arXiv:2508.03178v1 [cs.CL] 5 Aug 2025

Light-IF: Endowing LLMs with Generalizable Reasoning via Preview and Self-Checking for Complex Instruction Following

Chenyang Wang[1*†], **Liang Wen**[2†], **Shousheng Jia**[2], **Xiangzheng Zhang**[2‡], **Liang Xu**[3‡]

[1]Harbin Institute of Technology, [2]Qiyuan Tech, [3]CLUE
zhangxiangzheng@360.cn, contact@superclue.ai

Abstract

While advancements in the reasoning abilities of LLMs have significantly enhanced their performance in solving mathematical problems, coding tasks, and general puzzles, their effectiveness in accurately adhering to instructions remains inconsistent, particularly with more complex directives. Our investigation identifies lazy reasoning during the thinking stage as the primary factor contributing to poor instruction adherence. To mitigate this issue, we propose a comprehensive framework designed to enable rigorous reasoning processes involving preview and self-checking, essential for satisfying strict instruction constraints. Specifically, we first generate instructions with complex constraints and apply a filtering process to obtain valid prompts, resulting in three distinct prompt datasets categorized as hard, easy, and pass. Then, we employ rejection sampling on the pass prompts to curate a small yet high-quality dataset, enabling a cold-start initialization of the model and facilitating its adaptation to effective reasoning patterns. Subsequently, we employ an entropy-preserving supervised fine-tuning (Entropy-SFT) strategy coupled with token-wise entropy-adaptive (TEA-RL) reinforcement learning guided by rule-based dense rewards. This approach encourages the model to transform its reasoning mechanism, ultimately fostering generalizable reasoning abilities that encompass preview and self-checking. Extensive experiments conducted on instruction-following benchmarks demonstrate remarkable performance improvements across various model scales. Notably, our Light-IF-32B model surpasses both larger open-source models such as DeepSeek-R1 and closed-source models like Doubao-1.6.

Model — https://huggingface.co/qihoo360/Light-IF-32B

Introduction

Instruction following (Leike et al. 2018; Wei et al. 2021; Lou, Zhang, and Yin 2023) is a fundamental capability of large language models (LLMs), marking their transition from mere next-token predictors to practical and reliable assistants. Models adept at instruction following can generate controlled outputs, aligning closely with human intentions and proving beneficial across diverse tasks (Ouyang et al. 2022; Zhou et al. 2023; Zhang et al. 2024a). Conversely, models that fail to accurately interpret or adhere to instructions become unreliable, severely limiting their applicability in real-world domains such as healthcare (Singhal et al. 2023; Cascella et al. 2023; Singhal et al. 2025), autonomous driving (Ding et al. 2023; Shao et al. 2024; Cui et al. 2024), and agent-based systems (Wang et al. 2024b; Team et al. 2025; Luo et al. 2025a).

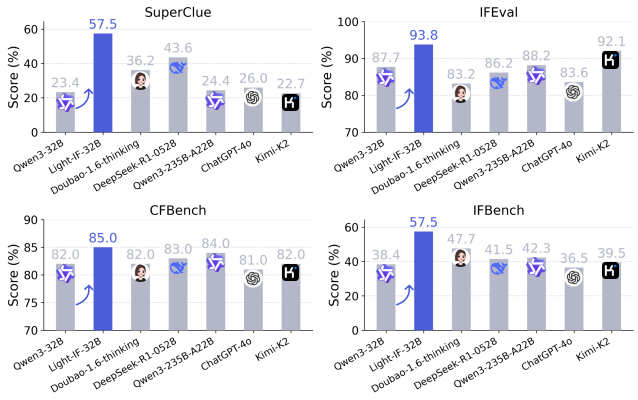


Figure 1: Main results of models on the instruction following benchmarks.

Numerous benchmarks have been proposed to evaluate instruction following abilities (Jing et al. 2023; Zhou et al. 2023; Xu et al. 2023; Jiang et al. 2023; He et al. 2024; Wen et al. 2024; Li et al. 2024; Zhang et al. 2024b; Qin et al. 2024; Pyatkin et al. 2025). For instance, IFeval (Zhou et al. 2023) introduced verifiable atomic instructions to construct prompts. SuperCLUE (Xu et al. 2023) is a Chinese benchmark that scores exact compliance to strict, constraint-heavy instructions on complex tasks. CFBench (Zhang et al. 2024b) highlighted realistic multi-constraint compliance, featuring the broadest constraint coverage and a model-based scoring scheme. On top of IFeval, IFBench (Pyatkin et al. 2025) introduced verifiable, out-of-domain constraints to reveal overfitting issues. Fig. 1 illustrates the evaluation performance of several mainstream LLMs on these benchmarks, exposing need for further improvements.

Previous studies (Sun et al. 2024; Wang et al. 2024a; Zhang et al. 2024c; Dong et al. 2024; Xu et al. 2024; Zhao et al. 2024; Huang et al. 2025; Ren et al. 2025) pri-

*This work was done during the author's internship at 360.
†These authors contributed equally.
‡Corresponding Authors.

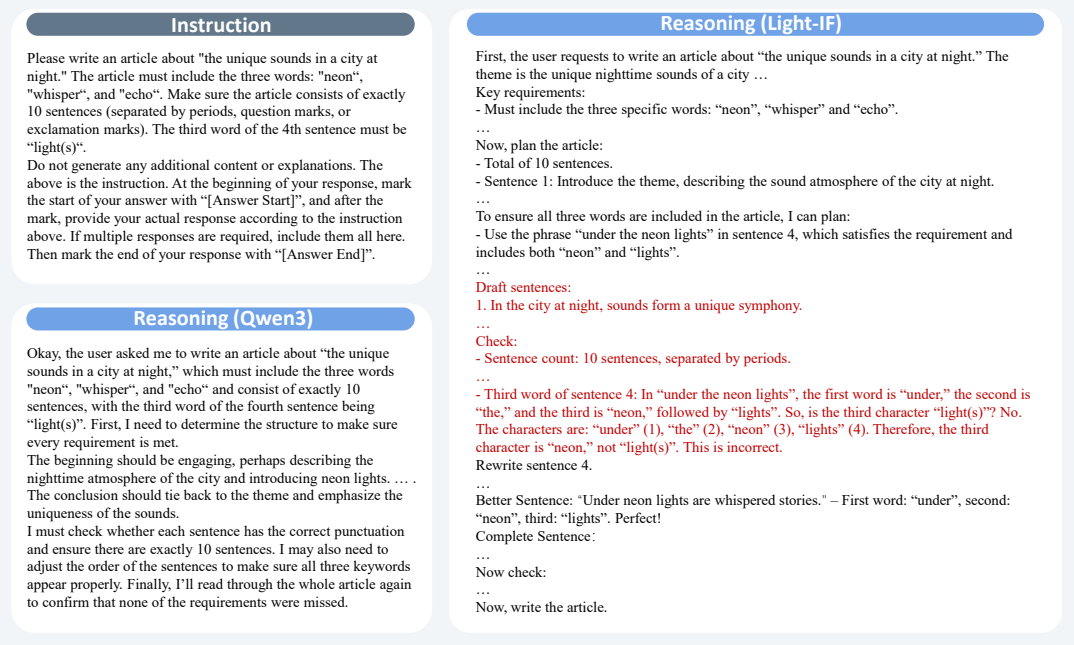


Figure 2: An example of different thinking patterns between Qwen3-32B and Light-IF-32B. Reasoning refers to the content between <think> and </think>. The original text is in Chinese and translated here for readability.

marily enhanced complex instruction-following capabilities during post-training via SFT or direct preference optimization (DPO) on collected or synthetic instructions. For example, Sun et al. (2024) leveraged GPT-4 to categorize ShareGPT prompts into five difficulty levels, subsequently fine-tuning models through a progressively challenging curriculum. Huang et al. (2025) created coarse and fine-grained contrastive pairs from the model itself, conducting multi-granularity self-contrastive DPO, while Ren et al. (2025) auto-generated soft-constraint data, and then trained models using DPO under a few-to-many constraints curriculum. Although effective, these methods heavily rely on extensive supervised data covering a broad spectrum of instructions, posing significant data collection challenges.

Recent advancements in reasoning LLMs (Guo et al. 2025; Yang et al. 2025; Ye et al. 2025; Wen et al. 2025; Wang et al. 2025; Bercovich et al. 2025; Abdin et al. 2025; Chen et al. 2025; Luo et al. 2025b; Sun et al. 2025) demonstrate impressive generalization capabilities of their reasoning processes across various tasks, even when trained on limited task domains like mathematics and coding. Inspired by these developments, we resort to eliciting effective reasoning using a relatively small dataset to address complex instruction-following tasks. Our core intuition is that when an LLM adopts an effective reasoning strategy for a limited set of instructions, this strategy generalizes to unseen instructions with varying constraints and intentions. To begin with, we examine the behavior of recent reasoning LLMs, discovering a prevalent lazy reasoning pattern when confronted with complex instructions, as exemplified in Fig. 2. This ineffective reasoning mode, characterized by simply restating instructions without genuine checking for compliance, hinders the model from strictly following the instructions of users.

We propose a comprehensive framework to address this issue. First, we generate distinct instruction sets categorized as hard, easy, and pass, each with carefully controlled difficulty levels. Subsequently, we conduct Zero-RL training on the lazy-thinking model to incentivize effective reasoning behaviors. Leveraging the Zero-RL model and optionally external APIs, we perform rejection sampling to obtain thousands of high-quality responses with preview and self-checking, which serve as the cold-start dataset for the base model. Finally, we apply Entropy-SFT for cold-start initialization and subsequently train with TEA-RL, equipping LLMs with generalizable reasoning for IF tasks. The main contributions of this work are summarized as follows:

- We identify and characterize the lazy reasoning pattern prevalent in current reasoning LLMs when handling complex constraints.
- We introduce an effective framework incorporating data collection, cold-start, and reinforcement learning stages, enabling generalizable and effective reasoning patterns characterized by previewing and self-checking mechanisms for complex instruction-following tasks.
- We propose innovative techniques to effectively control entropy during both SFT and RL stages, specifically entropy-preserving SFT and RL with token-wise entropy-adaptive regularization.
- Extensive experiments across multiple instruction-following benchmarks demonstrate superior performance of our Light-IF models.

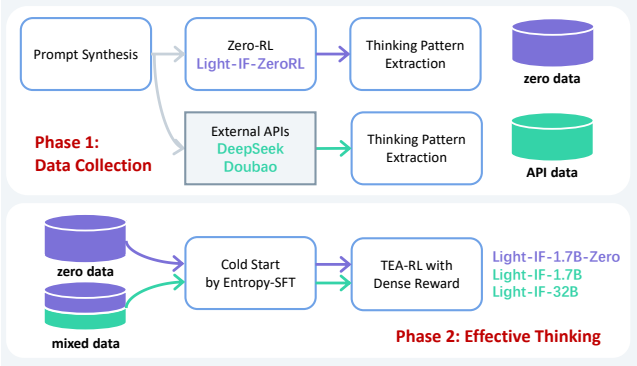


Figure 3: The overall framework of the proposed method.

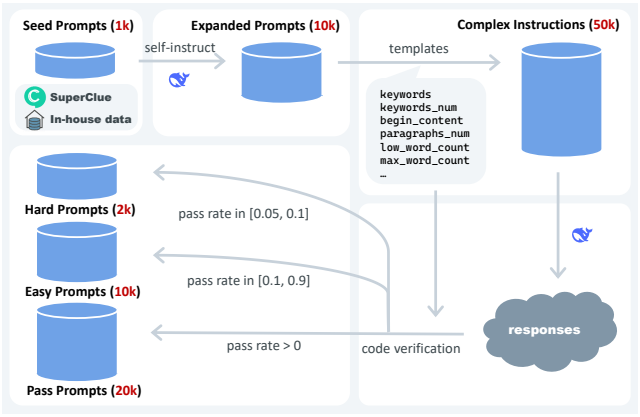


Figure 4: Pipeline of hardness-aware prompt synthesis.

Methods

The proposed framework for enabling generalizable reasoning comprises five components: prompt synthesis, zero-RL from the lazy-thinking model, thinking pattern extraction, entropy-preserving SFT, and TEA-RL with dense rewards. The overall framework is depicted in Fig. 3.

Hardness-aware Prompt Synthesis

Our pipeline for synthesizing hardness-aware prompts consists of four main steps: seed prompt collection, prompt expansion, construction of complex constraints, and prompt filtering. The overall pipeline is illustrated in Fig. 4.

Seed Prompt Collection. Seed prompts originate from two primary sources: historical evaluation data from SuperClue (Xu et al. 2023) (100+ Chinese prompts collected before October 2024) and our in-house evaluation datasets (200+ English prompts and 600+ Chinese prompts). **Prompt Expansion**. We employ the Self-Instruct (Wang et al. 2022) methodology to expand seed prompts with simple instructions, resulting in an expanded set of 10,000 prompts. **Construction of Complex Constraints**. In this stage, we exclusively incorporate limited verifiable constraints. Specifically, following AutoIF (Dong et al. 2024), we define an instruction template structured as a dictionary, including keys such as keyword frequency, word count, paragraph count, among others. The values for each instruction template are randomly sampled. By applying five different templates to each simple instruction, we generate 50,000 prompts with complex instructions. **Prompt Filtering**. To eliminate invalid prompts, we utilize an efficient LLM to generate ten outputs per prompt. Due to the verifiable nature of these instructions, we discard prompts whose outputs consistently fail code verification. This initial filtering process results in approximately 20,000 valid prompts, designated as pass prompts. Subsequently, we create two distinct datasets categorized by their difficulty levels: an easy prompt dataset with pass ratios ranging from $[0.1, 0.9]$ and a hard prompt dataset with pass ratios ranging from $[0.05, 0.1]$.

Zero-RL from Lazy-Thinking Model

To incentivize the effective reasoning pattern, we adopt R1-Zero-style reinforcement learning (Zero-RL) to post-train the lazy-thinking model. The key challenge in Zero-RL lies in the design of rewards. Specifically, the rewards in Zero-RL consist of two components: (1) correctness score R_c and (2) length reward R_l. The correctness reward (and other training designs) will be detailed in Section TAE-RL. In this section, we focus specifically on the length reward, defined as follows:

$$R_l = \begin{cases} -2, & \text{if } L \geq L_{max}, \\ 2 \cdot R_c \cdot \gamma(L), & \text{if } R_c \geq 0.2 \text{ and } L < L_{max}, \\ -\gamma(L), & \text{if } R_c < 0.2 \text{ and } L < L_{max}. \end{cases} \quad (1)$$

where $\gamma(L) = 0.5\left(1 - \cos\left(\pi \cdot \frac{L}{L_{max}}\right)\right)$. The length reward encourages correct and sufficiently long responses to explicitly mitigate lazy thinking. Simultaneously, it penalizes excessively long responses as well as incorrect and overly verbose outputs. The relationship between response length and correctness score R_c during Zero-RL is illustrated in Fig. 5. Apart from initial fluctuations, response length exhibits a strong positive correlation with correctness scores. The initial reduction also highlights the difficulty of using the correctness reward alone, where the model doesn't prefer longer responses when the length achieves 3,000, making it hard to learn effective thinking pattern. Moreover, we observed the emergence of preview and self-checking behaviors, aligning well with our expectations.

Thinking Pattern Extraction

To gather samples exhibiting effective reasoning patterns for cold-start, we design a data filtering pipeline for model responses to pass prompts (20K). The pipeline includes correctness check, thinking check, and fluency check (detailed in the Appendix C). After filtering, we collect the top 2,000 high-quality cold-start samples, evenly split between Chinese and English.

Depending on the use of external APIs, we construct two distinct cold-start datasets: 1) **zero data**: the acquirement of zero data doesn't rely on external guidance, where all responses are generated by the Zero-RL model. 2) **mixed data**: we mix responses from Zero-RL model (covering all

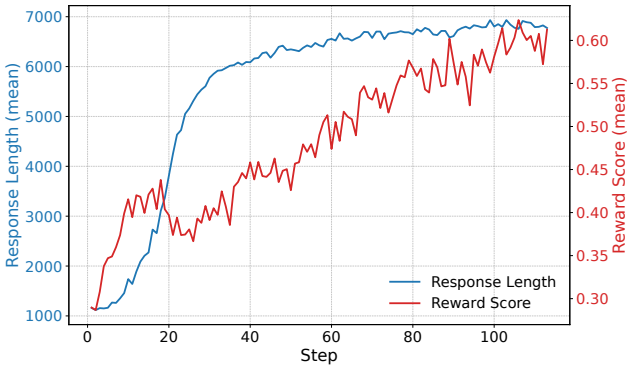


Figure 5: Response length and R_c during Zero-RL training.



(a) selected tokens (b) dropped tokens

Figure 6: Left: Top-100 tokens with high selection rank at the beginning of the cold-start stage. Right: Top-100 tokens with low selection rank at the end of the cold-start stage. Larger size indicates larger frequency and brighter color indicates higher rank.

pass prompts) and external APIs (DeepSeek-R1 for Chinese subset and Doubao-1.6 for English subset). These APIs demonstrate stable and effective reasoning patterns in certain scenarios, making them suitable candidates for sampling. The mixed responses are then filtered using the same pipeline described above. We train two separate models using these distinct cold-start datasets for the following reasons: 1) the Zero-RL-based cold start showcases the viability of eliciting effective reasoning patterns entirely from a lazy-thinking model, without relying on external guidance—offering a practical and self-contained solution for the community. 2) by leveraging the strong external APIs for cold-starting, we obtain a model with SOTA performance, offering the community a powerful open-source model that requires significantly fewer parameters.

Entropy-Preserving SFT for Cold-Start

Entropy (Cui et al. 2025; Wang et al. 2025; Yu et al. 2025; Fu et al. 2025) serves as a metric for exploration in LLMs during reinforcement learning and is considered a crucial indicator of their final performance. SFT aims to minimize the mean cross-entropy loss for each training sample, inevitably causing entropy reduction on the training data, given the large number of model parameters and limited training samples. Furthermore, we observe that this entropy reduction phenomenon generalizes to untrained samples as well, decreasing the initial entropy compared to the base model and negatively impacting subsequent reinforcement learning stages. Therefore, we propose entropy-preserving SFT to mitigate entropy loss during the cold-start phase.

Specifically, inspired by Wang et al. (2025), we think that not all tokens are necessary for loss computation during training. Thus, during the cold-start stage, we selectively choose only $r\%$ of tokens for each training step based on prediction entropy and cross-entropy loss. Formally, the prediction distribution for token o_t of sample s_i is defined as:

$$\boldsymbol{p_t^i} = \pi_\theta(\cdot|q^i, o^i_{<t}), \tag{2}$$

where π_θ represents the LLM parameterized by θ, q^i is the input query of sample s_i, and $o^i_{<t}$ denotes outputs prior to token o_t. The prediction entropy is defined as $H^i_t = -\sum_j (\boldsymbol{p_t^i})_j \log(\boldsymbol{p_t^i})_j$, and the cross-entropy loss is defined as $\mathrm{NLL}^i_t = -\log \boldsymbol{p_t^i}(o_t)$. For a batch of samples B, the selected tokens T_{sel} are determined as follows:

$$\begin{aligned} T_{all} &= \{o_t | o_t \in s_i \wedge s_i \in B\}, \\ S &= \{\mathrm{NLL}_t - \alpha H_t | o_t \in T_{all}\}, \\ T_{sel} &= \{o_t | o_t \in T_{all} \wedge \mathrm{NLL}_t - \alpha H_t > \mathrm{Quantile}_r(S)\}. \end{aligned} \tag{3}$$

The Entropy-SFT loss is then computed on selected tokens:

$$L_{Entropy-SFT} = \frac{1}{|T_{sel}|} \sum_{o_t \in T_{sel}} -\log \boldsymbol{p}_t(o_t). \tag{4}$$

This token selection strategy serves two purposes: 1) the first component prioritizes tokens with high negative log-likelihood, representing new information introduced by cold-start data not captured by the baseline model. As illustrated at the left of Fig. 6, tokens selected at the beginning typically relate to preview and checking pattern. Examples include sentences like "*Another* approach for the third bullet: [preview for the third bullet]" and "Wait, but in the *initial* response, [check constraints for responses]." 2) the second component accounts for token entropy, assigning lower selection ranks to high-entropy tokens to preserve the model's inherent entropy at the end of the cold-start. As depicted at the right of Fig. 6, tokens excluded at the end of training are either confidence tokens with low NLL (*e.g. user* and *plural*) or high entropy tokens (*e.g. Now* and *considering*). These tokens are appropriately dropped to avoid ineffective training and to preserve entropy.

TEA-RL with Dense Reward

The RL stage encounters two primary challenges: sparse reward and entropy collapse. The sparse reward issue arises because, although final instruction-following outcomes are verifiable, the traditional binary reward signal provides extremely sparse feedback when the model initially struggles to fully satisfy complex constraints. Given the nature that complex instructions consist of multiple constraints, we adopt a strategy of rewarding partial constraint satisfaction individually, as illustrated in Tab. 1. With this dense reward

Group	Item	Constraint Type	Reward
Keywords	Keyword 1	$[n_{\min}, n_{\max}] \wedge n_{max} < 5$	0.10
	Keyword 2	$[n_{\min}, n_{\max}] \wedge n_{max} \geq 5$	0.20
	Keyword 3	$\leq n_{\min}$	0.05
	Keyword 4	$\geq n_{\max}$	0.05
	Keyword 5	$= n$	0.10
Paragraphs	—	$= n$	0.10
Sentences	—	$= n$	0.20
Words	Word 1	$[n_{\min}, n_{\max}] \wedge \Delta > 50$	0.10
	Word 2	$[n_{\min}, n_{\max}] \wedge \Delta \leq 50$	0.20
	Word 3	$\leq n_{\min}$	0.05
	Word 4	$\geq n_{\max}$	0.05
Beginning	—	match	0.02
End	—	match	0.02
All	—	satisfy	+1.00

Table 1: Constraint types and corresponding rewards. If all constraints are met, the model receives an extra reward of 1.

strategy, the reward curve exhibits a smoother and more rapid increase (detailed in Appendix E).

Entropy collapse refers to the phenomenon where the model's entropy rapidly diminishes during training, significantly reducing exploration capability and resulting in suboptimal performance. A straightforward approach to address this problem is entropy regularization during the reinforcement learning phase. However, directly applying entropy regularization either leads to entropy explosion or fails to effectively halt entropy collapse. Building upon the theoretical insight that *changes in policy entropy are driven by the covariance between action probability and logit changes, proportional to advantages when employing policy gradient algorithms (Cui et al. 2025)*, we propose the token-level entropy-adaptive regularization term. Formally, for a rollout batch, the covariance for tokens T_r is calculated as:

$$\mathrm{Cov}_t = \left(\log \boldsymbol{p}_t(o_t) - \frac{1}{|T_r|} \sum_{o_j \in T_r} \log \boldsymbol{p}_t(o_j) \right) \cdot \left(A(o_t) - \frac{1}{|T_r|} \sum_{o_j \in T_r} A(o_j) \right). \quad (5)$$

The token-level regularization term is then defined as:

$$L_{TEA} = |T_r| \sum_{o_t \in T_r} \min \left(\frac{e^{\mathrm{Cov}_t/\tau}}{\sum_{o_j \in T_r} e^{\mathrm{Cov}_j/\tau}}, \frac{c}{|T_r|} \right) H_t, \quad (6)$$

where τ is the temperature and c is the max coefficient value. Consequently, the total loss for TEA-RL is given by:

$$L_{TEA-RL} = L_{GRPO} - \lambda L_{TEA}, \quad (7)$$

where λ represents the regularization coefficient.

TEA-RL training is conducted in two stages, following an easy-to-hard curriculum, with the model first trained on easy prompts and subsequently on hard prompts. Notably, to demonstrate the generalization capability of our framework, we only perform RL on the Chinese subset of each dataset.

Experiments

In this section, we first present evaluation results on instruction-following benchmarks. Next, we examine the performance progression at each stage of the Zero-RL pipeline. Subsequently, we validate the effectiveness of specific design choices within our framework through ablation studies. Lastly, we analyze the entropy dynamics.

Experimental Settings

Benchmarks. To evaluate the performance of various LLMs, we employ four distinct benchmarks: IFEval, CFBench, IFBench, and SuperCLUE-CPIF (2025) (abbreviated to SuperCLUE in the remaining). We adopt DeepSeek-V3 as the judge model for CFBench and use greedy decoding on all benchmarks.

Baselines. The baseline models include the Qwen3 series (Yang et al. 2025) (Qwen3-1.7B, Qwen3-32B, and Qwen3-235B-A22B), DeepSeek series (DeepSeek-V3 (Liu et al. 2024) and DeepSeek-R1 (Guo et al. 2025)), Doubao series (Doubao-1.5-pro (2025a), Doubao-1.6-thinking (2025b)), ChatGPT-4o (2024) and Kimi-K2 (Team et al. 2025). All comparison models are powerful, up-to-date, and actively deployed in production environments.

Implementation Details. For cold-start training, we utilize the LLaMA-Factory (Zheng et al. 2024) with our modified Entropy-SFT. For the subsequent RL stage, we adopt VeRL (Sheng et al. 2024) integrated with our proposed TEA regularization. The hyperparameters (r, α) for Entropy-SFT and (τ, λ, c) for TEA-RL are set to values $(80, 0.8)$ and $(1.0, 0.05, 100)$, respectively. Further implementation details are provided in the Appendix B.

Evaluation Results

In this section, we evaluate the performance of our models against recent strong reasoning and non-reasoning models. Specifically, we select three model variants for evaluation: Light-IF-32B-EntSFT-TEARL-s2, Light-IF-1.7B-EntSFT-TEARL-s2, and Light-IF-1.7B-EntSFT(ZR)-TEARL-s2, where EntSFT denotes entropy-preserving SFT, ZR denotes cold-starting from zero data, and s2 indicates that the model has undergone two stages of RL training. For brevity, we refer to these models as Light-IF-32B, Light-IF-1.7B, and Light-IF-1.7B-Zero, respectively.

As demonstrated by the results in Tab. 2, our models equipped with effective reasoning patterns show remarkable improvements over their corresponding base models (Qwen3-1.7B and Qwen3-32B). Notably, Light-IF-32B achieves the highest performance among all evaluated models, surpassing the next-best models by 13.9, 1.7, 1.0, and 9.8 points on SuperClue, IFEval, CFBench, and IFBench, respectively. Moreover, Light-IF-1.7B demonstrates impressive performance despite having significantly fewer parameters, outperforming Qwen3-235B-A22B and Qwen3-32B on SuperClue and IFEval, and closely matching them on IFBench. Furthermore, Light-IF-1.7B-Zero exhibits competitive performance relative to Light-IF-1.7B, which highlights the potential of the base model to independently enhance

Model	SuperClue	IFEval					CFBench				IFBench		
		LP	LI	SP	SI	AVG	CSR	ISR	PSR	AVG	PL	IL	AVG
Non-reasoning Models													
ChatGPT-4o	0.260	0.837	0.880	0.786	0.841	0.836	0.90	0.72	0.80	0.81	0.354	0.376	0.365
Deepseek-v3-0324	0.306	0.856	0.899	0.815	0.867	0.859	0.91	0.76	0.83	0.83	0.388	0.421	0.405
Doubao-1.5-pro	0.285	0.885	0.921	0.852	0.899	0.889	0.89	0.71	0.79	0.80	0.361	0.388	0.375
Kimi-K2	0.227	0.917	0.944	0.895	0.929	0.921	0.91	0.74	0.82	0.82	0.378	0.412	0.395
Reasoning Models													
Qwen3-1.7B	0.081	0.726	0.796	0.697	0.767	0.747	0.81	0.56	0.67	0.68	0.275	0.301	0.288
Qwen3-32B	0.234	0.871	0.914	0.834	0.887	0.877	0.91	0.74	0.82	0.82	0.364	0.403	0.384
Qwen3-235B-A22B	0.244	0.884	0.916	0.839	0.887	0.882	0.92	0.75	0.84	0.84	0.408	0.438	0.423
DeepSeek-R1-0528	0.436	0.857	0.905	0.814	0.873	0.862	0.91	0.74	0.83	0.83	0.405	0.424	0.415
Doubao-1.6-thinking	0.362	0.825	0.881	0.775	0.847	0.832	0.90	0.74	0.82	0.82	0.469	0.485	0.477
Our Models													
Light-IF-1.7B-Zero	0.232	0.872	0.917	0.836	0.901	0.882	0.84	0.61	0.72	0.72	0.375	0.407	0.391
Light-IF-1.7B	0.299	0.885	0.924	0.856	0.904	0.892	0.85	0.61	0.71	0.72	0.371	0.415	0.393
Light-IF-32B	**0.575**	**0.933**	**0.956**	**0.917**	**0.945**	**0.938**	**0.93**	**0.77**	**0.85**	**0.85**	**0.565**	**0.585**	**0.575**

Table 2: Evaluation results. LP: loose prompt, LI: loss instruction, SP: strict prompt, SI: strict instruction, PL: prompt level, IL: instruction level. Best are marked **bold** among all models.

Model	SuperClue	IFEval					CFBench				IFBench		
		LP	LI	SP	SI	AVG	CSR	ISR	PSR	AVG	PL	IL	AVG
Qwen3-1.7B	0.081	0.726	0.796	0.697	0.767	0.747	0.81	0.56	0.67	0.68	0.275	0.301	0.288
Qwen3-32B	0.234	0.871	0.914	0.834	0.887	0.877	0.91	0.74	0.82	0.82	0.364	0.403	0.384
Light-IF-1.7B-ZeroRL	0.157	0.843	0.886	0.804	0.855	0.847	0.82	0.59	0.68	0.70	0.327	0.355	0.341
Light-IF-1.7B-EntSFT(ZR)	0.148	0.841	0.888	0.797	0.860	0.847	0.84	0.59	0.70	0.71	0.350	0.388	0.369
Light-IF-1.7B-EntSFT(ZR)-TEARL	0.175	0.869	0.912	0.826	0.881	0.872	**0.85**	**0.62**	0.71	**0.73**	0.364	0.394	0.379
Light-IF-1.7B-EntSFT(ZR)-TEARL-s2	**0.232**	**0.872**	**0.917**	**0.836**	**0.901**	**0.882**	0.84	0.61	**0.72**	0.72	**0.375**	**0.407**	**0.391**

Table 3: Zero-RL model series performance. LP: loose prompt, LI: loss instruction, SP: strict prompt, SI: strict instruction, PL: prompt level, IL: instruction level. Best are marked **bold** among 1.7B models.

reasoning capabilities without reliance on external APIs. Importantly, these evaluation outcomes underscore the generalizable effectiveness of the reasoning patterns learned by our models. Despite being trained on a limited set of synthetic constraints, our models generalize successfully to more complex constraints (SuperClue and CFBench) and out-of-domain constraints (IFBench).

Zero-RL Performance

The Light-IF-1.7B-Zero pipeline starts from the base Qwen3-1.7B model, involving Zero-RL training, thinking-pattern extraction, a cold-start phase, and sequential easy-to-hard RL training. Through this process, we derive four models: 1) Light-IF-1.7B-ZeroRL, directly obtained after Zero-RL. 2) Light-IF-1.7B-EntSFT(ZR), trained from Qwen3-1.7B via extracted samples of Light-IF-1.7B-ZeroRL. 3) Light-IF-1.7B-EntSFT(ZR)-TEARL, obtained by applying TEA-RL to Light-IF-1.7B-EntSFT(ZR) on easy prompts. 4) Light-IF-1.7B-EntSFT(ZR)-TEARL-s2, obtained by TEA-RL to Light-IF-1.7B-EntSFT(ZR)-TEARL on hard prompts.

The results shown in Tab. 3 demonstrate that each step in the pipeline progressively enhances model performance, with the final model achieving the best results overall. Notably, the final Light-IF-1.7B-Zero exhibits instruction-following performance superior to the Qwen3-32B model on IFEval and IFBench, and comparable on SuperClue.

Ablation Study

To validate the effectiveness of each design component within our overall framework, we conduct an ablation study. Specifically, we consider two steps that can potentially be omitted: the cold-start phase and sequential RL phase with dense rewards. Additionally, the two novel components proposed, Entropy-SFT and TEA-RL, can be replaced by existing techniques. To comprehensively assess the contribution of each design, we compare seven models: 1) Qwen3-1.7B: base model. 2) Light-IF-1.7B-EntSFT: model without sequential RL. 3) Light-IF-1.7B-ZeroRL: model without cold-start. 4) Light-IF-1.7B-SFT-GRPO: model with SFT and GRPO. 5) Light-IF-1.7B-EntSFT-GRPO: model with EntropySFT and GRPO. 6) Light-IF-1.7B-SFT-TEARL: model with SFT and TEARL. 7) Light-IF-1.7B-EntSFT-TEARL: model with EntropySFT and TEARL.

The ablation study results presented in Tab. 4 clearly demonstrate the individual contributions of each step and method. By comparing the performance of models 1, 2, 3, and 7, we conclude that both the cold-start phase and the sequential RL stage significantly enhance the final performance. Furthermore, improvements observed when comparing model 4 with 5 and 6 validate the effectiveness of our proposed Entropy-SFT and TEA-RL over their counterparts, standard SFT and GRPO. Integrating all proposed components, Light-IF-1.7B-EntSFT-TEARL achieves the highest performance across all four benchmarks. Importantly, our

cold-start	RL stage	EntSFT	TEARL	Superclue	IFEval					CFBench				IFBench		
					LP	LI	SP	SI	AVG	CSR	ISR	PSR	AVG	PL	IL	AVG
		✓		0.081	0.726	0.796	0.697	0.767	0.747	0.81	0.56	0.67	0.68	0.275	0.301	0.288
✓		✓		0.065	0.828	0.881	0.784	0.850	0.836	0.84	0.59	**0.70**	0.71	0.328	0.350	0.339
	✓		✓	0.157	0.843	0.886	0.804	0.855	0.847	0.82	0.59	0.68	0.70	0.327	0.355	0.341
✓	✓			0.135	0.810	0.865	0.762	0.837	0.819	0.82	0.55	0.65	0.67	0.310	0.334	0.322
✓	✓	✓		0.153	0.815	0.872	0.776	0.842	0.826	**0.85**	0.59	**0.70**	0.71	0.323	0.361	0.342
✓	✓		✓	0.166	0.843	0.893	0.795	0.859	0.845	0.83	0.55	0.66	0.68	0.310	0.355	0.333
✓	✓	✓	✓	**0.231**	**0.880**	**0.917**	**0.845**	**0.891**	**0.883**	**0.85**	**0.60**	**0.70**	**0.72**	**0.344**	**0.391**	**0.368**

Table 4: Ablation Study. LP: loose prompt, LI: loss instruction, SP: strict prompt, SI: strict instruction, PL: prompt level, IL: instruction level. Best are marked **bold**.

results highlight that entropy plays a critical role in our framework—first during cold-start (via Entropy-SFT) and again during RL (via TEA-RL)—and their combined implementation is beneficial for instruction-faithful generation.

Entropy Dynamics within the Framework

Entropy control is one of the most prominent features of our framework. In this section, we demonstrate the effectiveness of our entropy management strategy through two aspects: the dynamics of entropy over training and the evolution of high-entropy tokens at different stages. The entropy dynamics experiment is conducted using RL stage 1 prompts with the 1.7B model, while the token dynamics experiment is conducted on easy prompts in English for readability.

Firstly, entropy dynamics across training steps are illustrated in Fig. 7. Results clearly show that both Entropy-SFT and TEA regularization effectively increase model entropy, promoting exploration. In contrast, standard SFT and conventional RL methods (standard GRPO, GRPO with entropy regularization, KL-Cov (Cui et al. 2025)) reduce entropy (detailed analyses and fine-grained entropy dynamics during RL stage 1 are provided in Appendix D). We argue that retaining relatively high entropy at the end of training remains beneficial, as overly deterministic outputs may still produce errors unrelated to high-entropy tokens. Moreover, excessively confident models are less desirable in practical settings. Maintaining higher entropy encourages exploration of alternative reasoning paths, enabling self-checking and reflection, thus enhancing the probability of correct outcomes.

Next, we examine high-entropy token dynamics as shown in Fig. 8. Two prominent trends emerge: 1) the overlap of high-entropy tokens between the base and the cold-start models is minimal. Transition words and tokens related to previewing and self-checking replace content-focused verbs and nouns, signaling the emergence of a new reasoning pattern. 2) In later stages, high-entropy token overlap significantly increases, indicating that changes during the RL stage are subtle—only a subset of tokens undergo entropy adjustments while preserving the overall distribution of high-entropy tokens. An intriguing observation is that certain tokens (e.g., *check*) initially become high-entropy tokens and subsequently excluded. The underlying reason is the increased prediction probability. Specifically, for tokens with low initial probability, an increase in prediction probability first raises their entropy and subsequently decreases it. This phenomenon further confirms that tokens like "check" become deeply integrated into the model's reasoning pattern.

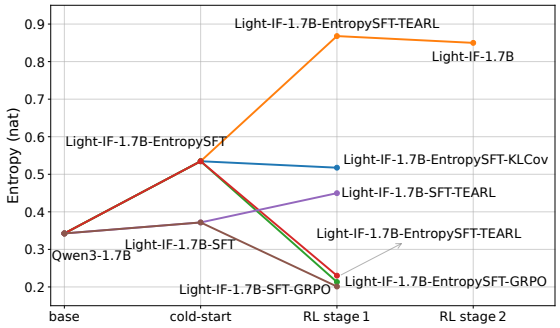


Figure 7: Entropy dynamics. Each point represents a model, and branching paths indicates the adoption of different training strategies in each stage.

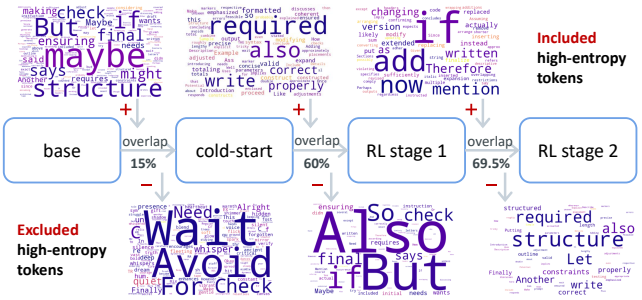


Figure 8: High-entropy token dynamics throughout training. Tokens with Top-200 average entropy and frequency more than 100 are viewed high-entropy.

Conclusion

In this paper, we propose a comprehensive framework to equip LLMs with generalizable reasoning for complex instruction following. Specifically, we first design a pipeline to synthesize complex prompts. Then, we post-train the base model via Zero-RL on synthetic prompts to encourage preview and self-checking behaviors. Leveraging responses from the Zero-RL model (optionally with external APIs), we collect effective reasoning samples and conduct Entropy-SFT to cold-start the base model. Subsequently, a two-stage easy-to-hard RL with TEA regularization fur-

ther enhances the reasoning capabilities. The resulting models, Light-IF-32B, Light-IF-1.7B, and Light-IF-1.7B-Zero, substantially outperform their respective base models. Notably, Light-IF-32B significantly surpasses powerful LLMs such as DeepSeek-R1 and Doubao-1.6, while the comparable performance between Light-IF-1.7B-Zero and Light-IF-1.7B demonstrates the promising solution for eliciting effective reasoning patterns without external guidance.

References

Abdin, M.; Agarwal, S.; Awadallah, A.; Balachandran, V.; Behl, H.; Chen, L.; de Rosa, G.; Gunasekar, S.; Javaheripi, M.; Joshi, N.; et al. 2025. Phi-4-reasoning technical report. *arXiv preprint arXiv:2504.21318*.

Bercovich, A.; Levy, I.; Golan, I.; Dabbah, M.; El-Yaniv, R.; Puny, O.; Galil, I.; Moshe, Z.; Ronen, T.; Nabwani, N.; et al. 2025. Llama-nemotron: Efficient reasoning models. *arXiv preprint arXiv:2505.00949*.

Cascella, M.; Montomoli, J.; Bellini, V.; and Bignami, E. 2023. Evaluating the feasibility of ChatGPT in healthcare: an analysis of multiple clinical and research scenarios. *Journal of medical systems*, 47(1): 33.

Chen, Q.; Qin, L.; Liu, J.; Peng, D.; Guan, J.; Wang, P.; Hu, M.; Zhou, Y.; Gao, T.; and Che, W. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

CLUE. 2025. SuperCLUE-CPIFOpen. https://github.com/CLUEbenchmark/SuperCLUE-CPIFOpen. Accessed: 2025-07-28.

Cui, C.; Ma, Y.; Yang, Z.; Zhou, Y.; Liu, P.; Lu, J.; Li, L.; Chen, Y.; Panchal, J. H.; Abdelraouf, A.; et al. 2024. Large language models for autonomous driving (llm4ad): Concept, benchmark, simulation, and real-vehicle experiment. *arXiv e-prints*, arXiv–2410.

Cui, G.; Zhang, Y.; Chen, J.; Yuan, L.; Wang, Z.; Zuo, Y.; Li, H.; Fan, Y.; Chen, H.; Chen, W.; et al. 2025. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*.

Ding, X.; Han, J.; Xu, H.; Zhang, W.; and Li, X. 2023. Hilm-d: Towards high-resolution understanding in multimodal large language models for autonomous driving. *arXiv preprint arXiv:2309.05186*.

Dong, G.; Lu, K.; Li, C.; Xia, T.; Yu, B.; Zhou, C.; and Zhou, J. 2024. Self-play with execution feedback: Improving instruction-following capabilities of large language models. *arXiv preprint arXiv:2406.13542*.

Fu, Y.; Chen, T.; Chai, J.; Wang, X.; Tu, S.; Yin, G.; Lin, W.; Zhang, Q.; Zhu, Y.; and Zhao, D. 2025. SRFT: A Single-Stage Method with Supervised and Reinforcement Fine-Tuning for Reasoning. *arXiv preprint arXiv:2506.19767*.

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

He, Q.; Zeng, J.; Huang, W.; Chen, L.; Xiao, J.; He, Q.; Zhou, X.; Liang, J.; and Xiao, Y. 2024. Can large language models understand real-world complex instructions? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 18188–18196.

Huang, H.; Liu, J.; He, Y.; Li, S.; Xu, B.; Zhu, C.; Yang, M.; and Zhao, T. 2025. Musc: Improving complex instruction following with multi-granularity self-contrastive training. *arXiv preprint arXiv:2502.11541*.

Jiang, Y.; Wang, Y.; Zeng, X.; Zhong, W.; Li, L.; Mi, F.; Shang, L.; Jiang, X.; Liu, Q.; and Wang, W. 2023. Followbench: A multi-level fine-grained constraints following benchmark for large language models. *arXiv preprint arXiv:2310.20410*.

Jing, Y.; Jin, R.; Hu, J.; Qiu, H.; Wang, X.; Wang, P.; and Xiong, D. 2023. Followeval: A multi-dimensional benchmark for assessing the instruction-following capability of large language models. *arXiv preprint arXiv:2311.09829*.

Leike, J.; Krueger, D.; Everitt, T.; Martic, M.; Maini, V.; and Legg, S. 2018. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*.

Li, Y.; Zheng, M.; Yang, F.; Dong, G.; Cui, B.; Chen, W.; Zhou, Z.; and Zhang, W. 2024. FB-Bench: A Fine-Grained Multi-Task Benchmark for Evaluating LLMs' Responsiveness to Human Feedback. *arXiv preprint arXiv:2410.09412*.

Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Lou, R.; Zhang, K.; and Yin, W. 2023. A comprehensive survey on instruction following. *arXiv preprint arXiv:2303.10475*, 1.

Luo, J.; Zhang, W.; Yuan, Y.; Zhao, Y.; Yang, J.; Gu, Y.; Wu, B.; Chen, B.; Qiao, Z.; Long, Q.; et al. 2025a. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*.

Luo, M.; Tan, S.; Wong, J.; Shi, X.; Tang, W. Y.; Roongta, M.; Cai, C.; Luo, J.; Zhang, T.; Li, L. E.; Popa, R. A.; and Stoica, I. 2025b. DeepScaleR: Surpassing O1-Preview with a 1.5B Model by Scaling RL. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2. Notion Blog.

OpenAI. 2024. GPT-4o System Card. https://openai.com/index/gpt-4o-system-card/. Accessed: 2025-07-28.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744.

Pyatkin, V.; Malik, S.; Graf, V.; Ivison, H.; Huang, S.; Dasigi, P.; Lambert, N.; and Hajishirzi, H. 2025. Generalizing Verifiable Instruction Following. *arXiv preprint arXiv:2507.02833*.

Qin, Y.; Song, K.; Hu, Y.; Yao, W.; Cho, S.; Wang, X.; Wu, X.; Liu, F.; Liu, P.; and Yu, D. 2024. Infobench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601*.

Ren, Q.; Zeng, J.; He, Q.; Liang, J.; Xiao, Y.; Zhou, W.; Sun, Z.; and Yu, F. 2025. Step-by-Step Mastery: Enhancing Soft Constraint Following Ability of Large Language Models. *arXiv preprint arXiv:2501.04945*.

Shao, H.; Hu, Y.; Wang, L.; Song, G.; Waslander, S. L.; Liu, Y.; and Li, H. 2024. Lmdrive: Closed-loop end-to-end driving with large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15120–15130.

Sheng, G.; Zhang, C.; Ye, Z.; Wu, X.; Zhang, W.; Zhang, R.; Peng, Y.; Lin, H.; and Wu, C. 2024. HybridFlow: A Flexible and Efficient RLHF Framework. *arXiv preprint arXiv: 2409.19256*.

Singhal, K.; Azizi, S.; Tu, T.; Mahdavi, S. S.; Wei, J.; Chung, H. W.; Scales, N.; Tanwani, A.; Cole-Lewis, H.; Pfohl, S.; et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972): 172–180.

Singhal, K.; Tu, T.; Gottweis, J.; Sayres, R.; Wulczyn, E.; Amin, M.; Hou, L.; Clark, K.; Pfohl, S. R.; Cole-Lewis, H.; et al. 2025. Toward expert-level medical question answering with large language models. *Nature Medicine*, 31(3): 943–950.

Sun, H.; Liu, L.; Li, J.; Wang, F.; Dong, B.; Lin, R.; and Huang, R. 2024. Conifer: Improving complex constrained instruction-following ability of large language models. *arXiv preprint arXiv:2404.02823*.

Sun, L.; Zhao, G.; Jian, X.; Wu, Y.; Lin, W.; Zhu, Y.; Zhang, L.; Wu, J.; Ran, J.; Hu, S.-e.; et al. 2025. Tinyr1-32b-preview: Boosting accuracy with branch-merge distillation. *arXiv preprint arXiv:2503.04872*.

Team, K.; Bai, Y.; Bao, Y.; Chen, G.; Chen, J.; Chen, N.; Chen, R.; Chen, Y.; Chen, Y.; Chen, Y.; et al. 2025. Kimi K2: Open Agentic Intelligence. *arXiv preprint arXiv:2507.20534*.

Volcengine. 2025a. doubao-1.5-pro-32k. https://www.volcengine.com/docs/82379/1554678. Accessed: 2025-07-28.

Volcengine. 2025b. Doubao 1.6 Thinking. https://www.volcengine.com/docs/82379/1593702. Accessed: 2025-07-28.

Wang, P.; Xu, A.; Zhou, Y.; Xiong, C.; and Joty, S. 2024a. Direct Judgement Preference Optimization. *arXiv preprint arXiv:2409.14664*.

Wang, S.; Yu, L.; Gao, C.; Zheng, C.; Liu, S.; Lu, R.; Dang, K.; Chen, X.; Yang, J.; Zhang, Z.; et al. 2025. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.

Wang, X.; Chen, Y.; Yuan, L.; Zhang, Y.; Li, Y.; Peng, H.; and Ji, H. 2024b. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

Wang, Y.; Kordi, Y.; Mishra, S.; Liu, A.; Smith, N. A.; Khashabi, D.; and Hajishirzi, H. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Wei, J.; Bosma, M.; Zhao, V. Y.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Wen, B.; Ke, P.; Gu, X.; Wu, L.; Huang, H.; Zhou, J.; Li, W.; Hu, B.; Gao, W.; Xu, J.; et al. 2024. Benchmarking complex instruction-following with multiple constraints composition. *Advances in Neural Information Processing Systems*, 37: 137610–137645.

Wen, L.; Cai, Y.; Xiao, F.; He, X.; An, Q.; Duan, Z.; Du, Y.; Liu, J.; Tang, L.; Lv, X.; et al. 2025. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. *arXiv preprint arXiv:2503.10460*.

Xu, C.; Sun, Q.; Zheng, K.; Geng, X.; Zhao, P.; Feng, J.; Tao, C.; Lin, Q.; and Jiang, D. 2024. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

Xu, L.; Li, A.; Zhu, L.; Xue, H.; Zhu, C.; Zhao, K.; He, H.; Zhang, X.; Kang, Q.; and Lan, Z. 2023. Superclue: A comprehensive chinese large language model benchmark. *arXiv preprint arXiv:2307.15020*.

Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Ye, Y.; Huang, Z.; Xiao, Y.; Chern, E.; Xia, S.; and Liu, P. 2025. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*.

Yu, Q.; Zhang, Z.; Zhu, R.; Yuan, Y.; Zuo, X.; Yue, Y.; Dai, W.; Fan, T.; Liu, G.; Liu, L.; et al. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Zhang, S.; Dong, L.; Li, X.; Zhang, S.; Sun, X.; Wang, S.; Li, J.; Hu, R.; Zhang, T.; Wu, F.; and Wang, G. 2024a. Instruction Tuning for Large Language Models: A Survey. arXiv:2308.10792.

Zhang, T.; Zhu, C.; Shen, Y.; Luo, W.; Zhang, Y.; Liang, H.; Yang, F.; Lin, M.; Qiao, Y.; Chen, W.; et al. 2024b. Cfbench: A comprehensive constraints-following benchmark for llms. *arXiv preprint arXiv:2408.01122*.

Zhang, X.; Yu, H.; Fu, C.; Huang, F.; and Li, Y. 2024c. IOPO: Empowering LLMs with Complex Instruction Following via Input-Output Preference Optimization. *arXiv preprint arXiv:2411.06208*.

Zhao, Y.; Yu, B.; Hui, B.; Yu, H.; Li, M.; Huang, F.; Zhang, N. L.; and Li, Y. 2024. Tree-instruct: A preliminary study of the intrinsic relationship between complexity and alignment. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 16776–16789.

Zheng, Y.; Zhang, R.; Zhang, J.; Ye, Y.; Luo, Z.; Feng, Z.; and Ma, Y. 2024. LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. Bangkok, Thailand: Association for Computational Linguistics.

Zhou, J.; Lu, T.; Mishra, S.; Brahma, S.; Basu, S.; Luan, Y.; Zhou, D.; and Hou, L. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A. Light Training Cost

The training cost for the Light-IF series models is relatively lightweight. Specifically, the Light-IF-32B model incurs a training cost of approximately $2,800, utilizing 11 nodes of A800 ×8 GPUs over 30 hours. Training the Light-IF-1.7B model costs around $342, utilizing 4 nodes of A800×8 GPUs over 10 hours.

B. Implementation Details

In this section, we provide detailed hyperparameters used during the cold-start stage, RL stage 1, and RL stage 2, as summarized in Tab. 5.

Model Names	LR	Batch Size	Length	Steps
SFT				
Light-IF-1.7B-Zero	1e-5	16	16K	375
Light-IF-1.7B	1e-5	16	16K	375
Light-IF-32B	5e-6	32	16K	187
RL Stage1				
Light-IF-1.7B-Zero	5e-6	128 (n=8, mini=32)	8K	100
Light-IF-1.7B	5e-6	128 (n=8, mini=32)	8K	100
Light-IF-32B	5e-6	352 (n=8, mini=88)	8K	100
RL Stage2				
Light-IF-1.7B-Zero	5e-6	128 (n=16, mini=32)	16K	30
Light-IF-1.7B	5e-6	128 (n=16, mini=32)	16K	30
Light-IF-32B	5e-6	352 (n=16, mini=88)	16K	75

Table 5: Training Hyperparameters. n refers to n responses per prompt during rollout, mini refers to mini batch size during RL policy optimization.

C. Cold-Start Data Filter

The filtering pipeline comprises three steps: correctness check, thinking check, and fluency check. In the correctness check step, we collect samples that strictly follow instructions, verified by automated code-based checks. In the subsequent thinking check step, we filter out samples exhibiting potentially lazy thinking patterns. Specifically, we apply a heuristic rule assuming that shorter thinking content generally indicates less thorough reasoning, discarding samples with fewer than 1,000 tokens. Finally, we remove samples with disfluent expressions by employing an LLM (ChatGPT-4o) as the evaluator. The prompt used for the LLM is provided in Tab. 7, and samples scoring below 8 are filtered out. After these steps, we select the top 2,000 samples ranked by reasoning length from the remaining candidates to form the final cold-start dataset.

D. Entropy Control during RL

We propose the Token-wise Entropy-Adaptive (TEA) method to maintain sufficient entropy during the RL stage. In this section, we compare TEA with existing entropy control methods from the literature, including entropy regularization and KL-Cov (Cui et al. 2025).

Experiments are conducted using the Light-IF-1.7B-EntSFT model with different entropy control strategies. The results presented in Tab. 6 indicate that TEA-RL consistently achieves the best performance across all four benchmarks. Additionally, the entropy dynamics over training steps are illustrated in Fig. 9. It is clear from the figure that TEA effectively maintains a high entropy level, whereas traditional entropy regularization methods tend to either cause entropy collapse or entropy explosion. Notably, despite sharing a similar theoretical foundation with KL-Cov, our token-wise entropy-adaptive regularization method demonstrates superior performance.

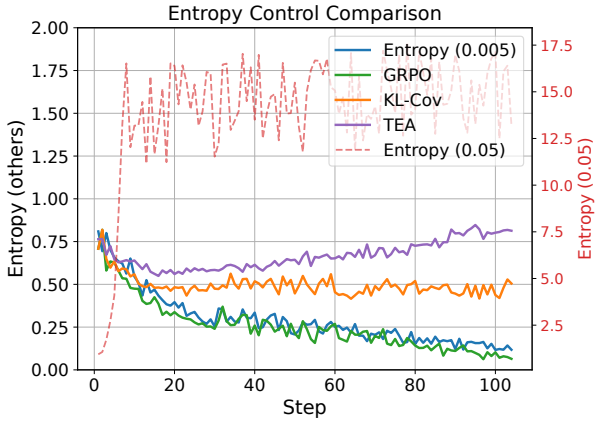


Figure 9: Entropy control during RL.

E. Dense Reward versus Sparse Reward

The multi-constraint nature of complex instruction-following tasks motivates the design of the dense reward. In this section, we compare the appearances of dense and sparse reward. Given the differences in absolute values between dense and sparse rewards, we normalize reward scores by dividing by the maximum possible reward value to obtain a relative reward score. The experiments are conducted using Light-IF-1.7B-EntSFT with TEA-RL.

As illustrated in Fig. 10, RL with sparse rewards exhibits slow improvement in reward scores, stagnates in validation accuracy, and displays considerable fluctuations. Conversely, RL with dense rewards demonstrates smooth and rapid improvements in both validation accuracy and relative reward scores, clearly indicating superior performance.

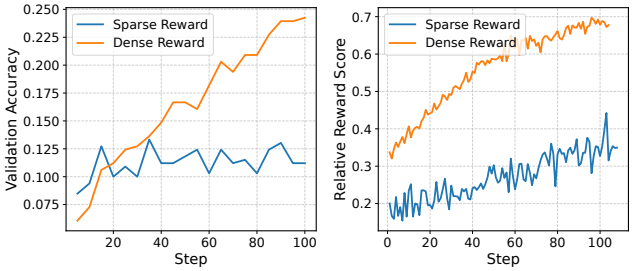


Figure 10: Appearances of the dense and sparse reward.

Method	Superclue	IFEval					CFBench				IFBench		
		LP	LI	SP	SI	AVG	CSR	ISR	PSR	AVG	PL	IL	AVG
GRPO	0.153	0.815	0.872	0.776	0.842	0.826	**0.85**	0.59	**0.70**	0.71	0.323	0.361	0.342
Entropy (0.005)	0.162	0.843	0.892	0.786	0.853	0.844	**0.85**	**0.60**	**0.70**	**0.72**	0.313	0.349	0.331
Entropy (0.05)	-	-	-	-	-	-	-	-	-	-	-	-	-
KL-Cov	0.209	0.861	0.903	0.826	0.878	0.867	0.83	0.57	0.68	0.69	**0.347**	0.388	**0.368**
TEA	**0.231**	**0.880**	**0.917**	**0.845**	**0.891**	**0.883**	**0.85**	**0.60**	**0.70**	**0.72**	0.344	**0.391**	**0.368**

Table 6: Comparisons of Entropy control strategies. LP: loose prompt, LI: loss instruction, SP: strict prompt, SI: strict instruction, PL: prompt level, IL: instruction level. Best are marked **bold**.

Please act as a reviewer and evaluate the quality of the model's responses as an AI assistant.
Your evaluation should prioritize the **fluency and readability** of the final answer. If the response is crafted solely to meet specific constraints and sacrifices natural flow or clarity, it should be rated poorly. For example, if the instruction requires the letter "n" to appear at least three times, and the response artificially includes multiple standalone instances of the letter "n" just to fulfill this requirement, such a response should receive a low score.
Next, consider whether the response **fulfills the instruction requirements**, and evaluate it strictly according to the **scoring criteria below**:
1) Score: 1–2, **Criteria:** The response is of poor quality, lacks fluency, and is written solely to satisfy the instruction constraints without regard for the overall quality of the reply.
2) Score: 3–5, **Criteria:** The response is generally readable but lacks overall fluency and smoothness.
3) Score: 6–8, **Criteria:** The response is of relatively high quality, with generally fluent and coherent language.
4) Score: 9–10, **Criteria:** The response is of high quality, with fluent, natural, and well-structured language.
Please provide the following: 1) A brief explanation to evaluate the quality of the AI assistant's response. If there are any issues with response quality or language fluency, please identify and briefly explain them;
2) Then provide an evaluation score, which must be strictly graded according to the following format: "[[rating]]", example: "Score:[[4]]".
`[Question]`
<question>
`[The Start of Assistant's Answer]`
<answer>
`[The End of Assistant's Answer]`

Table 7: Template for fluency check. Placeholders *<question>* and *<answer>* will be replaced with actual inputs.