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

Anonymous ACL submission

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. The improvement can even generalize to compositions of out-of-domain constraints. 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, training efficiency, and generalization abilities under four settings.

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).

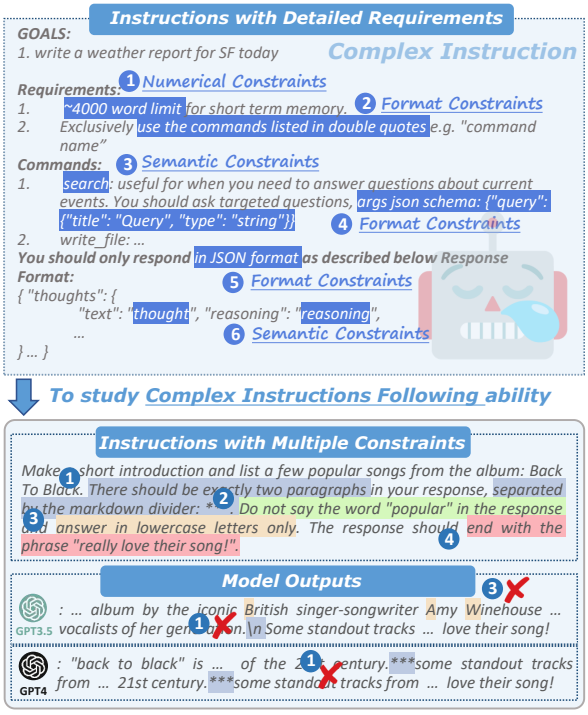

Figure 1: Real-world applications generally involve instructions with multiple constraints (i.e. *Complex Instructions*), posing challenges for models.

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

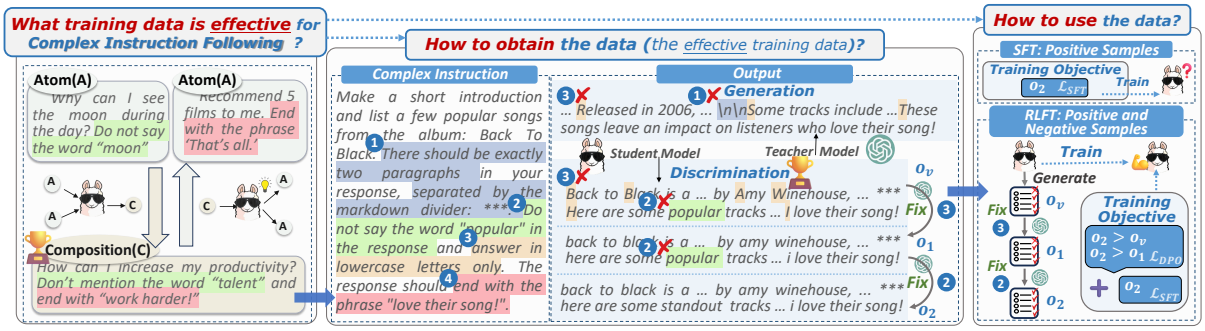

Figure 2: The framework of our study. We first study *what training data is effective* in enhancing complex instruction following abilities via an empirical study. Then, we design a discrimination-based method to address how to *obtain* the data. Finally, we propose a method for effectively *utilizing*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

biguous, 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., 2023; He et al., 2024; 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 (Jiang et al., 2023; 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 for this purpose 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 (§3), and can even generalize to the compositions of out-of-domain constraints found in §5.3.1.

To obtain high-quality compositional data, we generate initial output via a student model (vanilla model) and then correct 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 steps. It also demonstrates superior generalization across out-of-domain, in-domain, and adversarial settings while preserving overall 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 four settings.

2 Related Work

2.1 Instruction Following

There are various perspectives for assessing the ability of LLMs to follow instructions. A line of work perturbs the answer space to assess whether the model truly understands instructions or recites the answer (Zeng et al., 2023b; Li et al., 2023a; Wu et al., 2023). Another line of work exemplifies models' ability to follow instructions by incorporating verifiable constraints within them, such as lexical, numerical, format, and semantic constraints (Sun et al., 2023; Jiang et al., 2023). These constraints can be compositional, allowing one instruction to contain multiple constraints simultaneously (Aksu et al., 2023; Zhou et al., 2023b; Yao et al., 2023). Such complex instructions containing multiple user-specified constraints present 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 solely either focus on evaluation (Chen et al., 2024; Xia et al., 2024) or only consider instructions with few constraints (Chen et al., 2022; Zhang et al., 2023; Chen and Wan, 2023; Wang et al., 2024). Different from existing works, 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 instructions containing multiple 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?**

To enhance the LLM's capacity to follow complex instructions, two types of training data can be utilized: (1) Initially train models to understand atom constraints (atomic data), enabling them to resolve compositional constraints (compositional data) automatically. (2) Train models with compositional data, leading 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 the instructions in existing instructions following benchmarks (Zhou et al., 2023a; Jiang et al., 2023) into training and test sets. The training set contains atomic data (mostly with 1 constraint) and compositional data (mostly with over 3 constraints). Original benchmarks lack corresponding outputs, we first generate them via GPT-3.5-turbo. To improve the quality of the training set, we further filter the datasets to only keep outputs that satisfy all instruction constraints using GPT-3.5-turbo and rules for training. The remaining data forms the test set. Details on data construction and statistics are provided in the Appx. A.1.

We compare three methods: (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. To prevent models from catastrophic forgetting (McCloskey and Cohen, 1989), we mix training data with ShareGPT data (Chiang et al., 2023) for *Atom* and *Composition* checkpoint. 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., 2023):

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

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}).

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 backbone 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 among the open-source LLMs, respectively.

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 backbone 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-level and Constraint-level accuracy respectively.

The performance of the three methods on the test sets is shown in Tab. 1 and Tab. 2. First, with regard 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) can generally decrease performance compared to 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 (instructions with multiple constraints) 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 (Magister et al., 2022; Fu et al., 2023). Later in §5.3.1, we found that training with compositional data can even generalize to the compositions of out-of-domain constraints.

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., 2023) 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 had only 1 to 3 constraints, which were not complex enough. We construct 1500 complex instructions, each with 3 to 5 constraints from IFEval that are objective and can be automatically verified (Please refer to §4.1 for details). We leverage LLaMA2-13B-chat (Touvron et al., 2023) as the backbone and evaluate the performance of the three methods 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*) significantly outperform the outputs**

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 performance of different methods on IFEval.

generated by advanced LLMs directly (*Generation*). We believe this is because slight changes in the instruction (i.e. constraint) can cause substantial output differences, which the discrimination-based method captures better than the generation-based method.

4 Method

According to §3, we propose a discrimination-based method to obtain effective training data. A subsequent question is **how to effectively *utilize* the data obtained through the discrimination-based method?** Hence, we introduce a reinforcement learning fine-tuning (RLFT) based 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 written instructions covering diverse topics to facilitate instruction generation for new tasks. (3) *Super-Natural* (Wang et al., 2022b): A collection of natural language processing (NLP) tasks formatted with human instructions. We first exclude tasks with finite output sets using rules (e.g., classification, tagging), since the outputs are too simple for the corresponding instructions to incorporate constraints. This leaves us with 318 remaining tasks. Next, we randomly select one instruction for each task. From these three sources, we finally gather 1500 seed instructions.

Subsequently, we integrate constraints into these seed instructions. Initially, we randomly sample 3 to 5 constraints and utilize the provided scripts to resolve conflicts among the constraints provided by Zhou et al. (2023a). Next, given that, semantically equivalent but textually distinct instructions can substantially affect model outcomes (Yan et al., 2024; Chen et al., 2024), 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_i, 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, respectively. The number of constraints n ranges from 3 to 5.

4.2 Teacher Correction

As introduced 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. 9.

4.3 Contrastive Method

During §4.2, for each instruction I_c, we can gather positive sample set $\{o_f\}$ and negative samples set $\{o_1, ..., o_{f-1}\}$. 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).

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

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

where the reference model parameter π_{ref} is set to π_θ initially and remains fixed throughout training. β is a hyperparameter and σ is the sigmoid function. The goal of $\mathcal{L}_{\text{DPO}}$ is to maximize the log probability of 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. Therefore, we additionally 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 comprise popular open-source and close-source LLMs. With regard to our framework, utilizing synthesized complex instructions (§4.1), we compare three methods: (1) **Ours-13B-Generation** directly generates output with GPT-3.5-turbo and trains the backbone model via supervised fine-tuning (SFT). (2) **Ours-13B-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-13B-Contrastive** utilizes DPO for training to model positive and negative samples (§4.3). The backbone model for all three methods is LLaMA2-13B-Chat, with the instructions of training data being the same; only the output of training data and training paradigms differ. Specifically, continuous training may cause catastrophic forgetting (McCloskey and Cohen, 1989). To address this, we utilize the replay strategy (Ke and Liu, 2022), mixing the training data with 10000 ShareGPT data (Chiang et al., 2023) to maintain the general abilities of models during training.

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. All the constraints are objective and can be automatically verified, such as length constraints and detectable formats. The metrics are the same as §3.

5.2 Results

Overall Performance. The performance on IFEval is presented in Tab. 4. First, using the same backbone model, Ours-13B-Generation performs worse than many popular open-source models (Vicuna, WizardLM), even when the constraints in the test set have been seen in the instructions. This highlights the difficulty in obtaining high-quality output for complex instructions. Next, Ours-13B-discrimination achieves significant performance improvement, indicating that discrimination surpasses

Models	BaseModel	ChangeCase	Combination	Content	Format	Keywords	Language	Length	Punctuation	Startend	I-level	C-level
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
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
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
Ours-13B-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-13B-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-13B-Contrastive	LLaMA2	**65.17**	10.77	84.91	66.88	60.74	93.55	**47.55**	43.94	**86.57**	**48.24**	**59.71**
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	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	76.89	83.57

Table 4: The overall performance of models 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	ChangeCase	Combination	Content	Format	Keywords	Language	Length	Punctuation	Startend	I-level	C-level
LLaMA2-13B-Chat	17.86	00.00	68.42	58.54	61.43	27.27	34.43	00.00	27.03	09.50	42.27
WizardLM-13B-V1.2	16.67	13.64	56.58	53.66	64.29	100.00	40.98	17.39	48.65	14.00	47.20
OpenChat-13B-V3.2	25.00	00.00	76.32	56.71	61.43	86.36	35.25	15.22	55.41	16.50	49.07
Ours-13B-Discrimination	48.81	00.00	67.11	50.61	58.57	90.91	36.89	60.87	67.57	15.00	53.33
Ours-13B-Contrastive	35.71	04.55	63.16	50.61	65.00	86.36	47.54	63.04	79.73	**19.00**	**55.73**

Table 5: The performance of models on instructions within the same constraint category (each with 3 to 5 constraints) but with varying phrasing and detailed requirements, assessing our methods' in-domain generalization ability.

Models	ChangeCase	Combination	Content	Format	Keywords	Language	Length	Punctuation	Startend	I-level	C-level
LLaMA2-13B-Chat	25.71	08.70	67.44	47.41	60.71	28.00	26.92	02.38	21.90	01.00	40.15
WizardLM-13B-V1.2	28.57	00.00	54.26	50.00	66.67	72.00	34.62	15.48	52.38	07.00	46.60
OpenChat-13B-V3.2	31.43	04.35	62.79	56.03	60.71	72.00	31.73	23.81	49.52	07.30	47.64
Ours-13B-Discrimination	51.43	04.35	57.36	35.34	65.48	48.00	31.25	59.52	69.52	05.00	49.53
Ours-13B-Contrastive	40.95	08.70	50.39	45.69	72.22	64.00	37.50	55.95	74.29	**07.50**	**53.05**

Table 6: The performance of models on more challenging complex instructions with 6 to 7 constraints. The adversarial setting stress tests the generalization ability of LLMs in following complex instructions.

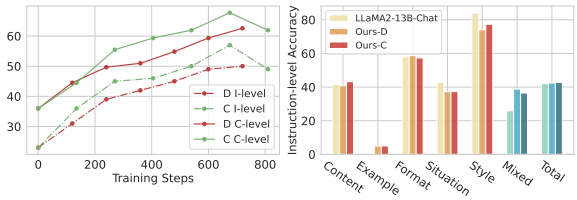

Figure 3: The performance of training efficiency (left) and out-of-domain generalization (right). D and C denote Ours-13B-Discrimination and Ours-13B-Contrastive respectively.

the generative paradigm in achieving high-quality output. Moreover, Ours-13B-contrastive performs the best, proving that our method excels in capturing subtle variations in complex instructions for the output.

Training Efficiency. We compare the training efficiency of Ours-13B-Discrimination and Ours-13B-Contrastive. Both use the same training data but employ different training methods: the former uses the next-token-prediction generation approach, while the latter uses our contrastive objective. As shown in Fig. 3 (left), Ours-13B-Contrastive achieves better performance with the same training steps and ultimately outperforms better than Ours-13B-Discrimination. This proves that our method utilizing both positive and negative samples can enhance complex instruction following ability more effectively and efficiently.

5.3 Generalization Experiments

We investigate the generalizability of our framework from four perspectives.

5.3.1 Out-of-Domain Generalization

We investigate whether the ability to follow complex instructions extends to unseen constraints. To achieve this, we evaluate models on another instruction-following benchmark FollowBench (Jiang et al., 2023), which has the following features to outline: (1) It contains almost entirely *different* constraints from IFEval, such as style scenario, and example constraints. (2) It includes com-

plex instructions of five difficulty levels. The difficulty level is denoted by incrementally increasing the *same* type of constraint to a seed instruction at each level. (3) Specifically, to mirror real-world scenarios, it introduces a *Mixed* Category. Instructions within this category encompass multiple constraints, akin to the compositional data in our study while incorporating different constraints.

As shown in Fig. 3 (right), first, the performance of our methods generally drops compared to the backbone model when tested on individual, unseen constraints. This suggests that models training with certain constraints can hardly generalize to unseen constraints directly. However, surprisingly, our methods show a remarkable 12.92% improvement in performance in the *Mixed* Category. This proves that tuning with compositional data enhances the models' capacity to follow instructions covering multiple constraints, even if these constraints differ greatly from those in the training set.

5.3.2 In-Domain Generalization

We construct a new test set to evaluate our methods' in-domain generalization, focusing on the same constraint but with varied wording and specific requirements. 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"). As shown in Tab. 5, Ours-13B-Contrastive remains the top performer. Additionally, the performance gap between Ours-13B-Contrastive and the best open-source model (OpenChat-13B-V3.2) has increased from 2.28 to 6.66. These results highlight the robustness of our method in handling complex instructions across different phrasing and detailed requirements within the same constraint category.

5.3.3 Adversarial Setting

We compare models' performance on more challenging complex instructions with increased constraints. This adversarial setting stress tests the generalization capacity of LLMs in following complex instructions. Specifically, we utilize the same 200 seed instructions from §5.3.2 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. As shown in Tab. 6, Ours-13B-Contrastive outperforms all other models and significantly performs better than Ours-13B-Discrimination. This demonstrates our method utilizing positive and negative samples generalizes better to complex instructions than SFT only utilizing positive samples.

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-13B-Discrimination	56.74	78.39	53.01	48.17	59.08
Ours-13B-Contrastive	57.76	79.95	53.79	48.15	59.91

Table 7: The performance of models on general tasks.

5.3.4 General Ability

We test whether training with our synthesized complex instructions compromises LLMs' general ability. To achieve this, we evaluate models on four widely adopted benchmarks, reflecting the models' 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. 7, our methods perform on par with other open-source LLMs, validating that our methods enhance the complex instructions following ability while maintaining 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 two empirical studies. Based on our findings, we introduce a method utilizing positive and negative samples to enhance LLMs' complex instruction-following capability. Our experiments show that our method more effectively and efficiently captures subtle instruction differences leading to significant output changes compared to the traditional supervised fine-tuning (SFT). Additionally, we evaluate the generalization capabilities of our framework through extensive experiments.

7 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 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*.

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.

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*.

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

Zixuan Ke and Bing Liu. 2022. Continual learning of natural language processing tasks: A survey. *arXiv preprint arXiv:2211.12701*.

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*.

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.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*.

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.

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 8: The statistic of the datasets constructed in the empirical study

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., 2023) 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. 8. 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.

A.2 Complex Structure Synthesis

As stated in §4.1, we employ GPT3.5-turbo to diversify the description for the same constraint. The corresponding prompt is shown in Tab. 10. 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. 11. 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 Case study

We present some cases of complex instruction following in Tab. 12 and Tab. 13. Given the complex instructions with multiple constraints, we present the outputs generated by GPT3.5-turbo and LLaMA2-13B-Chat and a list to indicate whether they have followed the specific constraint or not. Constraints incorporated in the instruction are underlined sequentially.

A.4 Implemention 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 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-5 and the gradient accumulation step set to 4. The warm-up 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.

/ 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 9: The prompts for correcting the response generated by the model to follow a specific constraint. The information that requires manual input is highlighted.

/ 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 10: 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.

/ 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 11: The prompts for brainstorming some related keywords of a given instruction. The information that requires manual input is highlighted.

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 12: 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 read exactly 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, False]

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

Models	ChangeCase	Combination	Content	Format	Keywords	Language	Length	Punctuation	Startend	I-level	C-level
LLaMA2-13B-Chat	51.69	15.38	83.02	67.52	67.48	41.94	47.55	9.09	58.21	41.22	53.00
Qwen-14B-Chat	58.43	23.08	75.47	58.60	60.12	83.87	36.36	25.76	74.63	40.11	53.00
Vicuna-13B-V1.5	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	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	58.43	35.38	88.68	71.34	68.10	90.32	58.04	24.24	74.63	51.02	62.59
Ours-13B-generation	66.29	26.15	66.04	73.25	59.51	35.48	49.65	27.27	82.09	46.03	57.31
Ours-13B-discrimination	69.66	12.31	79.25	67.52	62.58	96.77	49.65	54.55	80.60	50.83	61.27
Ours-13B-contrastive	69.66	16.92	84.91	68.15	66.87	93.55	51.05	57.58	88.06	52.13	63.91
GPT3.5-turbo	66.29	75.38	88.68	89.17	74.23	100.00	65.03	24.24	86.57	63.96	73.62

Table 14: The performance 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. Instruction-level accuracy and Constraint-level accuracy indicate the capacity of the model to follow the whole instruction and each constraint, respectively. The **bold** and underlined denote the first and second rankings, respectively.