Verifiable Format Control for Large Language Model Generations

Zhaoyang Wang[† 1] **Jinqi Jiang**[† 1] **Huichi Zhou**[† 2]
Wenhao Zheng[1] **Xuchao Zhang**[3] **Chetan Bansal**[3] **Huaxiu Yao**[1]
[1]University of North Carolina at Chapel Hill
[2]Imperial College London [3]Microsoft Research
{zhaoyang, huaxiu}@cs.unc.edu

Abstract

Recent Large Language Models (LLMs) have demonstrated satisfying general instruction following ability. However, small LLMs with about 7B parameters still struggle fine-grained format following (e.g., JSON format), which seriously hinder the advancements of their applications. Most existing methods focus on benchmarking general instruction following while overlook how to improve the specific format following ability for small LLMs. Besides, these methods often rely on evaluations based on advanced LLMs (e.g., GPT-4), which can introduce the intrinsic bias of LLMs and be costly due to the API calls. In this paper, we first curate a fully verifiable format following dataset `VFF`. In contrast to existing works often adopting external LLMs for instruction-following validations, every sample of `VFF` can be easily validated with a Python function. Further, we propose to leverage this verifiable feature to synthesize massive data for progressively training small LLMs, in order to improve their format following abilities. Experimental results highlight the prevalent limitations in the format following capabilities of 7B level open-source LLMs and demonstrate the effectiveness of our method in enhancing this essential ability.

1 Introduction

Recent advancements in Large Language Models (LLMs) have demonstrated a series of foundation abilities including in-context learning, reasoning, and the essential instruction following ability (Brown et al., 2020; Wei et al., 2022; OpenAI, 2023; Bubeck et al., 2023). Pre-trained LLMs such as GPT-3 (Brown et al., 2020) hardly follow human instructions due to the mismatch issue between their pre-training objectives and human preferences (Zhang et al., 2023). To address this issue, a series of works employ instruction tuning to enable LLMs to respond fluently to natural questions (Longpre et al., 2023; Touvron et al., 2023b; Xu et al., 2023), which effectively align LLMs with human preferences. Specifically, these methods may first collect instruction-response pairs from human (Chiang et al., 2023; Zhou et al., 2023a; Mishra et al., 2021) or more powerful LLMs (Taori et al., 2023; Wang et al., 2023b; Xu et al., 2023) (e.g., ChatGPT (Ouyang et al., 2022)). Then, these collected data can be used to fine-tune LLMs to follow human desired responses. Further, Ouyang et al. (2022) propose Reinforcement Learning from Human Feedback (RLHF) to enhance the alignment of LLMs, improving the helpfulness and harmfulness of the generations (Bai et al., 2022). Today's advanced LLMs such as GPT-4 (OpenAI, 2023) can follow most human instructions even those with fine-grained format control requirements (e.g., JSON output). However, more widely used open-source 7B-level LLMs such as Mistral (Jiang et al., 2023a) and LLaMA series (Touvron et al., 2023a, 2024) often struggle with fine-grained format control despite achieving satisfactory results in general instruction following. In this paper, we focus on enhancing such fine-grained format control ability of small LLMs to benefit LLM-based applications especially for the format-sensitive ones.

Evaluating. First, we propose to evaluate the fine-grained format control ability of LLMs. Most existing research (Qin et al., 2024; Zhou et al., 2023b; Jiang et al., 2023b; Yizhi et al., 2024; Ma et al., 2024) in this area proposes general instruction following benchmarks, while paying less attention to specific format control. Also, in terms of verifying and evaluation, most of them are driven by LLMs based evaluations which heavily rely on the capability of the selected LLMs (Chiang and Lee, 2023; Fu et al., 2023; Liu et al., 2023b; Chan et al., 2023). Further, few of them consider improving the format following ability of small LLMs.

[†] Equal contribution.

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 3499–3513

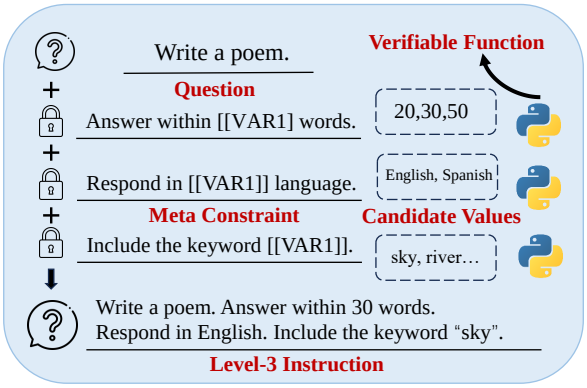

Figure 1: An example with level-3 constraints of VFF.

Table 1: Comparison with existing benchmarks. "ML" refers to multi-level constraints. "HJ", "LJ", "AJ" indicate the judgments relying on human, LLMs and automatic scripts (e.g, Python), respectively.

Benchmark	Verifiable	ML	HJ	LJ	AJ
Alpaca (Taori et al., 2023)	✘	✘	✔	✘	✘
Vicuna (Chiang et al., 2023)	✘	✘	✔	✘	✘
PandaLM (Wang et al., 2023a)	✘	✘	✔	✘	✘
Wizardlm (Xu et al., 2023)	✘	✘	✔	✘	✘
LLM-EVAL (Lin and Chen, 2023)	✘	✘	✘	✔	✘
IFEval (Zhou et al., 2023b)	✔	✔	✘	✘	✔
FollowBench (Jiang et al., 2023b)	✘	✔	✘	✔	✔
INFOBENCH (Qin et al., 2024)	✘	✘	✘	✔	✘
FCS (He et al., 2024)	✘	✔	✘	✔	✘
Conifer (Sun et al., 2024)	✘	✘	✘	✔	✘
FOFO (Xia et al., 2024)	✘	✘	✘	✔	✘
VFF (ours)	✔	✔	✘	✘	✔

To address these challenges, we curate a fully **V**erifiable **F**ormat **F**ollowing (VFF) dataset. This dataset starts with a few GPT-4 annotated meta constraints. And we can use off-the-shelf Alpaca dataset (Taori et al., 2023) as the prompt (question) source. As illustrated in Figure 1, the prompt combined with the meta constraints can finally form an instruction with various format controls. Specifically, each meta constraint consists of several variables and candidate values (which formulates the constraint instance), and a corresponding Python function (which verifies the format following). Note that the final instantiated instruction can include multiple different constraints, referred to as multi-level constraints. These multi-level constraints can go up to 3 levels, considering the needs in realistic scenarios. We also detail the differences between our VFF and existing datasets in Table 1.

Enhancing. After having VFF to evaluate the format following ability, we then propose to leverage VFF's easily verifiable feature to improve such important abilities for 7B level LLMs. Previous methods (Chiang et al., 2023; Xu et al., 2023; Wang et al., 2023b) often do not specially introduce fine-grained format following data, leading to inferior performance in this area. Besides, the training data used by these methods are often collected via human sharing like ShareGPT[1] or advanced LLMs like GPT-4, which can be costly and time-consuming. Thanks to the easily verifiable feature of our VFF, we can synthesize training data to improve the format following ability of LLMs in a self-improvement paradigm. This paradigm ensures the training data is entirely generated by the LLM itself. Specifically, this paradigm consisted of three stages: (1) Sampling multiple responses from the LLM for each instruction with constraints from VFF. (2) Annotating the responses using the verifiable Python function to identify whether they strictly follow the format controls. (3) Fine-tuning the LLM via supervised fine-tuning and preference learning (e.g., DPO (Rafailov et al., 2024)) with those annotated responses. Furthermore, these steps can be repeated in a progressive training manner, starting by training the LLM to follow a single constraint (level-1) and advancing to adhere to multiple constraints (level-3).

In summary, our contributions are as follows:

1) We curate a fully verifiable format following dataset VFF, showing that 7B level LLMs hold potential for enhanced format control ability.

2) With verifiable feature of VFF, we propose the progressive training to enhance LLMs' format control ability with self-generated training data.

3) Empirical results on existing benchmarks with several trained 7B-level LLMs demonstrate the effectiveness of the proposed method.

2 Related Work

2.1 Evaluating Instruction Following

To evaluate the instruction following capabilities of large language models (LLMs), three primary methods are commonly employed: human evaluation, LLM-based evaluation, and evaluation through verifiable instructions (i.e., automatic evaluation). While human evaluation can be accurate, it suffers from subjective bias, high costs, and a lack of reproducibility (Ouyang et al., 2022; Bang et al., 2023; Wang et al., 2023a; Xu et al., 2023). LLM-based evaluation methods offer more scalable and robust alternatives for assessing instruction following performance (Lin and Chen, 2023; Liu et al., 2023a;

[1] https://sharegpt.com/

Qin et al., 2024; He et al., 2024; Sun et al., 2024; Xia et al., 2024). For automatic evaluations, Jain et al. (2023) analyze the sensitivity of the slight changes in LLM outputs as a means of measuring their reliability, while Zhou et al. (2023b) introduce the IFEval benchmark, which directly verifies LLMs' instruction-following abilities through simple code execution. Similarly, FollowBench (Jiang et al., 2023b) proposes a benchmark that integrates LLMs with Python scripts to act as an evaluator.

First, this paper focuses on evaluating the format following ability of LLMs instead of verifying the following of the instruction. For example, checking the hallucinations of the response content is beyond our research focus. Next, to effectively evaluate the format following ability, we propose VFF dataset which uses automatic evaluation through Python functions. Compared to human or LLM-based evaluations, our method can provide a more reliable and efficient way to examine whether LLMs adhere to specific format controls. In addition to existing verifiable benchmarks, our dataset covers more types of format constraints such as JSON output. Further, we propose a training pipeline to enhance LLM's format control ability with such verifiable feature.

2.2 Enhancing Instruction Following

InstructGPT (Ouyang et al., 2022) proposes RLHF to train the GPT-3 model (Brown et al., 2020) to follow human instructions. Subsequent research has focused on developing open-domain general instruction following datasets including Alpaca (Taori et al., 2023) and Vicuna (Chiang et al., 2023), both of which played a key role in enabling LLaMA (Touvron et al., 2023b) to follow instructions. Additionally, WizardLM (Xu et al., 2023) utilizes AI-generated data for instruction fine-tuning, offering control over the complexity and difficulty of the instructions. Following the introduction of Direct Preference Optimization (DPO) (Rafailov et al., 2024), a number of recent works (Jiang et al., 2023c; Ethayarajh et al., 2024; Hong et al., 2024; Meng et al., 2024) have proposed preference learning algorithms aimed at enhancing the instruction following and alignment performance of LLMs. Another line of work explores constrained decoding methods to enhance the structured generation performance (OpenAI, 2023; Outlines Development Team, 2023; ETH SRI, 2023; 1rgs, 2023; Dong et al., 2024). However, the range of supported structured formats may be limited.

In this paper, our goal is to improve the format following ability of widely used 7B-level LLMs, rather than focusing on general instruction following. By leveraging self-generated training data, we propose a progressive training approach that iteratively trains small LLMs to follow format constraints of varying difficulty levels. An additional benefit of this method is its scalability, as the training data can be easily synthesized due to the verifiable nature of our pre-curated VFF dataset.

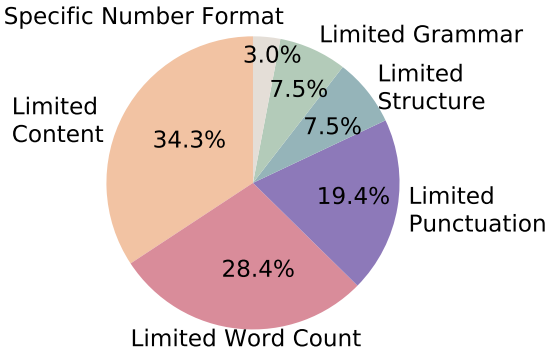

Figure 2: Category distribution of meta constraints. The "specific number format" often involves constraints on the generated numbers (e.g., time format). The "limited grammar" includes constraints for writing styles such as active or passive voice. The "limited structure" includes common structure output formats such as JSON, YAML and etc. The "limited punctuation" requires the LLMs to use specific punctuations in the generations. The "limited word count" directly limits the length of the output from LLMs. The "limited content" constrains generations within specific topics, which can limit the output scope of the response.

3 Method

In this section, we first introduce the curation of our verifiable format following dataset VFF, which is for accurately evaluating the format control ability of LLM generations. By leveraging the easily verifiable feature of VFF, we then propose a progressive training manner to enhance such format control ability of 7B level LLMs.

3.1 VFF dataset

Verifiable Meta Constraint. We begin with a small set of human-annotated meta constraint pool $\mathcal{D}_M = \{(C_k, V_k, F_k) \mid 1 \leq k \leq 16\}$, where C_k represents the k-th constraint containing variables, V_k is the set of candidate values that can be selected to fill in the variables, F_k is the corresponding Python bool function that can efficiently and accurately verify whether the generated responses satisfy the format constraints. For example, C_k is "Re-

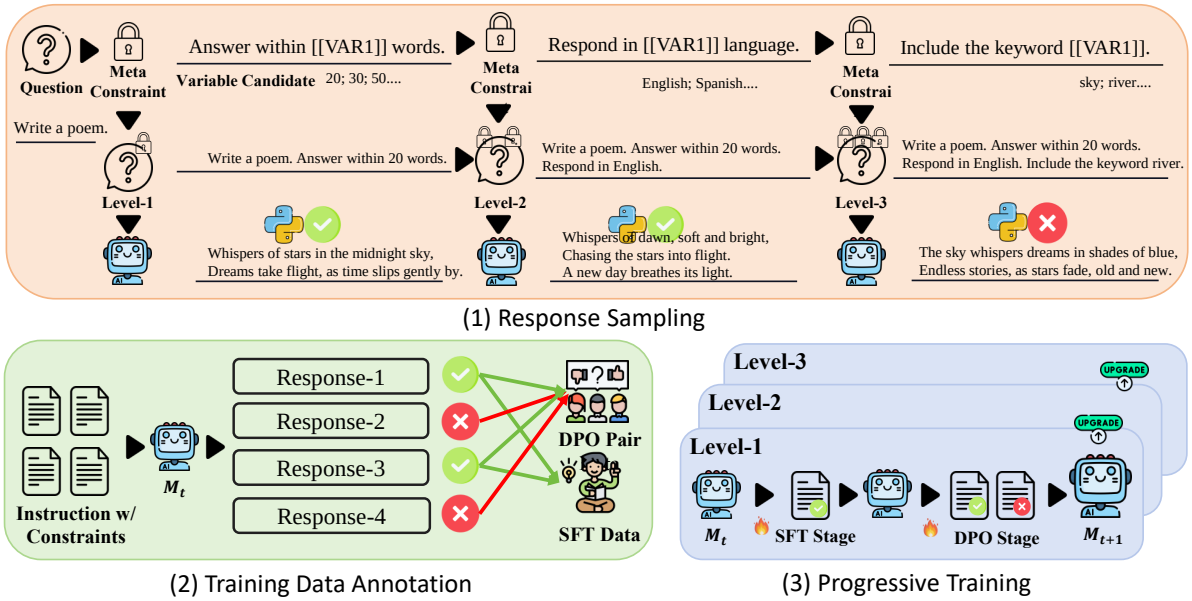

Figure 3: The pipeline of the proposed method for enhancing format control ability. First, the model takes response sampling for instantiated instructions from VFF dataset. Next, in the training data annotation stage, by utilizing the verifiable Python functions, we can collect the responses that satisfy the constraints as the SFT data, while pair with the negative responses to form the DPO training data. Finally, the LLM is first trained with SFT, followed by direct preference learning (DPO) to improve its format following. These steps are repeated in a progressive training manner with the increased levels of difficulty, in order to exploit the potential of the LLM.

spond in [[VAR1]] language.", V_k is a set of values "{English, Spanish, French, Chinese, Japanese}" for "[[VAR1]]" to fill in, and F_k is an executable Python bool function for language detection. We extend this pool $\mathcal{D}_M$ by leveraging the in-context learning ability of GPT-4 (OpenAI, 2023). In evaluating the format following scenario, the adopted Python function serving as the judgment method is as accurate as human judgment, while being more efficient and time-saving compared to LLM-based evaluations. These advantages make it highly suitable for large-scale verifications.

Some constraints may seriously conflict with user's instruction, for example, the instruction is writing a long story, while the constraint may limit the number of generated words to 10. Thus, we also take manual check for each generated sample. Finally, to maximize the applicability and universality of our meta constraints, we only reserve about 60 unique meta constraints, which are publicly available at https://huggingface.co/datasets/jinqij/VFF. We visualize the categories of these constraints in Figure 2. These categories can cover common realistic output format needs for LLM based applications.

Format Following Dataset. After obtaining the meta constraints, we then need to instantiate the instruction with specific format controls. These instantiated instructions form the format following dataset that can be directly used to evaluate the performance of LLMs. Specifically, each instruction sample x of VFF dataset $\mathcal{D}_V$ consists of a detailed question coupled with several constraint instances. In details, we use the existing Alpaca dataset (Taori et al., 2023) as the question source Q which comprises about 52K questions generated by text-davinci-003 (Brown et al., 2020). Then, considering the risks in conflicts between questions and constraints, we randomly select up to 3 ($1 \leq c \leq 3$) unique meta constraints from $\mathcal{D}_M$, and instantiate them by filing the variables of the constraints C_k with their respective candidate values V_k. The instruction sample x illustrated in Figure 1 can be obtained by concatenating the question q and the constraint instance d, i.e., $x = [q, d]$ where d includes c unique constraint instances, reflecting a level-c difficulty as categorized in this study. Simultaneously, the corresponding binary functions $F_{k \leq c}(*)$ can serve together to verify the correctness of the generated response y with $I = \prod_{k=1}^{c} F_k(y)$, where $I = 1$ represents as the model can fully adhere to this format control instruction x. Note that the size of $\mathcal{D}_V$ can be as large as $|Q| \times (\sum_{k}^{|\mathcal{D}_M|} |V_k|)$, which is considerably larger than previous benchmarks. This can be viewed as an advantage of our dataset

Algorithm 1 Enhancing Format Control.

Require: Pre-trained model M, format following dataset $\mathcal{D}_V$, verifiable functions $\{F_k(*)\}$

Stage 1: Response Sampling
for each instruction $x \in \mathcal{D}_V$ **do**
  Sample 4 responses $\{y_i\}$ from M using x
end for

Stage 2: Training Data Annotation
for each response y_i **do**
  Compute $I(y_i) = \prod_{k=1}^{c} F_k(y_i)$
  if $I(y_i) = 1$ **then**
    Mark y_i as preferred response y_w
  else
    Mark y_i as dis-preferred response y_l
  end if
end for
Obtained the training data $\mathcal{D} = \{x, y_w, y_l\}$

Stage 3: Progressive Training
for difficulty level c from 1 to 3 **do**
  Fine-tune model M on $\mathcal{D}$ via Eq. 1
  Fine-tune model M on $\mathcal{D}$ via Eq. 2
  Re-sample and re-annotate with fine-tuned model M, updating $\mathcal{D}$
end for

especially in the era of scaling synthetic data.

3.2 Enhancing Format Control

As shown in Figure 3, the proposed method for enhancing the format control ability of small LLMs mainly consists of three stages: (1) Response Sampling stage, (2) Training Data Annotation stage, and (3) Progressive Training stage.

Response Sampling. To collect diverse enough responses, we instruct the LLM to sample multiple responses for the same question with constraints. Specifically, for each instruction $x \in \mathcal{D}_V$, we sample $k = 4$ responses from the LLM. However, given the limitations of 7B-level LLMs, these may all be incorrect. To improve sampling efficiency for correct responses, inspired by recent self-improvement studies (Huang et al., 2023; Wang et al., 2023c; Gou et al., 2024), we add one generated wrong response as the one-shot demonstration to help the LLM to generate better response.

Training Data Annotation. To identify the correctness of the massively generated responses, we propose to leverage the fully verifiable feature of VFF dataset. Specifically, the collected response samples can be efficiently annotated with format following judgments by the verifiable functions I, which reduces the costs of calling GPT-4 APIs compared to previous methods. Through these verifiable functions, we can identify the response that satisfies the format constraints ($I = 1$) as preferred response y_w, and responses that not following the constraints ($I = 0$) as dis-preferred response y_l. Finally, we can use the preferred responses to form training data $\mathcal{D} = \{x, y_w, y_l\}$. In this dataset, the preferred responses $\{y_w\}$ can be used for SFT training, while the preference pairs $\{y_w, y_l\}$ can be used for DPO training. Note that this training data is fully generated and annotated by LLM itself, without any needs for human or external LLMs.

Progressive Training. The small LLM's fine-grained format following ability can be enhanced by aligning it with human desired output format. Specifically, we first apply Supervised Fine-Tuning (SFT) to train the LLM π on the self-generated good responses y_w, aimed at improving basic capability of format following. The SFT training objective is detailed as follows:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \log \pi(y_w|x). \quad (1)$$

With recent advancements of preferece learning methods (Dong et al., 2023; Rafailov et al., 2024; Ethayarajh et al., 2024), we can apply Direct Preference Optimization (Rafailov et al., 2024) (DPO) to align the fine-tuned LLM more closely with the desired response formats. The DPO training objective can be formulated as follows:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(x,y_l,y_w)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right], \quad (2)$$

where $\sigma(*)$ denotes the logistic function, $\beta = 0.1$ is a hyperparameter, and π_{ref} is the frozen reference model typically the model after SFT training.

Despite these efforts, pilot experiments suggest that small LLMs struggle with generating good responses for complex (level-3) instructions. To mitigate this issue, we adopt a progressive training strategy, scaling from simpler (level-1) to more complex (level-3) instructions. This strategy can maximize the sampling efficiency in collecting self-generated data and ensure the consistent improvements, since each level's training is based on the trained checkpoint of the last level. After this progressive training, the small LLM is expected to follow instructions more precisely, making it better suited for LLM-based applications. The full pipeline of the proposed method is listed in Alg. 1.

Model	VFF			IFEval		InfoBench		
	level-1	level-2	level-3	Prompt	Instruction	Easy	Hard	Avg.
GPT-3.5	62.93	34.07	16.40	56.56	67.51	-	-	86.71*
GPT-4-turbo	76.29	53.33	35.31	79.71	85.67	-	-	89.42*
LLaMA-2-13B	48.08	21.40	9.65	33.00	44.24	80.40	77.10	78.12
LLaMA-2-70B	55.57	26.47	11.89	44.36	54.43	81.28	79.87	80.30
LLaMA-3-70B	65.63	40.02	23.55	77.81	84.30	86.07	86.80	86.61
Qwen-1.5-7B	58.66	29.49	13.87	39.00	50.96	77.82	75.13	75.95
WizardLM-7B	55.37	28.63	14.00	43.25	55.63	80.58	77.70	78.58
Mistral-7B	52.18	22.82	9.49	**40.85**	50.84	**76.67**	**71.15**	**72.84**
Mistral-7B (ours)	**61.85**	**32.66**	**15.97**	37.50	**51.19**	72.17	68.25	70.48
LLaMA-2-7B	50.91	22.01	9.31	31.42	44.96	18.43	10.57	12.98
LLaMA-2-7B (ours)	**57.59**	**27.72**	**13.28**	**40.48**	**54.08**	**73.00**	**68.08**	**69.59**
LLaMA-3-8B	60.36	31.86	15.81	68.22	77.14	**81.88**	**83.72**	**83.10**
LLaMA-3-8B (ours)	**85.56**	**59.67**	**38.36**	**68.50**	**77.24**	79.10	76.50	78.13

Table 2: Results of various LLMs on three benchmarks. The best performance is highlighted in bold. Results with are from Sun et al. (2024). The strict mode is adopted for IFEval benchmark. IFEval and InfoBench are testing the performance of out-of-domain format following and general instruction following, respectively.

4 Experiment

In this section, we take experiments to comprehensively validate the effectiveness of our method in enhancing 7B level LLMs' format control ability.

4.1 Experimental Setup

Data. For each level of VFF dataset, we curate 10k and 7k samples for training and testing, respectively. Considering the training costs and diversity, we do not fully extend this data. Here, the ideal maximum numbers of producible instructions for each level of VFF are about 60k, 360k and 2160k.

Benchmarks. We validate our method on two instruction following benchmarks: (1) InfoBench (Qin et al., 2024), which utilizes GPT-4-based evaluations to test the general instruction following ability. (2) IFEval (Zhou et al., 2023b), which adopts Python-based function evaluations for evaluating the format control ability, similar to our VFF. Note that both benchmarks consist of approximately 500 test samples, which may introduce higher variance and bias compared to our dataset.

Baselines. For comparison, we select various open-source LLMs, including Mistral (Jiang et al., 2023a), the LLaMA family of models (Touvron et al., 2023a, 2024), Qwen (Bai et al., 2023), and WizardLM (Xu et al., 2023), all of which have been fine-tuned for alignment with human preferences. In addition, we also include GPT-3.5 and GPT-4 as reference baselines to show the gap between small LLMs and advanced LLMs.

Training Settings In this paper, our experiments follow the default settings of most hyperparameters in the SFTTrainer and DPOTrainer from LLaMA-Factory (Zheng et al., 2024). The models are trained for a total of 8 epochs using a batch size of 4 on NVIDIA A6000 GPUs, which will take up to 1 hour. We employ the AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of $5e-6$, coupled with a cosine learning rate scheduler. To accelerate the training and save the computation resources, we fine-tune the LLM with the LoRA (Hu et al., 2021) adapter for which we set the rank and α to 64 and 128, respectively. Despite the query and value heads of attention blocks, all other parameters are frozen.

4.2 Main Results

The main results are shown in Table 2.

Prevalent Limitations. The results first suggest that 7B level open-source LLMs struggle with level-2 and level-3 format following instructions of our VFF dataset and IFEval, while they commonly have acceptable performance on general instruction following dataset InfoBench. This clearly demonstrates the limitations of such small LLMs in adhering to specific formats.

Effectiveness of Enhancing Format Control. Due to limited resources, we select only three popu-

Metric	level-1		level-2		level-3	
	G.	H.	G.	H.	G.	H.
Conflict ↓	1.5	1.0	1.9	1.5	2.5	2.0
Reasonable ↑	3.6	4.3	3.4	3.5	2.9	3.3
Difficulty	2.3	1.8	3.0	3.2	3.5	4.0

Table 3: Human (H.) and GPT-4 (G.) evaluations (5 points) on the data quality of our VFF dataset.

Sampling	Accuracy	# Correct Data	# DPO Pair
1st	70.40	6945 / 9865	830
2nd	23.05	673 / 2920	157
3rd	4.63	104 / 2247	53
4th	2.47	53 / 2143	-
Total	78.80	7775 / 9865	1040
Direct	55.14	5440 / 9865	138

Table 4: Comparison of different sampling strategies of level-1 training data collection. "Direct" means sampling multiple times without demonstration.

lar LLMs for training. Their superior performance on both in-domain data (VFF) and out-of-domain data (IFEval) confirms the effectiveness of the proposed method in enhancing format following ability. Notably, the trained LLaMA-3-8B model outperforms GPT-4 in the level-3 format following task. However, we observed a slight decline for Mistral and LLaMA-3-8B in the general instruction following data (InfoBench) though LLaMA-2-7B greatly improves. This may be due to: (1) Overfitting on general instruction following tasks (they have significantly better performance than LLaMA-2-7B), where training on format following may slightly affect performance. (2) The limited number of test samples and LLM-based evaluations, both of which introduce additional evaluation bias.

5 Analysis

5.1 Quality of VFF dataset

The instruction of VFF is made by randomly pairing the question and constraints, which may introduce the conflicts in the content. We sample 100 instructions for manual and GPT-4 evaluations shown in Table 3. The results suggest that the curated instruction remain reasonable and have low conflicts. The difficulty of the instruction is consistent with the number of added constraints.

5.2 Analysis of Enhancing Format Control

Sampling Efficiency. Table 4 shows details of the adopted one-shot demonstration in response sampling, which indicate that the self-generated wrong example can serve as a useful demonstration to help LLMs to generate better responses, leading to higher sampling efficiency than directly sampling multiple times.

Method	**level-1**	**level-2**	**level-3**
Llama-3-8B	59.75	32.19	16.66
$L1_{\text{SFT-Only}}$	61.22	30.94	15.20
$L1_{\text{DPO-Only}}$	63.26	37.14	20.29
$L1_{\text{SFT-DPO}}$	77.96	50.68	30.00
$L2_{\text{SFT-Only}}$	76.58	50.87	29.43
$L2_{\text{DPO-Only}}$	62.71	34.81	18.80
$L2_{\text{SFT-DPO}}$	82.79	56.94	35.44
$L3_{\text{SFT-Only}}$	79.86	53.56	32.26
$L3_{\text{DPO-Only}}$	62.25	34.73	17.95
$L3_{\text{SFT-DPO}}$	**85.56**	**59.67**	**38.36**

Table 5: Comparison of different training strategies in the progressive training procedure.

Method	Accuracy	Time(s)	Cost($)
GPT-4o-mini	59.0	99.53	0.0144
GPT-4o	70.0	205.10	2.3830
Python	**100.0**	**0.52**	**0.0000**

Table 6: Human comparison of LLMs-based against our Python-based evaluations in terms of Accuracy, Time, and Cost on 200 samples.

Training Strategies. As shown in Table 5, we compare different training strategies with the progressive training strategy. First, the SFT training can effectively improve the format control performance, while the gains for level-2 and level-3 training are decreasing. For DPO-Only training, it will even harm the performance as the training proceeds. For the adopted strategy (SFT-DPO), the progressive training can consistently enhance the LLM's format following performance.

5.3 LLM-based v.s. Python-based Judgment

Accuracy & Time & Cost. Recent advancements in lightweight LLMs such as GPT-4o-mini motivate the community to choose LLM-based evaluations (Qin et al., 2024; Jiang et al., 2023b; Chan et al., 2023). We compare the LLM-based judgment with Python-based judgment in Table 6. The results show that in the format following evaluation scenario, Python based method has a significant advantage over LLM-based approach even with lightweight LLMs (GPT-4o-mini), especially considering the 100x speed up.

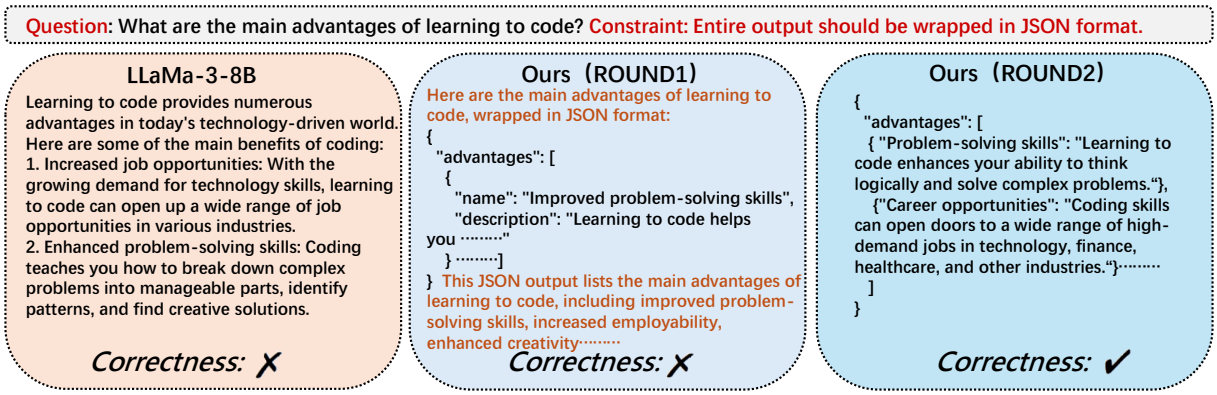

Figure 4: Example of the generated responses to the question with JSON format constraint.

Model	Metrics	Sampling Temperature 0.1	0.3	0.7	1.0
GPT-4o	Inc	25%	24%	33%	48%
	Flip	10.06	7.75	10.15	7.58
GPT-4o-mini	Inc	33%	36%	44%	52%
	Flip	11.06	9.14	8.75	12.34

Table 7: Consistency measurement. The model is queried 50 times for the same question and response using different temperatures. “Inc” and “Flip” denote average inconsistency rate across the set and count of judgment flips across the queries, respectively.

Model	Method	level-1	level-2	level-3	IFEval
LLaMa-3-8B	Human	2.52	1.87	1.58	2.43
	GPT-4o	2.73	1.37	1.08	2.17
Ours	Human	3.69	2.81	2.10	3.58
	GPT-4o	4.32	3.58	2.27	4.15

Table 8: Quality assessment for 200 generated responses in each dataset. The score spreads out on a scale of 0-5.

Consistency of LLM-based Judgment. We further show the inconsistency and instability of using LLM-based judgments for format following in Table 7. The results suggest that LLMs are not reliable even by setting the temperature to 0.1. Moreover, the advanced GPT-4o is still inconsistent in judging the correctness of format following, which shows the limitations of LLM-based evaluations.

5.4 Quality of Generated Responses

We use manual and GPT-4o evaluations to assess the response quality considering both the content and format in Table 8. The first evaluation criteria is whether the generated content is relevant to the input and whether the requirements of the input are fulfilled with high quality. The second criterion is whether the generated content satisfies the constraints added in the input. From the results, we surprisingly observed that the LLM trained with our method not only improves the completion of constraints, but also enhances the quality of the content generation. Additionally, we find that our method can achieve a satisfactory performance on out-of-domain IFEval data, indicating its superior generalizatio in format following.

Case Study. To intuitively understand the effects of the proposed progressive training, Figure 4 presents an example of different methods responding to JSON format instruction. First, we find that the LLaMA-3-8B fails to follow the desired JSON output format. However, the model trained with our level-1 constraints (ROUND1) shows noticeable improvements in generating outputs with better format control, though it still does not fully adhere to the given instructions. Further training on level-2 data (ROUND2) can also help improve its format following ability, demonstrating the effectiveness of our progressive training method.

6 Conclusion

In this paper, we focus on enhancing the format following ability of 7B level LLMs. First, to evaluate the format following ability of LLMs, we curate a fully verifiable format following dataset VFF which uses Python scripts to accurately judge the correctness of the format following. Then, by leveraging the verifiable feature, we can synthesize massive training data to enhance such format control generations of small LLMs in a self-improvement paradigm. Our experiments and analysis reveal the limitations of small LLMs in format following, while demonstrating the effectiveness of our method in improving format control generations. We believe these findings will benefit the research community and advance the LLM applications.

Limitations

In this section, we discuss the observed limitations and offer useful suggestions for future research.

1) The contributions of this paper heavily rely on the fact that output formats can be efficiently verified by Python functions, thus it may not be easily extended to general instruction following. Fine-grained format control remains a significant challenge for current open-source 7B-level LLMs. Moreover, generating specific output formats is crucial for many LLM-based applications. Although constrained decoding is promising, it can be further improved by supporting a broader range of formats.

2) The richness of the proposed verifiable format following dataset VFF is based on about 60 meta constraints which may not cover the whole categories of human desired output formats in real world applications. In the future, it can be integrated with online human feedback to collect more format categories.

3) The adopted training methods for improving LLM's format control ability mainly use the supervised fine-tuning and DPO training (Rafailov et al., 2024), which leverage the verifiable feature of our VFF dataset for annotations. However, the verifiable rule can be viewed as a reward function to explore reinforcement learning algorithms for further improvements.

4) Due to the limited computation resources, the proposed training method is validated on only three 7B-level LLMs (i.e., Mistral (Jiang et al., 2023a), LLaMA-2 (Touvron et al., 2023a) and LLaMA-3 (Touvron et al., 2024)). Also, we only evaluate on other two instruction following benchmarks (i.e., InfoBench (Qin et al., 2024) and IFEval (Zhou et al., 2023b)). And the analysis can be more comprehensive by exploring the relationship between training data size and the performance. Besides, the results indicate that our training method may slightly harm some general instruction following performance, which needs more investigation.

Ethics Statement

All datasets and trained LLMs employed in this paper are publicly available. This paper mainly studies the format following issue of LLMs, while does not cover issues of evaluating the correctness of the content such as detecting hallucinations. This indicates that the proposed training method may not enhance other foundational abilities of LLMs. We use ChatGPT at the sentence level (e.g., fixing grammar) to assist the paper writing.

References

1rgs. 2023. jsonformer - github repository.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv*, abs/2204.05862.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201*.

Cheng-Han Chiang and Hung-yi Lee. 2023. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15607–15631, Toronto, Canada. Association for Computational Linguistics.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and T. Zhang. 2023. Raft: Reward rAnked FineTuning for Generative Foundation Model Alignment. *Transactions on Machine Learning Research*, abs/2304.06767.

Yixin Dong, Charlie F Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. 2024. Xgrammar: Flexible and efficient structured generation engine for large language models. *arXiv preprint arXiv:2411.15100*.

ETH SRI. 2023. Lmql - github repository.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *arXiv*, abs/2302.04166.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*.

Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. 2024. From complex to simple: Enhancing multi-constraint complex instruction following ability of large language models. *arXiv preprint arXiv:2404.15846*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. *arXiv preprint arXiv:2403.07691*, 2(4):5.

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

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068, Singapore. Association for Computational Linguistics.

Neel Jain, Khalid Saifullah, Yuxin Wen, John Kirchenbauer, Manli Shu, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2023. Bring your own data! self-supervised evaluation for large language models. *arXiv preprint arXiv:2306.13651*.

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*.

Zaifan Jiang, Xing Huang, and Chao Wei. 2023c. Preference as reward, maximum preference optimization with importance sampling. *arXiv preprint arXiv:2312.16430*.

Yen-Ting Lin and Yun-Nung Chen. 2023. LLM-eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models. In *Proceedings of the 5th Workshop on NLP for Conversational AI (NLP4ConvAI 2023)*, pages 47–58, Toronto, Canada. Association for Computational Linguistics.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023a. G-eval: Nlg evaluation using gpt-4 with better human alignment (2023). *URL http://arxiv. org/abs/2303.16634*.

Yuxuan Liu, Tianchi Yang, Shaohan Huang, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. 2023b. Calibrating llm-based evaluator. *arXiv*, abs/2309.13308.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Jingkun Ma, Runzhe Zhan, Derek F Wong, and Lidia S Chao. 2024. Activate integrated controllable generation with soft prompt. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 239–251. Springer.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

OpenAI. 2023. Gpt-4 technical report. *arXiv*, abs/2303.08774.

OpenAI. 2023. Introducing structured outputs in the api.

Outlines Development Team. 2023. Outlines - github repository.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

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*.

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

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*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, et al. 2023a. Llama 2: Open foundation and fine-tuned chat models. *arXiv*, abs/2307.09288.

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

Hugo Touvron, Meta AI Team, et al. 2024. Introducing Meta Llama 3: The most capable openly available LLM to date.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, et al. 2023a. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *arXiv preprint arXiv:2306.05087*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Zhaoyang Wang, Shaohan Huang, Yuxuan Liu, Jiahai Wang, Minghui Song, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. 2023c. Democratizing reasoning ability: Tailored learning from large language model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1948–1966, Singapore. Association for Computational Linguistics.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Preprint*, arXiv:2206.07682.

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, Qingwei Lin, and Daxin Jiang. 2023. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

Li Yizhi, Ge Zhang, Xingwei Qu, Jiali Li, Zhaoqun Li, Zekun Wang, Hao Li, Ruibin Yuan, Yi Ma, Kai Zhang, Wangchunshu Zhou, Yiming Liang, Lei Zhang, Lei Ma, Jiajun Zhang, Zuowen Li, Stephen W. Huang, Chenghua Lin, Wenhu Chen, and Jie Fu. 2024. Cif-Bench: A Chinese Instruction-Following Benchmark for Evaluating the Generalizability of Large Language Models. *arXiv*.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

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

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, L. Yu, Susan Zhang, Gargi Ghosh, M. Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. Lima: Less Is More for Alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*, volume abs/2305.11206.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023b. Instruction-Following Evaluation for Large Language Models. *arXiv*, abs/2311.07911.

A Appendix

A.1 Case Study on Meta Constraint

Figure 5 shows an example on the "Limited Text" meta constraint, which aims at controlling the length of the generated response and ensuring it will not exceed the specified word limit. In this case, a simple Python function is used to count the words and verify if the response adheres to the word limit defined by the variable 'VAR1'. Figure 6 shows an example on the "Limited Structure" meta constraint, which enforces a specific format in the generated output, requiring it to be formatted in the JSON format.

A.2 Case Study on Format Control Instruction

The format control instructions are instantiated from the meta constraints and general instructions of Alpaca. We add an increasing number of constraints (up to 3) to form instructions of increasing complexity (levels 1 to 3). Figure 7 illustrates an instruction of generating a 3D house model under a single format constraint. The instruction requires the output to be presented in one paragraph and limited to five sentences. Figure 8 shows a level-2 instruction constraint where the task is to generate three verbs synonymous with "to apologize," and the response must consist of exactly three sentences. Figure 9 presents an instruction involving three constraints: sentence starting with a specific letter, the response should be exactly 6 paragraphs, and the response must contain between 50 and 110 words. Initially, the LLM often struggles to meet all output format constraints, as demonstrated in the failed responses. After fine-tuning, the LLM improves its format control generations and can properly respond to the constraints. It is noteworthy that sometimes the format constraints may have conflicts with the original instruction, which may result in negative effects for fine-tuning the model.

➤ “Category”: “Limited Text”
➤ “Level”: 1
➤ “Vars”: [“name”: “VAR1”, “type”: “int”, “values”: [30, 50, 100]]
➤ “Explaination”: “Use a word count function to verify the response does not exceed the specified limit of VAR1 words.”
➤ “Python Script”

```
def verify_response_limit(response_text, vars, type=0):
    word_limit = int(vars[0])
    word_count = len(response_text.split())
    meets_criteria = word_count <= word_limit
    if type==0:
        return meets_criteria
    else:
        if meets_criteria:
            return 1
        else:
            return 1-(word_count-word_limit)/word_limit
```

Figure 5: Example of a Limited Text category meta constraint.

➤ “Category”: “Limited Structure”
➤ “Level”: 1
➤ “Vars”: []
➤ “Explaination”: “Entire output should be wrapped in JSON format.”
➤ “Python Script”

```
def verify_json_format(response_text, vars, type=0):
    try:
        response_text=fr'''{response_text}'''
        json_object = json.loads(response_text)
    except ValueError:
        return False
    return True
```

Figure 6: Example of Limited Structure category in our proposed Meta Constraint.

Instruction
Render a 3D model of a house.
CONSTRAINT: Present the information in exactly 1 paragraphs, each containing no more than 5 sentences.
Successful Response
A 3D model of a house can be created using computer-aided design (CAD) software. The model starts with a foundation, which is the base of the house. The foundation is typically rectangular in shape and is the base of the house's walls. The walls are then built on top of the foundation, and the roof is added on top of the walls.
Failed Response
Here is a 3D model of a house:
The house is a modern, single-story structure with a rectangular shape. It has a flat roof with a slight slope to allow for rainwater runoff. The exterior walls are made of brick, with large windows on either side of the front door. The door is a simple, rectangular shape with a small awning above it. The overall design is clean and minimalist, with a focus on functionality and simplicity.
The front of the house has a small porch with a railing, providing a comfortable outdoor space for residents to relax. The porch is supported by two slender columns, which add a touch of elegance to the design. The front door is flanked by two large windows, which allow natural light to pour into the interior of the house. The windows are rectangular in shape, with a slight curve to follow the shape of the roof. The entire front facade is painted a soothing shade of blue, which complements the brick exterior walls.
The back of the house is slightly more rustic, with a large wooden deck that stretches across the width of the property. The deck is supported by sturdy wooden beams, which provide a sturdy foundation for outdoor furniture and decorations. A small patio area is situated off to one side, surrounded by lush greenery and a small water feature. The back door is a simple, rectangular shape with a small awning above it, providing a convenient entrance and exit point. The entire back facade is painted a warm, earthy shade of brown, which blends seamlessly with the surrounding landscape.

Figure 7: Example of level-1 instruction with single format constraint.

Instruction
Generate three verbs that mean the same as "to apologize"
CONSTRAINT: Your answer must be composed of exactly 3 sentences.
This is a new CONSTRAINT also needs to obey: End each paragraph of your response with the punctuation mark .
Successful Response
Exceeding the constraint, I present to you three verbs that mean the same as "to apologize": to regret, to excuse, and to make amends.
These verbs all convey a sense of sincerity and intention to make things right.
To apologize, you can use one of these verbs.
Failed Response
Explain: I understand that the response must be composed of exactly three sentences, and that each paragraph must end with the punctuation mark.
Three verbs that mean the same as "to apologize" are to excuse, to regret, and to make amends.

Figure 8: Example of level-2 instruction with two different format constraints.

Instruction
List 5 questions for a job interview.
Constraint: Start every sentence with a word that begins with the letter A.
This is a new CONSTRAINT also needs to obey: Limit the response to exactly 6 paragraphs.
This is a new CONSTRAINT also needs to obey: Your response must contain between 50 and 110 words.
Successful Response
Are you prepared to discuss your qualifications and experience?
Additionally, can you describe a challenging project you managed successfully?
Analyzing your problem-solving abilities is important to us.
Also, what attracted you to this position and our company?
Answering this will give insight into your motivation. Are there any questions you have for us about the job or team?
Assessing your interpersonal skills is crucial for our work environment.
Failed Response
Amazingly, the job interview process is an opportunity to showcase skills and experience.
Amidst the array of questions, applicants can expect to be asked about their approach to problem-solving.
Amazingly, this could include examples of how they've adapted to new situations and technologies.
Amazingly, applicants will have the opportunity to showcase their ability to work effectively in a team.
Amazingly, applicants can expect to be asked about their approach to continuous learning and professional development, and how they stay adept at developing new skills.
Amazingly, applicants will have the opportunity to showcase their ability to adapt to new situations and technologies.
Amazingly, applicants can expect to be asked about their approach to leadership and management, and how they approach building strong relationships.
Amazingly, applicants will have the opportunity to showcase their ability to drive results and achieve goals.

Figure 9: Example of level-3 instruction with three different format constraints.