From Complex to Simple: Enhancing Multi-Constraint Complex Instruction Following Ability of Large Language Models

Qianyu He[1]*, Jie Zeng[1]*, Qianxi He[1], Jiaqing Liang[2]†, Yanghua Xiao[1]†
[1]Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
[2]School of Data Science, Fudan University
{qyhe21, jzeng23, qxhe23}@m.fudan.edu.cn, {liangjiaqing, shawyh}@fudan.edu.cn

Abstract

It is imperative for Large language models (LLMs) to follow instructions with elaborate requirements (i.e. *Complex Instructions Following*). Yet, it remains under-explored how to enhance the ability of LLMs to follow complex instructions with multiple constraints. To bridge the gap, we initially study *what training data is effective* in enhancing complex constraints following abilities. We found that training LLMs with instructions containing multiple constraints enhances their understanding of complex instructions, especially those with lower complexity levels. Additionally, we further propose methods addressing how to *obtain* and *utilize* the effective training data. Finally, we conduct extensive experiments to prove the effectiveness of our methods in terms of overall performance and training efficiency. We also demonstrate that our methods improve models' ability to follow instructions generally and generalize effectively across out-of-domain, in-domain, and adversarial settings, while maintaining general capabilities. The datasets and code are publicly available at https://github.com/meowpass/FollowComplexInstruction.

1 Introduction

Large language models (LLMs) have become the backbone for real-world applications (Anil et al., 2023; Touvron et al., 2023; Achiam et al., 2023). Given natural language instructions, LLMs can solve unseen tasks with few or no examples (Brown et al., 2020). The capability of LLMs to accurately understand instructions and convey the desired output, known as *Instruction Following* (Lou et al., 2024), is crucial for the safety (Mu et al., 2023) and reliability (Zhou et al., 2023a) of LLMs.

It is imperative for LLMs to follow instructions with elaborate requirements (Yin et al., 2023; Xu et al., 2023) (i.e. *Complex Instructions*), such as formatting specifications outlined in Fig. 1. On one hand, the ability to follow detailed instructions alleviates the need for annotating samples, which can be costly and challenging for intricate tasks (Zeng et al., 2023a). On the other hand, complex instructions hardly appear in the training data (Zhou et al., 2024). Hence, the ability to follow complex instructions demonstrates models to have better generalization ability to unseen tasks (Yin et al., 2023).

Specifically, satisfying the multiple constraints in the instructions simultaneously (i.e. *Constraints Following*) poses a significant challenge in complex instruction following (Jiang et al., 2023b; He et al., 2024). As shown in Fig. 1, whether models

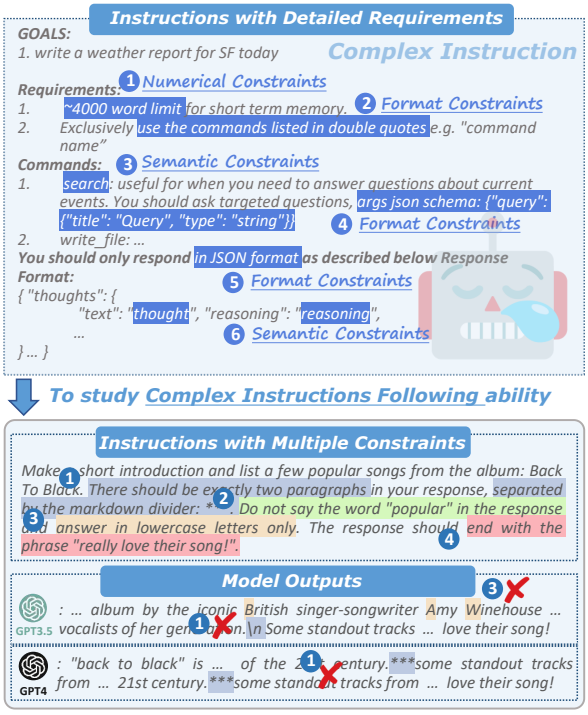

Figure 1: Real-world applications generally involve instructions with multiple constraints (i.e. *Complex Instructions*), posing challenges for models.

* Equal contribution.
† Corresponding author.

Findings of the Association for Computational Linguistics: EACL 2024, pages 10864–10882

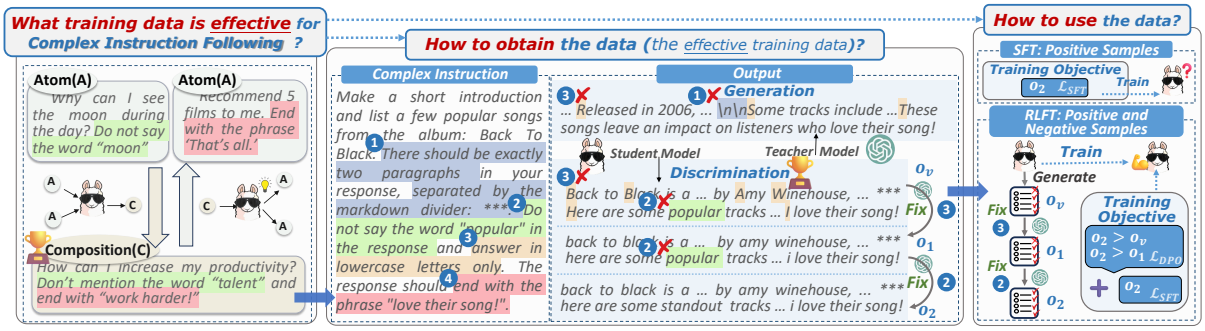

Figure 2: The framework of our study. We first study *what training data is effective* in enhancing complex instruction following abilities via an empirical study. Then, we design a discrimination-based method to address how to *obtain* the data. Finally, we propose a method for effectively *utilizing* positive and negative samples obtained.

can satisfy the multiple constraints in the instructions determines their ability to follow complex instructions. Hence, in our work, we explore complex instruction following by examining LLMs' ability to follow instructions with multiple constraints (Yin et al., 2023; Lou et al., 2024). On one hand, human instructions are subjective and ambiguous, while constraints within these instructions facilitate the automatic evaluation of instruction following ability (Zhou et al., 2023a; Wang et al., 2024). On the other hand, the compositional nature of constraints enables the automatic creation of instructions with unseen compositions of constraints (Zhou et al., 2023b; Yao et al., 2023). These instructions hardly appear in the training data, thus effectively assessing the model's ability to generalize to unseen tasks (Aksu et al., 2023).

Complex constraints following is a challenging task for LLMs (Jiang et al., 2023b; Qin et al., 2024). As shown in Fig. 1, even advanced LLMs struggle to meet the four specified constraints in complex instructions. However, it remains under-explored *how to enhance LLMs to follow multi-constraint complex instructions*. First, the existing works on constraints following mainly focus on *evaluation* without proposing methods for enhancement (Chen et al., 2024; Xia et al., 2024). Additionally, even when the improvement methods are proposed, they mainly consider instructions with *few* constraints, thereby failing to showcase the complexity of human instructions in practical applications (Chen et al., 2022; Zhang et al., 2023; Wang et al., 2024). Moreover, although some studies construct complex instructions with multiple constraints and fine-tune LLMs on them (Aksu et al., 2023; Sun et al., 2024), one key research question remains under-explored: **What *training data* is effective in enhancing complex constraint-following abilities?** This leads to two follow-up questions: (1) **How to *obtain* the effective training data?** and (2) **How to *utilize* the data effectively?**

In this work, we systematically study how to enhance the ability of LLMs to follow complex instructions, with the framework shown in Fig. 2. We initially explore the effective training data through an empirical study. We found that training LLMs on instructions containing multiple constraints (*compositional data*) enhances their understanding of complex instructions more effectively than training on atomic constraints (*atomic data*). Moreover, the improvement in performance is related to the number of constraints, the model size.

To obtain high-quality compositional data, we generate initial output via a student model (vanilla model) and then correct it via a teacher model (advanced model), termed the Discrimination method. This approach yields higher-quality output than using the teacher model to generate directly. To leverage the positive and negative samples collected during the Discrimination method, we introduce a contrastive method with reinforcement learning fine-tuning (RLFT) (Rafailov et al., 2023). Our method surpasses the SFT training paradigm on the instruction following benchmark (Zhou et al., 2023a) with fewer training tokens. Overall, our methods enhance models' ability to follow instructions generally. Our methods also generalize well across out-of-domain, in-domain, and adversarial settings while maintaining general capabilities.

Overall, our contributions are mainly three-fold: (1) We systematically improve LLMs' instruction-following ability by exploring effective training data. (2) We design a discrimination-based method to obtain effective training data. We also propose a

method for utilizing positive and negative samples obtained through this approach. (3) We conduct extensive experiments to prove the effectiveness and efficiency of our method. We also validate its generalization ability under five settings.

2 Related Work

2.1 Instruction Following

There are various ways to assess LLMs' ability to follow instructions. Some works study whether models understand the instructions by perturbing answer spaces (Zeng et al., 2023b; Li et al., 2023a; Wu et al., 2023). other works incorporates verifiable constraints (such as lexical, numerical and format) within instructions (Sun et al., 2023; Jiang et al., 2023b; Aksu et al., 2023; Zhou et al., 2023b; Yao et al., 2023). These constraints can be compositional, allowing one instruction to contain multiple constraints (Aksu et al., 2023; Zhou et al., 2023b; Yao et al., 2023). Such complex instructions pose greater challenges for LLMs to follow (He et al., 2024; Qin et al., 2024). Our work falls into this latter category. The existing works on constraints following either focus on evaluation (Chen et al., 2024; Xia et al., 2024) or consider instructions with few constraints (Zhang et al., 2023; Chen and Wan, 2023; Wang et al., 2024). Different from them, we systematically investigate how to enhance complex instructions with multiple constraints.

2.2 Complex Instruction Tuning

Complex Instructions can refer to instructions that involve more reasoning steps (Mukherjee et al., 2023), intricate input (Zhou et al., 2024), or multiple constraints (Luo et al., 2023a). Many studies have demonstrated that fine-tuning with complex instructions can boost performance in tasks such as instruction following (Xu et al., 2023), reasoning (Mitra et al., 2023), or code generation (Luo et al., 2023b). However, our work differs from these studies in two main aspects. First, we focus on improving LLMs' ability to follow complex constraints, which is crucial for the practicality and safety of LLMs (Zhou et al., 2023a; Mu et al., 2023). Furthermore, traditional supervised fine-tuning (SFT) uses only positive samples, whereas we use both positive and negative samples to enhance the complex instruction-following ability of LLMs effectively and efficiently.

3 Empirical Studies

A common approach to improve LLMs' ability to follow complex instructions is to construct corresponding instances and fine-tune the LLMs on them (Aksu et al., 2023; Sun et al., 2024). Yet, one key research question remains under-explored: ***What training data is effective*** **in enhancing complex constraint-following abilities?**

Two types of training data can be utilized: (1) Initially train models with atomic data, enabling them to handle compositional data automatically. (2) Train models with compositional data, enabling them to understand instructions with atomic or varying compositions of constraints spontaneously. Examples are shown in Fig. 2.

To compare these training data types, we split instructions from existing benchmarks on instruction following (Zhou et al., 2023a; Jiang et al., 2023b) into training and test sets. The training set includes atomic data (mostly with 1 constraint) and compositional data (mostly with over 3 constraints). Since original benchmarks lack corresponding outputs, we first generate them using GPT-3.5-turbo. To ensure quality, we further filter the training sets and retain only outputs that satisfy all instruction constraints by using GPT-3.5-turbo and predefined rules. The remaining data forms the test set[1].

We compare three methods[2]: (1) *Backbone*, the backbone model without further training. (2) *Atom* and (3) *Composition*, continue training the backbone model with atomic data and compositional data respectively. We leverage two backbone models (Zheng et al., 2024; Touvron et al., 2023) and adopt two accuracy metrics (Zhou et al., 2023a; Jiang et al., 2023b):

$$\text{acc}_{\text{ins}} = \frac{1}{m}\sum_{i=1}^{m}\prod_{j=1}^{n} c_i^j, \quad \text{acc}_{\text{con}} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} c_i^j,$$

where c_i^j equals 1 if the j-th constraint of the i-th instruction is satisfied, otherwise 0. Overall, achieving Instruction-level accuracy (acc_{ins}) is more challenging than Constraint-level accuracy (acc_{con}).

The performance of the three methods on the test sets is shown in Tab. 1 and Tab. 2. First, with regard

[1]Detailed data construction and statistics are provided in Appendix A.1.

[2]To prevent models from catastrophic forgetting (McCloskey and Cohen, 1989), we mix training data with 10k ShareGPT data (Chiang et al., 2023) for *Atom* and *Composition* checkpoint.

Backbone	Methods	Level 1	Level 2	Level 3	Level 4	Level 5	Avg.
Vicuna-7B-V1.5 (Zheng et al., 2024)	Backbone	39.07	44.71	37.28	**30.93**	**19.06**	34.21
	Atom	39.17	39.50	42.07	30.23	16.97	33.59
	Comp	**39.44**	**55.90**	**47.49**	22.27	16.65	**36.35**
LLaMA2-13B-Chat (Touvron et al., 2023)	Backbone	33.10	41.71	42.26	**23.89**	22.07	32.61
	Atom	**38.99**	39.78	36.61	20.74	14.83	30.19
	Comp	37.02	**44.66**	**42.55**	21.62	**22.36**	**33.64**

Table 1: The Instruction-level accuracy of models without further training (Backbone), training with atomic data (Atom), and compositional data (Comp) on FollowBench. Level x indicates there are x constraints in the instructions. Avg. indicates the average performance across 5 levels. The results are evaluated by GPT-4 using the FollowBench prompt template. The **bold** and underlined represent the first and second rankings.

Backbone	Methods	ChangeCase	Combination	Content	Format	Keywords	Language	Length	Punctuation	Startend	I-level	C-level
Vicuna-7B-V1.5	Backbone	27.87	15.91	**74.07**	44.09	**48.57**	**80.00**	30.69	10.71	40.00	26.89	37.47
	Atom	29.50	31.82	48.14	**63.44**	36.19	25.00	31.68	**16.07**	40.00	27.17	37.29
	Comp	**37.70**	**50.00**	40.74	55.91	36.19	25.00	**32.67**	14.29	**50.00**	**28.85**	**38.76**
LLaMA2-13B-Chat	Backbone	**42.62**	**11.36**	**81.48**	**55.91**	**45.71**	15.00	32.67	00.00	25.00	25.77	36.38
	Atom	**42.62**	00.00	37.04	54.84	42.86	35.00	34.65	12.50	37.50	**26.33**	35.83
	Comp	40.98	02.27	66.67	54.84	38.10	**50.00**	**36.63**	**16.07**	**40.00**	26.05	**37.84**

Table 2: The performance of models without further training (Backbone), training with atomic data (Atom), and compositional data (Comp) on IFEval. The I-level and C-level denote the Instruction and Constraint-level accuracy.

to the overall performance, training with compositional data generally surpasses both the backbone model and atomic data training. This demonstrates that **training with *compositional data* can generally enhance models' ability to follow complex instructions.** Surprisingly, according to Tab. 1, training with atomic data (mostly with 1 constraint) generally performs worse than the backbone model for instructions with more than 1 constraint. Also, training with compositional data (usually 3 to 5 constraints) boosts performance on instructions with 1 to 3 constraints significantly but shows less enhancement or even a decline for those with 4 to 5 constraints. This suggests that training with compositional data can better generalize to *lower-level* complex instructions (instructions with fewer constraints). Moreover, this effect is more pronounced in smaller LLMs (7B), likely due to their weaker generalization ability (Fu et al., 2023).

We have found that training with compositional data can better enhance LLM's ability to follow complex instructions compared with atomic data. A follow-up research question is **how to *obtain* high-quality compositional data?** Existing datasets either only provide compositional instructions without output (Zhou et al., 2023a; Jiang et al., 2023b) or directly generate responses using advanced LLMs and refine them manually (Sun et al., 2024).

We compare the outputs generated by three methods: (1) *Vanilla*: Output generated directly using backbone model. (2) *Generation*: Output generated directly using GPT-3.5-turbo. (3) *Discrimination*: First, we identify the constraints that Vanilla outputs failed to adhere to using test scripts (Zhou et al., 2023a). Then, we rectify the Vanilla outputs constraints by constraints using GPT-3.5-turbo (The framework is shown in Fig. 2 and please refer to §4.2 for details). With regard to the complex instructions, the instructions in IFEval (Zhou et al., 2023a) originally have only 1 to 3 constraints, which are not complex enough. We construct 1467 complex instructions, each comprising 3 to 5 constraints that can be automatically verified (Please refer to §4.1 for details). We leverage LLaMA2-13B-chat (Touvron et al., 2023) as the backbone and assess output quality using the test script from Zhou et al. (2023a).

As shown in Tab. 3, using the generation method, outputs from advanced LLMs (Generation) are of higher quality than those from weaker LLMs (Vanilla). However, **the outputs from weaker LLMs then refined by advanced LLMs (*Discrimination*) are of better quality than the outputs generated by advanced LLMs directly (*Generation*)**. This is because slight changes in the instruction (i.e. constraint) can cause substantial output differences, which the discrimination method captures better than the generation method.

4 Method

In §3, we propose a discrimination-based method to *obtain* effective training data. A subsequent question is **how to effectively *utilize* the obtained**

Methods	ChangeCase	Combination	Content	Format	Keywords	Language	Length	Punctuation	Startend	I-level	C-level
Vanilla	21.19	08.89	**77.26**	56.67	61.60	10.60	30.85	00.26	16.84	06.40	41.33
Generation	56.50	**30.37**	68.95	**74.96**	72.29	33.01	**52.91**	36.76	79.51	21.53	62.68
Discrimination	**66.56**	25.00	68.11	68.27	**77.32**	**81.95**	52.27	**70.90**	**85.60**	**35.04**	**68.30**

Table 3: The output quality evaluated by IFEval across different methods.

data? To address this, we introduce a method that leverages both positive and negative samples to improve complex instruction following. The framework is shown in Fig. 2.

4.1 Complex Instruction Synthesis

According to §3, the effective training data is complex instructions with multiple constraints (compositional data). To obtain compositional data, we first collect seed instructions from three widely used instruction-tuning datasets. Then, we rewrite the instructions to incorporate multiple constraints.

To ensure the *coverage* and *diversity* of the seed instructions, we consider three sources: (1) *Open Assistant* (Köpf et al., 2024): human-written instructions when interacting with chatbots. We only consider rank 0 instructions (annotated by humans as the highest quality) and the first turn of the conversation (Li et al., 2023b). (2) *Self-Instruct* (Wang et al., 2022a): 175 manually crafted instructions covering various topics to aid instruction generation for new tasks. (3) *Super-Natural* (Wang et al., 2022b): NLP tasks formatted with human instructions. Tasks with simple outputs (e.g., classification, tagging) are excluded, leaving 318 tasks. One instruction is randomly selected from each remaining task. From these sources, we gather a total of 1467 seed instructions.

Subsequently, we integrate constraints into these seed instructions. Initially, we randomly sample 3 to 5 constraints and use provided scripts to remove any conflicting constraints among those provided by Zhou et al. (2023a). Next, semantically equivalent but textually distinct instructions can substantially affect model outcomes (Yan et al., 2024; Chen et al., 2024). Hence, we employ eight diverse expressions to describe each type of constraint. Specifically, we manually select three common descriptions from the test set as seed descriptions, generate five similar descriptions using GPT-3.5-turbo, and refine them manually. For each sampled constraint c, we randomly select one description d_i from the description pool and append it to the instructions, formulated as:

$$I_c = \text{LLM}(I_s \oplus d_i \oplus ... \oplus d_n),$$

where I_s, I_c and d_i denote the seed instruction, its corresponding synthesized complex instruction, and appended constraint using a specific description. The number of constraints n ranges from 1 to 5, with a majority falling between 3 to 5[3].

4.2 Teacher Correction

In §3, we propose a discrimination-based approach for obtaining the output, shown to be more effective than directly generating output with advanced LLMs. The details of this approach are as follows.

Initially, we utilize LLaMA2-13B-Chat (Touvron et al., 2023) (student model) to generate results for our synthesized complex instructions. Then, we utilize the test scripts from Zhou et al. (2023a) to identify the constraints the model failed to follow since the constraints are objective and automatically verifiable. Finally, we adopt advanced LLMs (teacher model) GPT-3.5-turbo to correct the failed constraints one by one.

Specifically, each complex instruction I_c contains multiple constraints. In §4.2, we utilize the test script to pinpoint the f constraints $\mathcal{C} = \{c_1, c_2, ..., c_f\}$ that the student model's vanilla output o_v fails to follow. The teacher model sequentially corrects these failed constraints, yielding an output set $\mathcal{O} = \{o_v, o_1, o_2, ..., o_f\}$:

$$o_1 = \text{LLM}(o_v, c_1), \ldots, o_f = \text{LLM}(o_{f-1}, c_f),$$

where GPT-3.5-turbo is employed as the teacher model with prompts sourced from Tab. 16.

4.3 Contrastive Method

During §4.2, for each instruction I_c, we can gather positive sample set $\{o_f\}$ and negative samples set $\{o_v, o_1, ..., o_{f-1}\}$. Supervised fine-tuning (SFT) solely utilizes positive samples successfully meeting constraints specified in complex instructions (Radford et al., 2019; Howard and Ruder, 2018). However, negative samples from §4.2, failing to meet certain constraints, also offer valuable supervision signals. Hence, we leverage the positive and negative samples through reinforcement learning fine-tuning (Rafailov et al., 2023).

[3]The detailed statistics are shown in Tab. 11.

Specifically, given the output set $\mathcal{O} = \{o_v, o_1, o_2, ..., o_f\}$ for each complex instruction I_c, we form a training dataset $\mathcal{D}$ consisting of f contrastive triplets: $\mathcal{D} = \{(I_c, o_v, o_f), (I_c, o_1, o_f), ..., (I_c, o_{f-1}, o_f)\}$. In each triplet, the final corrected output o_f (positive sample) is preferred over o_i (negative sample) as o_f satisfies more constraints specified in the complex instruction I_c. To model this preference information, we apply Direct Preference Optimization (DPO) (Rafailov et al., 2023). The loss function involves a maximum likelihood objective for the language model parameters π_θ:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(I_c, o_f, o_i)\sim\mathcal{D}}[\log\sigma(\beta\log\frac{\pi_\theta(o_f|I_c)}{\pi_{\text{ref}}(o_f|I_c)} - \beta\log\frac{\pi_\theta(o_i|I_c)}{\pi_{\text{ref}}(o_i|I_c)})],$$

where the model parameter π_{ref} starts as π_θ and remains fixed during training. β is a hyperparameter, and σ denotes the sigmoid function. $\mathcal{L}_{\text{DPO}}$ maximizes the log probability of the preferred output o_f relative to the dispreferred output o_i.

However, solely relying on $\mathcal{L}_{\text{DPO}}$ may lead to low probabilities for both chosen and rejected outputs, yet with a significant disparity between them[4]. Therefore, we integrate the SFT loss $\mathcal{L}_{\text{SFT}}$ to constrain π_θ from deviating from the preferred data distribution (Xu et al., 2024; Hejna et al., 2023):

$$\mathcal{L}_{\text{SFT}}(\pi_\theta) = -\mathbb{E}_{(I_c, o_f)\sim\mathcal{D}}[\log \pi_\theta(o_f|I_c)].$$

Finally, our training procedure is to optimize $\mathcal{L}_{\text{DPO}}$ and $\mathcal{L}_{\text{SFT}}$ jointly:

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

5 Experiments

We conduct experiments to verify the effectiveness of our method, focusing on overall performance, training efficiency, and generalization ability.

5.1 Experiment Setup

Models. Our baselines include popular open and closed-source LLMs. We especially select models that excel in complex instruction following (Xu et al., 2023) and those that perform well on current instruction following benchmarks (Wang et al., 2023). Within our framework (§4.1), we compare three methods: (1) **Ours$_{\text{Generation}}$** directly generates output with GPT-3.5-turbo and trains the backbone model via supervised fine-tuning (SFT). (2) **Ours$_{\text{Discrimination}}$** generates output via the backbone model then refines with GPT-3.5-turbo (§4.2), and trains the backbone model via SFT. (3) **Ours$_{\text{Contrastive}}$** utilizes our advanced DPO for training (§4.3). Across all methods, the instructions in the training data are identical, differing only in output and training paradigms[5].

Evaluation. We evaluate all models on IFEval (Zhou et al., 2023a), a widely-used instruction-following benchmark. The test set consists of 541 samples, each containing 1 to 3 constraints. The metrics are the same as §3.

5.2 Results

Overall Performance. As shown in Tab. 4, for the same backbone, Ours$_{\text{Discrimination}}$ consistently outperforms Ours$_{\text{Generation}}$, and Ours$_{\text{Contrastive}}$ outperforms Ours$_{\text{Discrimination}}$, which proves the effectiveness of our methods. Next, using the same backbone model (LLaMA2), Ours-LLaMA2-13B$_{\text{Generation}}$ performs worse than many open-source models, even when the constraints in the test set have been seen during training. This highlights the importance in obtaining high-quality output for complex instructions.

Training Efficiency. To ensure fairness, we convert the checkpoints with the same number of training steps into the number of training tokens for the x-axis. As shown in Fig. 3 (top), Ours-LLaMA2-13B$_{\text{Contrastive}}$ achieves better performance with the same training tokens and ultimately outperforms Ours-LLaMA2-13B$_{\text{Discrimination}}$.

5.3 Analysis

5.3.1 Data Quality

Many studies have shown that data quality outweighs data quantity (Zhou et al., 2024). As shown in Tab. 3, our training data contains noise. Hence, we explore the performance using a subset of training data that has higher quality but lower quantity. Specifically, we first filter the full set of noisy training data (containing 1467 samples) to retain only those samples whose output satisfies all the

[4]We provide some cases in Appx. A.6.

[5]Continuous training may lead to catastrophic forgetting (McCloskey and Cohen, 1989). Hence, we employ a replay strategy mixing training data with 10k ShareGPT data (Chiang et al., 2023) to maintain general abilities during training.

Models	BaseModel	ChangeCase	Combination	Content	Format	Keywords	Language	Length	Punctuation	Startend	I-level	C-level
PaLM2-S* (Anil et al., 2023)	PaLM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	43.07	55.76
GPT3.5-turbo	GPT	58.43	70.77	88.68	88.54	71.17	98.35	53.85	18.18	76.12	58.96	68.47
GPT4 (Achiam et al., 2023)	GPT	75.28	70.77	96.23	94.27	84.05	96.77	73.43	66.67	95.52	76.16	82.97
ChatGLM3-6B (Du et al., 2021)	ChatGLM	14.61	16.92	67.92	42.68	50.92	51.61	34.97	28.79	49.25	27.36	39.33
Qwen-14B-Chat (Bai et al., 2023)	Qwen	57.30	23.08	75.47	57.96	58.28	83.87	33.57	21.21	68.66	37.89	51.08
LLaMA2-7B-Chat (Touvron et al., 2023)	LLaMA2	35.96	06.15	79.25	57.96	53.37	19.35	36.36	07.58	41.79	28.84	41.61
LLaMA2-13B-Chat (Touvron et al., 2023)	LLaMA2	37.08	07.69	83.02	60.51	57.06	25.81	37.76	00.00	29.85	29.94	42.21
LLaMA2-70B-Chat (Touvron et al., 2023)	LLaMA2	42.70	24.62	79.25	63.69	68.71	16.13	39.86	12.12	62.69	38.45	50.36
Vicuna-13B-V1.5 (Zheng et al., 2024)	LLaMA2	56.18	32.31	75.47	62.42	57.06	93.55	42.66	16.67	64.18	42.33	53.48
WizardLM-13B-V1.2 (Xu et al., 2023)	LLaMA2	49.44	16.92	75.47	67.52	66.26	83.87	46.85	15.15	64.18	43.07	54.56
OpenChat-13B-V3.2 (Wang et al., 2023)	LLaMA2	49.44	26.15	**88.68**	68.15	66.26	87.10	47.55	19.70	71.64	46.03	57.43
Mistral-7B-Instruct-v0.2 (Jiang et al., 2023a)	Mistral	61.80	21.54	**88.68**	75.16	**76.07**	58.06	50.35	16.67	74.63	51.02	61.03
Ours-LLaMA2-7B$_{\text{Generation}}$	LLaMA2	41.57	15.38	71.70	70.70	53.37	58.06	27.97	9.09	56.72	34.01	46.16
Ours-LLaMA2-7B$_{\text{Discrimination}}$	LLaMA2	49.44	06.15	77.36	64.97	53.99	74.19	34.27	07.58	73.13	38.82	48.56
Ours-LLaMA2-13B$_{\text{Generation}}$	LLaMA2	64.04	20.00	66.04	70.06	53.99	35.48	44.06	21.21	74.63	41.22	52.88
Ours-LLaMA2-7B$_{\text{Contrastive}}$	LLaMA2	76.40	13.85	75.47	70.06	50.92	67.74	37.76	24.24	82.09	42.88	54.68
Ours-LLaMA2-13B$_{\text{Discrimination}}$	LLaMA2	60.67	06.15	79.25	64.97	60.12	**96.77**	43.36	51.52	79.10	46.21	57.43
Ours-LLaMA2-13B$_{\text{Contrastive}}$	LLaMA2	65.17	10.77	84.91	66.88	60.74	93.55	47.55	43.94	86.57	48.24	59.71
Ours-Mistral-7B$_{\text{Generation}}$	Mistral	73.03	**47.69**	66.04	78.34	57.06	58.06	**51.05**	39.39	**89.55**	52.13	62.83
Ours-Mistral-7B$_{\text{Discrimination}}$	Mistral	**79.78**	16.92	71.70	80.89	61.35	93.55	44.76	**59.09**	85.07	53.79	64.27
Ours-Mistral-7B$_{\text{Contrastive}}$	Mistral	68.54	30.77	84.91	**85.35**	68.71	80.65	46.15	30.30	88.06	**53.97**	**64.99**

Table 4: The overall performance on IFEval (each with 1 to 3 constraints). The asterisk (*) indicates that the results are directly sourced from IFEval. N/A denotes that IFEval does not provide the results for specific constraints.

Models	BaseModel	ChangeCase	Combination	Content	Format	Keywords	Language	Length	Punctuation	Startend	I-level	C-level
Ours-LLaMA2-13B$_{\text{Discrimination-Random}}$	LLaMA2	58.43	06.15	73.58	63.69	50.92	87.10	33.57	10.61	77.61	39.00	49.40
Ours-LLaMA2-13B$_{\text{Discrimination-Selected}}$	LLaMA2	**71.91**	**15.38**	75.47	**75.16**	**61.96**	83.87	37.06	16.67	74.63	44.55	56.71
Ours-LLaMA2-13B$_{\text{Discrimination-All}}$	LLaMA2	60.67	06.15	**79.25**	64.97	60.12	**96.77**	**43.36**	**51.52**	**79.10**	**46.21**	**57.43**
Ours-LLaMA2-13B$_{\text{Contrastive-Random}}$	LLaMA2	46.07	10.77	81.13	68.79	**64.42**	83.87	39.86	**50.00**	79.10	44.55	56.71
Ours-LLaMA2-13B$_{\text{Contrastive-All}}$	LLaMA2	65.17	10.77	**84.91**	66.88	60.74	**93.55**	**47.55**	43.94	**86.57**	48.24	59.71
Ours-LLaMA2-13B$_{\text{Contrastive-Selected}}$	LLaMA2	**75.28**	**15.38**	79.25	**77.71**	58.90	74.19	37.76	45.45	85.07	**48.61**	**60.07**

Table 5: The performance using the full set of noisy data samples (All), a subset of high-quality data samples (Selected), and a randomly sampled subset of the same size as the selected high-quality data subset (Random).

constraints in the instructions and finally obtain 515 high-quality data samples. We also set a control group by randomly sampling a subset containing the same number of data samples as the selected high-quality data subset[6]. We finally train the LLaMA2-13B-Chat with these sampled data. As shown in Tab. 5, training with full set of noisy data performs better than training with the subset of noisy data. Also, the selected high-quality data achieves comparable performance to training with the full set of noisy data. This underscores the importance of selecting high-quality training data.

5.3.2 Constraints Type

We study the influence of constraint types on the effectiveness of the proposed methods. First, to ensure the consistency of the model's output, we report the performance across different constraint types at the checkpoints taking different training steps. As shown in Fig. 3 (bottom), certain constraints (e.g., Combination) are consistently more challenging to follow, while others (e.g., Startend) are easier. Hence, when synthesizing training data, certain challenging constraint types introduce more noise to our training data. As shown in Tab. 3, the quality of training data differs across various constraints, leading to variations in the effectiveness of our methods across different types of constraints.

Models	LIMA	Koala	AlpacaEval	Avg.
LLaMA2-13B-Chat	7.781	7.619	7.542	7.647
Ours-LLaMA2-13B$_{\text{Generation}}$	8.475	8.000	8.138	8.204
Ours-LLaMA2-13B$_{\text{Contrastive}}$	8.434	8.091	8.110	8.212
Ours-LLaMA2-13B$_{\text{Discrimination}}$	**8.552**	**8.097**	**8.204**	**8.284**

Table 6: The performance of models on general instruction following datasets.

5.4 Generalization Experiments

We investigate the generalizability of our framework from five perspectives.

5.4.1 General Instruction Following Ability

We evaluate models on three general instruction-following benchmarks, LIMA (Zhou et al., 2024), Koala (Geng et al., 2023) and AlpacaEval (Taori et al., 2023)[7]. They contain more general and diverse instructions than our training data. As shown in Tab. 6, compared to the backbone model, train-

[6]The statistics of the sampled data are shown in Tab. 11.

[7]The design of the scoring prompt is in Appx. A.3.1

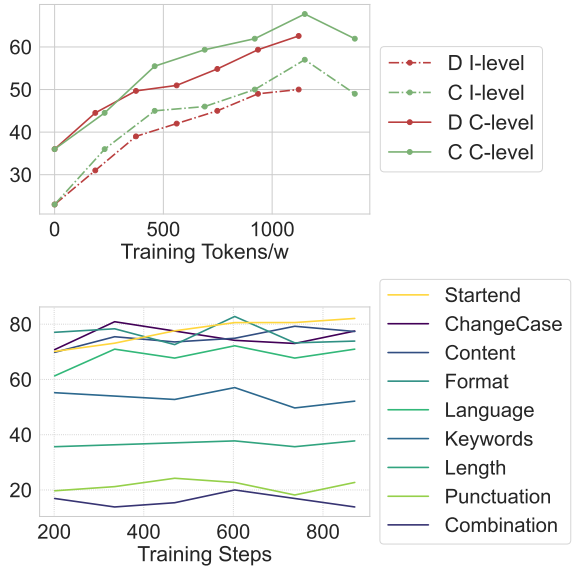

Figure 3: The performance of training efficiency (top) and each constraint type across different training tokens and training steps (bottom). D and C denote Ours-LLaMA2-13B$_{\text{Discrimination}}$ and Ours-LLaMA2-13B$_{\text{Contrastive}}$ respectively.

Models	Followbench		CELLO
	Mixed	Total	Total
LLaMA2-13B-Chat	25.88	42.04	40.20
Ours-LLaMA2-13B$_{\text{Contrastive}}$	36.47	**42.77**	44.20
Ours-LLaMA2-13B$_{\text{Discrimination}}$	**38.82**	42.43	**56.10**

Table 7: The performance of models on Followbench and CELLO.

ing with complex instructions improve models' ability to follow instructions generally.

5.4.2 Out-of-Domain Generalization

We evaluate models on another two complex instruction-following benchmarks FollowBench (Jiang et al., 2023b) and CELLO (He et al., 2024). They have the following features to outline: (1) They contain almost entirely *different* constraints from IFEval[8]. (2) To mirror real-world scenarios, FollowBench specifically introduces a *Mixed* Category. Instructions within this category encompass multiple constraints of different types. (3) CELLO is a *Chinese* complex instruction following benchmark mirroring real-world scenarios. The instructions are in a different language from our training data. As shown in Tab. 7, our methods significantly enhance the ability of LLMs to follow different types of constraints, even when applied to different languages. Interestingly, Ours$_{\text{Contrastive}}$ underperforms Ours$_{\text{Discrimination}}$ in some cases when applied to out-of-domain constraints, possibly due to DPO exhibiting lower generalization ability to out-of-preference data (Li et al., 2023c).

[8]The examples from these benchmarks are in Appx. A.3.2

Models	In-Domain		Adversarial	
	I-level	C-level	I-level	C-level
LLaMA2-13B-Chat	09.50	42.27	01.00	40.15
WizardLM-13B-V1.2	14.00	47.20	07.00	46.60
OpenChat-13B-V3.2	16.50	49.07	07.30	47.64
Ours-LLaMA2-13B$_{\text{Generation}}$	14.00	52.27	05.00	49.36
Ours-LLaMA2-13B$_{\text{Discrimination}}$	15.00	53.33	05.00	49.53
Ours-LLaMA2-13B$_{\text{Contrastive}}$	**19.00**	**55.73**	**07.50**	**53.05**

Table 8: The performance of models on instructions with varying phrasing (In-Domain) and on more challenging complex instructions (Adversarial).

Models	ARC (25-shot)	HellaSwag (10-shot)	MMLU (5-shot)	TruthfulQA (0-shot)	Avg.
LLaMA2-13B-Chat	59.04	81.94	54.64	44.12	59.94
WizardLM-13B-V1.2	59.04	82.21	54.64	47.27	60.79
OpenChat-13B-V3.2	59.64	82.68	56.68	44.49	60.87
Ours-LLaMA2-13B$_{\text{Discrimination}}$	56.74	78.39	53.01	48.17	59.08
Ours-LLaMA2-13B$_{\text{Contrastive}}$	57.76	79.95	53.79	48.15	59.91

Table 9: The performance of models on general tasks.

5.4.3 In-Domain Generalization

We evaluate models on 200 new instructions, the constraints of which fall into the same categories as the training data but have different wording and specific requirements[9]. As shown in Tab. 8 (In-Domain), Ours$_{\text{Contrastive}}$ remains the top performer. Also, the performance gap between Ours$_{\text{Contrastive}}$ and the best open-source model (OpenChat-13B-V3.2) has increased.

5.4.4 Adversarial Setting

We stress test the models on 200 more challenging complex instructions with increased constraints. The instructions in the test set contain 6 to 7 constraints while our training data contains 3 to 5 constraints[10]. As shown in Tab. 8 (Adversarial), Ours$_{\text{Contrastive}}$ outperforms all other models and significantly performs better than Ours$_{\text{Discrimination}}$.

5.4.5 General Ability

We evaluate models on four widely-used benchmarks, reflecting knowledge capability (MMLU (Hendrycks et al., 2020), TruthfulQA (Lin et al., 2021), ARC (Clark et al., 2018)), complex reasoning (HellaSwag (Zellers et al., 2019)). As shown in Tab. 9, our methods perform

[9]The construction process is detailed in the Appx. A.3.3

[10]The construction process is detailed in the Appx. A.3.4

on par with open-source LLMs, validating that our methods maintain the models' general ability.

6 Conclusion

In this paper, we systematically study how to enhance the ability of LLMs to follow complex instructions. Initially, we study effective training data and methods for obtaining high-quality data through empirical studies. Based on our findings, we introduce a method utilizing positive and negative samples to enhance LLMs' complex instruction-following ability. Our experiments show that our methods enhance models' ability to follow complex instructions more effectively and efficiently. Finally, extensive experiments demonstrate the generalization abilities of our framework.

Acknowledgments

This work is supported by National Natural Science Foundation of China (No.62102095), Science and Technology Commission of Shanghai Municipality Grant (No. 22511105902), Shanghai Municipal Science and Technology Major Project (No.2021SHZDZX0103). Yanghua Xiao is also a member of Research Group of Computational and AI Communication at Institute for Global Communications and Integrated Media, Fudan University. The computations in this research were performed using the CFFF platform of Fudan University.

Limitations

We analyze the limitations of our work as follows. First, we investigate complex instruction-following by testing LLMs' ability to adhere to instructions with multiple constraints. Even if the model meets all the constraints simultaneously, it may not fully follow complex instructions due to reasoning or knowledge limitations. However, we see complex constraint-following as a significant challenge worth studying. In constructing the training data, we primarily use hard constraints from IFEval, although real-world scenarios often include soft constraints like semantic constraints. We focus on hard constraints because they can be objectively and automatically evaluated, and we believe that experiments based on them can yield valuable insights into complex instruction-following.

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*.

Taha Aksu, Devamanyu Hazarika, Shikib Mehri, Seokhwan Kim, Dilek Hakkani-Tur, Yang Liu, and Mahdi Namazifar. 2023. Cesar: Automatic induction of compositional instructions for multi-turn dialogs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11709–11737.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Howard Chen, Huihan Li, Danqi Chen, and Karthik Narasimhan. 2022. Controllable text generation with language constraints. *arXiv preprint arXiv:2212.10466*.

Xiang Chen and Xiaojun Wan. 2023. A comprehensive evaluation of constrained text generation for large language models. *arXiv preprint arXiv:2310.16343*.

Yihan Chen, Benfeng Xu, Quan Wang, Yi Liu, and Zhendong Mao. 2024. Benchmarking large language models on controllable generation under diversified instructions. *arXiv preprint arXiv:2401.00690*.

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.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2021. Glm: General language model pretraining with autoregressive blank infilling. *arXiv preprint arXiv:2103.10360*.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pages 10421–10430. PMLR.

Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. Koala: A dialogue model for academic research. Blog post.

Qianyu He, Jie Zeng, Wenhao Huang, Lina Chen, Jin Xiao, Qianxi He, Xunzhe Zhou, Jiaqing Liang, and Yanghua Xiao. 2024. Can large language models understand real-world complex instructions? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18188–18196.

Joey Hejna, Rafael Rafailov, Harshit Sikchi, Chelsea Finn, Scott Niekum, W Bradley Knox, and Dorsa Sadigh. 2023. Contrastive prefence learning: Learning from human feedback without rl. *arXiv preprint arXiv:2310.13639*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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*.

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.

Shiyang Li, Jun Yan, Hai Wang, Zheng Tang, Xiang Ren, Vijay Srinivasan, and Hongxia Jin. 2023a. Instruction-following evaluation through verbalizer manipulation. *arXiv preprint arXiv:2307.10558*.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, Omer Levy, Jason Weston, and Mike Lewis. 2023b. Self-alignment with instruction backtranslation. *arXiv preprint arXiv:2308.06259*.

Ziniu Li, Tian Xu, and Yang Yu. 2023c. Policy optimization in rlhf: The impact of out-of-preference data. *arXiv preprint arXiv:2312.10584*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.

Renze Lou, Kai Zhang, and Wenpeng Yin. 2024. A comprehensive survey on instruction following. *Preprint*, arXiv:2303.10475.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023a. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023b. Wizardcoder: Empowering code large language models with evol-instruct. *Preprint*, arXiv:2306.08568.

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.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, et al. 2023. Orca 2: Teaching small language models how to reason. *arXiv preprint arXiv:2311.11045*.

Norman Mu, Sarah Chen, Zifan Wang, Sizhe Chen, David Karamardian, Lulwa Aljeraisy, Dan Hendrycks, and David Wagner. 2023. Can llms follow simple rules? *arXiv preprint arXiv:2311.04235*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

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*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

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*.

Jiao Sun, Yufei Tian, Wangchunshu Zhou, Nan Xu, Qian Hu, Rahul Gupta, John Wieting, Nanyun Peng, and Xuezhe Ma. 2023. Evaluating large language models on controlled generation tasks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3155–3168.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Fei Wang, Chao Shang, Sarthak Jain, Shuai Wang, Qiang Ning, Bonan Min, Vittorio Castelli, Yassine Benajiba, and Dan Roth. 2024. From instructions to constraints: Language model alignment with automatic constraint verification. *arXiv preprint arXiv:2403.06326*.

Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. 2023. Openchat: Advancing open-source language models with mixed-quality data. *arXiv preprint arXiv:2309.11235*.

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, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddhartha Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hannaneh Hajishirzi, and Daniel Khashabi. 2022b. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *Preprint*, arXiv:2204.07705.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2023. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. *arXiv preprint arXiv:2307.02477*.

Congying Xia, Chen Xing, Jiangshu Du, Xinyi Yang, Yihao Feng, Ran Xu, Wenpeng Yin, and Caiming Xiong. 2024. Fofo: A benchmark to evaluate llms' format-following capability. *arXiv preprint arXiv:2402.18667*.

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*.

Tianyi Yan, Fei Wang, James Y Huang, Wenxuan Zhou, Fan Yin, Aram Galstyan, Wenpeng Yin, and Muhao Chen. 2024. Contrastive instruction tuning. *arXiv preprint arXiv:2402.11138*.

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

Wenpeng Yin, Qinyuan Ye, Pengfei Liu, Xiang Ren, and Hinrich Schütze. 2023. Llm-driven instruction following: Progresses and concerns. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, pages 19–25.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023a. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*.

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

Qingru Zhang, Chandan Singh, Liyuan Liu, Xiaodong Liu, Bin Yu, Jianfeng Gao, and Tuo Zhao. 2023. Tell your model where to attend: Post-hoc attention steering for llms. *arXiv preprint arXiv:2311.02262*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.

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.

Benchmark	Type	Training Set						Test Set					
		L1	L2	L3	L4	L5	Avg.	L1	L2	L3	L4	L5	Avg.
FollowBench	Example	31	20	17	16	16	100	9	20	23	24	24	100
	Content	16	15	17	15	12	75	9	10	8	10	13	50
	Situation	14	13	13	13	13	66	8	9	9	9	9	44
	Style	19	19	18	18	16	90	11	11	12	12	14	60
	Format	20	19	17	18	16	90	10	11	13	12	14	60
	Mixed	14	10	11	7	6	48	3	7	6	10	11	37
	Total	114	96	93	87	79	469	50	68	71	77	85	351
IFEval	One-cons	-	-	-	-	-	92	-	-	-	-	-	213
	Multi-cons	-	-	-	-	-	92	-	-	-	-	-	144

Table 10: The statistic of the datasets constructed in the empirical study.

Data Selection Method	C_1	C_2	C_3	C_4	C_5	Total
All	61	54	431	493	428	1467
Select	60	48	192	136	79	515
Random	19	19	143	178	156	515

Table 11: The statistic of the data used in §5.3.1. C_i indicates that there are i constraints in the instruction.

A Appendix

A.1 Details of Empirical Studies

In §3, we first investigate what training data is effective in enhancing complex constraints following ability. To achieve this, we split the instructions in the existing instruction following benchmarks, i.e., Followbench (Jiang et al., 2023b) and IFEval (Zhou et al., 2023a) into the training and test sets. The training sets consist of two types of data: (1) Compositional data: From IFEval, we utilize all the instructions with more than one constraint and all level-4 and level-5 instructions from Followbench. (2) Atomic data: From IFEval, we use only one-constraint instructions. From Followbench, we use all level-1 and part of level-2 instructions to ensure an equal number of compositional and atomic data for fair comparison.

After collecting the instructions, we first employ GPT3.5-turbo to generate the answers to the corresponding instructions. To improve the quality of the training data, we filter the samples from Followbench by prompting GPT3.5-turbo (We use the evaluation prompt from the original paper) and those from IFEval via its provided test scripts.

The statistics of our training set and test set are provided in Tab. 10. It can be seen that there is a distribution shift between the training set and test set from FollowBench. This may be because we use outputs satisfying all instruction constraints judged by GPT-3.5-turbo for training, with the rest as the test set. Consequently, the test set can be more challenging than the training data, especially for instructions with more constraints (level 4, level 5). This can partially explain the results that training with compositional data boosts performance on instructions with 1 to 3 constraints but lowers it on those with 4 to 5 constraints.

Models	Format	Input	Task	Count	Average
LLaMA2-13B-Chat	64.00	34.20	28.00	67.40	40.20
Ours-LLaMA2-13B$_{\text{Contrastive}}$	54.90	44.00	41.30	60.70	44.20
Ours-LLaMA2-13B$_{\text{Discrimination}}$	62.70	49.00	55.00	71.30	56.10
GPT3.5-turbo	89.90	76.00	79.90	70.00	79.40
GPT-4	91.10	79.60	79.20	72.40	82.20

Table 12: The overall performance of models on CELLO. *Format*, *Task*, *Input*, *Count* denote the criteria *Answer format*, *Task-prescribed phrases*, *Input-dependent query*, and *Count limit* respectively.

Models	Content	Example	Format	Situation	Style	Mixed	Total
LLaMA2-13B-Chat	41.60	00.00	58.00	42.73	84.00	25.88	42.04
Ours-LLaMA2-13B$_{\text{Discrimination}}$	40.80	05.00	58.67	37.27	74.00	38.82	42.43
Ours-LLaMA2-13B$_{\text{Contrastive}}$	43.20	05.00	57.33	37.27	77.33	36.47	42.77

Table 13: Overall performance of models across different constraint categories on Followbench.

A.2 Complex Structure Synthesis

As stated in §4.1, we employ GPT-3.5-turbo to diversify the description for the same constraint. The corresponding prompt is shown in Tab. 14. It is worth noting that, for the *keyword* constraint, we prompt GPT3.5-turbo to brainstorm some keywords related to the instruction, shown in Tab. 15. Then, we randomly select one of them and incorporate it into the diversified description to form the final instruciton, e.g., your response should not include the word "architecture".

A.3 Generalization Experiments

A.3.1 General Instruction Following Ability

We adopt GPT-4 to compare and score the four candidates outputs given by LLaMA2-13B-Chat, Ours$_{\text{Generation}}$, Ours$_{\text{Discrimination}}$ and Ours$_{\text{Contrastive}}$ respectively. The score ranges from 1 to 10. To mitigate potential position bias in candidate outputs, we randomly shuffle the positions of the four candidate answers for each sample. The evaluation prompt is detailed in Tab. 17. Finally, we average the scores across all data samples.

A.3.2 Out-of-Domain Generalization

We provide detailed performance metrics and data examples for two out-of-domain complex instruction following benchmarks: FollowBench and CELLO. The detailed performance of FollowBench

/ Task prompt */*
You are provided with a <constraint> in an instruction. As a prompt engineer, your task is to rephrase the provided <constraint> to make it more diverse. You ought to provide five more variants of the <constraint>. Make sure your revision does not change the meaning of the original <constraint>.

/ Example */*
—INPUT—
<constraint>:
Your response should contain at least 3 sentences.
—OUTPUT—
variants:
1. Respond with at least three sentences
2. Use at least 3 sentences in your reply
3. Your entire response should include at least three sentences
4. Organize your entire response in at least 3 sentences
5. Please make sure the response is at least 3 sentences long

/ Input */*
—INPUT—
<constraint>:
{**Given_constraint**}
—OUTPUT—
variants:

Table 14: The prompts for diversifying the descriptions of a given constraint. We utilize one-shot in-context learning to enhance the performance. The information that requires manual input is highlighted in bold.

is shown in Tab. 13. Except for mixed categories, our methods typically exhibit decreased performance compared to the backbone model when evaluated against individual, unseen constraints. The declined performance in specific categories is reasonable. The complex instructions in specific categories (e.g., Style) from FollowBench (each has constraints from the same category) differ significantly from those in our training dataset (each contains constraints from multiple categories). We show some cases in the Tab. 21, with the constraints highlighted in bold. This suggests that models training with certain constraints can hardly generalize to unseen constraints directly. The detailed performance of CELLO is shown in Tab. 12. As demonstrated in Tab. 21, CELLO's constraints and language significantly differ from our training data.

A.3.3 In-Domain Generalization

We detail the test set construction process below. First, we select 200 instructions from the Open Assistant dataset (introduced in §4.1) not in our training set. Next, we randomly choose 3 to 5 constraints from IFEval, pair them with descriptions from our description pool (§4.1), and utilize GPT-3.5-turbo to paraphrase them, ensuring distinct descriptions from the training data. Additionally, we manually adjust specific requirements in the instructions, changing symbols (e.g., "separated by 6 asterisk symbols ******" to "separate the responses with 6 hash signs: ######") and formats (e.g., "wrap the entire output in JSON format" to "I want the entire output in XML format").

A.3.4 Adversarial Setting

We detail the test set construction process below. Specifically, we utilize the same 200 seed instructions from §5.4.3 and the method introduced in §4.1 to append 6 to 7 constraints to the seed instructions. These new instructions are challenging since our training data contains 3 to 5 constraints.

A.4 Case Study

We present some examples of various models following complex instructions in Tab. 18 and Tab. 19. Given the complex instructions with multiple constraints, we present the outputs generated by GPT3.5-turbo and LLaMA2-13B-Chat and the list indicating whether they have followed the specific constraint or not. Constraints in the instruction are underlined sequentially.

A.5 Implementation Details

We utilize 2 NVIDIA A800 80GB GPUs to conduct all the experiments. DeepSpeed ZeRO stage 1 is adopted for both SFT and DPO training. We use LORA(Hu et al., 2021) adaptor for effective training. We set the batch size to 4 for each GPU. All the methods utilizing SFT are trained for 2 epochs with the initial learning rate set to 3e-4 and the gradient accumulation step set to 4. The warm-up

/ Task prompt */*
You are provided with an <instruction>. Your object is to come up some keywords that may be used to answer the <instruction>. They are usually related to the task described in the <instruction>. you should output your thinking process and the keywords you come up with.

/ Example */*
—INPUT—
<instruction>:
Explain Generative Adversarial Networks (GANs) to me using bullet points. Do not contain any commas in your response.
—OUTPUT—
Thinking process:
The <instruction> asks to explain GANs, hence, "architecture", "training" and "generator" may be appropriate keywords to use in the answer.
Keywords:
["architecture", "training", "generator"]

/ Input */*
—INPUT—
<instruction>:
{**Given_instruction**}
—OUTPUT—

Table 15: The prompts for brainstorming some related keywords of a given instruction. The information that requires manual input is highlighted in bold.

steps are set to 300. As for the training of methods utilizing DPO, the beta value is set to 0.1. DPO is trained for 2 epochs with the initial learning rate set to 5e-4 and the gradient accumulation step also set to 4. We apply cosine learning rate scheduler and Adam optimizer to both models, and their maximum sequence length is set to 2048.

A.6 Ablation Study on the Contrastive Method

To prove the effectiveness of the proposed objective function in §4.3, we train LLaMA2-13B-Chat only utilizing naive DPO loss $\mathcal{L}_{\text{DPO}}$. As shown in Tab. 22, using only naive DPO loss causes models to output repeated constraints meaninglessly. This is probably because DPO Loss tend to overfit to the training data (Li et al., 2023c; Xu et al., 2024; Hejna et al., 2023). After incorporating the $\mathcal{L}_{\text{SFT}}$ loss into our objective function, the model can effectively follow complex instructions.

/ Task prompt */*
You are provided with a response which is generated by a LLM and a constraint that the response is asked to follow. Now, you have known that the response does not follow the constraint. You are designated as a corrector to correct the response. You should make as minimal revisions as possible so that it follows the constraint. For example, you should not change the case of the word if you are not asked. To fulfil this task, you are expected to provide your analysis and a revised response which has followed the constraint.

/ Example */*
—INPUT—
Response:
«Title»: ISO Code for Andorra. The International Organization for Standardization (ISO) code for Andorra is «ISO Code: 012». Andorra is a small, independent principality located in the Pyrenees mountains. The ISO code is a three-digit number that represents countries. I hope this information is helpful! Do you agree?
Constraint:
The very last sentence of your response should be "Hope you agree with me."
—OUTPUT—
Analysis:
The last sentence of the response is "Do you agree?". I need to change it to "Hope you agree with me." to follow the constraint.
Revised response:
«Title»: ISO Code for Andorra. The International Organization for Standardization (ISO) code for Andorra is «ISO Code: 012». Andorra is a small, independent principality located in the Pyrenees mountain. The ISO code is a three-digit number that represents countries. I hope this information is helpful! Hope you agree with me.

/ Input */*
—INPUT—
Response:
{**Given_response**}
Constraint:
{**Given_constraint**}
—OUTPUT—

Table 16: The prompts for correcting the response generated by the model to follow a specific constraint. The information that requires manual input is highlighted in bold.

/ Prompt */*
You are a helpful assistant who reviews a debate among four other assistants in evaluating the quality of the outputs for a given instruction. The four assistants, Assistant (LLaMA2-13B-Chat), Assistant (Ours-LLaMA2-13B$_{\text{Generation}}$), Assistant (Ours-LLaMA2-13B$_{\text{Discrimination}}$), and Assistant (Ours-LLaMA2-13B$_{\text{Contrastive}}$), are given an instruction. Output (LLaMA2-13B-Chat), Output (Ours-LLaMA2-13B$_{\text{Generation}}$), Output (Ours-LLaMA2-13B$_{\text{Discrimination}}$), and Output (Ours-LLaMA2-13B$_{\text{Contrastive}}$) are generated by four different AI chatbots respectively. Assistants have conflicting evaluations. Your goal is to rate each output, assigning higher scores to the assistants whose responses better fulfill the given instruction.
Here are some rules of the evaluation:
1) You should prioritize evaluating whether the output honestly, precisely, and closely executes the instruction, then consider its helpfulness, accuracy, level of detail, harmlessness, etc.
2) Outputs should NOT contain more or less than what the instruction asks for; as such outputs do NOT precisely execute the instruction.
Rate each output from 1 to 10, then output your final verdict using this format: [[LLaMA2-13B-Chat-x]], [[Ours-LLaMA2-13B$_{\text{Generation}}$-x]], [[(Ours-LLaMA2-13B$_{\text{Discrimination}}$)-x]], [[(Ours-LLaMA2-13B$_{\text{Contrastive}}$)-x]], where x is the score you assigned to each assistant.

/ Instruction */*
{**Given_instruction**}

/ Random order of four model outputs */*
<The Start of Assistant's Answer>
{**Model_output**}
<The End of Assistant's Answer>

Table 17: The prompts for scoring the outputs generated by different models to a general instruction. The information that requires manual input is highlighted in bold.

Instruction

Describe the content of the article in a brief manner. A study published earlier this year by Zee and her team examined the role of light in sleep for healthy adults in their 20s. Sleeping for only one night with a dim light, such as a TV set with the sound off, raised the blood sugar and heart rate of the young people during the sleep lab experiment. An elevated heart rate at night has been shown in prior studies to be a risk factor for future heart disease and early death, while higher blood sugar levels are a sign of insulin resistance, which can ultimately lead to type 2 diabetes. (1) The answer should be in at least **4** sections with each section starting with section X (where X is 1, 2, 3,...). The answer should be (2) written in English, with all letters lowercased. Respond with (3) less than **73** words.

GPT3.5-turbo

/ Output */*
section 1: study by zee and team on light's impact on sleep for young adults
section 2: sleeping with dim light raised blood sugar and heart rate
section 3: higher blood sugar levels indicate insulin resistance and risk of type 2 diabetes.

/ Followed list */*
[False, True, True]

LLaMA2-13B-Chat

/ Output */*
section 1: **I**ntroduction
the article discusses the impact of light on sleep and its effects on healthy adults in their 20s.
section 2: **E**xperiment **F**indings
the study found that sleeping with a dim light, such as a **TV** with the sound off, raised blood sugar and heart rate levels during the sleep lab experiment.
section 3: **R**isk **F**actors
elevated heart rate at night has been linked to future heart disease and early death, while higher blood sugar levels can lead to insulin resistance and type 2 diabetes.
section 4: **C**onclusion
the study highlights the importance of a dark, quiet sleep environment for optimal sleep and overall health.

/ Followed list */*
[True, False, False]

Table 18: The complex instructions and corresponding results of GPT3.5-turbo and LLaMA2-13B-Chat (Case 1).

Instruction

Summarize a meeting from the given list of bullet points. Be sure to convert shorthand into a first-hand account. Rose: Analyze data and presents findings
John: propose new idea
Jane: appointed to head project
Tom: need more time to fix a software bug.
(1) End it with a postscript starting with P.S.. (2) The **very end** of your entire response should be like: That is all you need!.
Your answer must also (3) contain at least **2** placeholders (an example of a placeholder is [address]).

GPT3.5-turbo

/ Output */*
During the meeting, Rose analyzed data and presented her findings. John proposed a new idea for the project, while Jane was appointed to head the project. Tom mentioned that he needed more time to fix a software bug.
P.S. [Address] That is all you need!

/ Followed list */*
[True, True, False]

LLaMA2-13B-Chat

/ Output */*
Sure, I'd be happy to help! Here's a summary of the meeting based on the provided bullet points:
The meeting began with Rose presenting her findings after analyzing the data. She shared some insightful observations and conclusions, which were well-received by the team.
Next, John proposed a new idea that he believed would benefit the project. He explained his reasoning and provided some supporting evidence for his proposal.
After John's proposal, Jane was appointed as the head of the project. She expressed her gratitude for the opportunity and outlined her vision for the project's success.
Tom then reported that he needed more time to fix a software bug. He explained that the bug was more complex than initially thought and required additional time to resolve.
In conclusion, the meeting was productive and successful. The team made progress on several fronts, including data analysis, proposal development, and project leadership.
P.S. That is all you need!
[Address] - the physical location of the meeting [Data] - the specific data that Rose analyzed
[Software bug] - the technical issue that Tom is working to resolve

/ Followed list */*
[True, False, True]

Table 19: The complex instructions and corresponding results of GPT3.5-turbo and LLaMA2-13B-Chat (Case 2).

Models	BaseModel	ChangeCase	Combination	Content	Format	Keywords	Language	Length	Punctuation	Startend	I-level	C-level
PaLM2-S	PaLM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	46.95	59.11
GPT3.5-turbo	GPT	66.29	75.38	88.68	89.17	74.23	100.00	65.03	24.24	86.57	63.96	73.62
GPT4	GPT	78.65	72.31	96.23	94.27	88.34	96.77	76.92	69.70	95.52	78.74	85.13
ChatGLM3-6B	ChatGLM	16.85	21.54	67.92	45.86	56.44	54.84	38.46	34.85	56.72	30.31	43.41
Qwen-14B-Chat	Qwen	58.43	23.08	75.47	58.60	60.12	83.87	36.36	25.76	74.63	40.11	53.00
LLaMA2-7B-Chat	LLaMA2	47.19	12.31	79.25	58.60	62.58	29.03	43.36	16.67	56.72	36.60	48.68
LLaMA2-13B-Chat	LLaMA2	51.69	15.38	83.02	67.52	67.48	41.94	47.55	09.09	58.21	41.22	53.00
LLaMA2-70B-Chat	LLaMA2	49.44	27.69	79.25	65.61	72.39	22.58	48.25	21.21	70.15	43.44	55.40
Vicuna-13B-V1.5	LLaMA2	60.67	44.62	75.47	64.97	61.35	93.55	48.95	22.73	67.16	46.95	58.03
WizardLM-13B-V1.2	LLaMA2	57.30	21.54	75.47	70.70	70.55	93.55	55.94	25.76	71.64	49.72	60.55
OpenChat-13B-V3.2	LLaMA2	58.43	35.38	88.68	71.34	68.10	90.32	58.04	24.24	74.63	51.02	62.59
Mistral-7B-Instruct-v0.2	Mistral	68.54	26.15	88.68	77.71	77.30	80.65	56.64	27.27	79.10	56.19	65.95
Ours-LLaMA2-7B$_{\text{Generation}}$	LLaMA2	57.30	16.92	71.70	70.70	60.12	61.29	33.57	19.70	65.67	40.67	51.92
Ours-LLaMA2-7B$_{\text{Discrimination}}$	LLaMA2	55.06	09.23	77.36	64.97	61.35	74.19	40.56	21.21	79.10	43.99	53.48
Ours-LLaMA2-13B$_{\text{Generation}}$	LLaMA2	66.29	26.15	66.04	73.25	59.51	35.48	49.65	27.27	82.09	46.03	57.31
Ours-LLaMA2-7B$_{\text{Contrastive}}$	LLaMA2	77.53	15.38	75.47	70.70	55.83	67.74	46.85	31.82	89.55	46.95	58.75
Ours-LLaMA2-13B$_{\text{Discrimination}}$	LLaMA2	69.66	12.31	79.25	67.52	62.58	96.77	49.65	54.55	80.60	50.83	61.27
Ours-LLaMA2-13B$_{\text{Contrastive}}$	LLaMA2	69.66	16.92	84.91	68.15	66.87	93.55	51.05	57.58	88.06	52.13	63.91
Ours-Mistral-7B$_{\text{Generation}}$	Mistral	76.40	50.77	66.04	78.98	61.35	58.06	55.94	46.97	92.54	54.90	66.07
Ours-Mistral-7B$_{\text{Contrastive}}$	Mistral	70.79	35.38	84.91	85.35	68.71	80.65	50.35	40.91	89.55	55.82	67.27
Ours-Mistral-7B$_{\text{Discrimination}}$	Mistral	82.02	20.00	71.70	81.53	63.19	96.77	55.24	62.12	85.07	56.75	67.39

Table 20: The loose accuracy score (defined by (Zhou et al., 2023a)) of models on different constraints of the IFEval. To alleviate this false negative problem, following (Zhou et al., 2023a), we use three variants of the model response to calculate a more loose accuracy score.

Source	Category	Instruction
FollowBench	Style	..., **Position yourself as a sagacious detective**, ..., **Respond with the whimsical humor** and **imaginative wit typical of Lewis Carroll**, ..., placing emphasis on refined language and meticulous attention to detail **in a manner befitting the social and literary norms of the early 19th century.**
FollowBench	Mixed	Lost, found vodka, drank to forget.\n \nAccording to the above prompt, write a **four-sentence** story that describes a man. However, **the word "man" should not appear** in the story. Please write **using an introspective narrative tone**.You should also **describe something about the bad weather**.
CELLO	Meta	模仿以下格式，出一道题目："input": "一个物体质量2千克，以10米/秒的速度运动，它的动能是？" "output": "动能公式...
CELLO	Structure	给定以下SQL文本，记录主键为f的薪水是多少？ "' 主键 性别 年龄 姓名 身高 a 女 70 q 213 ... e 女 55 u 170
Our Training data	N/A	For the following request, please **repeat the request itself** exactly as it is, ..., Create a list of subtopics for the given topic: Music theory. **Response should in all lowercase**. Your answer **must have a title** contained in double angular brackets, such as «title». **Refrain from using commas** in your response.

Table 21: Data examples from FollowBench, CELLO and our training data.

Instruction
Can you write a poem about the pros and cons of playing a lot of video games? Please make sure it's at least 40 sentences long (don't forget to add punctuation). You must highlight some words or phrases in your response, like *highlighted phrase*.
Output (w $\mathcal{L}_{\text{DPO}}$)
highlighted phrase "the pros and cons of playing a lot of video games" highlighted phrase* "the benefits and drawbacks of gaming" highlighted phrase* "the advantages and disadvantages of playing video games" highlighted phrase* "the positive and negative aspects of gaming" highlighted phrase* "the good and bad sides of gaming" highlighted phrase* "the pros and cons of gaming" highlighted phrase* "the benefits and drawbacks of gaming"...(repeated)
Output (w $\mathcal{L}_{\text{Ours}}$)
Sure, I'd be happy to help! Here's a poem about the pros and cons of playing a lot of video games: Pros of playing video games: * Fun and entertainment * Improved cognitive skills * Enhanced problem-solving abilities * Develops hand-eye coordination * Social connections with other gamers * Stress relief and relaxation * Escapism from real-life problems * Opportunities for creativity and self-expression Cons of playing video games: * Addiction and negative impact on mental health * Sleep deprivation and poor physical health * Social isolation and lack of face-to-face interaction * Increased aggression and violent behavior * Distraction from important responsibilities and tasks * Exposure to inappropriate content and negative influences * Wasting time and not being productive It's important to note that while playing video games can have both positive and negative effects, the key is to find a balance and enjoy them responsibly. *Highlighted phrase:* It's crucial to be aware of the potential downsides and take steps to mitigate them, such as setting limits on playtime, engaging in physical activity, and prioritizing real-life relationships and responsibilities.
Instruction
Generate a forum thread about several people waiting to hear the latest local news. All sentences should be short. Refrain from using any commas. Use placeholders to represent different usernames. Use square brackets for placeholders, like [username1], [username2]. Please include at least 20 placeholders in the thread.
Output (w $\mathcal{L}_{\text{DPO}}$)
[username1] [username2] [username3] [username4] [username5] [username6] [username7] [username8] [username9] [username10] [username11] [username12] ...(repeated)
Output (w $\mathcal{L}_{\text{Ours}}$)
Title: Waiting for the Latest Local News [username1]: Hey everyone, has anyone heard the latest local news? [username2]: No, have you? [username3]: Not yet, I'm waiting for the evening news. [username4]: Me too. I hope it's not more bad news. [username5]: I heard there might be a new development in the local project. [username6]: Really? Do you know anything about it? [username7]: No, I just heard it from a friend. [username8]: I hope it's something positive. We need some good news around here. [username9]: Agreed. I'm getting tired of all the negative news. [username10]: I know, it's been a tough few months. [username11]: I heard the local government is working on a new plan to improve the economy. [username12]: That sounds promising. Do you know what the plan is? [username13]: No, I don't have any details. But it's good to hear they're taking action. [username14]: Definitely. I'm looking forward to hearing more about it. [username15]: Me too. I hope it makes a difference. [username16]: I'm sure it will. The community needs something positive to focus on. [username17]: Absolutely. I'm feeling more hopeful already. [username18]: That's the spirit! We just need to stay positive and support each other. [username19]: Agreed. Let's keep each other updated on any news we hear. [username20]: Great idea! Let's keep the conversation going.

Table 22: The outputs generated by vanilla DPO loss and our training objective in §4.3.