Instructions are all you need: Self-supervised Reinforcement Learning for Instruction Following

Qingyu Ren[1]*, Qianyu He[1]*, Bowei Zhang[2], Jie Zeng[1], Jiaqing Liang[2]†, Yanghua Xiao[1]†
Weikang Zhou[3], Zeye Sun[3], Fei Yu[3]
[1]Shanghai Key Laboratory of Data Science, College of Computer Science and Artificial Intelligence, Fudan University, [2]School of Data Science, Fudan University, [3]Ant Group
{qyren24,qyhe21,bwzhang24, jzeng23}@m.fudan.edu.cn, {liangjiaqing, shawyh}@fudan.edu.cn

Abstract

Language models often struggle to follow multi-constraint instructions that are crucial for real-world applications. Existing reinforcement learning (RL) approaches suffer from dependency on external supervision and sparse reward signals from multi-constraint tasks. We propose a label-free self-supervised RL framework that eliminates dependency on external supervision by deriving reward signals directly from instructions and generating pseudo-labels for reward model training. Our approach introduces constraint decomposition strategies and efficient constraint-wise binary classification to address sparse reward challenges while maintaining computational efficiency. Experiments show that our approach generalizes well, achieving strong improvements across 3 in-domain and 5 out-of-domain datasets, including challenging agentic and multi-turn instruction following. The data and code are publicly available at https://github.com/Rainier-rq/verl-if.

1 Introduction

Instruction following capabilities (i.e., the ability to follow multiple constraints simultaneously) are crucial to ensure practical use of language models in real-world scenarios (Zhang et al., 2025; Li et al., 2025b). On one hand, real-world conversations with human users often contain multiple constraints in the instructions (Deshpande et al., 2025; Wen et al., 2024). On the other hand, reliable instruction following is essential for models in complex agentic tasks (Qi et al., 2025). However, instruction-tuned models often demonstrate inadequate adherence to user instructions (Team, 2024). Meanwhile, reasoning models, though specifically trained for various reasoning abilities (OpenAI, 2024; Guo et al., 2025; Seed et al., 2025), tend to sacrifice instruction-following abilities.

Existing methods for improving instruction-following abilities focus on obtaining high-

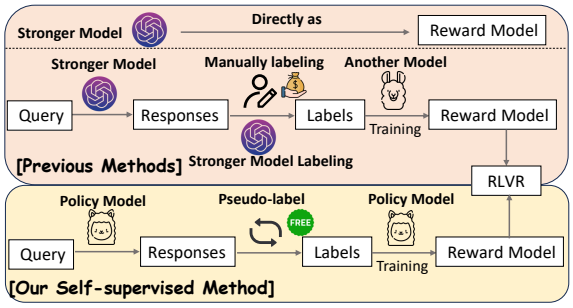

Figure 1: Comparison between our self-supervised RL method and previous methods. Our method does not rely on external sources to generate outputs or labels, which is both effective and efficient.

* Equal contribution.
† Corresponding author.

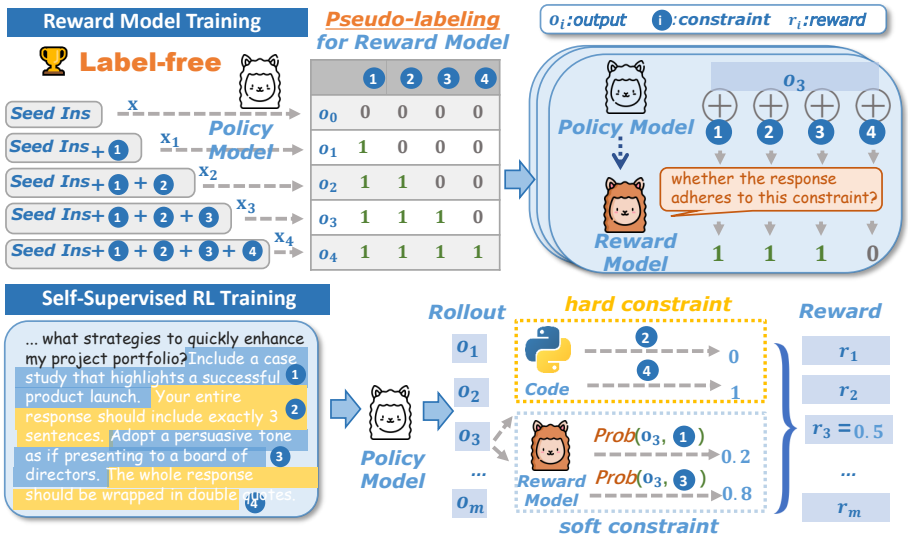

Figure 2: Overview of our self-supervised RL framework: from label-free instructions, we generate pseudo-labels for constraint-wise reward model training and then train the policy model with both hard constraints via rule-based verification and soft constraints via the reward model.

quality external supervision signals and outputs for training (Qin et al., 2025; Ren et al., 2025). These methods typically employ supervised fine-tuning (SFT) with distilled data from stronger models (Sun et al., 2024) or direct preference optimization (DPO) with collected pairwise preference data (He et al., 2024; Huang et al., 2025). However, reinforcement learning with verifiable rewards (RLVR) presents a more suitable paradigm for multi-constraint instruction following tasks. First, RLVR only requires instructions with verifiable reward signals, rather than high-quality outputs from external models. Moreover, constraints in complex instructions are inherently verifiable (Yang et al., 2025; Peng et al., 2025), making RLVR suited for our framework.

Existing RLVR approaches have following limitations. First, as shown in Fig. 1, the supervision signal problem: existing approaches rely heavily on stronger models, either directly as reward models or as sources for distilling reward model training data, or on manual data annotation (Qin et al., 2025). Stronger models are often proprietary or computationally expensive, and their inherent ability can also restrict the improvement of other models. Moreover, the sparse reward signal problem: multi-constraint instructions are difficult to satisfy completely, leading to sparse rewards that hinder effective learning (Yu et al., 2025a). Furthermore, generative reward models are computationally heavy, causing slow training, particularly for tasks with multiple constraints (Yu et al., 2025b).

To address these challenges, we propose an efficient self-supervised RL framework that enhances instruction following capabilities using only label-free instructions as inputs. Specifically, to tackle the supervision signal challenge, we develop a pseudo-label generation mechanism that the model can derive reward signals directly from instructions, achieving true label-free training without external depen-

dencies. By label-free, we emphasize that our approach requires no human annotations, preference datasets, or external reward models—the model generates all supervision signals autonomously from the instructions themselves. For the sparse reward issue, we introduce a multi-constraint decomposition strategy that breaks down multi-constraint instructions into simpler sub-tasks with incrementally increasing complexity, establishing dense learning signals throughout the training process (Yu et al., 2025a). Additionally, we design an efficient constraint-wise binary classification approach that evaluates each constraint independently, ensuring computational efficiency while maintaining high effectiveness. Our framework shows remarkable generalization abilities, showing consistent improvements across 3 in-domain and 5 out-of-domain datasets, including challenging agentic and multi-turn instruction following.

Overall, our contributions are as follows: (1) We propose a self-supervised RL framework that enhances instruction following capabilities using only label-free instructions as inputs. (2) We design an efficient and accurate reward modeling approach that handles multi-constraint scenarios through constraint-wise binary classification. (3) Extensive experiments demonstrate that our trained models achieve significant improvements in instruction following abilities with strong generalization to out-of-domain tasks, while maintaining general capabilities.

2 Related Work

2.1 Complex Instruction Following Improvement

Some works distill responses from stronger models through supervised fine-tuning (Sun et al., 2024; Qin et al., 2025) or collecting pairwise preference data for direct preference optimization (He et al., 2024; Qi et al., 2024). Others adopt self-play approaches that enhance model capabilities through code generation verification (Dong et al., 2024) or training additional refiner models (Cheng et al., 2024). Unlike these methods, we do not rely on stronger models and instead train exclusively through reinforcement learning based on the model's own abilities.

2.2 Reinforcement Learning for Complex Instruction Following

Some works use rule-based rewards for hard constraints (Lambert et al., 2024; Pyatkin et al., 2025), but these methods cannot generalize to soft constraints. Other works either rely on a stronger model as the reward model or utilize the stronger model to distill data for reward model training. Additionally, existing approaches often suffer from computational inefficiency. Different from these works, our reward model is independent of stronger reasoning models, requires no distillation, and operates more efficiently.

3 Method

Our framework consists of two main stages. First, as illustrated in Fig. 2, we construct a multi-constraint instruction dataset and decompose complex instructions into incremental constraint curricula to provide dense training signals. We then train constraint-wise binary classification reward models using pseudo-labels generated directly from the instructions. Second, we then apply GRPO (Shao et al., 2024) to optimize the policy model using the composite reward signals.

3.1 Dataset Construction

Our training dataset consists of two main components: (1) synthetically generated multi-constraint instructions, and (2) general reason-

ing data from math and science domains to maintain the model's overall capabilities.

Complex Instruction Synthesis. To ensure diversity in our multi-constraint instruction dataset, we cover both hard and soft constraint types (Ren et al., 2025). We begin by collecting a diverse set of 3,000 seed instructions from different sources (Köpf et al., 2024; Wang et al., 2022a,b; Li et al., 2025a). Subsequently, we use GPT-4o to add multiple constraints to these seed instructions. Our self-supervision supervises the outputs and labels, not the instructions. For hard constraints, we include 23 types, such as JSON format and frequency of all-capital words. For soft constraints, we include 25 types, such as style and role-based constraints. Details of the constraint types can be found in Appx. A.1.2.

General Reasoning Data Integration. To maintain the general abilities of the model, we integrate math and science data into our dataset. Specifically, we select 4,501 math problems from the DeepScaleR-Preview-Dataset (Luo et al., 2025) and 1,929 science questions from SciKnowEval (Feng et al., 2024).

Incremental Constraint Curriculum. Complex instruction-following is challenging (Zhang et al., 2024), leading to sparse reward signals during RL training (Yu et al., 2025a). We decompose complex instructions for progressive learning. Given a multi-constraint instruction x with constraints $\{c_1 \oplus c_2 \oplus \ldots \oplus c_n\}$, we create curriculum levels $\mathcal{L}_k$ where each level k contains sub-instruction x_k with the first k constraints: $x_k = x$ with constraints $\{c_1, c_2, \ldots, c_k\}$. This creates a progressive curriculum from single-constraint ($\mathcal{L}_1$) to full multi-constraint instructions ($\mathcal{L}_n$). Statistical details are in Tab. 1.

Curriculum	# Instruct.	# Cons.	# Soft.	#Hard.
$\mathcal{L}_1$	2806	2806	1578	1228
$\mathcal{L}_2$	2745	5490	3203	2287
$\mathcal{L}_3$	2700	8100	4833	3267
$\mathcal{L}_4$	2700	10800	6481	4319
$\mathcal{L}_5$	2619	13095	8101	4994

Table 1: Statistics of curricula. #Instruct, #Constraints, #Soft, and #Hard refer to the number of instruction constraints, total constraints, soft constraints, and hard constraints, respectively.

	Kendall Tau Coefficient	Position Consistency
Our Dataset	94.0	97.0

Table 2: Agreement between constructed dataset and human annotation. The detailed evaluation setup is provided in Appx. A.1.3

3.2 Reward Modeling

To model constraint satisfaction, we design different reward mechanisms for hard and soft constraints to produce constraint-level rewards.

Hard Constraint Modeling. For hard constraints that can be directly verified using explicit rules (Pyatkin et al., 2025), we adopt programmatic verification. For an input example (o, c) with response o and constraint c, we define a binary constraint-level reward function:

$$R_h(o, c) = \begin{cases} 1, & \text{if } o \text{ satisfies constraint } c \\ 0, & \text{otherwise} \end{cases}$$

Soft Constraint Modeling. To model soft constraints that cannot be verified through rules, avoid external supervision, and achieve efficiency, we train a binary classification reward model using pseudo-labels constructed from the instructions themselves without external labels.

During constraint decomposition, a natural relationship emerges: for constraint c_k, the response o_k (generated for instruction with constraint c_k) is likely to satisfy it, while o_{k-1}

(generated for instruction without c_k) does not. This allows us to construct pseudo-labels: (1) *Positive sample* $(o_k, c_k, label = 1)$: response satisfies the constraint, (2) *Negative sample* $(o_{k-1}, c_k, label = 0)$: response does not satisfy the constraint. We define a constraint-level reward function $f(o, c) \to [0, 1]$ that estimates the probability that response o satisfies constraint c. The model is trained to minimize the Binary Cross-Entropy loss:

$$\mathcal{L} = -\sum_{k=1}^{n} \left[\log f(o_k, c_k) + \log\left(1 - f(o_{k-1}, c_k)\right)\right].$$

As shown in Tab. 2, our self-supervised dataset demonstrates high consistency with humans.

3.3 RL Training

With the constraint-level reward signals established, we introduce how to utilize these models during reinforcement learning training to predict sample-level rewards for policy optimization.

Reward Model Usage During Training. For soft constraints, the trained reward model $f(o, c)$ takes a response o and constraint c as input, producing logits over two classes (satisfy/not satisfy). These logits are converted to probabilities:

$$R_s(o, c) = \frac{\exp(logits[1])}{\exp(logits[0]) + \exp(logits[1])}$$

This gives a scalar reward value $R_s(o, c) \in [0, 1]$ representing the probability that response o satisfies soft constraint c.

Constraint-Level Reward Prediction. We aggregate constraint-level rewards into sample-level rewards. For instruction x_k with constraints $\{c_1 \oplus c_2, \ldots, c_k\}$ and policy-generated response o_k, the sample-level reward is:

$$R_f = \frac{1}{k}\sum_{i=1}^{k} r_i, \quad r_i = \begin{cases} R_s(o_k, c_i), & \text{if } c_i \text{ is soft} \\ R_h(o_k, c_i), & \text{if } c_i \text{ is hard} \end{cases}$$

For reasoning tasks, we assign $R_f = 1$ for correct answers and $R_f = 0$ otherwise. This composite reward $R_f \in [0, 1]$ captures the overall constraint satisfaction of the response and serves as the reward signal for GRPO optimization.

4 Experiment

4.1 Experiment Setup

We experiment on both non-reasoning models (Qwen2.5-1.5B/7B-Instruct, Llama-3.1-8B-Instruct (Team, 2024; Grattafiori et al., 2024)) and reasoning models distilled from R1 (R1-Distill-Qwen-1.5B/7B, R1-0528-Qwen3-8B (Guo et al., 2025)). **IF** denotes models trained with our method. We select several models specifically optimized for instruction following as our baselines[1], and additionally include the strong models GPT-4o (Hurst et al., 2024) and QwQ-32B (Qwen, 2025) as baselines.

4.2 In-Domain Performance

As shown in Tab. 3, our method can significantly improve the constraint-following capability of models with different architectures and sizes on in-domain tasks. The largest gain is from Qwen2.5-1.5B-Instruct on IFEval (+21.6%). Our method outperforms instruction-following baselines. After training, Llama-3.1-8B-Instruct, initially weaker than SPAR-8B-DPO and VERIF-8B, surpasses or matches them[2]. The performance drop of 0528-Distill-Qwen3-8B-IF on WritingBench occurs because the base model has already been extensively

[1]Details about the baselines are provided in Appx. A.3.2.

[2]Detailed results are provided in Appx. A.3.6.

Models	Base Model	In-Domain			Out-of-Domain				
		IFEval	CFBench	FollowBench	ComplexBench	WritingBench	Collie	AgentIF	MultiChallenge
		Pr.(L)	ISR	HSR	Overall	Avg.	Avg.	CSR	Overall
GPT-4o	GPT	84.8	65.3	70.4	71.6	7.5	49.8	58.5	12.9
QwQ-32B	Qwen-32B	83.9	68.0	62.2	73.3	7.9	52.4	58.1	38.5
IR-1.5B	Qwen-1.5B	57.7	22.0	37.8	44.4	4.0	16.3	38.7	12.5
Conifer-7B-DPO	Mistral-7B	52.3	18.0	50.0	48.1	3.2	17.8	44.3	8.0
Crab-7B-DPO	Mistral-7B	57.7	25.0	49.4	59.0	4.5	19.6	47.2	14.1
SPAR-8B-DPO	LLaMA-8B	82.4	37.0	56.1	63.8	4.7	27.7	53.6	17.1
VERIF-8B	LLaMA-8B	87.1	41.0	56.9	54.7	5.1	28.3	56.6	15.0
Qwen2.5-1.5B-Instruct	Qwen-1.5B	43.6	22.0	34.6	45.9	4.5	13.0	42.8	12.0
Qwen2.5-1.5B-Instruct-IF	Qwen-1.5B	65.2(+21.6)	29.0(+7.0)	39.0(+4.4)	48.6(+2.7)	4.6(+0.1)	16.7(+3.7)	48.2(+5.4)	13.3(+1.3)
Qwen2.5-7B-Instruct	Qwen-7B	73.9	47.0	55.1	66.1	5.7	36.3	54.2	15.2
Qwen2.5-7B-Instruct-IF	Qwen-7B	78.9(+5.0)	52.0(+5.0)	57.5(+2.4)	68.7(+2.6)	5.8(+0.1)	38.0(+1.7)	56.7(+2.5)	15.6(+0.4)
Distill-Qwen-1.5B	Qwen-1.5B	42.3	17.0	22.7	39.8	3.9	14.0	37.5	12.0
Distill-Qwen-1.5B-IF	Qwen-1.5B	58.8(+16.5)	20.0(+3.0)	26.4(+3.7)	43.3(+3.5)	4.1(+0.2)	14.5(+0.5)	39.7(+2.2)	13.1(+1.1)
Distill-Qwen-7B	Qwen-7B	61.7	36.0	41.7	55.2	5.3	25.2	47.2	13.9
Distill-Qwen-7B-IF	Qwen-7B	71.7(+10.0)	42.0(+6.0)	49.1(+7.4)	60.7(+5.5)	5.6(+0.3)	27.0(+2.0)	46.9(-0.3)	16.3(+2.4)
Llama-3.1-8B-Instruct	LLaMA-8B	73.8	34.0	53.8	63.6	4.7	46.5	53.4	16.2
Llama-3.1-8B-Instruct-IF	LLaMA-8B	83.0(+9.2)	40.0(+6.0)	57.5(+3.7)	64.5(+0.9)	4.8(+0.1)	50.2(+3.7)	55.1(+1.7)	19.1(+2.9)
0528-Distill-Qwen3-8B	Qwen-8B	79.7	66.0	60.4	68.5	7.6	36.9	57.4	21.4
0528-Distill-Qwen3-8B-IF	Qwen-8B	87.1(+7.4)	68.0(+2.0)	63.8(+3.4)	71.1(+2.6)	7.1(-0.5)	37.1(+0.2)	63.5(+6.1)	28.7(+7.3)

Table 3: Overall performance on In-Domain and Out-of-Domain benchmarks. Pr.(L) denotes loose prompt.

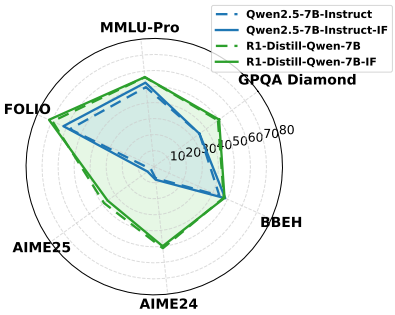

Figure 3: Performance on general benchmarks.

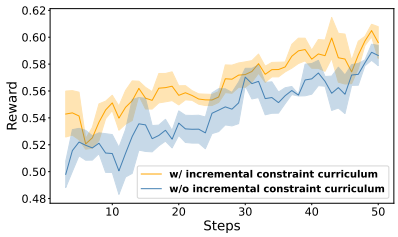

Figure 4: Comparison of reward density.

trained on writing data, and further training may lead to interference with its writing abilities.

4.3 Generalizability

Out-of-Domain Generalization. We select benchmarks containing constraints different from those in the training data to evaluate the model's out-of-domain generalization ability, including writing tasks, agent tasks, and multi-turn dialogues. As shown in Tab. 3, our method improves the model's instruction-following ability on out-of-domain tasks. Specifically, after training, 0528-Distill-Qwen3-8B achieves performance gains of 6.1 and 7.3 on AgentIF and MultiChallenge, respectively, and achieves state-of-the-art performance on AgentIF.

General Abilities. Besides instruction-following ability, we also evaluate the model's

Method	IFEval	CFBench	FollowBench	ComplexBench	Collie
	Pr. (L)	ISR	HSR	Overall	Avg.
R1-Distill-Qwen-7B-IF	71.7	42.0	49.1	60.7	27.0
Ablation Study					
w/o rule_based reward	-4.2	-2.0	-2.4	-1.7	-6.9
w/o incremental constraint curriculum	-4.0	-2.0	-3.6	-3.0	-1.9

Table 4: Ablation study results on reward modeling and incremental constraint curriculum.

Reward Model	4 constraint				5 constraint				Evaluation on Trained Policy Model		
	KT↑	PC↑	Time↓	Speedup↑	KT↑	PC↑	Time↓	Speedup↑	IFEval	CFBench	FollowBench
QwQ-32B	76.7	88.7	23.0s	×1.0	73.2	87.2	35.7s	×1.0	69.9	45.0	45.8
IF-Verifier-7B	58.7	80.7	5.2s	×4.4	**61.2**	82.0	7.4s	×4.8	69.1	41.0	41.6
BT Training Model	59.3	81.7	0.3s	×76.7	48.8	78.8	0.4s	×89.3	65.6	40.0	39.4
Our Reward Model	**62.7**	**83.3**	**0.2s**	**×115.0**	**61.2**	**83.4**	**0.3s**	**×119.0**	**71.7**	**42.0**	**49.1**

Table 5: Comparison of reward models on human-alignment (Kendall's Tau, Position Consistency), inference speed under varying constraints, and RL training performance on Distill-Qwen-7B.

general ability[3]. As shown in Fig. 3, we evaluate AIME using Avg@30 method, which refers to the average score across 30 sampled responses per question. The max output tokens is set to 32k. The results show that our method can maintain the model's general abilities. On certain benchmarks, the model's performance declines because instruction-following ability and general reasoning ability are two distinct capabilities — fine-tuning to enhance one often comes at the expense of the other.

4.4 Ablation Studies

Incremental Constraint Curriculum. To validate the effectiveness of incremental Constraint Curriculum, we conduct ablation studies under the following setting: w/o incremental constraint curriculum, which directly trains on multi-constraint instructions without decomposition. As shown in Tab. 4, this leads to the performance drop. As illustrated in Fig. 4, the model trained w/o incremental constraint curriculum receives sparser rewards, which hinders its ability to learn to follow constraints.

[3]Details about the benchmarks are in Appx. A.3.3.

Reward Modeling. We conduct ablation studies for reward modeling under the following settings: (1) **w/o rule_based reward**, which uses only the reward model to provide rewards. (2) **Comparison of different modeling methods**. We manually annotated 50 groups in the 5-constraint setting and 50 groups in the 4-constraint setting. In each group, an instruction is paired with a fixed number of constraints, along with candidate responses that satisfy from 1 up to all of the given constraints. We then compare the correlation between human-labeled rankings and the rankings produced by three modeling methods: (a) LLM-as-a-judge, (b) reward model trained with Bradley-Terry (BT) Loss, and (c) our reward model. We use two metrics **Kendall Tau coefficient** and **Position Consistency** to measure the consistency. As shown in Tab. 5, our reward model demonstrates better alignment with humans and achieves faster inference speed and better performance.

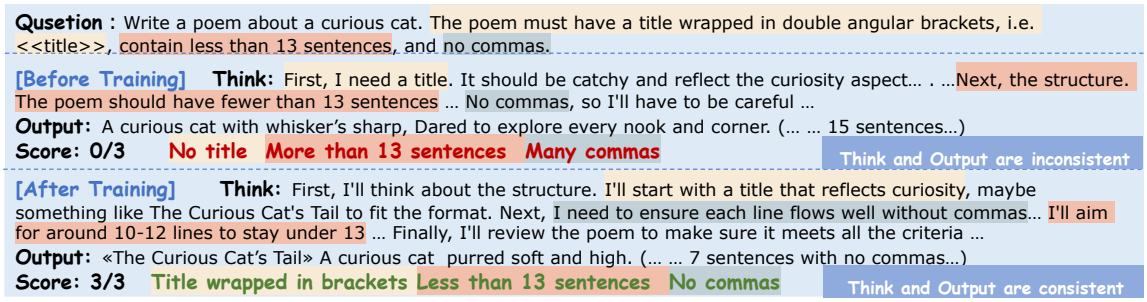

Figure 5: Qualitative analysis of consistency between model's thinking and outputs before and after training.

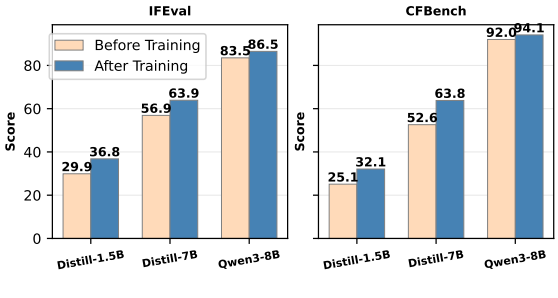

Figure 6: Quantitative analysis of consistency scores before and after model training.

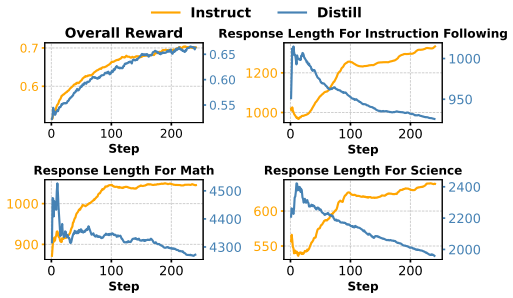

Figure 7: The reward and response length dynamics of Qwen2.5-7B-Instruct and R1-Distill-Qwen-7B.

4.5 Analysis

4.5.1 Consistency Analysis

We also analyze the reasons for performance improvement. From the **qualitative** perspective, as shown in Fig. 5, before training, the model's thinking process could take all constraints into account, but when generating answers it diverged from that thinking, resulting in none of the constraints being followed. After training, however, the model's answers remain consistent with its thinking process, and all constraints are followed. From the **quantitative** perspective, as shown in Fig. 6, we use GPT-4.1 to assess the consistency between the model's reasoning process and its responses on IFEval and CFBench. After training, our method can significantly enhance the consistency between the model's thinking and its responses, and this capability applies to a wide range of tasks, which also explains the effectiveness and strong generalization of the method.

4.5.2 Training Dynamics

As shown in Fig. 7, we analyze the dynamics during training on Qwen2.5-7B-Instruct and R1-Distill-Qwen-7B. First, rewards consistently increase during training for both models until convergence. Second, the two models exhibit different trends in response length on instruction-following tasks. The instruct model shows a continuous increase in response length, while the distill model first increases and then decreases. The former's length growth occurs because the instruct model initially lacks reasoning ability,

but RL gradually encourages such ability, leading to longer responses. In contrast, the latter's decline arises because the distilled reasoning model has been extensively reasoning tasks, which constrain its search space.

5 Conclusion

We propose a self-supervised RL framework that enhances instruction following without external supervision signals. Our approach features: (1) instruction decomposition for dense training signals, (2) self-supervised reward modeling using pseudo-labels generated directly from the instructions themselves, and (3) constraint-wise binary classification for efficiency. Experiments show significant improvements in instruction following while preserving general capabilities.

6 Limitations

Due to computational resource limitations, we have not validated our method on larger-scale models (e.g., 32B parameters), though our experiments on smaller models provide strong evidence of the method's effectiveness and scalability potential. Additionally, as our primary focus centers on reward modeling design, the construction of multi-constraint datasets remains relatively limited in diversity, and the current dataset construction process could benefit from incorporating a broader range of constraint types, domains, and complexity levels. Despite these limitations, our results provide compelling evidence for the effectiveness of the proposed approach, establishing a solid foundation for future scaling and enhancement efforts.

References

Jiale Cheng, Xiao Liu, Cunxiang Wang, Xiaotao Gu, Yida Lu, Dan Zhang, Yuxiao Dong, Jie Tang, Hongning Wang, and Minlie Huang. 2024. Spar: Self-play with tree-search refinement to improve instruction-following in large language models. *arXiv preprint arXiv:2412.11605*.

Kaustubh Deshpande, Ved Sirdeshmukh, Johannes Baptist Mols, Lifeng Jin, Ed-Yeremai Hernandez-Cardona, Dean Lee, Jeremy Kritz, Willow E Primack, Summer Yue, and Chen Xing. 2025. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 18632–18702.

Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. 2024. Self-play with execution feedback: Improving instruction-following capabilities of large language models. *arXiv preprint arXiv:2406.13542*.

Kehua Feng, Keyan Ding, Weijie Wang, Xiang Zhuang, Zeyuan Wang, Ming Qin, Yu Zhao, Jianhua Yao, Qiang Zhang, and Huajun Chen. 2024. Sciknoweval: Evaluating multi-level scientific knowledge of large language models. *arXiv preprint arXiv:2406.09098*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, and 1 others. 2022. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*.

Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. 2024. From complex to simple: Enhancing multi-constraint complex instruction

following ability of large language models. *arXiv preprint arXiv:2404.15846*.

Hui Huang, Jiaheng Liu, Yancheng He, Shilong Li, Bing Xu, Conghui Zhu, Muyun Yang, and Tiejun Zhao. 2025. Musc: Improving complex instruction following with multi-granularity self-contrastive training. *arXiv preprint arXiv:2502.11541*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2023. Followbench: A multi-level fine-grained constraints following benchmark for large language models. *arXiv preprint arXiv:2310.20410*.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K Jain, Virginia Aglietti, Disha Jindal, Peter Chen, and 1 others. 2025. Big-bench extra hard. *arXiv preprint arXiv:2502.19187*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, and 1 others. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Jijie Li, Li Du, Hanyu Zhao, Bo-wen Zhang, Liangdong Wang, Boyan Gao, Guang Liu, and Yonghua Lin. 2025a. Infinity instruct: Scaling instruction selection and synthesis to enhance language models. *arXiv preprint arXiv:2506.11116*.

Xiaomin Li, Zhou Yu, Zhiwei Zhang, Xupeng Chen, Ziji Zhang, Yingying Zhuang, Narayanan Sadagopan, and Anurag Beniwal. 2025b. When thinking fails: The pitfalls of reasoning for instruction-following in llms. *arXiv preprint arXiv:2505.11423*.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, and 1 others. 2025. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*.

MAA. 2024. American invitational mathematics examination - aime. Accessed in February 2024.

MAA. 2025. American invitational mathematics examination - aime. Accessed in February 2025.

OpenAI. 2024. Learning to reason with llms.

Hao Peng, Yunjia Qi, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. 2025. Verif: Verification engineering for reinforcement learning in instruction following. *arXiv preprint arXiv:2506.09942*.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. 2025. Generalizing verifiable instruction following. *arXiv preprint arXiv:2507.02833*.

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 preprint arXiv:2505.16944*.

Yunjia Qi, Hao Peng, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. 2024. Constraint back-translation improves complex instruction following of large language models. *arXiv preprint arXiv:2410.24175*.

Yulei Qin, Gang Li, Zongyi Li, Zihan Xu, Yuchen Shi, Zhekai Lin, Xiao Cui, Ke Li, and Xing Sun. 2025. Incentivizing reasoning for advanced instruction-following of large language models. *arXiv preprint arXiv:2506.01413*.

Team Qwen. 2025. Qwq-32b: Reward model for factuality and harmlessness. Accessed: 2025-05-13.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Qingyu Ren, Jie Zeng, Qianyu He, Jiaqing Liang, Yanghua Xiao, Weikang Zhou, Zeye Sun, and Fei Yu. 2025. Step-by-step mastery: Enhancing soft constraint following ability of large language models. *Preprint*, arXiv:2501.04945.

ByteDance Seed, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang, Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, and 1 others. 2025. Seed1. 5-thinking: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*.

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*.

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. 2024. Conifer: Improving complex constrained instruction-following ability of large language models. *arXiv preprint arXiv:2404.02823*.

Qwen Team. 2024. Qwen2 technical report. *arXiv preprint arXiv:2412.15115*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022a. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, and 1 others. 2022b. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxing Xu, and 1 others. 2024. Benchmarking complex instruction-following with multiple constraints composition. *Advances in Neural Information Processing Systems*, 37:137610–137645.

Yuning Wu, Jiahao Mei, Ming Yan, Chenliang Li, Shaopeng Lai, Yuran Ren, Zijia Wang, Ji Zhang, Mengyue Wu, Qin Jin, and 1 others. 2025. Writingbench: A comprehensive benchmark for generative writing. *arXiv preprint arXiv:2503.05244*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Howard Chen, Austin W Hanjie, Runzhe Yang, and Karthik Narasimhan. 2023. Collie: Systematic construction of constrained text generation tasks. *arXiv preprint arXiv:2307.08689*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025a. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Zhuohao Yu, Jiali Zeng, Weizheng Gu, Yidong Wang, Jindong Wang, Fandong Meng, Jie Zhou, Yue Zhang, Shikun Zhang, and Wei Ye. 2025b. Rewardanything: Generalizable principle-following reward models. *arXiv preprint arXiv:2506.03637*.

Junjie Zhang, Ruobing Xie, Yupeng Hou, Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2025. Recommendation as instruction following: A large language model empowered recommendation approach. *ACM Transactions on Information Systems*, 43(5):1–37.

Tao Zhang, Chenglin Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, and 1 others. 2024. Cfbench: A comprehensive constraints-following benchmark for llms. *arXiv preprint arXiv:2408.01122*.

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 preprint arXiv:2311.07911*.

A Appendix

A.1 Dataset Analysis

A.1.1 Constraint Distribution

Our dataset includes instruction-following, mathematical, and scientific tasks. For instruction-following tasks, the constraints added in the instructions can be classified into soft constraints and hard constraints. Each instruction contains 1 to 5 constraints. As shown in Fig. 8, we present statistics on the amount of data across different domains, the number of soft and hard constraints, and the distribution of instructions with varying numbers of constraints. As shown in Tab. 6 and Tab. 7, we present examples of soft and hard constraints in the dataset.

A.1.2 Constraint Types

As shown in Tab. 10, our dataset includes various types of constraints, covering multiple domains and tasks. For each seed instruction, we use the prompt shown in Tab. 11 to construct constraints, resulting in multi-constraint instructions.

A.1.3 Evaluation Setup

We manually label 50 groups of preference data, each consisting of an instruction with five constraints and five corresponding responses that satisfy 1 to 5 constraints respectively. We then compare the correlation between the human-labeled ranking and the ranking.

A.2 Reward Model Training

We perform full fine-tuning on two models, Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct, for a binary classification task aimed at determining whether a response satisfies a given constraint. Each training example consists of a prompt that concatenates the response and its associated constraint. Fine-tuning is conducted using the HuggingFace Trainer framework with FP16 precision enabled and Deepspeed optimization (Stage 2 ZeRO) configured via a JSON file specifying automatic batch size and gradient accumulation steps, as well as support for the adamw_torch optimizer. Training uses a batch size of 1, a learning rate of 5e-6, and runs for 3 epochs. Accuracy is employed as the evaluation metric. The Qwen2.5-1.5B reward model provides reward signals for reinforcement learning (RL) training of 1.5B models, while the Qwen2.5-7B reward model serves as a general reward model for other base models. All training is performed on 8 NVIDIA A100 80GB GPUs. As shown in Tab. 9, we present examples of the reward model training data. For the reward model training, we totally use 14,058 samples.

As shown in Tab. 8, for the reward model training data, we performed manual annotation and trained the reward model under the same setting, with 10% of the data split off as the test set. The accuracy of the reward model trained with manually labeled data showed no significant improvement than our original reward model on the test set. Although the reward model's training data may contain noisy labels, this does not significantly compromise its overall performance. This demonstrates the advantage of our pseudo-labeled reward model training data construction:

/ Seed instruction */*
Assign a category to each text snippet identifying the main emotion expressed. Categories may include “joy,” “anger,” “sadness,” or “fear.” Sentence: NAME_2 was overwhelmed with tears of happiness as she accepted the award on stage.
/ Constraints */*
Must include the word “exuberant” in at least one category description.
Incorporate a scenario where a character is experiencing mixed emotions.
Limit the response to a maximum of 25 words.
Write the response in the style of a psychological evaluation report.
Tailor the answer for an audience of high school psychology students.

Table 6: Examples of instructions with soft constraints.

/ Seed instruction */*
Given a question, generate a paraphrase of that question without changing the meaning of it. Your answer should reword the given sentence, but not add information to it or remove information from it. The answer to your question should be the same as the answer to the original question. Input: Question: who does the voice of carl in phineas and ferb?
/ Constraints */*
Answer with the letter `"i"` appearing at least 4 times.
Highlight at least 2 sections of your response in markdown such as `*highlighted section*`.
Make sure your reply is in English and all capital letters.
Put your entire response inside double quotation marks.

Table 7: Examples of instructions with hard constraints.

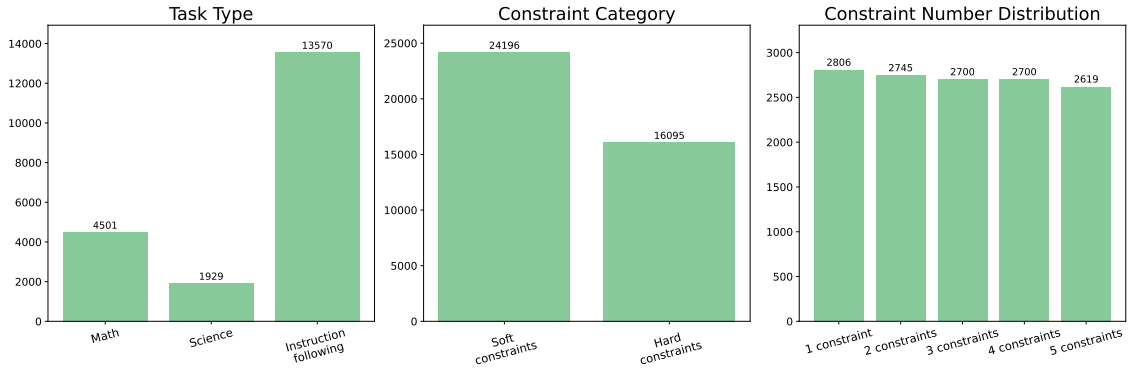

Figure 8: Sample Distribution across Task Types, Constraint Categories, and Number of Constraints

Model Name	Test Accuracy
Original Reward Model	85.3%
Manually Annotated Model	86.1%

Table 8: Comparison of the performance of reward models trained on pseudo-labeled data and human-labeled data.

it achieves performance comparable to human-annotated baselines while eliminating the need for manual labeling.

A.3 RL Training

A.3.1 Implementation Details

We apply the GRPO training using the VeRL framework. We use a distributed training setup across 3 nodes, each equipped with 8 NVIDIA A100 80GB GPUs, for a total of 24 NVIDIA A100 80GB GPUs. Prompts and responses are truncated to a maximum length of 8192 tokens. The RL algorithm employs the GRPO advantage estimator with no KL penalty (kl_coef set to 0.0). The optimizer uses a learning rate of 1e-6 with a weight decay of 1e-2, and no learning rate warm-up. We leverage Fully Sharded Data Parallel (FSDP) with full sharding enabled, rank0 parameter initialization, and parameter offloading to optimize GPU memory usage. The training batches are organized with a global batch size of 96, micro-batches of size 2 for updates, and micro-batches of size 16 for experience generation. Gradient clipping is applied with a max norm of 1.0. Rollouts are performed with a temperature of 1.0 and a group size of 5. Tensor parallelism of size 2 is applied. We train for a total of 260 steps.

A.3.2 Baselines

IR-1.5B (Qin et al., 2025): A systematic method to boost LLMs in dealing with complex instructions via incentivizing reasoning for test-time compute scaling.

Conifer-7B-DPO (Sun et al., 2024): The method first performs SFT training on a constructed dataset arranged from easy to hard, and then performs DPO training using an open-source preference dataset.

Crab-7B-DPO (Qi et al., 2024): the method adopts Llama3-70B-Instruct to back-translate constraints and create a high-quality, complex instruction-response dataset for DPO training.

SPAR-8B-DPO (Cheng et al., 2024): SPaR is a self-play framework integrating tree-search self-refinement to yield valid and comparable

/ Positive Example */*
Question: Introduce Time Constraints: Impose a time limit for the role-play to unfold or for certain actions to be completed.
Answer: Characters identified in the story: 1. Kathy 2. Sue. The main conflict between the characters is the tension and argument that arises due to Kathy spreading rumors about Sue and Sue discovering and reacting to these rumors. To introduce a time constraint for the role-play, we can say: The events in the story unfold over the course of a week. Kathy starts the rumors on Monday, Sue finds out by Wednesday, they have the argument on Thursday, and by the end of the week, Sue stops speaking to Kathy. This time constraint adds a sense of urgency and progression to the narrative.
Label: 1
/ Negative Example */*
Question: Introduce Time Constraints: Impose a time limit for the role-play to unfold or for certain actions to be completed.
Answer: The characters available in the given story are Kathy and Sue. The main conflict between the characters is the tension and argument that arises due to Kathy's rumors about Sue, which leads to Sue finding out and subsequently never talking to Kathy again.
Label: 0

Table 9: Examples of reward model training data.

preference pairs free from distractions. By playing against itself, an LLM employs a tree-search strategy to refine its previous responses with respect to the instruction while minimizing unnecessary variations.

VERIF (Peng et al., 2025): VERIF is a verification method that combines rule-based code verification with LLM-based verification from a large reasoning model (RLVR).

A.3.3 Benchmarks

We evaluate instruction-following ability on various benchmarks:

IFEval (Zhou et al., 2023): It focuses on a set of "verifiable instructions" such as "write in more than 400 words" and "mention the keyword of AI at least 3 times". IFEval contains 25 types of those verifiable instructions and around 500 prompts, with each prompt containing one or more verifiable instructions.

CFBench (Zhang et al., 2024): It is a large-scale Comprehensive Constraints Following Benchmark for LLMs, featuring 1,000 curated samples that cover more than 200 real-life scenarios and over 50 NLP tasks. CFBench meticulously compiles constraints from real-world instructions and constructs an innovative systematic framework for constraint types, which includes 10 primary categories and over 25 subcategories, and ensures each constraint is seamlessly integrated within the instructions.

FollowBench (Jiang et al., 2023): FollowBench comprehensively includes five different types (i.e., Content, Situation, Style, Format, and Example) of fine-grained constraints. To enable a precise constraint following estimation on diverse difficulties, it introduces a Multi-level mechanism that incrementally adds a single constraint to the initial instruction at each increased level.

ComplexBench (Wen et al., 2024): ComplexBench is a benchmark for comprehensively evaluating the ability of LLMs to follow complex instructions composed of multiple constraints. It proposes a hierarchical taxonomy for complex instructions, including 4 constraint types, 19 constraint dimensions, and 4 composition types, and manually collect a high-quality

Constraint Type	Definition	Example
Lexical content constraint	Must include specific terms or symbols with precise placement.	"...must include the word 'beautiful.'"
Element constraint	Include specific entities or scenarios.	"...highlights the Great Wall."
Semantic constraint	Focus on themes, tone, or stance.	"Write a poem about London."
Word Count	Limit the number of words.	"A 50-word poem."
Sentence Count	Limit the number of sentences.	"...three sentences."
Paragraph Count	Limit the number of paragraphs.	"...divided into 3 sections."
Document Count	Limit the number of documents.	"...list 3 articles."
Tone and emotion	Conform to specific emotional tone.	"Write a letter in an angry and sarcastic tone."
Form and style	Use specified stylistic form and perception.	"Write a passage in an encyclopedic style."
Audience-specific	Tailored to a specific audience group.	"Write a poem for a 6-year-old."
Authorial style	Emulate specific authors' styles.	"Write a passage in the style of Shakespeare."
Fundamental format	Follow standard formats like JSON, HTML, etc.	"Output in JSON format."
Bespoke format	Use custom formatting protocols.	"Bold the main idea and output in unordered list."
Specialized format	Tailored for specific applications or domains.	"Convert to electronic medical record format."
Pragmatic constraint	Adapt to context like dialects or language policy.	"Output in English, classical Chinese, etc."
Syntactic constraint	Follow specific phrase and clause structures.	"Use imperatives with nouns and verb phrases."
Morphological constraint	Control over affixes, roots, and word formation.	"Output all content in lowercase English."
Phonological constraint	Focus on sounds, tone, and intonation.	"Single-syllable tongue twisters."
Role-based constraint	Respond with specific role identity.	"You are Confucius, how do you decide?"
Task-specific constraint	Address a defined situational task.	"Work from home, how to report?"
Complex context constraint	Involve multi-faceted and nested reasoning.	"On the left, 10 total, what to do?"
Example constraint	Conform to patterns from example pairs.	"input:x..., output:...; input:y..., output?"
Inverse constraint	Narrow response space via exclusions.	"No responses about political topics."
Contradictory constraint	Combine requirements that are hard to satisfy simultaneously.	"A five-character quotation, 1000 words."
Rule constraint	Follow symbolic or logical operation rules.	"Each answer adds 1+1=3, then 2+2=5."

Table 10: Constraint Types (Zhang et al., 2024).

[Task Description]

1. I currently have a seed question, but the seed questions are relatively simple. To make the instructions more complex, I want you to identify and return five atomic constraints that can be added to the seed question.

2. I will provide [Seed Question] and [Constraint References], and you can use these references to propose five constraints that would increase the difficulty of the seed question.

3. [Constraint References] are just suggestions. You may choose one or more constraints from the list or propose new ones if needed.

4. Do not modify or rewrite the seed question. Your task is only to generate new constraints that can be added to it.

5. Return the added constraints in the following JSON format:

```
json
{
   "c1": "<first constraint>",
   "c2": "<second constraint>",
   "c3": "<third constraint>",
   "c4": "<fourth constraint>",
   "c5": "<fifth constraint>"
}
```

6. Do not return anything else. No explanation, no reformulated question, no analysis—only the JSON structure.

[Constraint References]

1. Lexical content constraint : {Definition} {Example}

2. Word Count : {Definition} {Example}

. . .

25. Rule Constraint : {Definition} {Example}

[Seed Question]

{raw_question}

Table 11: Prompt for generating constraints.

dataset accordingly.

WritingBench (Wu et al., 2025): It is a comprehensive benchmark designed to evaluate LLMs across 6 core writing domains and 100 subdomains, encompassing creative, persuasive, informative, and technical writing.

Collie (Yao et al., 2023): A grammar-based framework that allows the specification of rich, compositional constraints with diverse generation levels (word, sentence, paragraph, passage) and modeling challenges (e.g.,language understanding, logical reasoning, counting, semantic planning).

AgentIF (Qi et al., 2025): It is the first benchmark for systematically evaluating LLM instruction following ability in agentic scenarios. AgentIF features three key characteristics: (1) Realistic, constructed from 50 real-world agentic applications. (2) Long, averaging 1,723 words with a maximum of 15,630 words. (3) Complex, averaging 11.9 constraints per instruction, covering diverse constraint types, such as tool specifications and condition constraints.

Multichallenge (Deshpande et al., 2025): It is a pioneering benchmark evaluating large language models (LLMs) on conducting multi-turn conversations with human users, a crucial yet underexamined capability for their applications. MultiChallenge identifies four categories of challenges in multi-turn conversations that are not only common and realistic among current human-LLM interactions, but are also challenging to all current frontier LLMs. All 4 challenges require accurate instruction-following, context allocation, and in-context reasoning at the same time.

We assess general reasoning and knowledge capabilities with the following datasets:

GPQA-Diamond (Rein et al., 2024): GPQA-Diamond is a specialized subset of the GPQA (Graduate-Level Google-Proof Q&A) benchmark, comprising 198 meticulously crafted multiple-choice questions in biology, chemistry, and physics. These questions are designed to be exceptionally challenging, even for domain experts, making them a rigorous test for AI models.

BBEH (Kazemi et al., 2025): BBEH stands for BIG-Bench Extra Hard, a benchmark introduced by Google DeepMind to evaluate the advanced reasoning capabilities of large language models (LLMs). It serves as an extension to the previous BIG-Bench Hard (BBH) benchmark, addressing the saturation observed as state-of-the-art models achieved near-perfect scores on many tasks in BBH.

AIME2024 (MAA, 2024): The AIME 2024 dataset consists of problems from the American Invitational Mathematics Examination (AIME) 2024 and is commonly used to evaluate the mathematical reasoning ability of large language models (LLMs). AIME2025 (MAA, 2025): The AIME 2025 dataset is a benchmark designed to evaluate the mathematical reasoning capabilities of large language models (LLMs). It comprises problems from the American Invitational Mathematics Examination (AIME) 2025, a prestigious mathematics competition for high school students in the United States.

FOLIO (Han et al., 2022): FOLIO (First-Order Logic Inference Over Text) is a benchmark dataset developed to assess the logical reasoning capabilities of large language models (LLMs). It consists of human-annotated examples that require deductive reasoning grounded in first-order logic (FOL).

MMLU-Pro (Wang et al., 2024): MMLU-Pro (Massive Multitask Language Understanding - Pro) is an advanced benchmark designed to evaluate the reasoning and language understanding capabilities of large language models (LLMs). It builds upon the original MMLU benchmark by

introducing more complex, reasoning-intensive questions and expanding the answer choices from four to ten, significantly increasing the difficulty and reducing the chance of success through random guessing.

A.3.4 Analysis of Model Response Improvement

Tab. 12 demonstrates the model's improvement after training. Before training, the model mainly generates a detailed thought process instead of directly producing the poem, resulting in an output that does not meet the specified constraints and has a low score. After training, the model outputs a concise poem that strictly follows all requirements—including the title format, sentence limit, and punctuation rules—with fluent and vivid content. This clear difference in output quality and adherence to constraints, along with the improved score, shows that the training effectively enhances the model's performance.

A.3.5 Full Training Dynamics Analysis

As shown in Fig. 9, we present the model's reward and output length changes during training under the setting without incremental constraint curriculum (w/o incremental constraint curriculum).

A.3.6 Full Performance on Soft and Hard Constraints

As shown in Tab. 13 and Tab. 14, we present the model's performance on the soft and hard constraint benchmarks.

Case	Content
Before Training	**Input:** Write a poem about a curious cat. The poem must have a title wrapped in double angular brackets, i.e. «title», contain less than 13 sentences, and no commas. Don't forget to add other punctuations.<think> **Think:** Alright, so the user has asked me to write a poem about a curious cat. They've given me specific guidelines: the title needs to be in double angular brackets, the poem should have less than 13 sentences, no commas, and include other punctuations. Hmm, okay, let me break this down. First, I need a title. It should be catchy and reflect the curiosity aspect. Maybe something like "Whisker's Quest" or "Paws of Curiosity." I'll go with "Curious Cat's Journey" since it clearly states the theme. Next, the structure. The poem should have fewer than 13 sentences, so maybe around 11 to keep it concise. I'll aim for a consistent rhyme scheme to make it flow smoothly. Let's try AABB or ABAB, but I'll see as I go. No commas, so I'll have to be careful with sentence structure. Instead, I'll use other punctuation like periods, exclamation points, and question marks to add rhythm and emphasis. Maybe some internal rhymes to make it catchy. Thinking about the content, I want to highlight the cat's curiosity and adventure. Maybe describe the cat exploring places, meeting new things, and the excitement of discovery. Including sensory details like sights, sounds, smells can make the poem vivid. I'll start each line with a different image related to curiosity—like whiskers, paws, exploring, meeting new friends, etc. Then, gradually build up the story of the cat's journey. Each stanza can focus on a different aspect: meeting a friend, discovering a new place, the thrill of exploration. I should also include a question to engage the reader, something like "What's next for our curious friend?" to leave a lasting impression. Ending with a playful line about the cat chasing its own shadow adds a nice touch and a bit of whimsy. Now, putting it all together, I'll draft each line, ensuring it flows without commas and uses the required punctuation effectively. I'll count the sentences to make sure it's under 13. Let me check each line for rhythm and how well it contributes to the overall theme of curiosity and adventure. Finally, I'll review the poem to ensure it meets all the user's guidelines and conveys the intended emotions and imagery. Making sure it's engaging and captures the essence of a curious cat's journey. </think> **Output:** A curious cat with whisker's sharp, Dared to explore every nook and corner. "Where will this adventure lead?" she purred, A playful purr as she met a new friend. A tail as long as the sun's golden rays, A heart full of tales to share and share again. "Will you help me climb that tall tree?" she asked, A leap in leap with a playful skip and jerk. The meow that echoes through the night, A symphony of sounds that never quite fit. A tail as quiet as a gentle sigh, A mind full of wonders that never die. The world her nose so small a dot, With curiosity as her constant companion. A journey through shadows and bright light, A tale of tails and tails that light the night. What's next for our curious friend? A pawsome quest with no map in sight? A tail as long as the stars above, A heart full of stories that never end. So here's to curious cats like her, Who explore and learn and never fear. A tail that twists and turns with flair, A soul full of adventure and flair. A shadow to chase and a story to tell, A cat with curiosity that knows no rule. A tail that wags and a heart that's free, A friend to all who dare to be true. So here's to curious cats like her, With tails that dance and hearts that never tire. A tail that tells a tale of tails, A story of wonder that never ends. **Score:** 0.333
After Training	**Input:** Write a poem about a curious cat. The poem must have a title wrapped in double angular brackets, i.e. «title», contain less than 13 sentences, and no commas. Don't forget to add other punctuations.<think> **Think:** Alright, let's break down the user's query. They want a poem about a curious cat with specific constraints. The title needs to be in double angular brackets, less than 13 sentences, no commas, and include other punctuations. First, I'll think about the structure. The poem should be concise, so each line should pack a punch. I'll start with a title that reflects curiosity, maybe something like "The Curious Cat's Tail" to fit the format. Next, I need to ensure each line flows well without commas. I'll use periods or other punctuation like exclamation marks or question marks to maintain rhythm. I'll describe the cat's actions and environment to make it vivid. I'll consider the imagery. A curious cat might explore, interact with other animals, and show personality. Including elements like yarn, yarn balls, and sun can add depth. I'll make sure each line is short and impactful. Now, checking the sentence count. I'll aim for around 10-12 lines to stay under 13. I'll also ensure that the poem doesn't use any commas, so I'll structure the sentences to avoid them naturally. Finally, I'll review the poem to make sure it meets all the criteria: title format, sentence count, no commas, and appropriate punctuation. This should create a engaging and concise poem that highlights the cat's curiosity. </think> **Output:** «The Curious Cat's Tail» A curious cat purred soft and high Curious as a yarn ball on a hook It meandered through the sunlit yard Exploring every nook and every pore Meeting friends old and new along the way With curiosity in its bright eyes It spun tales of adventure untold Of yarn and games and endless joy The cat's tail wagged like a flag Pointing to wonders both near and far In each glance and in each sigh The curious cat's spirit did thrive A poem of paws and a heart full of glee Wrapped in warmth and filled with glee **Score:** 1.0

Table 12: Comparison of the outputs of DeepSeek-R1-Distill-Qwen-7B before and after training for the same

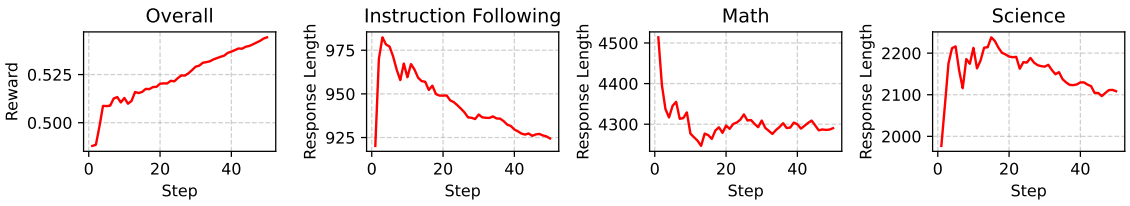

Figure 9: The reward and response length of w/o incremental constraint curriculum.

Method	Easy Set			Hard Set			Full Set		
	CSR	ISR	PSR	CSR	ISR	PSR	CSR	ISR	PSR
Qwen2.5-7B-Instruct	79.0	52.0	58.0	66.0	24.0	37.0	72.0	38.0	48.0
Qwen2.5-7B-Instruct-R	80.0	52.0	58.0	65.0	20.0	34.0	72.0	36.0	46.0
Qwen2.5-7B-Instruct-IF	81.0	55.0	63.0	71.0	24.0	41.0	76.0	40.0	52.0
R1-Distill-Qwen-1.5B	58.0	24.0	29.0	48.0	10.0	18.0	53.0	17.0	24.0
R1-Distill-Qwen-1.5B-IF	64.0	30.0	35.0	53.0	11.0	20.0	59.0	20.0	28.0
R1-Distill-Qwen-7B	79.0	50.0	57.0	66.0	22.0	38.0	72.0	36.0	48.0
R1-Distill-Qwen-7B-IF	83.0	56.0	62.0	71.0	27.0	41.0	77.0	42.0	52.0
R1-0528-Qwen3-8B	95.0	83.0	86.0	84.0	50.0	64.0	89.0	66.0	75.0
R1-0528-Qwen3-8B-IF	94.0	83.0	86.0	85.0	54.0	66.0	90.0	68.0	76.0

Table 13: Full results on CFBench. CSR, ISR, and PSR refer to Constraint Satisfaction Rate, Instruction Satisfaction Rate, and Priority Satisfaction Rate, respectively.

Method	Pr. (S)	Ins. (S)	Pr. (L)	Ins. (L)	Avg.
Qwen2.5-7B-Instruct	71.5	79.1	73.9	81.3	76.5
Qwen2.5-7B-Instruct-R	71.9	78.7	72.3	79.0	75.5
Qwen2.5-7B-Instruct-IF	83.7	88.7	83.9	88.8	86.3
R1-Distill-Qwen-1.5B	38.1	50.4	42.3	54.3	46.3
R1-Distill-Qwen-1.5B-IF	54.0	65.8	58.8	69.8	62.1
R1-Distill-Qwen-7B	56.7	68.3	61.7	72.5	64.8
R1-Distill-Qwen-7B-IF	65.8	75.2	71.7	80.2	73.2
R1-0528-Qwen3-8B	74.9	82.4	79.7	85.9	80.7
R1-0528-Qwen3-8B-IF	83.5	88.7	87.1	91.2	87.6

Table 14: Full results on IFEval. "Pr." and "Ins." refer to prompt-level and instruction-level metrics, respectively, while "S" and "L" denote strict and loose evaluation.