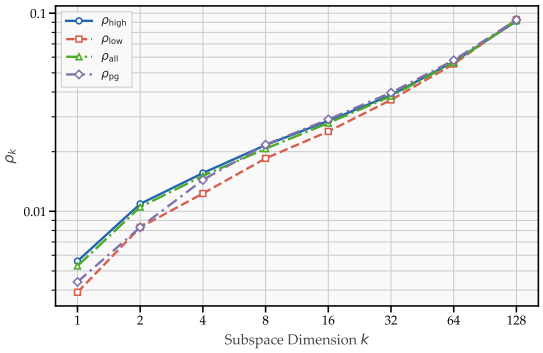

Figure 6: Block energy ρ_k as a function of the subspace dimension k, computed with Qwen2.5-Math-1.5B on MATH500 over 1,380 positive and 1,380 negative samples drawn from 359 prompts. Each curve is averaged over 197 weight matrices.

We compute the gradient from the advantage-weighted log-likelihood loss

$$l = -\frac{A}{|o|} \sum_{t=1}^{|o|} m_t \log \pi_\theta(o_t \mid o_{<t}, q), \quad \boldsymbol{G} = \frac{\partial l}{\partial \boldsymbol{W}}, \tag{4}$$

where o is the sampled trajectory, q the question, $|o|$ the trajectory length, A the advantage, and $m_t \in \{0, 1\}$ a token mask that selects which tokens contribute to the gradient. We compare four settings. The first three isolate the effect of a token subset: we fix $A = 1$, accumulate the gradient over both positive and negative samples, and vary only the mask—(1) **high-prob.** tokens, (2) **low-prob.** tokens, and (3) **all tokens** ($m_t = 1$). The fourth, (4) **policy gradient**, sets $m_t = 1$ and uses the group-relative advantage ($A = 1$ for correct and $A = -1$ for incorrect trajectories), so that positive and negative trajectories enter with opposite signs.

Fig. 6 reveals three patterns: (i) the all-token curve closely tracks the high-prob. token curve; (ii) for small k, high-prob. tokens attain a substantially larger ρ_k than low-prob. tokens; and (iii) at $k = 1, 2$, the policy-gradient curve nearly coincides with the low-prob. token curve.

These patterns support our hypothesis. High-prob. tokens align far more strongly with the top singular directions than low-prob. tokens (observation (ii)), so updating on them perturbs precisely the directions we associate with the model's core capability—and since the all-token gradient is dominated by these tokens (observation (i)), naive single-rollout training cannot avoid this. Under group-based optimization, however, the high-prob. gradients of positive and negative trajectories are top-K-aligned but carry opposite-sign advantages, and therefore largely cancel; the policy-gradient curve consequently collapses onto the low-prob. curve at $k = 1, 2$ (observation (iii)). Grouping thus shields the dominant directions from over-penalization, preventing collapse.

3 Methodology

3.1 Batch-level Advantage Computation

Notation. Let π_θ be the current policy with parameter θ and $\pi_{\theta_{\text{old}}}$ the behavior policy used to collect rollouts, and let $\rho(o_t) = \pi_\theta(o_t)/\pi_{\theta_{\text{old}}}(o_t)$ denote the per-token importance ratio. For a query q, each sampled trajectory is labeled positive (o^+) or negative (o^-); in RLVR with binary verifiable rewards this label is simply the correctness of the trajectory. We write N^+ and N^- for the number of positive and negative trajectories in a mini-batch. For continuous rewards, the positive and negative samples are determined by the value baseline estimated by group/batch average.

Base RL techniques with batch-level advantage estimation. We mainly verify our proposed techniques on two RL techniques with batch-level advantage estimation.

REINFORCE++ (RF++). RF++ (Hu et al., 2025) forms the advantage by z-score normalizing rewards within a mini-batch B,

$$A = \frac{r - \mu_B}{\text{std}_B}. \tag{5}$$

Contrastive-REINFORCE (C-RF). For binary rewards with batch success rate p, we have $\mu_B = p$

and $\mathrm{std}_B = \sqrt{p(1-p)}$. Wu et al. (2025) and Li et al. (2025) show that the surrogate with z-score normalization yields [3]

$$\mathcal{J} = \sqrt{p(1-p)}\, \mathbb{E}_q\big[\, l^+(o^+ \mid q) - l^-(o^- \mid q)\,\big]\,, \tag{6}$$

where the scalar $\sqrt{p(1-p)} = \mathrm{std}_B$ is the *standard deviation* of the batch success indicator, which might lead to an undesired weighting effect in RLVR setting – amplifying at $p = \frac{1}{2}$ and attenuating as $p \to 0$ or $p \to 1$. This might be more pronounced in agentic settings, where the initial success rate is near zero.

Therefore, we also verify on another objective, W-REINFORCE (Zhu et al., 2026), without the weighting coefficient:

$$\begin{aligned}
\mathcal{L} &= -\frac{1}{2}\left(\frac{1}{N^+}\sum_{i=1}^{N^+} l^+(o_i^+) \;-\; \frac{1}{N^-}\sum_{j=1}^{N^-} l^-(o_j^-)\right),\\
l^+(o) &= \frac{1}{|o|}\sum_{t=1}^{|o|} \min\big(\rho(o_t),\, 1+\epsilon\big),\\
l^-(o) &= \frac{1}{|o|}\sum_{t=1}^{|o|} \max\big(\rho(o_t),\, 1-\epsilon\big).
\end{aligned} \tag{7}$$

We denote it as Contrastive-REINFORCE (C-RF), as the weighting coefficient is removed.

3.2 Negative Token Filtering (NTF)

A negative trajectory is penalized token by token, but not every token is to blame for the failure. Many of its high-probability tokens are generic or are also produced during correct behavior. Penalizing them indiscriminately injects noise into the update and destabilizes training. We therefore concentrate the negative gradient on the tokens most plausibly responsible for the failure.

Concretely, given a masking fraction $\tau \in [0, 1]$, we rank the tokens of each negative trajectory by their probability under the current policy and mask the top-τ fraction (the highest-probability tokens). Let $\mathcal{K}(o)$ denote the set of tokens that remain. The negative term is still normalized by the full sequence length,

$$l^-(o) = \frac{1}{|o|}\sum_{t\in\mathcal{K}(o)} \max\big(\rho(o_t),\, 1-\epsilon\big), \tag{8}$$

so masked tokens contribute nothing while still counting toward the denominator. This shields high-probability tokens from spurious penalties while keeping the update focused on the tokens that drive the negative outcome.

4 Related Work

Group-Based RL Algorithms Most current RL algorithms for LLMs are group-based; they require multiple rollouts per prompt. GRPO and RLOO are two representative members of this family. ReMax uses two rollouts per prompt and can therefore also be regarded as group-based. We note that REINFORCE++ performs a single rollout in the RLHF setting but multiple rollouts in the RLVR setting. In all of these methods, the group of rollouts is used to estimate the value function, thereby reducing variance. Recent work offers an alternative interpretation in which group-based methods are viewed as a form of contrastive learning (Wu et al., 2025; Zhu et al., 2026), where the primary role of grouping is to supply contrastive pairs.

Group-Free RL Algorithms PPO (Schulman et al., 2017), the most widely used RL algorithm, requires only a single rollout per prompt. However, it must maintain a separate LLM as the critic. More recently, SPO has been proposed as a critic-free, group-free method. SPO leverages historical information to estimate the success rate of each prompt, which serves as the value function. To obtain accurate estimates, SPO requires a pre-rollout phase before training, incurring additional computational cost. In this paper, we instead focus on uncovering the hidden mechanism behind grouping. Our proposed negative filtering enables stable single-rollout training without any additional computational overhead.

Token Masking in RL Wang et al. (2025) propose improving RL post-training by only optimizing high-entropy minority tokens. While this idea is philosophically related to our proposed NTF, the two methods differ in both motivation and implementation. NTF applies only to negative trajectories, whereas Wang et al. (2025) are not restricted to negative samples and targets high-entropy tokens.

5 Experiment

In this section, we conduct empirical experiments to validate the efficacy of our proposed NTF on

[3] Conditioning on the outcome, $A^+ = \sqrt{(1-p)/p}$ with probability p and $A^- = -\sqrt{p/(1-p)}$ with probability $1-p$; multiplying by the respective outcome probabilities gives the shared coefficient $p\,A^+ = -(1-p)\,A^- = \sqrt{p(1-p)}$.

Table 1: Mathematical reasoning results under different rollout budgets. All methods are trained on the 7.5K-prompt DAPO-Math subset and evaluated with Mean@32 accuracy (%) on five held-out benchmarks. All G=1 rows use the same log-probability-based negative-token filtering ratio τ=0.1. "–" indicates a training crash.

Model	Algorithm	G	Benchmarks (Mean@32 ↑)					Avg.
			MATH-500	AMC 2023	Minerva	AIME 2025	Olympiad	
Qwen2.5-Math-1.5B	GRPO	16	71.04	57.11	18.22	10.00	24.86	36.25
	GRPO	2	68.81	52.19	16.79	8.13	23.52	33.89
	RF++	8	67.48	54.77	17.36	6.46	22.87	33.79
	RF++	1	–	–	–	–	–	–
	RF++ w/ NTF	1	68.21	55.47	16.31	2.60	21.68	32.86
	C-RF	1	–	–	–	–	–	–
	C-RF w/ NTF	1	69.63	55.47	18.35	9.38	23.89	35.34
Qwen2.5-Math-7B	GRPO	16	76.94	68.91	25.67	15.94	28.66	43.22
	GRPO	2	77.43	64.84	21.95	14.58	29.86	41.73
	RF++	8	71.53	58.36	23.39	6.88	24.74	36.98
	RF++	1	–	–	–	–	–	–
	RF++ w/ NTF	1	64.94	68.91	21.97	6.15	24.92	37.37
	C-RF	1	–	–	–	–	–	–
	C-RF w/ NTF	1	74.08	61.95	23.82	12.19	26.54	39.72

two RL base techniques with batch-level advantage estimation: RF++ and C-RF.

Experimental Setup. We evaluate on mathematical reasoning and agentic decision-making. The reasoning experiments train on a 7.5K-prompt subset of DAPO-Math (Yu et al., 2025; Wu et al., 2025) and evaluate on five held-out math benchmarks; the agentic experiments follow He et al. (2026) on ALFWorld and WebShop. We use verl (Sheng et al., 2024) for experiment framework. Full baseline, dataset, metric, reward, and hyper-parameter details are provided in Appendix A.

5.1 Reasoning Task

Table 1 reports mathematical reasoning results under different rollout budgets. The main observation is that single-rollout training becomes viable when negative trajectories are filtered. In our experiments, RF++ and C-RF with $G=1$ can be trained stably with our proposed NTF, while the same setting without filtering collapses during training. This suggests that directly penalizing all tokens in failed trajectories is harmful for long-form reasoning, Filtering mitigates this issue by selectively reducing negative updates on high-confidence tokens and thus preserves useful reasoning patterns.

Compared to RF++ w/ NTF, C-RF w/ NTF reaches a better overall performance, outperforming the low-budget GRPO baseline with $G = 2$ on the 1.5B model and approaches GRPO with $G = 16$, with only a 0.91 gap in average for the 1.5B model. On the 7B model, group-based GRPO remains stronger, but our method still substantially stablize the training of RF++/C-RF, while using only one rollout per prompt.

We attribute this advantage to a stronger and more reliable training signal in the single-rollout setting: after filtering harmful negative updates on supporting tokens, our objective can make better use of the remaining sparse reward signal than vanilla batch-level advantage computation.

5.2 Agentic Task

In agentic tasks, we evaluate C-RF with NTF only, as we were unable to successfully train RF++ with NTF in these settings. We attribute this to the coefficient $\sqrt{p(1-p)}$ discussed in Eq. 6: because agentic tasks are substantially harder in the early stages of training, p is small, so this coefficient sharply attenuates the learning signal and makes training difficult.

Table 2 reports results on ALFWorld and WebShop, which test whether our single-rollout training principle generalizes beyond math reasoning to agentic tasks under sparse, delayed rewards. RL training substantially outperforms prompting-based methods in both environments, confirming that environment feedback supplies useful supervision for agentic behavior. We compare primarily against PPO, RLOO, and GRPO, representing critic-based and group-based critic-free training.

Table 2: Agentic decision-making results on ALFWorld and WebShop. We compare prompting baselines and RL-trained methods on Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct. ALFWorld reports in-distribution and out-of-distribution success rates, while WebShop reports task score and task success. "–" indicates a training crash. Baseline results are from He et al. (2026).

Model	Type	Method	ALFWorld		WebShop	
			In-Success	Out-Success	Task Score	Task Success
Qwen2.5-1.5B-Instruct	Prompting	Qwen2.5	4.1	–	23.1	5.2
	Prompting	ReAct	12.8	–	40.1	11.3
	Prompting	Reflexion	21.8	–	55.8	21.9
	RL Training	PPO (with critic)	54.4	–	73.8	51.5
	RL Training	RLOO	69.7	68.7	73.9	52.1
	RL Training	GRPO	72.8	70.1	**75.8**	56.8
	RL Training	C-RF	–	–	–	–
	RL Training	C-RF w/ NTF	**87.4**	**77.9**	72.4	**65.7**
Qwen2.5-7B-Instruct	Prompting	Qwen2.5	14.8	–	26.4	7.8
	Prompting	ReAct	31.2	–	46.2	19.5
	Prompting	Reflexion	42.7	–	58.1	28.8
	RL Training	PPO (with critic)	77.1	76.2	**81.4**	68.7
	RL Training	RLOO	77.9	74.0	80.3	65.7
	RL Training	GRPO	78.6	76.8	79.3	66.1
	RL Training	C-RF	–	–	–	–
	RL Training	C-RF w/ NTF	**90.5**	**86.3**	80.2	**74.1**

On ALFWorld, our method is strong at both model scales. Qwen2.5-1.5B-Instruct reaches 87.38 in-distribution and 77.86 out-of-distribution success, exceeding GRPO on both. Qwen2.5-7B-Instruct further improves these to 90.48 and 86.32, surpassing PPO, RLOO, and GRPO. The single-rollout objective is thus effective in long-horizon settings where reward is observed only after a complete trajectory.

WebShop shows a similar advantage on task success: our method raises success over GRPO from 56.8 to 65.73 (1.5B) and from 66.1 to 74.07 (7B). Results on task score are more mixed—our method trails GRPO on the 1.5B model and PPO/RLOO on the 7B model, though it remains competitive. This suggests our approach chiefly improves final task completion, while fine-grained score optimization remains sensitive to the environment's reward structure.

The stronger performance of C-RF w/ NTF on agentic tasks suggests that group-based cancellation may be less reliable in complex search spaces, where positive and negative trajectories share relatively fewer supporting tokens due to the large search spaces. This further highlights the importance of selectively controlling negative-sample updates. Extending NTF to group-based RL algorithms is a promising direction for future work.

6 Conclusion

We revisited the role of multi-rollout sampling and group-based advantage in critic-free RLVR and showed that their function extends beyond value-baseline estimation. In particular, groups stabilize training by allowing positive and negative rollouts from the same prompt to cancel harmful negative updates on shared supporting tokens. This perspective explains why single-rollout training is especially vulnerable to negative samples, which can falsely penalize useful reasoning patterns, grammatical tokens, and structural components.

Motivated by this insight, we proposed negative token filtering (NTF), which masks high-probability tokens in negative trajectories and concentrates the negative update on lower-probability tokens. NTF enables stable single-rollout, critic-free, and group-free training. Experiments on mathematical reasoning and agentic decision-making show that our method avoids collapse and achieves competitive or stronger performance than representative group-based baselines. These results suggest that stable RLVR does not inherently require group-based advantage computation, provided that negative-token updates are handled selectively.

7 Limitation

Our analysis is primarily static: we do not conduct dynamic experiments that track metrics throughout RL training. This is because our weight analysis relies on SVD, which is computationally expensive even at small scale, making spectrum analysis over the course of training impractical. In addition, our experiments focus on the RLVR (binary reward) setting and do not cover ordinal or continuous rewards. We believe RLVR is representative enough to validate our findings, and we leave the exploration of continuous-reward settings to future work.

We use LLMs for polishing writing.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267.

Tianhao Cheng, Zeyu Huang, Zihan Qiu, Yu Cheng, Edoardo Ponti, Yinghui Xu, Ivan Titov, and Zenglin Xu. 2026. The cancellation hypothesis in critic-free rl: From outcome rewards to token credits. *arXiv preprint arXiv:2605.08666*.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2026. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *Preprint*, arXiv:2402.14008.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. 2026. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. *arXiv preprint arXiv:2602.22817*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. 2025. Reinforce++: Stabilizing critic-free policy optimization with global advantage normalization. *arXiv preprint arXiv:2501.03262*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and 1 others. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2025. Tulu 3: Pushing frontiers in open language model post-training. *Preprint*, arXiv:2411.15124.

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. *Preprint*, arXiv:2206.14858.

Gang Li, Ming Lin, Tomer Galanti, Zhengzhong Tu, and Tianbao Yang. 2025. Disco: Reinforcing large reasoning models with discriminative constrained optimization. *Advances in Neural Information Processing Systems*, 38:57304–57331.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. 2023. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint arXiv:2310.10505*.

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. *Preprint*, arXiv:2305.20050.

Math-AI Team. 2025. Amc 2023. Hugging Face dataset. Available at: `https://huggingface.co/datasets/math-ai/amc23`.

MathArena. 2026. Aime 2025. Hugging Face dataset. Dataset constructed from the 2025 American Invitational Mathematics Examination.

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, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv 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*.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. *Preprint*, arXiv:2010.03768.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, and 1 others. 2025. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. In *Advances in Neural Information Processing Systems*.

Yihong Wu, Liheng Ma, Lei Ding, Muzhi Li, Xinyu Wang, Kejia Chen, Zhan Su, Zhanguang Zhang, Chenyang Huang, Yingxue Zhang, and 1 others. 2025. It takes two: Your grpo is secretly dpo. *arXiv preprint arXiv:2510.00977*.

Zhongwen Xu and Zihan Ding. 2026. Single-stream policy optimization. In *Int. Conf. Learn. Represent.*

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2023a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Preprint*, arXiv:2207.01206.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. 2026. The surprising effectiveness of negative reinforcement in llm reasoning. *Advances in Neural Information Processing Systems*, 38:126546–126573.

A Experimental Details

A.1 Datasets

For mathematical reasoning, all RL methods are trained on the same 7.5K-prompt subset of DAPO-Math (Yu et al., 2025) and evaluated on MATH-500 (Hendrycks et al., 2021; Lightman et al., 2023), AMC 2023 (Math-AI Team, 2025), Minerva Math (Lewkowycz et al., 2022), AIME 2025 (Math-Arena, 2026), and Olympiad Bench (He et al., 2024). We use binary verifiable rewards based on extracted final-answer correctness and report Mean@32 accuracy, computed from 32 sampled generations per problem.

For agentic tasks, we follow the ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2023a) protocol of He et al. (2026). We train our method on a 1024-task ALFWorld subset and a 1024-instruction WebShop subset, and evaluate with the same metrics as the reported baselines: in-/out-of-distribution success for ALFWorld, and task score/task success for WebShop. Rewards are sparse and delayed environment feedback with no per-step shaping; for WebShop, we use binary task success as the training reward.

A.2 Baseline Details

We provide additional details about the baselines used in our experiments. The baselines are selected to compare our method with representative RLVR (Lambert et al., 2025) training paradigms under different rollout budgets. In particular, we use REINFORCE++ (Hu et al., 2025) with $G = 1$ as the closest single-rollout baseline, GRPO (Shao et al., 2024) with small and large group sizes as group-based references, and standard prompting/RL baselines for agentic tasks.

Mathematical reasoning. For mathematical reasoning, we compare with GRPO and REINFORCE++. GRPO is a representative group-based critic-free RLVR method, where multiple responses are sampled for the same prompt and relative advantages are estimated within each group. We report GRPO with $G = 16$ as a strong multi-rollout baseline and GRPO with $G = 2$ as a lower-budget group-based baseline. These two settings allow us to examine how our single-rollout method compares with group-based training under different rollout costs.

REINFORCE++ is used as the main batch-normalized critic-free baseline. We report REINFORCE++ with $G = 8$ as a multi-rollout batch-normalized baseline and REINFORCE++ with $G = 1$ as the closest comparison to our method. The $G = 1$ setting is especially important because it shares the same one-rollout-per-prompt constraint as our method. For all single-rollout runs conducted by us, including REINFORCE++ with $G = 1$ and our method, we apply the same log-probability-based negative-token filtering with ratio $\tau = 0.1$. Therefore, the comparison between REINFORCE++ with $G = 1$ and our method evaluates whether our single-rollout training rule can make better use of the same filtered reward signal, rather than merely testing the effect of adding token filtering.

Agentic tasks. For agentic tasks, we compare with both prompting-based and RL-based baselines. The prompting baselines include direct Qwen2.5 (Qwen et al., 2025) prompting, ReAct (Yao et al., 2023b), and Reflexion (Shinn et al., 2023), which measure the performance of the base model without RL training. The RL baselines include PPO (Schulman et al., 2017), RLOO (Ahmadian et al., 2024), and GRPO. PPO represents critic-based RL training, while RLOO and GRPO represent critic-free methods based on relative or grouped trajectory comparison.

The baseline numbers for agentic tasks are taken from He et al. (2026), which reports results on ALFWorld and WebShop using Qwen2.5-Instruct backbones. Specifically, ALFWorld is evaluated by in-distribution and out-of-distribution success rates, while WebShop is evaluated by task score and task success rate. Our method is evaluated under the same task protocol and metrics for comparison. Unlike the group-based baselines, our method uses only one rollout per prompt and does not require a critic, a value tracker, or prompt-level rollout groups.

A.3 Hyper-parameters

See Table 3 and Table 4.

Table 3: Training hyper-parameters of our method. Values that are shared across all three tasks are merged into a single cell; columns differ only where shown.

Hyper-parameter	**Math**	**ALFWorld**	**WebShop**
Backbone			
Base model	Qwen2.5-Math-{1.5, 7}B	Qwen2.5-{1.5, 7}B-Instruct	
Optimizer			
Optimizer	AdamW		
Learning rate	10^{-6}		
LR scheduler	Cosine	Exponential	Exponential
Warm-up steps	10		
Weight decay	0.1		
Gradient clipping	1.0		
RL objective			
Advantage estimator	RF++, C-RF	C-RF	C-RF
Rollouts per prompt (G)	1		
Clip ϵ_{low}	0.2		
Clip ϵ_{high}	10		
Negative-token filter ratio τ	0.1		
KL penalty in reward	disabled		
KL loss in objective	disabled		
Batching & sequence lengths			
Rollout batch size (prompts)	512	64	64
PPO mini-batch size	128	64	64
Max prompt length	2048		
Max response length	4096	4096 (all turns)	4096 (all turns)
Rollout sampling			
Temperature	1.0		
Top-p	1.0		
Environment / reward			
Reward type	verifiable binary	binary success	binary success
Max steps per episode	–	50	15

Table 4: Evaluation hyper-parameters.

Hyper-parameter	**Math**	**ALFWorld**	**WebShop**
Sampling temperature	0.7	1.0	1.0
Top-p	0.7	0.7	0.7
Samples per prompt	32	3	3
Reported metric	Mean@32 accuracy	In/out-distribution success rate	Task score, success rate
Eval set	MATH-500, AMC 23, Minerva, AIME 25, Olympiad	ALFWorld in/out-distribution splits	WebShop test set