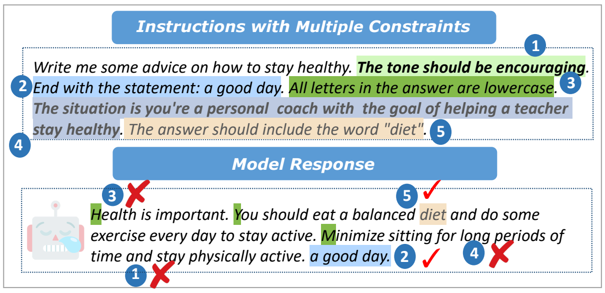
Step-by-Step Mastery: Enhancing Soft Constraint Following Ability of Large Language Models

Qingyu Ren[1,3], Jie Zeng[1], Qianyu He[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, jzeng23, qyhe21}@m.fudan.edu.cn, {liangjiaqing, shawyh}@fudan.edu.cn

Abstract

It is crucial for large language models (LLMs) to follow instructions that involve multiple constraints. In real-world scenarios, user instructions often contain soft constraints, which are semantically related and cannot be rule-based verified, posing challenges for LLMs. To enhance the soft constraint following ability of LLMs, we initially design a pipeline to construct datasets with high-quality outputs for instructions containing soft constraints automatically. Additionally, to fully utilize the positive and negative samples generated during the data construction process, we choose Direct Preference Optimization (DPO) as the training method. Furthermore, taking into account the difficulty of soft constraints indicated by the number of constraints, we design a curriculum learning training paradigm based on the constraint quantity. We experimentally evaluate the effectiveness of our methods in improving LLMs' soft constraint following ability and analyze the factors driving the improvements. The datasets and code are publicly available at https://github.com/Rainier-rq/FollowSoftConstraint.



Write me some advice on how to stay healthy. **The tone should be encouraging**. End with the statement: a good day. All letters in the answer are lowercase. **The situation is you're a personal coach with the goal of helping a teacher stay healthy**. The answer should include the word "diet".

Health is important. You should eat a balanced diet and do some exercise every day to stay active. Minimize sitting for long periods of time and stay physically active. a good day.

Figure 1: In real-world scenarios, user instructions contain many soft constraints, posing challenges for LLMs. We use **bold** to represent soft constraints.

1 Introduction

In the application of LLMs, the instruction following ability is of paramount importance, especially when the instructions involve multiple constraints (Lou et al., 2024; Zeng et al., 2023; Zhou et al., 2023a). The capability of LLMs plays a critical role in aligning with human preferences, ensuring the reliability and helpfulness of the models' outputs (Wang et al., 2023a; Song et al., 2024).

Following instructions with soft constraints is imperative for LLMs (Jiang et al., 2023b; Qin et al., 2024). Constraints can be categorized into soft and hard constraints. Hard constraints can be explicitly expressed as specific rules and directly verified through programming methods (Zhou et al., 2023a; He et al., 2024). For example, Python can parse JSON data to verify whether it follows specific format constraints. However, instructions in real-world applications often contain semantic-level limitations, which can be categorized as soft constraints. Soft constraints include restrictions related to content (Liang et al., 2024; Zhang et al., 2023), specific backgrounds (Shanahan et al., 2023; Liu et al., 2023), and the style of expressions (Sigurgeirsson and King, 2024; Mukherjee et al., 2024). They are difficult to verify automatically through programming methods. As shown in Fig. 1, following soft constraints is challenging for LLMs.

Existing methods for improving soft constraint following ability of LLMs have following limitations: First, much of the existing work focuses on evaluating LLMs' soft constraint following ability (Chen et al., 2024a; Qin et al., 2024) rather than how to improve the ability. Second, when constructing soft constraint datasets, current methods typically introduce all constraints at once and directly prompt advanced models to generate responses (Xu et al., 2023; Chiang et al., 2023). However, these responses may not satisfy all constraints. For example, GPT-4 demonstrates a constraint satisfaction rate of only 74.4% on FollowBench (Jiang et al., 2023b), making the assurance of high-quality outputs difficult. Third, in terms

[*] Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2025, pages 19581–19596
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

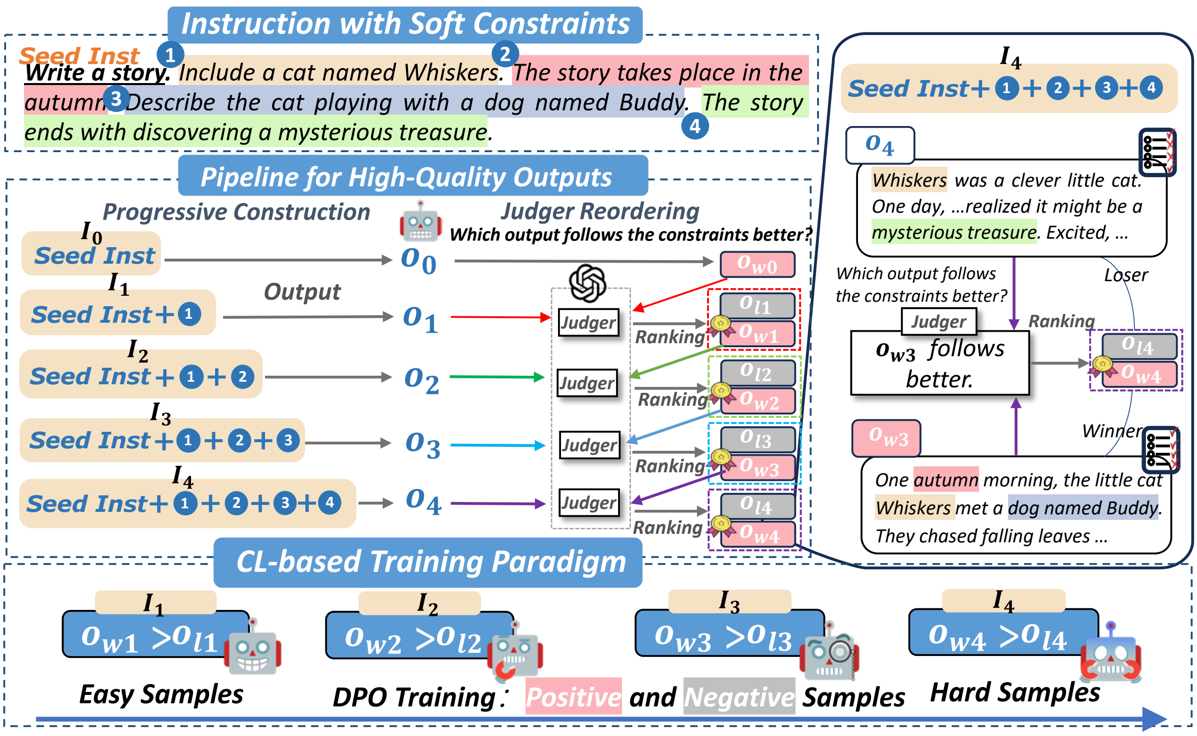

Instruction with Soft Constraints
Seed Inst
Write a story. Include a cat named Whiskers. The story takes place in the autumn. Describe the cat playing with a dog named Buddy. The story ends with discovering a mysterious treasure.
1
2
3
4
Pipeline for High-Quality Outputs
Progressive Construction
Judger Reordering
Which output follows the constraints better?
I_0 Seed Inst
I_1 Seed Inst+1
I_2 Seed Inst+1+2
I_3 Seed Inst+1+2+3
I_4 Seed Inst+1+2+3+4
Output
o_0
o_1
o_2
o_3
o_4
Judger
Ranking
o_{w0}
o_{l1} o_{w1}
o_{l2} o_{w2}
o_{l3} o_{w3}
o_{l4} o_{w4}
I_4 Seed Inst+1+2+3+4
o_4
Whiskers was a clever little cat. One day, ...realized it might be a mysterious treasure. Excited, ...
Which output follows the constraints better?
Judger
o_{w3} follows better.
Ranking
Loser
Winner
o_{w3}
One autumn morning, the little cat Whiskers met a dog named Buddy. They chased falling leaves ...
CL-based Training Paradigm
I_1 $o_{w1} > o_{l1}$
I_2 $o_{w2} > o_{l2}$
I_3 $o_{w3} > o_{l3}$
I_4 $o_{w4} > o_{l4}$
Easy Samples
DPO Training : Positive and Negative Samples
Hard Samples

Figure 2: The framework of our study. We first design a pipeline to automatically construct datasets with high-quality outputs for soft constraint following. Then, we propose a reinforcement learning training paradigm based on curriculum learning. CL denotes curriculum learning.

of the training paradigm, existing work ignores the difficulty of constraint following indicated by the number of constraints (He et al., 2024; Qi et al., 2024). Many studies show that organizing training data in a curriculum-based order—starting from simpler examples and gradually increasing complexity—leads to better performance than random shuffling (Sun et al., 2024; Lee et al., 2024). Therefore, a pipeline constructing datasets with high-quality outputs and an efficient training paradigm are required.

In this work, we systematically investigate strategies to enhance the ability of LLMs to follow soft constraints, with the framework shown in Fig. 2, including constructing datasets with high-quality outputs and proposing a reinforcement learning training paradigm based on curriculum learning. To construct datasets with high-quality outputs for soft constraint following, we progressively add constraints to the instructions and incorporate Judger to reorder the outputs based on constraint following. To fully leverage the positive and negative samples generated during Judger reordering, we apply Direct Preference Optimization (DPO) (Rafailov et al., 2024) as the training method. Since soft constraints cannot be verified through rule-based methods, Judger reordering can ensure that the constructed preference dataset adheres to the correct constraint-following behavior. To account for the difficulty of soft constraints indicated by the number of constraints, we propose a training paradigm that constructs curricula based on the number of constraints in the instruction. In this paradigm, the model starts by learning simpler tasks (fewer constraints) and progresses to more complex ones (more constraints). Our methods can improve LLMs' soft constraint following ability effectively. Moreover, our method is also effective for hard constraints.

Our contributions are summarized as follows: (1) We design a pipeline to automatically construct datasets with high-quality outputs for soft constraint following. (2) We introduce a reinforcement learning training paradigm that constructs curricula based on the number of constraints. (3) We conduct extensive experiments to validate the effectiveness of our methods and analyze the reasons for the performance improvements.

2 Related Work

Soft Constraint Following Existing research on soft constraint following focuses on evaluating the ability of LLMs by constructing benchmarks (Jiang et al., 2023b; Qin et al., 2024). These benchmarks include a variety of fine-grained constraint types (Zhang et al., 2024). These constraints can be categorized into several types: (1) Content soft constraints involve restrictions on the scope or depth of the responses (Zhou et al., 2023b; Ashok and Poczos, 2024). (2) Situation soft constraints refer to the background limitations (Wang et al., 2023b; Shao et al., 2023). (3) Style soft constraints limit the manner of expressions (Tao et al., 2024; Pu et al., 2024). Some works directly utilize responses generated by GPT-4 to construct datasets (Sun et al., 2024; Peng et al., 2023). Different from these, we propose a pipline constructing datasets with high-quality outputs to improve LLMs' soft constraint following ability.

Curriculum Learning Curriculum learning is a training strategy that mimics the learning process of humans from simpler to more complex tasks (Soviany et al., 2022; Wang et al., 2021). Current research on LLMs' curriculum learning can be categorized into two primary paradigms: (1) Learning Based on Data Difficulty: This approach organizes the training data sequence according to various evaluation metrics. Metrics such as sequence length (Pouransari et al., 2024), perplexity (Liu et al., 2024) have been employed to guide this process. LLMs can also construct curricula by organizing the training data sequence in a strategic way (Ryu et al., 2024). (2) Learning Based on Task Difficulty: This paradigm focuses on changing the training tasks (Chen et al., 2024b) or adjusting the training objectives (Zhao et al., 2024b; Lee et al., 2024). However, our work organizes the curriculum based on the number of constraints .

3 Method

In this section, we provide a detailed explanation of how to obtain datasets with high-quality outputs and how to leverage the dataset by establishing a curriculum learning training paradigm. The framework is shown in Fig. 2.

3.1 Dataset Construction

We first construct a multi-constraint instruction following dataset. Adding all constraints at once increases the complexity of the instructions rapidly, making it difficult for the model to understand all constraints (He et al., 2024). To addresss this, we propose a progressive construction method, adding one constraint at a time, which allows the model to understand and learn to follow each constraint effectively. Moreover, existing works in dataset construction rely on advanced models to directly generate the outputs (Sun et al., 2024). However, even GPT-4 is struggling to follow the instructions with complex soft constraints (Jiang et al., 2023b; Qin et al., 2024). Since soft constraints cannot be verified for compliance through rules, we use Judger to reorder the outputs based on the extent of constraint following to obtain high-quality outputs. Overall, our pipeline consists of two successive steps: **Progressive Construction** and **Judger Reordering**.

3.1.1 Progressive Construction

To enable the model to effectively learn how to follow each constraint, we propose a progressive construction method. Specifically, we add only one constraint at a time, enabling the model progressively learn to follow each constraint during the training process.

We begin by collecting seed instructions from three sources. We first collect instructions from Open Assistant (Köpf et al., 2024), which includes instructions generated by users interacting with chatbots. We select rank 0 instructions and those from the first turn of conversations. Next, we gather manually created instructions from the Self-Instruct (Wang et al., 2022a). The third source is Super-Natural (Wang et al., 2022b), from which we select instructions after filtering out tasks with simple outputs. These three sources together provide a total of 1,500 seed instructions, offering a broad range of coverage across diverse tasks.

Subsequently, we construct different types of soft constraints. Initially, we categorize the soft constraints into three types: content, situation, and style. Next, we randomly select constraints for each seed instruction. For the soft constraint type we select, GPT-4o is employed to generate corresponding descriptions. The prompt used to construct soft constraints is detailed in the Appx. A.1. For the hard constraint type, we select the description from a predefined description list.

To obtain multi-constraint instructions, we adopt a progressive construction approach. As shown in Fig. 2, we add only one constraint to the instruction at a time, allowing the model to focus on learning to follow each constraint. This process helps the model gradually adapt to the increasing complexity of the constraint following task and balance multiple constraints effectively. Specifically, for seed instruction I_0, we progressively add one constraint each time to form the instruction set $I = \{I_1, I_2, \ldots, I_n\}$, where n denotes the maximum number of constraints. For each instruction I_k with k constraints $(k = 1, 2, \ldots, n)$, we use GPT-4o to generate the corresponding output $O_k = \text{LLM}(I_k)$. After performing inference on all the instructions in the instruction set I, we obtain the output set $O = \{O_1, O_2, \ldots, O_n\}$.

3.1.2 Judger Reordering

In §3.1.1, we progressively increase the constraints, but the quality of the outputs may not improve incrementally. To address this, we introduce Judger to reorder the outputs based on constraint following to ensure the quality of outputs.

During the progressive construction process in §3.1.1, as new constraints are progressively added, the model's responses may overlook previously added constraints, leading to a decrease in the output quality. As shown in Fig. 2, to obtain high-quality outputs, we introduce Judger, where GPT-4o is prompted to compare two outputs before and after adding the new constraint, to determine which better follows the updated instruction. The two outputs in each comparison are recorded, and the one deemed better by Judger is used for the next round of comparison. By iteratively ranking the outputs, the constructed data is consistent with constraint following, thereby improving the output quality.

Specifically, when a new constraint is added into the instruction I_{k-1} to form I_k , the model's response O_k may not fully follow the constraints in I_k. To obtain high-quality outputs, we use Judger to rank the new output O_k with the previous output $O_{w_{k-1}}$ that more follows I_{k-1} to determine which one better follows the current instruction I_k: $O_{w_k}, O_{l_k} = \text{Judger}\ (I_k, O_{w_{k-1}}, O_k)$.

In each ranking, we can obtain the output O_{w_k} which follows the current instruction I_k better and the output O_{l_k} which follows less. Finally, after completing all n rankings, we obtain the positive set $O_w = \{O_{w_1}, O_{w_2}, \ldots, O_{w_n}\}$, which consists of outputs that follow their respective instructions better. We also obtain the negative set $O_l = \{O_{l_1}, O_{l_2}, \ldots, O_{l_n}\}$, which contains outputs that less follow. The prompt used to reorder outputs and reordering cases are detailed in the Appx. A.2.

3.2 Curriculum Learning Training Paradigm

In §3.1.2, we use Judger to obtain the positive set O_w and the negative set O_l. As shown in Fig. 2, to fully leverage both the positive and negative sets, we apply DPO as the training method. To account for the difficulty of soft constraints indicated by the number of constraints , we establish a curriculum learning training paradigm based on the number of constraints in the instruction.

Given the positive set and the negative set, we can construct the training dataset with n triplets: $(I_1, O_{w_1}, O_{l_1}), (I_2, O_{w_2}, O_{l_2}), \ldots, (I_n, O_{w_n}, O_{l_n})$. In each triplet, the output from O_w is preferred than the output from O_l. To fully leverage these samples, we apply DPO to train the model. To enable the model to learn from easy to hard tasks during training, we propose a curriculum learning training paradigm based on the number of constraints, starting with simpler tasks involving fewer soft constraints and progressively advancing to more complex tasks involving more soft constraints.

Specifically, for curriculum k, the training dataset D_k contains the triplet (I_k, O_{w_k}, O_{l_k}), where I_k represents the instruction with k constraints, and O_{w_k} and O_{l_k} denote the corresponding outputs in preference learning. The model begins with the simplest dataset, D_1, and sequentially progresses through D_2 to D_n, gradually enhancing its ability to handle more soft constraints. The complete curriculum is defined as $D = \{D_1, D_2, \ldots, D_n\}$. This stepwise approach ensures that the model first builds a strong foundation by learning from simpler datasets with fewer soft constraints, and gradually adapts to more complex datasets with more constraints. To prevent catastrophic forgetting during training (McCloskey and Cohen, 1989), we mix each curriculum's data with a proportion of 10k ShareGPT examples based on its data size (Chiang et al., 2023). Based on the curriculum learning training paradigm, the loss function of DPO training is as follows:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(I_k, O_{w_k}, O_{l_k}) \sim D_k} \Big[\log \sigma \Big(\beta \log \frac{\pi_\theta(O_{w_k}|I_k)}{\pi_{\text{ref}}(O_{w_k}|I_k)} - \beta \log \frac{\pi_\theta(O_{l_k}|I_k)}{\pi_{\text{ref}}(O_{l_k}|I_k)} \Big) \Big].$$

where π_θ represents the current model, and π_{ref} denotes the reference model.

To ensure training stability (Xu et al., 2024), we add the SFT loss into the DPO loss function:

$$\mathcal{L}_{\text{Ours}} = \mathcal{L}_{\text{DPO}} + \mathcal{L}_{\text{SFT}}.$$

Curriculum	# Constraints	# Preference Pairs	Avg Length
Curri.1	1	3714	369
Curri.2	2	3494	422
Curri.3	3	3387	461
Curri.4	4	3300	503
Curri.5	5	3148	516

Table 1: Statistics of curricula. # Constraints refers to the number of constraints in each instruction. # Preference Pairs refers to the number of preference pairs. "Avg Length" denotes the average instruction length.

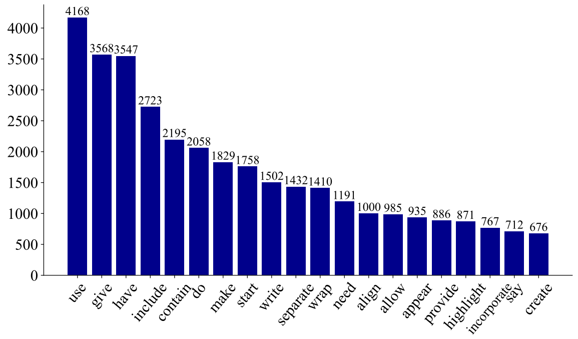


Figure 3: The verb frequency in the instructions.

where SFT loss is as follows:

$$\mathcal{L}_{\text{SFT}}(\pi_\theta) = -\mathbb{E}_{(I_k, O_{w_k}) \sim D_k}[\log \pi_\theta(O_{w_k}|I_k)].$$

3.3 Analysis and Comparison

3.3.1 Data Statistics

We present a statistical analysis of different curricula in Tab. 1. The results show that the number of constraints and instruction length in the curriculum continuously increase. Each curriculum contains a large scale of preference data. To show the diversity of our dataset, we analyze the frequency of verbs in the instructions. As shown in Fig. 3, the instructions contain a variety of verbs, reflecting diverse linguistic patterns. This diversity is crucial for enhancing the model's ability to generalize across different types of constraints.

3.3.2 Comparison with Other Works

As shown in Tab. 2, we compare our dataset with related works. Our dataset is large in scale compared to these methods. In terms of constraint categories, it includes both soft and hard constraints, enhancing the model's ability to learn to follow different types of constraints. Additionally, we use Judger for pairwise comparisons of outputs, improving the overall quality of the dataset. Moreover, our dataset is progressively constructed.

Method	Nums.	Cons.	Reor.	Prog.
Conifer (Sun et al., 2024)	13606	H/S	×	✓
Suri (Pham et al., 2024)	20000	S	×	×
AutoIF (Dong et al., 2024)	-	H	×	×
Complex to Simple (He et al., 2024)	12939	H	×	×
Ours	17043	H/S	✓	✓

Table 2: A detailed comparison of related works. Ours represents our dataset. "Nums.", "Cons.", "Reor.", and "Prog." denote the number of preference pairs, constraint types, whether to perform output reordering, and whether the dataset is progressively constructed.

4 Experiments

We conduct extensive experiments to validate the effectiveness of our method on improving LLMs' soft constraint following ability.

4.1 Experiment Setup

Models. We conduct experiments on several widely recognized base LLMs, including the LLaMA series (Dubey et al., 2024), Mistral series (Jiang et al., 2023a), and Qwen2.5 series (Yang et al., 2024). Within our experimental framework, we compare three approaches: (1) **BASE** directly utilizes the base model to generate outputs. (2) **SFT+Judger** applies supervised fine-tuning on LLMs using the instruction-response pairs (I_n, O_{w_n}) generated by Judger (§3.1.1,§3.1.2). (3) **DPO+Judger+CL** utilizes Judger to obtain high-quality training data, which is then used for DPO training following the curriculum learning training paradigm (§3.1.2, §3.2). For baseline comparisons, we include proprietary models such as GPT-4 (Achiam et al., 2023) and GPT-3.5 Turbo. Additionally, we also select models specifically designed to enhance the ability to follow multi-constraint instructions.

Settings. For each seed instruction, we progressively add five constraints. In the curriculum setting, we combine curriculum 1 to 3 into a simple curriculum and curriculum 4 to 5 into a difficult curriculum. The model is first trained on the simple curriculum, and then on the difficult one.

Evaluation Benchmarks. FollowBench (Jiang et al., 2023b) is a benchmark that evaluates the ability of models to follow both soft and hard constraints across multiple levels of granularity. CF-Bench (Zhang et al., 2024) is a benchmark that evaluates the ability of models to follow soft constraints. Each example needs to be evaluated by

Model	FollowBench (HSR)						IFEval				
	L1	L2	L3	L4	L5	Avg	[S]P	[S]I	[L]P	[L]I	Avg
GPT-4 (Achiam et al., 2023)*	84.7	75.6	70.8	73.9	61.9	73.4	76.9	83.6	79.3	85.4	81.3
GPT-3.5 Turbo*	80.3	68.0	68.6	61.1	53.2	66.2	-	-	-	-	-
WizardLM-v1.2-13B (Xu et al., 2023)	56.4	49.2	37.0	33.1	24.2	40.0	43.6	54.4	48.4	59.1	51.4
Vicuna-13B-v1.5 (Chiang et al., 2023)	56.2	42.9	32.3	32.1	24.6	37.6	43.1	53.6	46.6	58.0	50.3
Conifer-7B (Sun et al., 2024)	54.3	49.5	49.3	40.8	30.5	44.9	45.8	57.1	50.8	62.0	53.9
Conifer-7B-DPO (Sun et al., 2024)	60.3	53.6	48.0	47.1	41.0	50.0	48.1	59.1	52.3	63.3	55.7
Mistral$_{\text{CRAB}}$+DPO (Qi et al., 2024)	66.1	53.6	53.4	42.4	31.7	49.4	49.7	61.5	57.7	68.5	59.3
Mistral-7B-ShareGPT (Jiang et al., 2023a)	51.6	45.8	38.3	25.8	20.7	36.5	43.6	53.5	47.3	57.8	50.6
Llama3$_{\text{CRAB}}$+DPO (Qi et al., 2024)	64.6	49.0	41.6	35.8	36.8	45.5	40.3	52.0	47.7	58.9	49.7
Mistral-7B-Instruct-v0.2$_{\text{BASE}}$	58.1	52.1	44.6	39.8	29.9	44.9	44.5	56.4	49.2	61.5	52.9
Mistral-7B-Instruct-v0.2$_{\text{SFT+Judger}}$	58.1	50.5	44.6	36.0	36.2	45.1	44.2	56.5	47.5	60.2	52.1
Mistral-7B-Instruct-v0.2$_{\text{DPO+Judger+CL}}$	58.8	54.8	46.3	39.4	35.0	46.9	46.6	58.5	52.1	63.7	55.2
Mistral-7B-Instruct-v0.3$_{\text{BASE}}$	61.0	49.3	49.0	39.7	35.0	46.8	47.0	58.0	52.1	62.7	55.0
Mistral-7B-Instruct-v0.3$_{\text{SFT+Judger}}$	58.7	52.4	42.5	37.2	35.6	45.3	56.8	67.8	60.6	71.3	64.1
Mistral-7B-Instruct-v0.3$_{\text{DPO+Judger+CL}}$	63.3	56.0	47.5	40.0	36.7	48.7	53.4	63.5	57.1	67.5	60.4
LLaMA3-8B-Instruct$_{\text{BASE}}$	67.8	54.5	46.6	**50.6**	39.1	51.7	67.5	76.1	72.8	80.9	74.3
LLaMA3-8B-Instruct$_{\text{SFT+Judger}}$	66.3	55.4	50.1	49.7	39.8	52.3	70.4	77.8	73.2	80.1	75.4
LLaMA3-8B-Instruct$_{\text{DPO+Judger+CL}}$	**69.2**	**59.6**	**50.8**	48.9	**44.6**	**54.6**	**72.5**	**80.3**	**77.1**	**84.1**	**78.5**

Table 3: The overall performance on FollowBench and IFEval. We use **bold** for the best results and underlined for the second-best results among the models ranging from 7B to 13B parameter sizes.

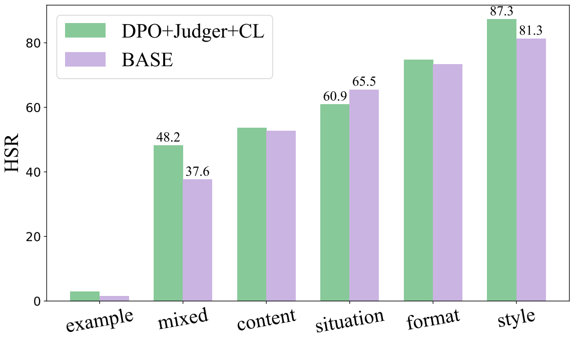


Figure 4: Results across various constraint categories.

GPT-4. IFEval (Zhou et al., 2023a) is a benchmark designed to evaluate the ability to follow hard constraints.

4.2 Main Results

Our method (DPO+Judger+CL) significantly enhances the model's ability to follow soft constraints across different models. Moreover, our method is also effective for hard constraints. As shown in Tab. 3, on FollowBench, which includes both soft and hard constraints, our method improves the model's performance. The improvement is significant on difficult constraint following tasks, especially at the L5 difficulty level in FollowBench. Specifically, LLaMA-3-8B-Instruct shows an improvement of 5.5% at the L5 difficulty level. For soft constraints, as shown in Tab. 4 and Tab. 5, we validate the effectiveness of our method using the soft constraint subset of FollowBench and CF-Bench. The results show that our method is also effective in improving performance on soft constraints. For hard constraints, model's performance on IFEval which includes only hard constraints demonstrates the effectiveness of our method. After training with SFT+Judger, there may be a decline in the performance. This drop is attributed to the model's integration of various specialized training techniques during its initial training phase.

From the perspective of constraint types, as shown in Fig. 4, our method significantly improves the model's performance across different types of constraints. The most notable improvement is observed in the Mixed category, which is defined as a composition of multiple constraint categories to simulate real-world scenarios. Our method enhances the model's performance by 10.6% in this category, suggesting a notable enhancement in the model's ability to handle complex constraints in real-world scenarios. Moreover, our method improves model's performance by 6% in the Style category which contains only soft constraints. In the category of situation constraints, the decrease in performance is due to the fact that our constructed situation constraints mainly focus on soft constraints. As a result, our method struggles with hard constraints in this category, such as complex situational reasoning.

4.3 Generalization Experiments

Besides the ability to follow soft constraints, we also assess the model's general instruction following abilities on AlpacaEval2.0 (Zhao et al., 2024a) and MT-Bench (Zheng et al., 2023). We first perform SFT on LLaMA3-8B-Instruct, followed by

Model	Hard Satisfaction Rate						Soft Satisfaction Rate					
	L1	L2	L3	L4	L5	Avg	L1	L2	L3	L4	L5	Avg
Mistral-7B-Instruct-v0.2$_{\text{BASE}}$	70.5	58.4	50.6	51.7	39.0	54.0	70.5	67.5	66.1	64.1	61.2	65.9
Mistral-7B-Instruct-v0.2$_{\text{DPO+Judger+CL}}$	70.3	65.4	55.8	47.4	41.5	56.1	70.3	71.3	67.1	64.1	63.9	67.3
LLaMA3-8B-Instruct$_{\text{BASE}}$	81.9	60.5	60.7	55.4	52.0	62.1	81.9	66.4	69.5	68.7	69.3	71.2
LLaMA3-8B-Instruct$_{\text{DPO+Judger+CL}}$	81.2	67.5	64.9	56.3	49.9	64.0	81.2	73.3	73.3	70.3	69.3	73.5
Qwen2.5-32B-Instruct$_{\text{BASE}}$	89.6	78.5	**79.6**	70.3	63.7	76.3	89.6	85.0	82.0	**82.0**	75.1	82.7
Qwen2.5-32B-Instruct$_{\text{DPO+Judger+CL}}$	**91.2**	**82.6**	78.6	**72.2**	**69.0**	**78.7**	**91.2**	**85.1**	**84.1**	79.5	**80.4**	**84.1**

Table 4: The overall performance on the soft constraint subset of FollowBench. We use **bold** for the best results and underlined for the second-best results.

Models	Easy Set			Hard Set			Full Set			Avg.
	CSR	ISR	PSR	CSR	ISR	PSR	CSR	ISR	PSR	
BASE	0.770	0.450	0.530	0.640	0.160	**0.320**	0.700	0.300	0.430	0.478
DPO+judger+CL	**0.790**	**0.480**	**0.560**	**0.670**	**0.190**	0.320	**0.730**	**0.330**	**0.440**	**0.501**

Table 5: The overall performance of LLaMA3-8B-Instruct on CFBench. We use **bold** for the best results.

Model	BaseModel	AlpacaEval2.0	MT-Bench
GPT-4-0613*	GPT	30.2	9.18
GPT-3.5-Turbo-0613*	GPT	22.4	8.39
LLaMA-3.1-70B-Instruct*	LLaMA3	39.3	8.22
WizardLM-13B-v1.2*	LLaMA2	14.5	7.20
Vicuna-13B-v1.5*	LLaMA2	10.5	6.57
BASE	LLaMA3	21.6	6.78
DPO+Judger+CL	LLaMA3	22.0	6.80

Table 6: Results of the length control win rate of AlpacaEval2.0 and the score of MT-Bench. * indicates that the results are directly sourced from the original leaderboards.

Model	FollowBench (HSR)			IFEval
	L1 - L3	L4 - L5	Avg	Avg
BASE	56.3	44.9	51.7	74.3
SFT	59.5	38.4	51.0	73.8
SFT+Judger	57.3	44.8	52.3	75.4
DPO+Judger	58.8	44.6	53.1	78.4
DPO+Judger+CL	59.9	46.8	**54.6**	**78.5**

Table 7: Ablation study results on FollowBench and IFEval.

Method	FollowBench (HSR)					
	L1	L2	L3	L4	L5	Avg
BASE	67.8	54.5	46.6	50.6	39.1	51.7
w/o progressive	69.8	56.7	50.8	42.4	47.6	53.5
w/ progressive	69.2	59.6	50.8	48.9	44.6	**54.6**

Table 8: Results of different construction methods.

DPO+Judger+CL. Specifically, we use precomputed outputs of GPT-4 Turbo on AlpacaEval as reference outputs and GPT-4o as evaluators. As shown in the Tab. 6, our method improves the model's general instruction following ability on both banchmarks.

4.4 Ablation Studies

In this section, we conduct ablation experiments to study the impact of Judger and curriculum learning on LLaMA3-8B-Instruct. As shown in Tab. 7, directly using the constructed data for SFT without Judger reordering underperforms the SFT+Judger method on both benchmarks, even worse than the base model. The performance decreases significantly at the L4-L5 levels of FollowBench. This suggests that Judger plays a critical role in ranking the model's responses to challenging instructions. The model trained with DPO outperforms the SFT baseline, especially on IFEval, emphasizing the effectiveness of DPO over SFT in constraint following tasks. Additionally, the curriculum learning training paradigm improves the model's ability to follow constraints on both benchmarks, particularly those at higher difficulty levels (L4-L5) of FollowBench. This validates the necessity of curriculum learning paradigm for enhancing the model's ability to follow soft constraints.

4.5 Analysis

4.5.1 The Role of Progressive Construction

In this section, we analyze the role of progressively constructing the dataset. We compare two methods: w/o progressive and w/ progressive. w/o progressive means replacing both the instruction field and the chosen field with the corresponding ones from samples that include all constraints in the pro-

Ranking Method	Kendall Tau Coefficient	Position Consistency
w/o Judger	0.847	0.743
w/ Judger	**0.862**	**0.794**

Table 9: Results on Judger's effectiveness.

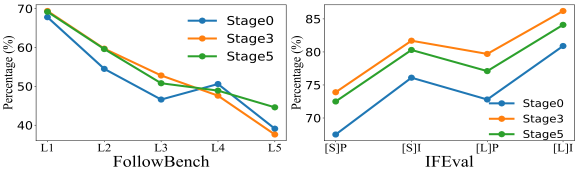


Figure 5: Results of the model across different training stages in curriculum learning.

gressively constructed DPO dataset. As shown in Tab. 8, our progressive construction method enhances the model's performance on FollowBench, especially at the difficlut level L4. Introducing all constraints at once makes the training process unstable. By progressively adding constraints, the model can focus on learning to follow one constraint at a time and learn how to balance constraints effectively.

	Judger Ranking Case
Instruction	Please write a description of quantum physics, including "wave function" and "superposition state", avoiding complex formulas, under 100 words. Ensure this description is easy to understand.
Responses	(a) Quantum physics studies the behavior of tiny particles. The *wave function* describes a particle's state, while *superposition state* means a particle can exist in multiple states at once. (b) Quantum physics studies the behavior of tiny particles. In the quantum world, particles don't have fixed positions or states; they can be in many states at once, and only take a definite state when measured.
Judger	Output (a) is better. It includes required terms, explains them simply, and follows all constraints. Output (b) misses "wave function" and is less precise.

Table 10: The case for Judger ranking outputs.

4.5.2 The Role of Judger

In this section, we investigate the role of Judger in obtaining high-quality outputs. Specifically, we randomly select 100 seed instructions and rank the outputs at each step of progressive construction. We evaluate the rankings in three scenarios: (1) w/o Judger, (2) w/ Judger and (3) rankings annotated by human experts, which serve as the reference standards. As shown in Tab. 10, Output(a) represents the model's initial response generated without the newly introduced constraint "Ensure that this description is easy to understand" while Output(b) shows the response produced after incorporating this additional constraint. In the w/o Judger setting, it assumes Output(b) to be superior by default, without performing reordering. To assess the similarity between the rankings, we employ two metrics. The first is the **Kendall Tau Coefficient** (Kendall, 1938), which measures the correlation between two rankings by assessing the agreement in the order of paired items. Formally, given two rankings R_1 and R_2 over the same set of n items. Kendall Tau Coefficient is defined as

$$\tau(R_1, R_2) = \frac{\sum_{1 \leq i < j \leq n} \operatorname{sgn}(R_1(i) - R_1(j)) \operatorname{sgn}(R_2(i) - R_2(j))}{\frac{1}{2} n(n-1)}$$

The second metric is **Position Consistency**, which quantifies the proportion of elements that occupy the same relative positions in both rankings. If $\operatorname{pos}_{R_1}(i)$ and $\operatorname{pos}_{R_2}(i)$ denote the positions of item i in R_1 and R_2, respectively, then

$$\text{PC}(R_1, R_2) = \frac{1}{n} \left| \{ i \mid \operatorname{pos}_{R_1}(i) = \operatorname{pos}_{R_2}(i) \} \right|$$

As shown in Tab. 9, the rankings adjusted by the Judger exhibit greater alignment with human-annotated rankings. This suggests that Judger enhances the consistency of outputs with human judgments, thereby improving their quality.

4.5.3 The Role of Curriculum Learning

We analyze the effects of the curriculum learning paradigm. Specifically, we compare the performance of LLaMA3-8B-Instruct with the DPO+Judger+CL method across three training stages. Stage0 represents the base model, while Stage3 and Stage5 represent stages where the model completes the easy curriculum and the hard curriculum, respectively.

As shown in Fig. 5, our training paradigm progressively enhances the model's constraint following capability across various training stages. Specifically, after easy curriculum learning, the model trained in Stage3 shows superior performance compared to the base model across L1-L3. The model's performance at L4-L5 in Stage3 is lower than Stage0. The reason is Stage3 has not adequately prepared for the complexity of L4-L5.

The gap between these difficulty levels leads to the performance drop. Subsequentially, when the model progresses to Stage5, after hard curriculum learning, the average performance improves significantly at the difficlut levels L4-L5. The results on IFEval further support this conclusion. Stage0 has the lowest average performance across all indicators. After curriculum learning, there is a significant improvement in the model's performance on IFEval. Although Stage 5 performs slightly worse than Stage 3 on IFEval, Stage 5 shows a significant performance improvement on FollowBench, indicating the enhancement in the model's ability to follow soft constraints.

5 Conclusion

In this paper, we systematically study how to improve LLMs' ability to follow soft constraints. Initially, we design a pipeline to automate the construction of datasets with high-quality outputs for soft constraint following. Based on the pipeline, we introduce a reinforcement learning training method utilizing positive and negative samples generated during the pipeline. Moreover, we propose a new training paradigm that leverages curriculum learning to enhance LLMs' soft constraint following ability. The experiment results show that our methods enhance models' ability to follow soft constraints effectively.

6 Limitations

We discuss the limitations of our study as follows. First, we improve the model's ability to follow soft constraints, thereby improving its overall instruction following capability. However, even when the model's output meets all the specified constraints, it may still struggle to fully comply with complex instructions due to limitations in reasoning ability or the knowledge it masters. Additionally, while the dataset constructed encompasses a diverse set of tasks, it may still not cover some task types in the long tail. We consider these as key directions for future research.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Dhananjay Ashok and Barnabas Poczos. 2024. Controllable text generation in the instruction-tuning era. *arXiv preprint arXiv:2405.01490*.

Yihan Chen, Benfeng Xu, Quan Wang, Yi Liu, and Zhendong Mao. 2024a. Benchmarking large language models on controllable generation under diversified instructions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17808–17816.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024b. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

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*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93.

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

JoonHo Lee, Jae Oh Woo, Juree Seok, Parisa Hassanzadeh, Wooseok Jang, JuYoun Son, Sima Didari, Baruch Gutow, Heng Hao, Hankyu Moon, et al. 2024. Improving instruction following in language models through proxy-based uncertainty estimation. *arXiv preprint arXiv:2405.06424*.

Xun Liang, Hanyu Wang, Yezhaohui Wang, Shichao Song, Jiawei Yang, Simin Niu, Jie Hu, Dan Liu, Shunyu Yao, Feiyu Xiong, et al. 2024. Controllable text generation for large language models: A survey. *arXiv preprint arXiv:2408.12599*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Yinpeng Liu, Jiawei Liu, Xiang Shi, Qikai Cheng, Yong Huang, and Wei Lu. 2024. Let's learn step by step: Enhancing in-context learning ability with curriculum learning. *arXiv preprint arXiv:2402.10738*.

Renze Lou, Kai Zhang, and Wenpeng Yin. 2024. Large language model instruction following: A survey of progresses and challenges. *Computational Linguistics*, pages 1–10.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Sourabrata Mukherjee, Atul Kr Ojha, and Ondřej Dušek. 2024. Are large language models actually good at text style transfer? *arXiv preprint arXiv:2406.05885*.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Chau Minh Pham, Simeng Sun, and Mohit Iyyer. 2024. Suri: Multi-constraint instruction following for long-form text generation. *arXiv preprint arXiv:2406.19371*.

Hadi Pouransari, Chun-Liang Li, Jen-Hao Rick Chang, Pavan Kumar Anasosalu Vasu, Cem Koc, Vaishaal Shankar, and Oncel Tuzel. 2024. Dataset decomposition: Faster llm training with variable sequence length curriculum. *arXiv preprint arXiv:2405.13226*.

Xiao Pu, Tianxing He, and Xiaojun Wan. 2024. Style-compress: An llm-based prompt compression framework considering task-specific styles. *arXiv preprint arXiv:2410.14042*.

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*.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. Infobench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Kanghyun Ryu, Qiayuan Liao, Zhongyu Li, Koushil Sreenath, and Negar Mehr. 2024. Curricullm: Automatic task curricula design for learning complex robot skills using large language models. *arXiv preprint arXiv:2409.18382*.

Murray Shanahan, Kyle McDonell, and Laria Reynolds. 2023. Role play with large language models. *Nature*, 623(7987):493–498.

Yunfan Shao, Linyang Li, Junqi Dai, and Xipeng Qiu. 2023. Character-llm: A trainable agent for role-playing. *arXiv preprint arXiv:2310.10158*.

Atli Sigurgeirsson and Simon King. 2024. Controllable speaking styles using a large language model. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 10851–10855. IEEE.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2024. Preference ranking optimization for human alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18990–18998.

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. 2022. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565.

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*.

Zhen Tao, Dinghao Xi, Zhiyu Li, Liumin Tang, and Wei Xu. 2024. Cat-llm: Prompting large language models with text style definition for chinese article-style transfer. *arXiv preprint arXiv:2401.05707*.

Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576.

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, et al. 2022b. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. 2023a. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966*.

Zekun Wang, Ge Zhang, Kexin Yang, Ning Shi, Wangchunshu Zhou, Shaochun Hao, Guangzheng Xiong, Yizhi Li, Mong Yuan Sim, Xiuying Chen, et al. 2023b. Interactive natural language processing. *arXiv preprint arXiv:2305.13246*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. *arXiv preprint arXiv:2401.08417*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2023. Evaluating large language models at evaluating instruction following. *arXiv preprint arXiv:2310.07641*.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2023. A survey of controllable text generation using transformer-based pre-trained language models. *ACM Computing Surveys*, 56(3):1–37.

Tao Zhang, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, Bin Cui, et al. 2024. Cfbench: A comprehensive constraints-following benchmark for llms. *arXiv preprint arXiv:2408.01122*.

Hao Zhao, Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. 2024a. Long is more for alignment: A simple but tough-to-beat baseline for instruction fine-tuning. *arXiv preprint arXiv:2402.04833*.

Zirui Zhao, Hanze Dong, Amrita Saha, Caiming Xiong, and Doyen Sahoo. 2024b. Automatic curriculum expert iteration for reliable llm reasoning. *arXiv preprint arXiv:2410.07627*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023a. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023b. Controlled text generation with natural language instructions. In *International Conference on Machine Learning*, pages 42602–42613. PMLR.

A Appendix

A.1 Details of Soft Constraints

We utilize GPT-4o to construct soft constraints. The three categories of soft constraints that we define are as follows:

- **Soft Constraints in Content**: Content soft constraints refer to limitations associated with the data itself. These constraints govern the elements of information, the logical relationships between them, and the scope of topics that need to be covered in the response. When multiple content soft constraints are imposed, the model is required to not only generate comprehensive and coherent content but also ensure that the response aligns with the specific logical definitions and boundaries outlined by the instruction. This presents a significant challenge, as it demands both the integration of diverse elements and the maintenance of internal consistency. To address this challenge, we define the following tasks for constructing and applying content soft constraints:

  1. **Inclusion of Key Elements**: The response must incorporate the key points specified in the instruction. This requires the model to effectively extract and integrate relevant information, ensuring that the essential components are included without omitting critical details.
  2. **Topic Focus**: The model must narrow the discussion to a specific subtopic, avoiding broad generalizations or irrelevant tangents. This task emphasizes the importance of maintaining focus and precision within the scope defined by the instruction.
  3. **Strict Structure**: The generated content must adhere to a predefined structure, such as being organized into coherent paragraphs, utilizing subheadings, or following a specific format. This task imposes a higher demand on the model's ability to generate well-organized and structured outputs, aligning with the required presentation structure.

  We provide the prompt template for constructing the Content Soft Constraint in Tab. 11 and Tab. 12.

- **Soft Constraints in Situation**: Situation soft constraints are those related to the context within which the response is situated. These constraints require the response to be adjusted according to the context or assumptions specified in the instruction, ensuring that the content is appropriate to the given background. Such adjustments may involve factors like a particular time or location, the assumption of a specific role, or drawing conclusions based on certain premises. The response must dynamically adapt to situational changes and maintain consistency with the contextual elements. The tasks defined by these constraints can be categorized as follows:

  1. **Role-Playing**: The response must be framed from the perspective of a specific role or persona, ensuring alignment with the contextual expectations associated with that role.
  2. **Decision Support**: The response should provide advice or recommendations that support decision-making within a particular context.
  3. **Storytelling**: The response should construct a narrative that is situated within a defined time, location, or background, maintaining coherence with the provided contextual elements.

  We provide the prompt template for constructing the Situation Soft Constraint in Tab. 13, Tab. 14, and Tab. 15.

- **Soft Constraints in Style**: Style soft constraints pertain to the mode of expression, encompassing factors such as the formality or informality of tone, the level of conciseness in language, and the emotional tenor. These constraints require the response to adjust its style in accordance with the given requirements, adapting to different linguistic contexts. The following task types are defined under this category:

  1. **Tone Requirement**: The generated content must adopt a specific tone, such as formal, humorous, or otherwise defined.
  2. **Language Complexity Control**: The complexity of the language used must adhere to specific standards, such as maintaining conciseness and clarity or employing academic expressions.

3. **Emotional Expression**: The response must convey a particular emotion, such as positivity or sadness, as dictated by the context.

We provide the prompt template for constructing the Style Soft Constraint in Tab. 16.

A.2 Details of Judger Reordering

We utilize GPT-4o to reorder the outputs. We provide the prompt of Judger ranking in Tab. 17 and examples of how the Judger ranks responses in Tab. 10.

A.3 Implementation Details

We train LLaMA-3-8B-Instruct, Mistral-7B-Instruct-v0.3, and LLaMA2-13B-Chat-HF using LLaMA-Factory (Zheng et al., 2024) on 4 NVIDIA A100 80GB GPUs, applying LoRA (Hu et al., 2021) for efficient training. The lora target is set to all, with all models training for 3 epochs. The per device train batch size is set to 1, and gradient accumulation steps is set to 8. The warm-up ratio is set to 0.1. For SFT, LLaMA-3-8B-Instruct has a learning rate of 1.0e-4, Mistral-7B-Instruct-v0.3 uses 5.0e-7, and LLaMA2-13B-Chat-HF uses 1.0e-7. For DPO, the learning rate is 5.0e-6 with a beta value of 0.1. We apply cosine learning rate scheduler. For the benchmark evaluation, we adopt the online serving inference approach using vLLM.

You are an Instruction Rewriting Expert. You need to rewrite #Given Instruction# based on #Rewriting Requirement#, in order to obtain a #Rewritten Instruction#. Basically, #Rewritten Instruction# should adhere to the following guidelines:
1. Your rewriting cannot omit the non-text parts such as the table and code in #Given Instruction#.
2. #Rewritten Instruction# must be reasonable and must be understood and responded to by humans.
3. You should try your best not to make the #Rewritten Instruction# become verbose, #Rewritten Instruction# can only add 10 to 20 words into #Given Instruction#.
/* The Given Instruction */
{Given Instruction}
/* Rewriting Requirement */
Please add one proper content constraint to the #Given Instruction#. The content constraints include but are not limited to:
1. Add a Subtask or Another Related Question.
2. Narrow Down the Topic: Instead of a general theme or topic, provide a more specific subset.
3. Set a Higher Standard: Raise the bar for what's considered acceptable or successful.
4. Limit Resources: Restrict the number or type of resources someone can use.
5. Introduce Specific Criteria: Mandate particular components or features that must be included.
6. Specifying Sequence: Dictate the order in which certain steps or actions should be taken.
Please output in JSON format with the fields 'modified_instruction' for the modified instruction and 'added_constraint' for the added constraint.

Table 11: The prompt template for constructing the open-ended question answering task in Content Soft Constraint (Jiang et al., 2023b).

You are an Instruction Rewriting Expert. You need to rewrite #Given Instruction# based on #Rewriting Requirement#, in order to obtain a #Rewritten Instruction#. Basically, #Rewritten Instruction# should adhere to the following guidelines:
1. Your rewriting cannot omit the non-text parts such as the table and code in #Given Instruction#.
2. #Rewritten Instruction# must be reasonable and must be understood and responded to by humans.
3. You should try your best not to make the #Rewritten Instruction# become verbose, #Rewritten Instruction# can only add 10 to 20 words into #Given Instruction#.
/* The Given Instruction */
{Given Instruction}
/* Rewriting Requirement */
Please add one proper content constraint to the #Given Instruction#. The content constraints include but are not limited to:
1. Specify Language Complexity: Determine whether the text should use simple, intermediate, or advanced language.
2. Control Output Length: Set limits on the text's length, such as maximum word count or number of paragraphs.
3. Restrict Vocabulary: Include or exclude specific words or phrases, or limit the range of vocabulary.
4. Mandate Structure: Require a specific format, such as headings, bullet points, or a particular narrative style.
Please output in JSON format with the fields 'modified_instruction' for the modified instruction and 'added_constraint' for the added constraint.

Table 12: The prompt template for constructing the language limitations in Content Soft Constraint.

You are an Instruction Rewriting Expert. You need to rewrite #Given Instruction# based on #Rewriting Requirement#, in order to obtain a #Rewritten Instruction#. Basically, #Rewritten Instruction# should adhere to the following guidelines:
1. Your rewriting cannot omit the non-text parts such as the table and code in #Given Instruction#.
2. #Rewritten Instruction# must be reasonable and must be understood and responded to by humans.
3. You should try your best not to make the #Rewritten Instruction# become verbose, #Rewritten Instruction# can only add 10 to 20 words into #Given Instruction#.
/* The Given Instruction */
{Given Instruction}
/* Rewriting Requirement */
Please add one proper situation constraint to the #Given Instruction#. The situation constraints include but are not limited to:
1. Define the Context: Specify a particular situation or environment that the suggestions should be relevant to.
2. Introduce a Specific Problem: Focus on addressing a distinct problem or challenge that needs suggestions.
3. Impose Urgency: Include a time constraint or urgency for when the suggestions should be applied.
4. Limit Options: Restrict the scope of potential suggestions to a narrower set of choices.
5. Add Dependencies: Require that suggestions consider certain conditions or prerequisites.
6. Prioritize Outcomes: Highlight specific outcomes or goals that the suggestions should aim to achieve.
Please output in JSON format with the fields 'modified_instruction' for the modified instruction and 'added_constraint' for the added constraint.

Table 13: The prompt template for constructing the suggestion generation task in Situation Soft Constraint.

You are an Instruction Rewriting Expert. You need to rewrite #Given Instruction# based on #Rewriting Requirement#, in order to obtain a #Rewritten Instruction#. Basically, #Rewritten Instruction# should adhere to the following guidelines:
1. Your rewriting cannot omit the non-text parts such as the table and code in #Given Instruction#.
2. #Rewritten Instruction# must be reasonable and must be understood and responded to by humans.
3. You should try your best not to make the #Rewritten Instruction# become verbose, #Rewritten Instruction# can only add 10 to 20 words into #Given Instruction#.
/ The Given Instruction */*
{Given Instruction}
/ Rewriting Requirement */*
Please add one proper situation constraint to the #Given Instruction#. The situation constraints include but are not limited to:
1. Specify a Role: Clearly define the role or persona to be taken on during the role-play.
2. Define the Setting: Outline the environment or context in which the role-play should occur.
3. Add Conflict or Challenge: Introduce a specific problem, conflict, or challenge that must be addressed within the role-play.
4. Limit the Actions: Restrict the types or number of actions that can be taken during the role-play.
5. Set Specific Goals: Define clear objectives that the role-player must achieve.
6. Introduce Time Constraints: Impose a time limit for the role-play to unfold or for certain actions to be completed.
Please output in JSON format with the fields ’modified_instruction’ for the modified instruction and ’added_constraint’ for the added constraint.

Table 14: The prompt template for constructing the role-playing task in Situation Soft Constraint.

You are an Instruction Rewriting Expert. You need to rewrite #Given Instruction# based on #Rewriting Requirement#, in order to obtain a #Rewritten Instruction#. Basically, #Rewritten Instruction# should adhere to the following guidelines:
1. Your rewriting cannot omit the non-text parts such as the table and code in #Given Instruction#.
2. #Rewritten Instruction# must be reasonable and must be understood and responded to by humans.
3. You should try your best not to make the #Rewritten Instruction# become verbose, #Rewritten Instruction# can only add 10 to 20 words into #Given Instruction#.
/ The Given Instruction */*
{Given Instruction}
/ Rewriting Requirement */*
Please add one proper situation constraint to the #Given Instruction#. The situation constraints include but are not limited to:
1. Define Character Archetypes: Specify certain archetypes or roles characters should fulfill, such as a hero, mentor, or antagonist.
2. Include Specific Plot Points: Mandate the inclusion of certain events or plot twists that must occur.
3. Moral Dilemmas: Introduce a scenario where the characters must make a tough decision that involves competing ethical principles or risks.
Please output in JSON format with the fields ’modified_instruction’ for the modified instruction and ’added_constraint’ for the added constraint.

Table 15: The prompt template for constructing the story generation task in Situation Soft Constraint.

You are an Instruction Rewriting Expert. You need to rewrite #Given Instruction# based on #Rewriting Requirement#, in order to obtain a #Rewritten Instruction#. Basically, #Rewritten Instruction# should adhere to the following guidelines: 1. Your rewriting cannot omit the non-text parts such as the table and code in #Given Instruction#. 2. #Rewritten Instruction# must be reasonable and must be understood and responded to by humans. 3. You should try your best not to make the #Rewritten Instruction# become verbose, #Rewritten Instruction# can only add 10 to 20 words into #Given Instruction#.
/ The Given Instruction */*
{Given Instruction}
/ Rewriting Requirement */*
Please add one proper style constraint to the #Given Instruction#. The style constraints include but are not limited to:
1. Tone and Emotion: Specify the desired emotional tone for the response.
2. Writing Style: Ask the AI to mimic a specific author’s writing style.
3. Contradiction: Ask the AI to provide a response that contradicts the previous statement or take a stance opposite to its prior response.
4. Ambiguity: Instruct the AI to create responses with intentional ambiguity or double meanings.
5. Humor or Satire: Request that the response be humorous or satirical, requiring the AI to generate jokes or witty remarks.

Table 16: The prompt template for constructing the Style Soft Constraint (Jiang et al., 2023b).

You are a helpful assistant who reviews a debate between two other assistants in evaluating the quality of the outputs for a given instruction.The two assistants, Assistant (a) and Assistant (b), are given an instruction. Output (a) and Output (b) are generated by two different AI chatbots respectively. Assistant (a) and Assistant (b) have conflicting evaluations. Your goal is to review their evaluations and give your final decision on which output is better. Here are some rules of the evaluation:
(1) You should prioritize evaluating whether the output honestly/precisely/closely executes the instruction, then consider its helpfulness, accuracy, level of detail, harmlessness, etc.
(2) Outputs should NOT contain more/less than what the instruction asks for, as such outputs do NOT precisely execute the instruction.
(3) You should avoid any potential bias and your judgment should be as objective as possible. For example, the order in which the outputs were presented should NOT affect your judgment, as Output (a) and Output (b) are **equally likely** to be the better.
Output your final verdict by strictly following this format: "[[A]]" if Output (a) is better, "[[B]]" if Output (b) is better, and "[[C]]" for a tie.
/ Given instruction */*
{question}
/ The Start of Output (a) */*
{answer of assistant a}
/ The End of Output (a) */*
/ The Start of Output (b) */*
{answer of assistant b}
/ The End of Output (b) */*

Table 17: The prompt template for Judger to reorder the responses (Zheng et al., 2023)

.