Mitigating Exploration Bias in RL for Multi-Instruction Following

Mian Zhang[1]*, Yueqin Yin[2]*,
Kaiyu He[1], Peilin Wu[1], Xinlu Zhang[3], Mingyuan Zhou[2], Zhiyu Zoey Chen[1]
[1]UT Dallas, [2]UT Austin, [3]UC Santa Barbara

Abstract

RL has emerged as a powerful paradigm for enhancing the instruction following capabilities of LLMs. While existing training recipes achieve substantial gains, we find that they suffer from **exploration bias towards easy instructions** when the training data has multiple instructions in a prompt. This bias is caused by two main reasons: 1) **the policy model's initial ability to satisfy hard instructions is too low** to trigger successful exploration during RL training, so the optimization is biased to easy instructions and 2) canonical RL training recipes typically employ a cumulative reward (the number of instructions fulfilled), **treating all the instructions equally**; the policy model is biased to fulfill the easy instructions to get the same amount of rewards. To address these, we first propose two metrics to measure the exploration bias of instruction following and then introduce a two-stage framework to alleviate it: 1) Behavioral Bootstrapping: a lightweight rejection sampling fine-tuning before RL to activate hard instructions and 2) Scarcity-Aware Rewards: a new RL reward function that assigns rewards to instructions based on their empirical scarcity. Experiments show that the proposed metrics are highly correlated with model performance and our methods unleash the potential of RL training: our best models outperform the baselines by a significant margin across three verifiable instruction following benchmarks. We release code at https://github.com/mianzhang/MulIF.

1 Introduction

Since the advent of ChatGPT (Ouyang et al., 2022), instruction following has become a cornerstone capability of Large Language Models (LLMs). Reinforcement Learning (RL) has emerged as a powerful paradigm for enhancing the instruction-following capabilities of LLMs. Following Pyatkin et al. (2025) and Peng et al. (2025), RL for Instruction Following (RLIF) can be formulated as optimizing a policy π_θ to generate a response y that adheres to a specific set of instructions (or constraints)[1] $\mathcal{I} = \{I_1, I_2, \ldots, I_N\}$ while fulfilling a core task (prompt) x. This optimization problem is modeled as maximizing the expected return of a reward function $R(y, \mathcal{I})$:

$$\max_\theta \mathbb{E}_{(x,\mathcal{I})\sim\mathcal{D}} \left[\mathbb{E}_{y\sim\pi_\theta(\cdot|x,\mathcal{I})}[R(y,\mathcal{I})]\right]$$

Here, $\mathcal{D}$ represents the distribution of prompt-instruction pairs, and $R(y, \mathcal{I})$ can be either deterministic programs or reward models.

Canonical RLIF training recipes typically employ a cumulative reward function (Pyatkin et al., 2025; Guo et al., 2025):

$$R(y, \mathcal{I}) = \sum_{j=1}^{N} \nVdash(I_j, y)$$

where $\nVdash(I_j, y)$ is an indicator function denoting whether the j-th instruction is satisfied in the rollout y. While intuitive, this objective introduces a *exploration bias* towards easy instructions. Since easy instructions are more frequently satisfied during initial exploration, they provide a dense and stable gradient signal that dominates the optimization process. As a result, the model gradually prioritizes low-hanging fruit to maximize immediate cumulative rewards while neglecting hard instructions. This imbalance leads to a stagnation in exploring the sparse frontier of complex constraints, hindering the model's ability to achieve an "all-pass" state in complex scenarios.

We train Qwen3-1.7B using GRPO (Shao et al., 2024) with the cumulative reward function on IF-Train (Pyatkin et al., 2025), which consists of prompts containing up to five instructions and track the performance gain by comparing the accuracies of the base model against the RL-trained model

*Equal contribution.

[1]We use "instructions" and "constraints" interchangeably.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 16624–16633
October 24-29, 2026 ©2026 Association for Computational Linguistics

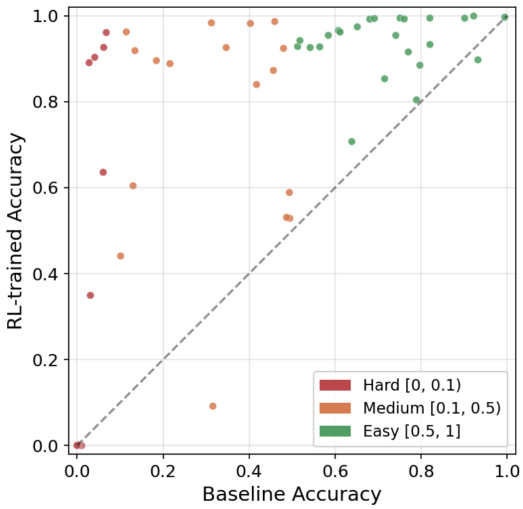

Figure 1: Instruction Accuracy: Before vs After RL

across various instruction types. As illustrated in Figure 1, the impact of RL training varies significantly across different instructions. We categorize instruction types into Easy, Medium, and Hard based on their baseline performance. The results provide *empirical evidence* of the exploration bias: RL provides more consistent and robust improvements for Easy instructions; in contrast, performance gains for Medium and Hard instructions exhibit high variance. We also notice that **certain instructions with very low initial accuracy benefit minimally from RL training**; this suggests another reason for the exploration bias that extends beyond the limitations of the cumulative reward.

To address the exploration bias in RLIF, we first introduce two diagnostic metrics: Variation of Instruction Accuracy, which measures the imbalance in individual instruction mastery, and Variance of Synergy Accuracy, which assesses the model's uneven capacity to satisfy diverse instructions simultaneously. We experimentally demonstrate that both metrics exhibit a strong negative correlation with the prompt-level "all-pass" rate. Building on these insights, we propose a two-stage framework to alleviate the exploration bias: 1) Behavioral Bootstrapping, a lightweight Rejection Sampling Fine-Tuning phase designed to **activate hard instructions before reinforcement learning**; and 2) Scarcity-Aware Rewards, a reward mechanism that **prioritizes the discovery of rare instructions and combinations** based on their empirical scarcity.

We conduct extensive experiments using Qwen3-1.7B and Qwen2.5-7B-Instruct across three verifiable instruction following benchmarks. Our results demonstrate that models integrated with Behavioral Bootstrapping and Scarcity-Aware Rewards significantly outperform the standard cumulative reward baseline by an average of 4.5 points across all evaluated benchmarks and achieve substantial gains of up to 9.2 points on our primary test set, IFBench, enabling these relatively small models to match the performance of much larger off-the-shelf models such as Llama-3.1-70B and GPT-OSS-20B. Finally, we conduct ablation studies to reveal the impact of our proposed framework under varying configurations, providing practical insights for their optimal implementation.

2 Related Work

2.1 Instruction Following

Precisely following instructions is a fundamental capability of LLMs (Zhou et al., 2023; Pyatkin et al., 2025). Unlike general instruction tuning (Chung et al., 2022; Wang et al., 2023), which focuses on teaching models basic skills to understand user intent and solve tasks, precise instruction following is a more rigorous challenge. It requires LLM agents to adhere strictly to specific constraints within a prompt, a factor critical for the reliability and evolution of autonomous agents (Wu et al., 2024; Fu et al., 2025). As tasks grow in complexity, the number of instructions per task continues to increase (Qi et al., 2025; He et al., 2025; Pyatkin et al., 2025), yet empirical evidence suggests that model performance often degrades as constraint density rises (Jaroslawicz et al., 2025; Guo et al., 2025). RL has proven effective for improving instruction following (Bercovich et al., 2025; Pyatkin et al., 2025; Peng et al., 2025). Our work focuses on alleviating the exploration bias in RL optimization when the model is confronted with multiple, diverse instructions.

2.2 Bias in RL Training

RL training of LLMs is known to introduce systematic biases. A well-studied instance is *length bias*, where policies exploit verbosity as a reward shortcut (Singhal et al., 2024; Chen et al., 2024). RL fine-tuning also reduces output diversity through *mode collapse* (Kirk et al., 2024), and in the verifiable-reward regime, *entropy collapse* causes the policy to lose exploratory capacity early in training (Yu et al., 2026). When training involves multiple objectives or tasks, optimization imbalance becomes more pronounced. Wu et al.

(2026) show that gradient magnitudes across tasks can differ by up to 33× in RL post-training, causing easy tasks to dominate, and multi-objective RLHF methods must carefully balance conflicting alignment objectives to avoid one dominating others (Dai et al., 2024; Moskovitz et al., 2024). Recent work on RLVR has further identified *exploration bottlenecks*, where low initial accuracy on hard problems yields sparse rewards and creates a self-reinforcing loop that prevents learning (Chen et al., 2026). While these works address biases at the reward, algorithm, or task level, the exploration bias *within* a single prompt, where instructions of varying difficulty coexist, remains unexplored.

3 Measure of Exploration Bias

Given a prompt with a set of instructions $\mathcal{I} = \{I_1, I_2, \ldots, I_N\}$, we define the marginal satisfaction probability $P(I_j)$ as the probability that a response y sampled from the policy π_θ satisfies I_j:

$$P(I_j) = \mathbb{P}_{y \sim \pi_\theta}(\mathbb{1}(I_j, y) = 1)$$

Standard evaluations of instruction following typically focus on maximizing *prompt accuracy* or more granular *instruction accuracy*. However, neither metric adequately captures the model's exploration bias. We propose two metrics based on the intuition that a model should maintain balanced capability in satisfying instructions and synergizing possible instruction pairs within a single response.

Variation of Instruction Accuracy (VIA). VIA serves as a metric for the imbalance in a model's individual instruction mastery. Given a prompt with N instructions, VIA is calculated as the *variance of the marginal satisfaction probabilities* $\{P(I_j)\}_{j=1}^{N}$ *across the instruction set*. A high VIA indicates that the model consistently excels at certain instructions while failing others.

Variance of Synergy Accuracy (VSA). While VIA focuses on individual instructions, VSA measures the imbalance in the model's capability to synergize instruction pairs. We first define the pairwise synergy probability $P(I_j, I_k)$ for any pair of instructions (I_j, I_k) as the ratio of their joint satisfaction probability to their individual bottleneck:

$$P(I_j, I_k) = \frac{P(I_j \cap I_k)}{\min(P(I_j), P(I_k))} \quad (1)$$

where $P(I_j \cap I_k)$ is the joint probability that a single response satisfies both I_j and I_k. VSA is then calculated as the *variance of the pairwise synergy probabilities across all unique instruction pairs* $\mathcal{P} = \{(j,k) \mid 1 \leq j < k \leq N\}$ *where* $P(I_j) > 0$ *and* $P(I_k) > 0$. A high VSA indicates that the model is significantly better at synergizing certain instruction pairs while failing others.

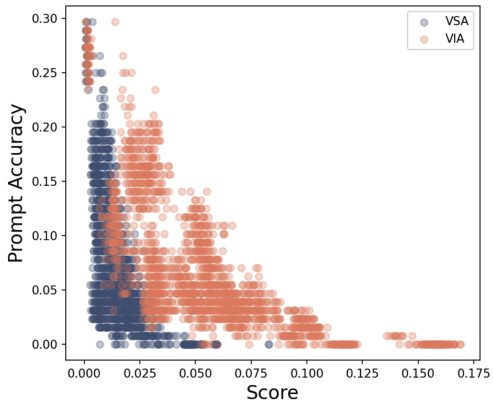

Figure 2: VIA and VSA vs Prompt Accuracy

Relationship with Prompt Accuracy To investigate the relationship between VIA, VSA, and prompt accuracy, we conduct a controlled experiment. We utilize test prompts from AdvancedIF (He et al., 2025), which contain diverse multiple instructions for each prompt. We construct a comprehensive response pool by aggregating outputs from a wide range of popular models, including GPT-5.4, GPT-5.4-mini, GPT-5, GPT-4o, GPT-OSS-120B, GPT-OSS-20B, Qwen-3.5-35B-A3B, Qwen-3.5-9B, Qwen-3-32B, Qwen-3-8B, Qwen-3-1.7B, Olmo-3.1-32B-Instruct, Olmo-3-7B-Instruct, Llama-3.2-3B, and Llama-3.1-8B. We then repeatedly sample response groups from this pool and **only keep those satisfying a fixed number of instructions** (same instruction accuracy). For each sampled group of size K, we derive a verification matrix $\mathbf{M} \in \{0,1\}^{K \times N}$, where $\mathbf{M}_{i,j} = 1$ if the i-th response satisfies the j-th instruction, and 0 otherwise. Based on $\mathbf{M}$, we estimate the marginal satisfaction probability $P(I_j)$ and the pairwise synergy probability $P(I_j \cap I_k)$ using the following empirical frequencies:

$$P(I_j) \approx \frac{n_j}{K}, \quad P(I_j \cap I_k) \approx \frac{m_{j,k}}{K}$$

where $n_j = \sum_{i=1}^{K} \mathbf{M}_{i,j}$ represents the success count for instruction I_j, and $m_{j,k} = \sum_{i=1}^{K} (\mathbf{M}_{i,j} \cdot \mathbf{M}_{i,k})$ denotes the joint success count for the instruction pair (I_j, I_k). The pairwise synergy probability is calculated by substituting these frequency-

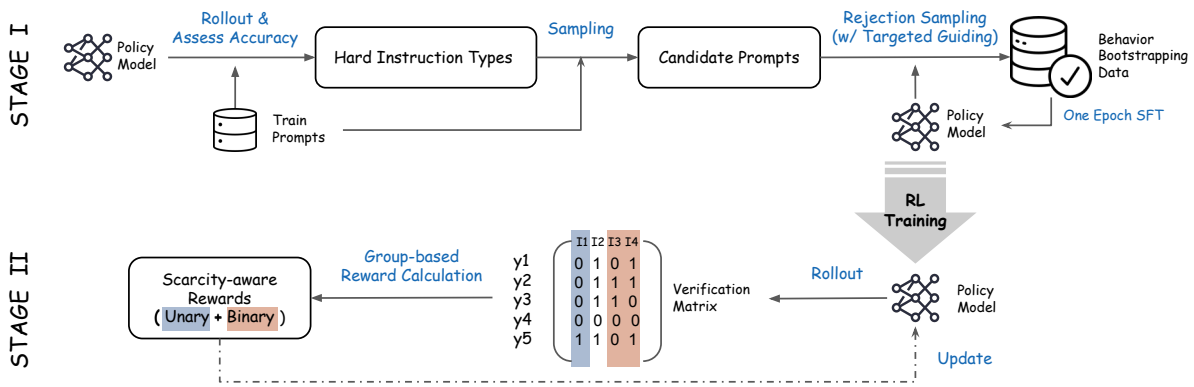

Figure 3: Overview of the framework for mitigating exploration bias. **Top** (Behavioral Bootstrapping): The policy model is first evaluated on the training set to identify hard instruction types. Prompts containing at least one hard instruction are then sampled, and a subsequent rejection sampling phase is conducted to collect bootstrapping data. Finally, the policy model is supervised fine-tuned for one epoch on this collected data. **Bottom** (Scarcity-Aware Rewards): The bootstrapped model is fine-tuned via RL using Scarcity-Aware Rewards (Unary + Binary), which are calculated across a response group to incentivize the exploration of rare instructions and combinations.

based estimates into Eq. 1. We employ a group size of $K = 128$ and fix the instruction accuracy at 0.68 ± 0.01, which corresponds to the mean instruction accuracy across the evaluated models. As illustrated in Figure 2, **both VIA and VSA exhibit a consistent inverse relationship with prompt accuracy**: a reduction in variance, whether in individual instruction mastery or pairwise synergy, strongly correlates with higher prompt-level success rates. These results empirically validate that **models with minimized exploration bias are significantly more likely to fulfill the entire set of instructions within a prompt simultaneously**.

4 Mitigating Exploration Bias

As mentioned in Section 1, two factors contribute to the exploration bias of RLIF: the policy model's low initial capability regarding certain instructions and a standard cumulative reward formulation that treats all instructions equally. To alleviate these, we propose a two-stage framework including 1) lightweight behavioral bootstrapping to first activate difficult instructions and 2) a new reward design inspired by the metrics proposed in Section 3 that assigns rewards based on the empirical scarcity of instructions. Figure 3 provides an overview.

4.1 Behavioral Bootstrapping

In multi-instruction scenarios, complex constraints often suffer from low initial accuracy, rendering standard RL ineffective due to the lack of successful exploration. To bridge this gap, we introduce Behavioral Bootstrapping via a lightweight Rejection Sampling Fine-Tuning phase. We identify seed instruction types $\mathcal{T}_{seed}$ that fall below a predefined accuracy threshold τ and curate a minimal dataset $\mathcal{D}_{seed}$ of successful rollouts for them. For each target type t, we sample K responses per prompt, selecting those that satisfy the target instruction while maximizing the coverage of other instructions in a prompt. If the base policy fails to generate a valid response, we employ a *Targeted Guiding* prompting by appending a suffix *"Focus on fulfilling [Instruction] first"* to nudge the model toward the target instruction. After collecting $\mathcal{D}_{seed}$ with at most N_{target} for each selected instruction type, the initial policy is supervised fine-tuned on $\mathcal{D}_{seed}$ for a single epoch to boost the initial capability of hard instructions. This establishes an ideal base for subsequent RL training. The whole process is detailed in Algo. 1.

4.2 Scarcity-Aware Rewards

Standard cumulative reward treats instructions equally during RL training. However, instructions vary in difficulty, causing the model to develop a bias toward exploring easier instructions because fulfilling an easy instruction receives the same reward as a difficult one. To address this, inspired by the metrics introduced in Section 3, we design a new reward function that encourages the model to explore rare instructions or rare combinations.

The intuition of our reward design is **hard instructions satisfaction should receive more rewards and easy ones should receive less** and we use *empirical scarcity* in a response group

Algorithm 1: Behavioral Bootstrapping

Input : Initial policy π_θ, Data $\mathcal{D}_{train}$, Threshold τ, Sample size K, Target count N_{target}
Output : Bootstrapped policy $\pi_{\theta'}$

▷ *Step 1: Evaluation of initial policy*
Evaluate π_θ on $\mathcal{D}_{train}$ to compute accuracy $\text{Acc}(t)$ for each instruction type $t \in \mathcal{T}$;
Identify seed types $\mathcal{T}_{seed} = \{t \in \mathcal{T} \mid \text{Acc}(t) < \tau\}$;
$\mathcal{D}_{seed} \leftarrow \emptyset$;
▷ *Step 2: Seed data curation via RSFT*
foreach instruction type $t \in \mathcal{T}_{seed}$ **do**
  while $|\mathcal{D}_{seed,t}| < N_{target}$ **do**
    Sample a prompt x from $\mathcal{D}_{train}$ containing an instruction $I \in t$;
    Generate $\mathcal{Y} = \{y_1, \ldots, y_K\} \sim \pi_\theta(\cdot|x)$;
    $\mathcal{Y}_{valid} = \{y \in \mathcal{Y} \mid y \text{ satisfies } I\}$;
    if $\mathcal{Y}_{valid} = \emptyset$ **then**
      ▷ *Targeted guiding*
      $x \leftarrow x +$ *"Focus on fulfilling [I] first."* ;
      $\mathcal{Y}_{valid} \leftarrow \{y \sim \pi_\theta(\cdot|x) \mid y \text{ satisfies } I\}$;
    if $\mathcal{Y}_{valid} \neq \emptyset$ **then**
      ▷ *Maximize auxiliary coverage*
      Select y^* with highest instruction coverage;
      $\mathcal{D}_{seed} \leftarrow \mathcal{D}_{seed} \cup \{(x, y^*)\}$;
Supervised fine-tune π_θ on $\mathcal{D}_{seed}$ for 1 epoch $\rightarrow \pi_{\theta'}$;

during training to estimate the difficulty. Specifically, we first let the model generate K responses for a prompt containing a set of instructions $\mathcal{I} = \{I_1, I_2, \ldots, I_N\}$, and employ verification functions to obtain a verification matrix $\mathbf{M} \in \{0,1\}^{K \times N}$. We then estimate the model's marginal satisfaction probability for an instruction, $P(I_j)$, and the pairwise synergy probability, $P(I_j, I_k)$, based on current model performance (see Section 3). The reward for instruction I_j is defined as $1 - P(I_j)$, and for an instruction pair as $1 - P(I_j, I_k)$. This design ensures that RL optimization treats instructions differently. The final reward for rollout R_i is formulated as:

$$\mathcal{R}(R_i) = \underbrace{\frac{1}{N}\sum_{j=1}^{N} \mathbf{M}_{i,j}\left(1 - \frac{n_j}{K}\right)}_{\text{Unary Reward}} + \underbrace{\frac{\alpha}{|\mathcal{P}_i|}\sum_{(j,k)\in\mathcal{P}_i}\left(1 - \frac{\sum_{m=1}^{K}\mathbf{M}_{m,j}\mathbf{M}_{m,k}}{\min(n_j, n_k)}\right)}_{\text{Binary Reward}} \tag{2}$$

where $n_j = \sum_{i=1}^{K} \mathbf{M}_{i,j}$ denote the total successes for instructions j in the response group. $\mathcal{P}_i = \{(j,k) \mid j < k, \mathbf{M}_{i,j} \cdot \mathbf{M}_{i,k} = 1\}$ is the set of satisfied instruction pairs in a response. If $n_j = 0$ or $n_k = 0$, the binary reward is set to 0. α is a balancing factor. The Unary Reward scales the contribution of each satisfied instruction by its empirical scarcity; instructions that are rarely satisfied by the current policy receive higher weights. The Binary Reward focuses on the difficulty of instruction "co-occurrence." By normalizing the intersection against the minimum of individual successes $\min(n_j, n_k)$, it specifically penalizes pairs that are easy to satisfy separately but difficult to satisfy together, thereby encouraging the model to overcome synergy bottlenecks. Please note that while our proposed reward can theoretically be integrated into any policy gradient method, group-based approaches like GRPO (Shao et al., 2024) are a more natural choice because they utilize a response group to estimate advantages for optimization. Our reward can be derived directly from these samples with negligible additional cost.

5 Experiments

5.1 Experimental Settings

We select *verifiable instruction following* as the testbed to evaluate our proposed methods. This choice is motivated by the fact that verification can be executed via deterministic functions, yielding 100% accuracy. In contrast, soft constraints require a model-based verifier, which frequently suffers from reliability issues, even when state-of-the-art models are employed (Wen et al., 2026), and thus interfering with the evaluation process.

Training We utilize the Qwen3-1.7B (Yang et al., 2025) and Qwen2.5-7B-Instruct (Qwen et al., 2024) as the primary base models for our experiments. Training is conducted using IFTrain (Pyatkin et al., 2025), a dataset with prompts containing rule-based instructions. To specifically investigate model behavior in multi-instruction scenarios, we filter the original dataset to retain only prompts containing 3 to 5 instructions, resulting in a specialized training set of 45,374 prompts. Regarding Behavioral Bootstrapping, we use an instruction selection threshold τ of 0.5 and the target seed count N_{target} of 100 ($\mathcal{D}_{seed} = 900$). For the RL phase, we employ GRPO with a response group size of 16 and a total batch size of 512. We use a learning rate of 1×10^{-6} and set the reward balance factor α (Eq. 2) to 2. Based on initial runs, the maximum training duration is set to 400 steps. All experiments are executed on four NVIDIA A100 GPUs.

Model	*IFBench*		*IFEval*		*Multi-IF*			Average
	Loose	Strict	Loose	Strict	Turn 1	Turn 2	Turn 3	
Off-the-shelf								
GPT-OSS-120B	65.2	59.6	86.6	83.7	76.4	65.1	54.8	70.2
Llama-3.1-70B	35.0	29.8	83.7	81.8	80.5	71.9	63.1	63.7
Qwen3-32B	31.9	28.0	89.0	86.1	84.1	74.3	60.6	64.8
Gemma-3-27B	33.3	25.6	82.9	80.2	82.5	72.9	63.7	63.0
GPT-OSS-20B	40.0	37.5	82.8	79.6	61.4	52.4	42.1	56.5
Qwen3-8B	31.2	26.6	85.7	82.4	82.6	73.1	64.4	63.7
Mistral-7B-Instruct-v0.3	17.1	14.3	45.8	42.3	44.0	30.5	19.6	30.5
Gemma-3-4B	28.4	22.4	77.0	73.1	76.3	63.7	53.0	56.3
Qwen3-1.7B (Qwen1.7B)								
Qwen1.7B	23.1	18.5	70.2	67.6	67.3	55.2	44.2	49.4
Qwen1.7B-CR	41.1	35.1	86.8	84.1	81.2	59.2	46.7	62.0
Qwen1.7B-BeBoot-CR*	$43.1^{\uparrow 2.0}$	$39.2^{\uparrow 4.1}$	$90.0^{\uparrow 3.2}$	$88.3^{\uparrow 4.2}$	$78.3^{\downarrow 2.9}$	$61.5^{\uparrow 2.3}$	$48.9^{\uparrow 2.2}$	$64.2^{\uparrow 2.2}$
Qwen1.7B-BeBoot-SaR*	$\mathbf{50.1}^{\uparrow 9.0}$	$\mathbf{44.3}^{\uparrow 9.2}$	$\mathbf{90.7}^{\uparrow 3.9}$	$\mathbf{89.0}^{\uparrow 4.9}$	$\mathbf{85.4}^{\uparrow 4.2}$	$\mathbf{62.9}^{\uparrow 3.7}$	$\mathbf{50.4}^{\uparrow 3.7}$	$\mathbf{67.5}^{\uparrow 5.5}$
Qwen2.5-7B-Instruct (Qwen7B)								
Qwen7B	30.5	27.7	75.0	71.9	75.0	59.0	47.6	55.2
Qwen7B-CR	42.4	39.5	91.1	90.0	**88.0**	68.2	55.9	67.9
Qwen7B-BeBoot-CR*	$48.4^{\uparrow 6.0}$	$43.1^{\uparrow 3.6}$	$92.6^{\uparrow 1.5}$	$91.2^{\uparrow 1.2}$	$84.7^{\downarrow 3.3}$	$68.3^{\uparrow 0.1}$	$\mathbf{57.7}^{\uparrow 1.8}$	$69.4^{\uparrow 1.5}$
Qwen7B-BeBoot-SaR*	$\mathbf{52.6}^{\uparrow 10.2}$	$\mathbf{48.4}^{\uparrow 8.9}$	$\mathbf{94.0}^{\uparrow 2.9}$	$\mathbf{92.9}^{\uparrow 2.9}$	$86.5^{\downarrow 1.5}$	$\mathbf{69.3}^{\uparrow 1.1}$	$56.6^{\uparrow 0.7}$	$\mathbf{71.5}^{\uparrow 3.6}$

Table 1: Performance on Instruction Following Benchmarks. All off-the-shelf model results are reproduced by us. For the Qwen3 series, we use their non-thinking mode, while for GPT-OSS models, we use the default (medium) thinking effort. The best performance within each column for the same base model is boldfaced and the performance comparison of our methods with the baselines (Qwen1.7B-CR and Qwen7B-CR) is annotated.

For further details on the sensitivity of τ, N_{target} and α, please refer to Section 5.3.

Evaluation The model's performance is evaluated on three precise instruction following benchmarks: IFBench (Pyatkin et al., 2025), IFEval (Zhou et al., 2023), and Multi-IF (He et al., 2024). IFBench serves as our primary test set, featuring out-of-domain instruction types relative to the training data. In contrast, IFEval functions as an in-domain benchmark, as its instruction types overlap with those in the training set. Multi-IF assesses the models' proficiency in following multi-turn and multilingual instructions. Consistent with the settings recommended by the Qwen team for optimal output quality, all generations are conducted with a temperature of 0.7. We follow the original papers to report loose and strict prompt accuracy for IFBench and IFEval and an average of four types of accuracy scores on each turn.

5.2 Main Results

Model Performance Table 1 shows the model performance on the benchmarks. The proposed mitigation methods for exploration bias, Behavioral Bootstrapping (BeBoot) and Scarcity-Aware Rewards (SaR), significantly enhance the instruction-following capabilities of the RL-trained models, outperforming the cumulative reward (CR) baselines **with an average gain of 5.5 on 1.7B base model and 3.6 on 7B base model across the evaluated benchmarks**. Most notably, our best models, Qwen1.7B-BeBoot-SaR and Qwen7B-BeBoot-SaR, achieve scores of 44.3 and 48.4 strict accuracy, respectively, **with substantial gains of 9.2 and 8.9 points over the baselines on our primary test set IFBench**. This indicates that the models trained with BeBoot and SaR have better generalization to unknown instruction types. This allows our small models to match the overall performance of much larger off-the-shelf models like Llama-3.1-70B and GPT-OSS-120B. We can also see that while BeBoot provides a critical initial boost by addressing early exploration failures, the integration of SaR proves essential for peak performance, as evidenced by the additional performance increase when comparing -BeBoot-SaR to -BeBoot-CR. These findings validate that mitigating exploration bias through targeted bootstrapping and scarcity-aware RL is effective for mastering complex instruction sets.

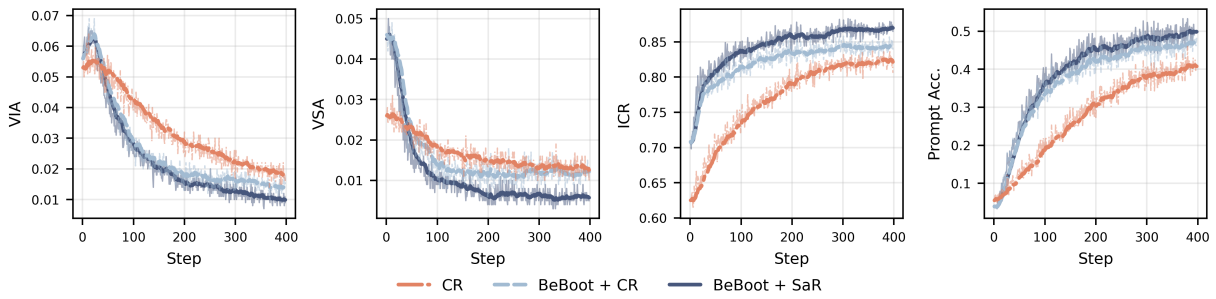

Figure 4: The training curves of VIA, VSA, ICR, and Prompt Accuracy.

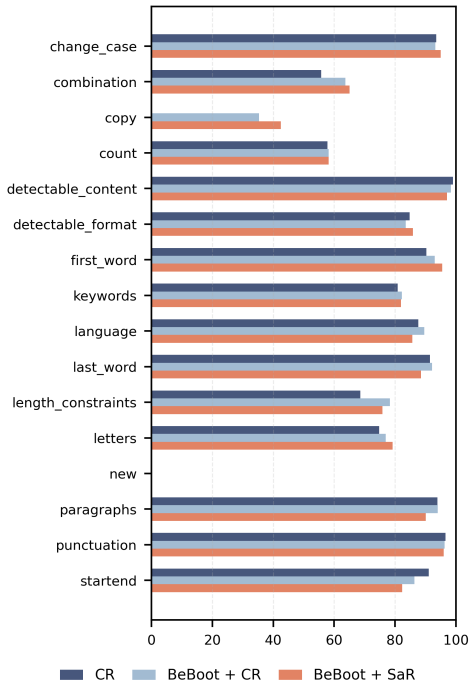

Figure 5: Accuracy Shift by Instruction Type

Exploration Bias The training curves in Figure 4 illustrate how our proposed framework mitigates exploration bias compared to the standard cumulative reward (CR) baseline. We estimate the Variation of Instruction Accuracy (VIA) and Variance of Synergy Accuracy (VSA) for each prompt based on the response groups as described in Section 3. Additionally, we visualize the Instruction Coverage Rate (ICR), which measures the proportion of instructions fulfilled at least once by the response group, and prompt accuracy. We can see that while the standard CR approach exhibits a slow, gradual reduction in VIA and VSA, it consistently maintains higher levels of instructional imbalance. In contrast, methods utilizing BeBoot begin at a significantly higher ICR and prompt accuracy, successfully activating hard instructions that standard reinforcement learning typically neglects. Notably, **BeBoot+SaR achieves the fastest and most substantial decline in both VIA and VSA, reaching the lowest variance levels across all tested methods**. This reduction signifies a more balanced exploration of the instruction space, which directly facilitates the model's ability to satisfy multiple diverse instructions simultaneously. The trajectories for ICR and prompt accuracy further confirm that **BeBoot+SaR enables the model to address more instructions per prompt, resulting in superior prompt-level success on the training set**. Figure 5 provides a detailed decomposition of the pass rate shift across instruction types. We observe that **BeBoot and SaR yield significant improvements over the CR baseline on challenging categories** such as `combination`, `copy`, and `length_constraint`. Furthermore, the integration of both BeBoot and SaR outperforms the individual application of BeBoot. We also notice that all models achieve a 0% pass rate on the `new` category; this is likely due to extremely difficult instructions, such as: *"Copy the span of words that lies between (and including) index 13 and 43, where the indices refer to character positions."*

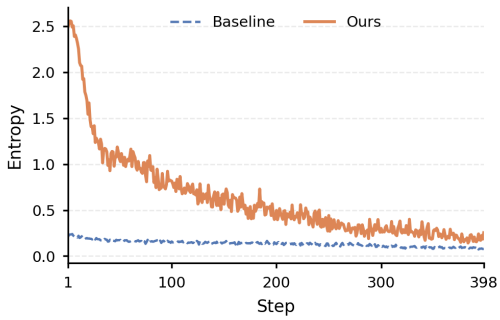

Figure 6: Policy Entropy

Entropy Figure 6 visualizes the entropy of the policy models, revealing that the baseline suffers from *entropy collapse*, losing the capacity to explore the sparse frontier of complex constraints. In contrast, our method maintains significantly higher

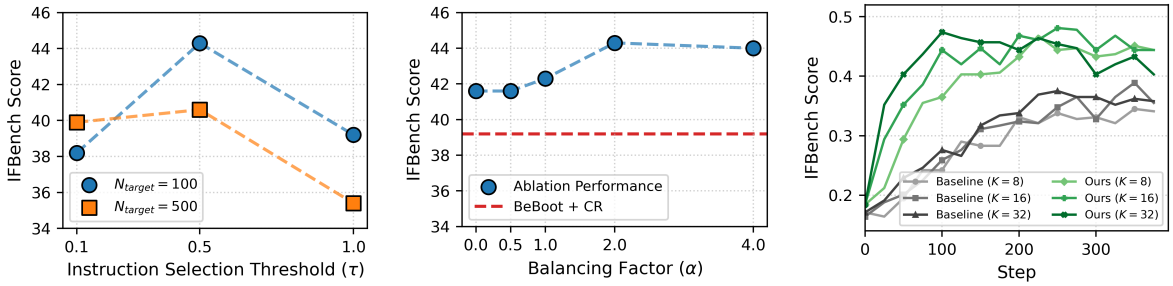

Figure 7: Ablation results on BeBoot data (left), reward balancing factor (middle), and response group size (right).

entropy: the model keeps exploring diverse responses, increasing the probability of finding a "all-pass" response. This is consistent with the better prompt accuracy of our models on training and evaluation benchmarks, further demonstrating the effectiveness of BeBoot and SaR.

5.3 Ablation

Impact of the Behavior BootStrapping Data According to our primary findings, Behavioral Bootstrapping (BeBoot) successfully activates instructions with low initial accuracy, leading to improved optimization efficiency during the reinforcement learning phase. We further investigate how the composition and scale of the bootstrapping data impact RL performance by conducting an ablation study on two critical hyperparameters: the instruction selection threshold τ and the target data count N_{target}. τ determines the data diversity and N_{target} determines data size. The sample size K for generating rollouts is fixed at 32, as we observed that if the model fails to produce a valid rollout within 32 attempts, the probability of successful generation remains exceedingly low. We evaluate τ at three levels: 0.1 (targeting hard instructions), 0.5 (targeting both hard and medium instructions), and 1.0 (targeting all the instructions). Additionally, we vary N_{target} between 100 and 500 to compare the effects of small-scale versus larger-scale seed data. Figure 7 (left) illustrates the model performance under different data settings. We observe that an instruction selection threshold of $\tau = 0.5$ consistently achieves superior results compared to $\tau = 0.1$ or $\tau = 1.0$. This indicates that **including a moderate diversity of non-easy instruction types is optimal for behavioral activation**. Furthermore, the smaller data scale ($N_{target} = 100$) generally outperforms the larger scale ($N_{target} = 500$). This suggests that **behavioral activation is most effective when restricted to a lightweight activation phase; excessive data likely leads to over-optimization**, which may limit the model's flexibility during the subsequent reinforcement learning stage.

Analysis of the Balancing Factor Figure 7 (middle) illustrates the sensitivity of the model's performance to the balancing factor α. We first observe that even when $\alpha = 0$ (employing only unary rewards), the model significantly outperforms the BeBoot+CR baseline, indicating the effectiveness of the scarcity-aware reward design. As the weight of the binary reward increases, performance remains stable from $\alpha = 0.0$ to 0.5 before trending upward to a peak score at $\alpha = 2.0$. This trend underscores that while small values may have little impact on training and excessively large values can lead to over-optimization of instruction combinations, properly balancing unary and binary rewards can effectively optimize the model performance.

Response Group Size The scarcity-aware rewards are calculated based on the entire response group. We study the effect of the response group size K by setting it to 8, 16, and 32. As shown in Figure 7 (right), our method robustly improves upon the corresponding baselines with the same group sizes. While larger group sizes lead to faster convergence, they may cause over-optimization, which we attribute to assigning too large rewards to rare instructions due to the large group size.

6 Conclusion

This work studies exploration bias in multi-instruction following with RL, where models prioritize easy instructions over complex ones. We introduce metrics to quantify the imbalanced ability of models and propose a two-stage framework with a lightweight behavioral bootstrapping phase to activate difficult instructions and scarcity-aware rewards to incentivize rare instruction satisfaction during RL training. Our results show the frame-

work enables small models to match the performance of larger off-the-shelf models.

Limitations

While our methods significantly mitigate exploration bias in multi-instruction following, several limitations remain:

- Our Scarcity-Aware Rewards design currently focuses on unary and binary rewards to optimize for individual instructions and pairwise synergies. This approach is based on the intuition that fulfilling a complex set of instructions can be effectively decomposed into mastering various instruction pairs. However, this may not explicitly address higher-order dependencies where three or more instructions interact simultaneously, potentially leaving some co-satisfaction bottlenecks unaddressed in extremely dense constraint scenarios. Theoretically, our framework can be generalized to n-ary scarcity rewards, which we leave for future investigation.

- Our framework requires identifying difficult instruction for behavior bootstrapping, which makes verifiable instruction following a natural testbed, where each instruction is associated with an explicit instruction type. For more open-ended instructions, instruction types could in principle be discovered via clustering, but reliable difficulty estimation and reward assignment would still depend on model-based verifiers. Because such verifiers are often less reliable and can introduce noise, we validate our framework only on verifiable instruction following in this work. Extending the method to broader open-ended instructions (e.g., style, politeness, and factuality) is an important direction for future research.

Acknowledgment

This research was supported by the OpenAI Researcher Access Program. The authors are solely responsible for the contents of the paper and the opinions expressed in this publication do not reflect those of the funding agencies.

References

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, and 117 others. 2025. Llama-nemotron: Efficient reasoning models. *arXiv [cs.CL]*.

Lichang Chen, Chen Zhu, Davit Soselia, Jiuhai Chen, Tianyi Zhou, Tom Goldstein, Heng Huang, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Odin: Disentangled reward mitigates hacking in rlhf. *ICML*.

Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2026. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *NeurIPS*, 38:57654–57689.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2022. Scaling instruction-finetuned language models. *arXiv [cs.LG]*.

Juntao Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. 2024. Safe rlhf: Safe reinforcement learning from human feedback. In *ICLR*, volume 2024, pages 50750–50777.

Tingchen Fu, Jiawei Gu, Yafu Li, Xiaoye Qu, and Yu Cheng. 2025. Scaling reasoning, losing control: Evaluating instruction following in large reasoning models. *arXiv [cs.CL]*.

Zhengkang Guo, Wenhao Liu, Mingchen Xie, Jingwen Xu, Zisu Huang, Muzhao Tian, Jianhan Xu, Yuanzhe Shen, Qi Qian, Muling Wu, Xiaohua Wang, Changze Lv, He-Da Wang, Hu Yao, Xiaoqing Zheng, and Xuanjing Huang. 2025. RECAST: Expanding the boundaries of LLMs' complex instruction following with multi-constraint data. *arXiv [cs.AI]*.

Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, Shruti Bhosale, Chenguang Zhu, Karthik Abinav Sankararaman, Eryk Helenowski, Melanie Kambadur, Aditya Tayade, Hao Ma, Han Fang, and Sinong Wang. 2024. Multi-IF: Benchmarking LLMs on multi-turn and multilingual instructions following. *arXiv [cs.CL]*.

Yun He, Wenzhe Li, Hejia Zhang, Songlin Li, Karishma Mandyam, Sopan Khosla, Yuanhao Xiong, Nanshu Wang, Xiaoliang Peng, Beibin Li, Shengjie Bi, Shishir G Patil, Qi Qi, Shengyu Feng, Julian Katz-Samuels, Richard Yuanzhe Pang, Sujan Gonugondla, Hunter Lang, Yue Yu, and 6 others. 2025. AdvancedIF: Rubric-based benchmarking and reinforcement learning for advancing LLM instruction following. *arXiv [cs.CL]*.

Daniel Jaroslawicz, Brendan Whiting, Parth Shah, and Karime Maamari. 2025. How many instructions can LLMs follow at once? *arXiv [cs.AI]*.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2024. Understanding the effects of rlhf on llm generalisation and diversity. In *ICLR*, volume 2024, pages 20620–20653.

Ted Moskovitz, Aaditya Singh, DJ Strouse, Tuomas Sandholm, Ruslan Salakhutdinov, Anca Dragan, and Stephen McAleer. 2024. Confronting reward model overoptimization with constrained rlhf. In *ICLR*, volume 2024, pages 21998–22025.

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 E Miller, Maddie Simens, Amanda Askell, P Welinder, P Christiano, J Leike, and Ryan J Lowe. 2022. Training language models to follow instructions with human feedback. *Adv. Neural Inf. Process. Syst.*, abs/2203.02155.

Hao Peng, Yunjia Qi, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. 2025. VerIF: Verification engineering for reinforcement learning in instruction following. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 30312–30327, Stroudsburg, PA, USA. Association for Computational Linguistics.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. 2025. Generalizing verifiable instruction following. *arXiv [cs.CL]*.

Yunjia Qi, Hao Peng, Xiaozhi Wang, Amy Xin, Youfeng Liu, Bin Xu, Lei Hou, and Juanzi Li. 2025. AGENTIF: Benchmarking instruction following of large language models in agentic scenarios. *arXiv [cs.AI]*.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2024. Qwen2.5 technical report. *arXiv [cs.CL]*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y K Li, Y Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv [cs.CL]*.

Prasann Singhal, Tanya Goyal, Jiacheng Xu, and Greg Durrett. 2024. A long way to go: Investigating length correlations in rlhf. *COLM*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Bosi Wen, Yilin Niu, Cunxiang Wang, Xiaoying Ling, Ying Zhang, Pei Ke, Hongning Wang, and Minlie Huang. 2026. IF-RewardBench: Benchmarking judge models for instruction-following evaluation. *arXiv [cs.CL]*.

Runzhe Wu, Ankur Samanta, Ayush Jain, Scott Fujimoto, Jeongyeol Kwon, Ben Kretzu, Youliang Yu, Kaveh Hassani, Boris Vidolov, and Yonathan Efroni. 2026. Imbalanced gradients in rl post-training of multi-task llms. In *Findings of EACL*, pages 3137–3150.

Xiaodong Wu, Minhao Wang, Yichen Liu, Xiaoming Shi, He Yan, Xiangju Lu, Junmin Zhu, and Wei Zhang. 2024. LIFBench: Evaluating the instruction following performance and stability of large language models in long-context scenarios. *arXiv [cs.CL]*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv [cs.CL]*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2026. Dapo: An open-source llm reinforcement learning system at scale. *NeurIPS*, 38:113222–113244.

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