Fine-Grained Constraint Generation-Verification for Improved Instruction-Following

Anonymous ACL submission

Abstract

The ability of Large Language Models (LLMs) to follow natural language instructions is crucial. However, numerous studies have demonstrated that LLMs still struggle to follow instructions with complex constraints, limiting their application in other areas. Meanwhile, obtaining high-quality instruction-following data often requires substantial manual annotation, which is both time-consuming and labor-intensive. In this work, we present `FiGV`, a fine-grained constraint generation-verification strategy for synthesizing instruction-following data. `FiGV` employs LLM-driven processes to generate fine-grained constraints and check the legality of the synthetic instructions. Subsequently, LLMs are utilized to perform nuanced, constraint-level verification to determine whether the generated responses adhere to the synthetic instructions, with LLM-generated functions incorporated for auxiliary validation tailored to the types of constraints. Experiments on 7B to 70B models demonstrate that `FiGV` consistently achieves strong performance across various benchmarks designed to evaluate the instruction-following capabilities of LLMs.

1 Introduction

The field of large language models (LLMs) has witnessed remarkable advancements in recent years, demonstrating a wide range of impressive capabilities (Zhao et al., 2024a). Among these, instruction-following stands out as one of the most critical requirements for LLMs, as it directly influences how effectively these models align with human intentions (Wang et al., 2023), serving as a key factor in ensuring the safety and reliability of LLMs. (Huang et al., 2023).

Although the instruction-following capability of LLMs is crucial, current models still exhibit limitations in following instructions with complex constraints (Zhou et al., 2023b; Jiang et al., 2024; Qin et al., 2024). To enhance the instruction-following capability of LLMs, current measures typically focus on instruction-tuning (Wei et al., 2022; Liu et al., 2023; Zhang et al., 2024a) the models using instruction-response pairs, where the former represents the human-provided instruction, and the latter denotes the desired response that aligns with the given instruction . The data used in this instruction-tuning phase is mainly obtained through manual annotation or the synthesis of complex instructions. For manual annotation, the high cost, low efficiency, and uncertain quality of human-labeled data make it difficult to scale, thus failing to meet the large-scale data requirements of current LLMs (Long et al., 2024). Regarding the synthesis of complex data, previous work (He et al., 2024; Sun et al., 2024) has primarily focused on incorporating multiple constraints into instructions and then using exisiting LLMs like GPT-4 to generate responses. While this approach yields promising results, the quality of the synthesized complex instructions is

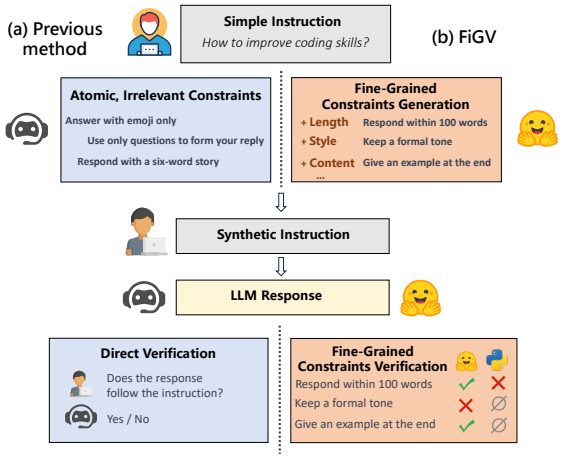

Figure 1: Comparison between the previous method for generating instruction-following data and `FiGV`. `FiGV` adopts a fine-grained constraint generation-verification strategy to ensure data quality.

hard to control, and the reliability of the distilled data cannot be guaranteed (Cui et al., 2024).

In this work, we address these issues by introducing a **Fi**ne-grained Constraints **G**eneration-**V**erification method for automatically synthesizing instruction-following data, named **FiGV**, which support both Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO) algorithm (Rafailov et al., 2023). To generate high-quality complex instruction-following data, FiGV incorporates several key components, including fine-grained constraints generation, instruction verification, and verified response generation to ensure that the instructions are diverse, realistic, and comprehensive, while responses remain reliable and aligned with the given instructions. During the constraints generation step, LLMs are prompted to generate fine-grained constraints based on the original instructions, considering multiple categories. In the instruction verification process, validity analysis is conducted on the synthesized instructions to ensure their reasonableness and verify that the added constraints do not conflict with one another. In the verified response generation phase, we employ LLMs to generate responses for the synthetic instructions and conduct fine-grained constraint-level verification to ensure that the generated responses align with each constraint in the instructions. To enhance the verification process, LLM-generated functions are introduced for auxiliary validation based on the types of constraints. By operating entirely under LLM supervision, FiGV demonstrates both automation and scalability.

A series of experiments are conducted to validate the effectiveness of FiGV by training LLMs ranging from 7B to 70B parameters, including models from the Qwen2 (Qwen, 2024), LLaMA3 (Meta, 2024), and GLM4 (GLM, 2024) series, across both SFT and DPO training algorithms. The effectiveness of our methodology is assessed using widely adopted instruction-following benchmarks, including IFEval (Zhou et al., 2023b), FollowBench (Jiang et al., 2024), and InFoBench (Qin et al., 2024). The results on these three instruction-following benchmarks demonstrate that FiGV significantly enhances LLMs' performance in complex instruction-following tasks. Experiments on MT-Bench (Zheng et al., 2023) and AlpacaEval (Dubois et al., 2024) further demonstrate that the models trained using our method exhibit performance comparable to their respective alignment models in general instruction-following abilities.

2 Related Work

2.1 Instruction Following

Instruction-following is one of the essential capabilities of LLMs. Previous studies (Weller et al., 2020; Mishra et al., 2022) has demonstrated that fine-tuning LLMs with annotated instructional data can enhance their ability to follow general language instructions. However, recent studies (Qin et al., 2024; Zhou et al., 2023b; Jiang et al., 2024) indicates that LLMs still struggle to follow complex instructions effectively. To address this issue, recent research (Sun et al., 2024; He et al., 2024) suggests that increasing the number and variety of constraints can enhance the complexity of instructions, thereby improving the model's ability to follow complex instructions. Typically, such studies (Zhang et al., 2024b; Dong et al., 2024; Sun et al., 2024) involve collecting a series of seed instructions, generating constraints, and subsequently creating responses based on these instructions and constraints using advanced LLMs. These efforts have demonstrated that constraint-based instruction tuning can significantly improve LLMs' instruction-following performance. In this work, we introduce an approach to enhance LLMs' instruction-following capabilities by generating and verifying fine-grained constraints.

2.2 Synthetic Data

Training LLMs on synthetic data is a promising approach for enhancing their capability to solve a wide range of tasks (Long et al., 2024; Liu et al., 2024a). Recent studies, such as Alpaca (Taori et al., 2023) and WizardLM (Xu et al., 2024), have utilized synthetic data for instruction tuning of LLMs. Compared to manually annotated instruction tuning data, synthetic data offers mainly two advantages: it is faster and more cost-effective to generate task-specific synthetic data, and its quality and variety often exceed what human annotators can produce (Zhang et al., 2024a). In the field of instruction-following, some studies (Sun et al., 2024; He et al., 2024; Dong et al., 2024) have employed synthetic data to enhance the instruction-following capabilities of LLMs, yielding promising results. However, they often lack effective evaluation and filtering for the instructions and responses. In this work, we propose a method that effectively supervises the quality of synthesized instruction-following data, enabling us to obtain high-quality instruction-following data.

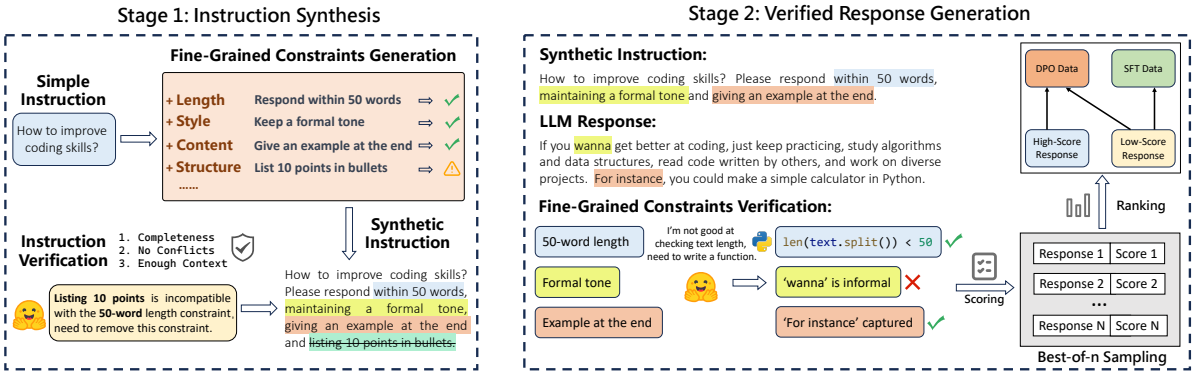

Figure 2: An overview of FiGV: The left section illustrates the Instruction Synthesis stage (Section 3.1), where fine-grained constraints are derived from original instructions and their legitimacy is verified. The right section presents the Verified Response Generation stage (Section 3.2), where responses are generated from synthetic instructions and verified at the constraint level to ensure adherence.

3 Method

In this section, we provide a detailed explanation of the methodologies employed in FiGV for constructing the instruction-following dataset. This process comprises two primary stages: the synthesis of instructions from original instructions (Section 3.1) and the generation of verified responses to these synthetic instructions (Section 3.2).

3.1 Instruction Synthesis

Building on insights from previous work (Dong et al., 2024; He et al., 2024; Sun et al., 2024), we identify the integration of diverse, realistic, and well-balanced combinations of constraints as the key to constructing high-quality instruction-following datasets.

In the instruction synthesis stage, FiGV begins with leveraging the supervisor model to generate fine-grained constraints derived from the original instructions. These constraints are then combined with the original instructions to create synthetic instructions. To ensure the quality of the generated data, FiGV incorporate a verification process to confirm that the constraints are non-conflicting and that the resulting instructions are coherent and reasonable. This systematic process allows us to produce high-quality synthetic instructions adapted to diverse scenarios.

Fine-Grained Constraints Generation This stage aims to generate realistic, detailed, and contextually relevant constraints across multiple categories. To achieve this, we first analyze a large corpus of open-source, real user instructions to identify comprehensive types of constraints. These constraints are then refined by human experts into several distinct categories. For further clarity and guidance, we include example constraints under each category, which were generated by GPT-4 (OpenAI, 2023).

To prompt the supervisor model for constraint generation, we randomly provide it with a subset of the predefined constraint categories. The supervisor model then proposes constraints relevant to the original instructions, tailored to the selected categories. This approach generates constraints that are more relevant and realistic compared to using specific atomic constraints alone (Dong et al., 2024). By synthesizing fine-grained constraints across multiple aspects, we generate synthetic instructions that are both complex and comprehensive, capturing a wide range of constraints and scenarios..

Instruction Verification The synthetic instructions generated by the supervisor model may not always be reliable. For instance, the added constraints might be contradictory, or the synthetic instruction could lack important content from the original instruction. Therefore, it is necessary to validate the synthetic instructions produced in the previous step.

During the validation process, the supervisor model evaluates the synthetic instruction to ensure it meets three key criteria: completeness, non-conflicting constraints, and sufficient contextual information to support a meaningful query. Only instructions satisfy these requirements are deemed valid. Following this process, we obtain the filtered synthetic instruction, denoted as I_S.

3.2 Verified Response Generation

After constructing fine-grained constraints and synthesizing complex instructions, obtaining high-quality responses that strictly adhere to these constraints is critical for effective model fine-tuning. Previous studies (Jiang et al., 2024; Sun et al., 2024) have employed LLMs to evaluate whether responses comply with instructional constraints. However, research has also identified significant limitations in LLM-based evaluations. For instance, (Kamoi et al., 2024) highlighted that LLMs often provide unreliable explanations, particularly when detecting errors. Similarly, in our experiments, we observed frequent inaccuracies in evaluating specific criteria, such as output length and keyword frequency.

To address these limitations, FiGV employs a hybrid strategy that combines direct LLM evaluation with verification functions also generated by LLMs. This integrated approach enhances the accuracy of evaluations by complementing the subjective assessments of LLMs with objective verification mechanisms, ensuring that responses more consistently adhere to the instructional constraints.

Constraints Classification In this step, we classify the constraints in each synthetic instruction into two categories based on their verifiability: those requiring automated functions due to limitations in LLM performance, and those that LLMs can evaluate effectively. This classification yields a synthetic instruction set with extracted constraints, denoted as $D_1 = \{I_S, C_F, C_L\}$, where C_F represents constraints that are more reliably verified by automated functions, such as text length or keyword existence, which LLMs struggle to evaluate accurately. On the other hand, C_L includes constraints that LLMs can evaluate well, often involving nuanced or contextual aspects of the instruction. This classification allows us to apply the most appropriate verification strategy for each constraint type, improving overall reliability and consistency.

Verification Function Generation In this part, we utilize the supervisor model to generate verification functions for the constraints identified in the previous steps as effectively verifiable by functions. To ensure the quality of these functions, we adopt the cross-validation method from AutoIF (Dong et al., 2024) to validate the quality of these verification functions. As a result, we extend the synthetic instruction set to include the generated verification functions, denoted as $D_2 = \{I_S, C_F, C_L, F\}$, where F represents the set of verification functions corresponding to C_F.

Response Generation & Verification After obtaining the synthetic instructions, we generate corresponding responses and evaluate their adherence to the specified constraints. To achieve this, we employ best-of-n sampling, generating multiple responses for each synthetic instruction. These responses are then evaluated and scored by both the supervisor model and LLM-generated functions to assess adherence to each constraint. The constraint-following score (CF) can be calculated as follows:

$$\text{CF} = \frac{1}{m}\sum_{j=1}^{m}\left(\mathbb{I}_{f_j}\cdot\frac{S_j^F + S_j^L}{2} + (1-\mathbb{I}_{f_j})\cdot S_j^L\right) \quad (1)$$

where m is the total number of constraints in the synthetic instruction. S_j^L represents the adherence to the j-th constraint as evaluated by the LLM supervisor model (boolean: 0 or 1), while S_j^F denotes the adherence score for the same constraint as assessed by the LLM-generated function (ranging from 0 to 1). The indicator function $\mathbb{I}_{f_j}$ determines whether the j-th constraint can be evaluated by a function, with a value of 1 if applicable and 0 otherwise. This scoring method allows for a fine-grained verification of constraint adherence.

For Supervised Fine-Tuning (SFT), we select the response with the highest CF score, provided that it exceeds a specified threshold. This ensures that synthetic instructions with conflicting constraints are further filtered out. For Direct Preference Optimization (DPO) (Rafailov et al., 2023), we use the SFT model to perform another round of best-of-n sampling. In this step, both high and low CF-scoring responses are selected to construct preference data, enabling the model to learn from comparative responses effectively.

4 Experiments

We conduct comprehensive experiments to evaluate the effectiveness of FiGV, mainly focus on the instruction-following performance.

4.1 Experimental Setup

Datasets We utilized LMSYS-Chat-1M [1] as the initial seed dataset. To ensure data quality, user instructions in the raw dataset were assessed across

[1] https://huggingface.co/datasets/lmsys/lmsys-chat-1m

Model	IFEval				FollowBench		InFoBench		
	Pr. (S)	Ins. (S)	Pr. (L)	Ins. (L)	HSR-Avg	SSR-Avg	Easy	Hard	Overall
GPT-3.5-Turbo-1106†	60.4	69.5	63.8	72.8	66.2	72.5	90.4	85.1	86.7
GPT-4-1106-Preview†	76.9	83.6	79.3	85.3	73.4	77.2	90.1	89.1	89.4
GPT-4o-2024-0513	81.1	86.7	85.4	89.6	76.7	79.4	89.2	92.1	90.7
GLM-4-0520	79.1	85.0	83.7	88.7	70.5	75.3	85.7	87.8	87.1
Qwen2-7B(LMSYS-Chat)	37.9	48.8	39.2	50.2	41.3	54.3	77.5	75.7	76.3
Qwen2-7B-Instruct	50.8	60.9	55.3	64.6	55.5	63.7	83.3	81.0	81.8
AutoIF-Qwen2-7B-DPO†	44.0	55.0	46.6	57.9	-	56.6	-	-	-
FiGV-Qwen2-7B-SFT	64.9	74.3	69.9	78.7	55.7	63.2	84.3	82.0	82.7
FiGV-Qwen2-7B-DPO	**67.5**	**77.0**	**71.7**	**80.5**	**57.0**	**65.1**	**84.6**	**83.7**	**84.0**
LLaMA3-8B(LMSYS-Chat)	42.9	52.2	44.0	53.3	41.5	56.1	78.9	74.3	75.7
LLaMA3-8B-Instruct	69.9	78.2	77.6	84.4	59.4	67.3	**83.4**	**84.0**	**83.8**
AutoIF-LLaMA3-8B-DPO†	28.8	42.4	43.1	56.0	-	49.9	-	-	-
FiGV-LLaMA3-8B-SFT	67.7	76.7	72.6	80.5	57.8	67.0	80.5	80.0	80.2
FiGV-LLaMA3-8B-DPO	**74.1**	**81.5**	**77.1**	**84.1**	**60.5**	**67.4**	82.5	81.9	82.3
GLM4-9B(LMSYS-Chat)	41.3	52.2	42.3	53.1	43.5	57.9	76.4	74.8	75.3
GLM4-9B-Chat	69.7	77.8	71.0	79.1	59.5	66.9	82.3	81.7	81.9
FiGV-GLM4-9B-SFT	67.1	76.3	70.4	79.0	58.5	66.7	83.8	81.7	82.2
FiGV-GLM4-9B-DPO	**73.9**	**81.2**	**77.3**	**83.8**	**61.5**	**69.3**	**85.4**	**84.1**	**84.5**
Qwen2-72B-Instruct	77.1	80.5	84.3	86.9	**68.9**	**73.2**	85.2	85.0	85.0
AutoIF-Qwen2-72B-Instruct-DPO†	80.2	**86.1**	82.3	88.0	-	67.5	-	-	-
FiGV-Qwen2-72B-SFT	78.6	84.7	82.6	87.9	64.9	69.8	87.4	87.3	87.4
FiGV-Qwen2-72B-DPO	**81.0**	85.4	**84.5**	**88.3**	67.1	72.5	**89.6**	**89.0**	**89.4**
LLaMA3-70B-Instruct	77.6	84.4	84.8	89.6	64.7	69.0	87.5	88.1	88.0
AutoIF-LLaMA3-70B-Instruct-DPO†	80.2	**86.7**	85.6	90.4	-	66.5	-	-	-
FiGV-LLaMA3-70B-SFT	77.3	83.6	82.7	86.3	63.2	68.9	85.2	85.8	85.6
FiGV-LLaMA3-70B-DPO	**81.4**	86.2	**85.9**	**90.7**	**64.9**	**69.1**	**89.2**	**88.9**	**89.0**

Table 1: Main results on three instruction-following benchmarks: IFEval, FollowBench and InFoBench. Pr. and Ins. denote prompt and instruction levels, respectively. S and L represent strict and loose metrics for IFEval. We use bold text for the best results and underline for the second-best results within the same model. Results with † are directly sourced from original papers or benchmarks.

dimensions such as clarity, specificity, answerability, and reasonableness, with only high-scoring instructions selected as seed data. Our training dataset is generated using the method described in Section 3, with GLM-4-0520 (GLM, 2024) serving as the supervisor model. Specifically, we used 20% of the prompts in the LMSYS-Chat dataset after filtration as seed data, resulting in a total of 28k SFT data and 7k DPO data. We employed the LLM decontaminator (Yang et al., 2023) to check potential data contamination between our training data and the testing sets and subsequently removed any contaminated data from the training set.

Implementation Details We conduct experiments on three open-source base models series: Qwen2 (Qwen2-7B and Qwen 2-72B) (Qwen, 2024), LlaMA3 (LlaMA3-8B and LLaMA3-70B) (Meta, 2024), and GLM-4 (GLM-4-9B) (GLM, 2024). We use the dataset above to train our SFT model from the base model and then further train the DPO model using the preference data we constructed on top of the SFT model.

The baseline includes alignment models (e.g., Qwen2-7B-Instruct) and base models (e.g., Qwen2-7B) fine-tuned using the original LMSYS-Chat dataset, with responses in the dataset rewritten by the supervisor model GLM-4-0520. The AutoIF (Dong et al., 2024) series are included for comparison, with experimental settings kept consistent with ours to ensure fairness.

Evaluation To assess the effectiveness of our approach in enhancing the model's instruction-following capabilities, we evaluate `FiGV` using three instruction-following benchmarks: **IFEval** (Zhou et al., 2023b), **FollowBench** (Jiang et al., 2024), and **InFoBench** (Qin et al., 2024).

IFEval includes 25 instruction types and 541 instructions that can be automatically validated using Python scripts, focusing on objective and reproducible metrics. For IFEval, we report the strict and loose accuracy at both the prompt and instruction levels. FollowBench is a fine-grained instruction-

following benchmark with five difficulty levels (L1 to L5) based on the number of constraints per instruction. Using advanced LLMs like GPT-4, it evaluates responses for constraint satisfaction. For FollowBench, we report the average of Hard Satisfaction Rate for fully satisfied instructions and the Soft Satisfaction Rate for individual constraint satisfaction. InFoBench evaluates LLMs' instruction-following ability by breaking down complex instructions into simpler tasks and leverages GPT-4 for assessment. For InfoBench, we report success rates across easy and hard sets, along with the overall success rate.

4.2 Main Results

The main results of our experiments on IFEval, FollowBench, and InFoBench are presented in Table 1. The models trained using FiGV method demonstrate excellent performance on both three instruction-following benchmarks.

Compared to models trained on the LMSYS-Chat dataset, our SFT models perform better across all instruction-following benchmarks, demonstrating enhanced instruction-following capabilities across diverse tasks. Furthermore, the DPO model trained with FiGV-constructed preference data often outperforms both corresponding alignment models and the AutoIF series trained from alignment models on all three benchmarks.

The significant improvements observed in the DPO model compared to the SFT model can be attributed to the method used for constructing the preference data. In FiGV, constraint-level verification is conducted to assess whether the generated responses adhere to the synthetic instructions, with LLM-generated functions integrated for auxiliary validation tailored to specific constraint types. By sampling responses from the SFT model and scoring them, a substantial number of positive and negative sample pairs are generated for DPO training. This enables the DPO model to effectively address the shortcomings identified during the SFT stage, thereby significantly enhancing its instruction-following capabilities.

Due to the fine-grained constraints from multiple aspects in our training dataset, our models demonstrate exceptional capabilities in handling complex combination of constraints, particularly evident in their performance on level 4 and level 5 of FollowBench and the hard set of InFoBench. For instance, Qwen-2-7B-DPO outperformed Qwen-2-7B-Instruct on levels 4 and 5 of FollowBench, and GLM-4-9B-DPO surpassed GLM-4-9B-Chat on the hard set of InFoBench. These results underscore the effectiveness of our approach in enhancing the models' ability to follow instructions in complex and challenging tasks.

4.3 Analyses

4.3.1 Ablation Studies

Model	IFEval	FollowBench	InFoBench
	Pr.(S)	HSR-Avg	Overall
GLM-4-9B SFT			
- *w/o* Verify	62.1	56.8	80.9
- *w* Direct Verify	63.6	57.5	81.9
- *w* Fine-grained	64.9	58.2	82.0
- *w* Func + Fine-grained	67.1	58.5	82.2
GLM-4-9B DPO			
- *w* Direct Verify	66.0	56.7	82.0
- *w* Fine-grained	71.3	60.9	83.7
- *w* Func + Fine-grained	73.9	61.5	84.5

Table 2: Model's performance on IFEval, FollowBench, and InFoBench with different strategies for response verification.

Model	Supervisor Model	IFEval	FollowBench
		Pr.(S)	HSR-Avg
Qwen2-7B	GPT-4o-0513	**65.9**	**57.0**
	GLM-4-0520	64.9	55.7
LLaMA3-8B	GPT-4o-0513	**68.2**	**58.5**
	GLM-4-0520	67.7	57.8
GLM-4-9B	GPT-4o-0513	**67.7**	**59.5**
	GLM-4-0520	67.1	58.5

Table 3: SFT model's performance on instruction following benchmarks with different supervisor models. Bold text indicates the best result within the same base model.

The models trained using FiGV exhibited exceptional performance across all three instruction-following benchmarks. A critical factor contributing to this success is our strategy of jointly employing LLMs and LLM-generated functions to verify whether responses adhere to each constraint in the instructions. To assess the effectiveness of the fine-grained constraints verification strategy within FiGV, we conducted an ablation study at both the SFT and DPO training stages of GLM4-9B. The results of this study are detailed in Table 2. In this context, Direct Verify uses the supervisor model to assess if the response follows the entire instruction without checking each constraint individually. Fine-grained examines if each specific constraint is met,

while Func + Fine-grained uses LLM-generated functions to assist in this process.

The results presented in Table 2 clearly demonstrate the impact of various response verification strategies on model performance. A consistent improvement in performance metrics is observed when moving from no verification to LLM Direct Verification, with further enhancements noted when employing the Fine-Grained Verification strategy. Notably, the LLM + Function Fine-Grained Verification approach achieved the highest scores across all benchmarks. This trend underscores the importance of fine-grained verification of constraints and indicates that evaluating responses for adherence to the constraints within instructions is crucial for constructing high-quality data for instruction-following.

We also conducted ablation experiments during the data synthesis phase using different supervisory models. As shown in Table 3, the stronger supervisor model GPT-4o-0513 demonstrates slightly better performance compared to GLM-4-0520. This is consistent with the observation that stronger models also serve as more effective synthetic data generators (Kim et al., 2024).

4.3.2 Complexity and Quality

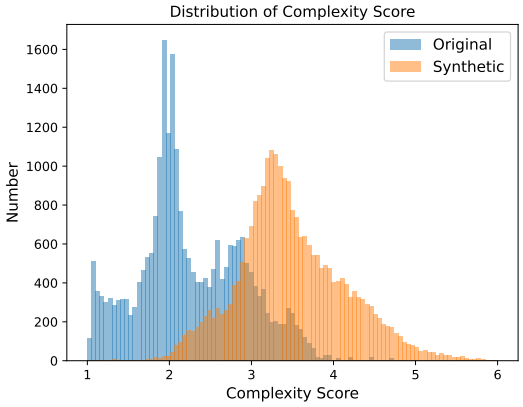

Figure 3: The distribution of complexity scores for original instructions and synthetic instructions. The instructions enhanced by FiGV demonstrate greater complexity compared to the original ones.

It is widely accepted that lengthy, challenging, and complex data samples yield greater benefits for instruction tuning (Zhao et al., 2024b). For instance, WizardLM (Xu et al., 2024) prompt ChatGPT to "evolve" data samples by deliberately enhancing their complexity, which led to improvements in LLM performance. To further investigate the improvement in complexity of our dataset compared to original LMSYS-Chat dataset, we employed the deita-complexity-scorer (Liu et al., 2024b) to evaluate the instructions originally present in LMSYS-Chat and those enhanced using FiGV. As illustrated in the Figure 3, the instructions enhanced by FiGV exhibit higher complexity compared to the original ones. This demonstrates the superiority of our synthesized data for instruction tuning.

Category	Win Rate (%)
Verified Response	54.28
Tie	11.58
Unverified Response	34.14

Table 4: Quality comparison between verified and unverified response.

During the instruction-tuning phase, the quality of the response is also crucial for the alignment of the model (Zhou et al., 2023a; Liu et al., 2024b). To validate that our evaluation of responses not only ensures adherence to complex constraints specified in the instructions but also maintains the overall quality of the responses, we prompted GPT-4 using the pairwise comparison prompt from MT-Bench (Zheng et al., 2023). This was employed to compare the highest-scoring responses after instruction-following evaluation with those directly output without evaluation. As illustrated in Table 4, the responses filtered through the instruction-following evaluation exhibit higher general quality. This demonstrates that our data is also beneficial for aligning with general human preferences.

4.3.3 General Abilities

Model	AlpacaEval	MT-Bench	IFEval
	LC WinRate	Score	Pr.(S)
Qwen2-7B-Instruct	32.6	8.49	50.8
Qwen2-7B-DPO	33.2	8.28	66.9
LLaMA3-8B-Instruct	31.1	7.96	69.9
LLaMa3-8B-DPO	36.2	7.56	73.6
GLM-4-9B-Chat	38.5	8.54	69.7
GLM-4-9B-DPO	37.2	8.49	71.1

Table 5: Model's performance on the AlpacaEval and MT-Bench for general instruction-following ability evaluation.

To verify that our synthetic data is effective not only for the instruction-following task but also in enhancing general capabilities, we also conduct evaluations using two widely recognized benchmarks **AlpacaEval** (Dubois et al., 2024) and **MT-**

Bench (Zheng et al., 2023) that assess LLMs' general ability to align with human preferences. AlpacaEval is an LLM-based automatic benchmark for evaluating response quality by comparing it against GPT-4's reference output and calculating the win rate. We use GPT4-1106-Preview (OpenAI, 2023) as evaluator and adopt AlpacaEval 2.0 Length-Adjusted win rate as our metric. MT-Bench (Zheng et al., 2023) is a multi-turn conversational benchmark consisting of 80 questions, where the model responds to an initial question followed by a predefined subsequent question, with GPT-4 rating the responses on a scale from 1 to 10.

As shown in Table 5, our DPO models not only demonstrate excellent performance in instruction-following evaluations, but they also achieve scores that are comparable to or even exceed those of corresponding alignment models on MT-Bench and Alpaca-eval. This indicates that our models not only enhance instruction-following capabilities but also effectively retain general-purpose abilities, demonstrating consistent improvements in aligning with general human preferences. The underlying reason for this phenomenon, as discussed in Section 4.3.2, is that the data generated by FiGV exhibits excellent complexity and quality. Additionally, the inclusion of fine-grained constraints from different aspects adds diversity to the data. This matches previous research (Liu et al., 2024b) indicating that good data for alignment requires such characteristics.

4.3.4 Scaling Anlysis

Stage	Data Amount	IFEval	FollowBench
		Pr.(S)	HSR-Avg
SFT	LMSYS-Chat(28k)	41.3	43.5
SFT	28k (100%)	67.1	58.5
SFT	14k (50%)	65.8	57.4
SFT	7k (25%)	63.7	56.3
SFT	3.5k (12.5%)	60.5	54.6
DPO	7k (100%)	73.9	61.5
DPO	3.5k (50%)	72.0	60.4
DPO	1.75k (25%)	71.7	59.3
DPO	0.875k (12.5%)	70.1	57.6

Table 6: Model's performance on IFEval, FollowBench, and InFoBench with different amounts of training data.

In the current trend of scaling language models, increasing the size of the training dataset is one of the key strategies (Muennighoff et al., 2023). To validate the potential of FiGV in terms of scalability for instruction-following tasks, we trained GLM-4-9B using 100%, 50%, 25%, and 12.5% of the SFT and DPO datasets, respectively. We then evaluated the fine-tuned model's performance across the three aforementioned instruction-following benchmarks.

As observed in Table 6, the model's performance increases with the amount of data used. However, even with a reduced dataset, the model maintains relatively high performance. Notably, the model trained with only 12.5% of the data exhibits exceptional performance across all three benchmarks, achieving over 70% prompt strict accuracy on IFEval and significantly outperforming the model fine-tuned with the original LMSYS-Chat dataset. This finding underscores the superiority of the data synthesized by FiGV and further validates the critical importance of data quality in instruction fine-tuning.

5 Conclusion

In this work, we introduced FiGV, a fine-grained constraints generation-verification method for synthesizing high-quality instruction-following data. Our method integrates fine-grained constraints generation, instruction verification, and verified response generation, all conducted under LLM supervision to ensure a fully automated pipeline that produces diverse, realistic, and reliable data for instruction-following tasks. Experimental results on IFEval, FollowBench, and InFoBench demonstrate that our approach significantly improves LLMs' ability to follow complex instructions. We also conduct extensive analytical experiments to evaluate the effectiveness, scalability, and potential of our method.

6 Limitations

We identify the limitations of our work in the following aspects. First, the LLM supervisor model generates constraints for the original instruction based on the predefined constraint categories. While this approach allows for the creation of diverse and realistic constraints, it may still fail to fully capture the wide distribution of constraints present in real-world scenarios. Second, during the response verification stage, although LLM-generated functions are introduced to assist the evaluation, the process fundamentally relies on the LLM-as-a-Judge paradigm. Developing more robust, objective, and reliable methods is necessary to further enhance the accuracy and credibility of the verification process.

References

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Ultrafeedback: Boosting language models with scaled ai feedback. *Preprint*, arXiv:2310.01377.

Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. 2024. Self-play with execution feedback: Improving instruction-following capabilities of large language models. *Preprint*, arXiv:2406.13542.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *Preprint*, arXiv:2404.04475.

Team GLM. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

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. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10864–10882, Miami, Florida, USA. Association for Computational Linguistics.

Xiaowei Huang, Wenjie Ruan, Wei Huang, Gaojie Jin, Yi Dong, Changshun Wu, Saddek Bensalem, Ronghui Mu, Yi Qi, Xingyu Zhao, Kaiwen Cai, Yanghao Zhang, Sihao Wu, Peipei Xu, Dengyu Wu, Andre Freitas, and Mustafa A. Mustafa. 2023. A survey of safety and trustworthiness of large language models through the lens of verification and validation. *Preprint*, arXiv:2305.11391.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688, Bangkok, Thailand. Association for Computational Linguistics.

Ryo Kamoi, Sarkar Snigdha Sarathi Das, Renze Lou, Jihyun Janice Ahn, Yilun Zhao, Xiaoxin Lu, Nan Zhang, Yusen Zhang, Haoran Ranran Zhang, Sujeeth Reddy Vummanthala, Salika Dave, Shaobo Qin, Arman Cohan, Wenpeng Yin, and Rui Zhang. 2024. Evaluating LLMs at detecting errors in LLM responses. In *First Conference on Language Modeling*.

Seungone Kim, Juyoung Suk, Xiang Yue, Vijay Viswanathan, Seongyun Lee, Yizhong Wang, Kiril Gashteovski, Carolin Lawrence, Sean Welleck, and Graham Neubig. 2024. Evaluating language models as synthetic data generators. *Preprint*, arXiv:2412.03679.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Preprint*, arXiv:2304.08485.

Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi Peng, Diyi Yang, Denny Zhou, and Andrew M. Dai. 2024a. Best practices and lessons learned on synthetic data. *Preprint*, arXiv:2404.07503.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2024b. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On llms-driven synthetic data generation, curation, and evaluation: A survey. *Preprint*, arXiv:2406.15126.

Meta. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. *Preprint*, arXiv:2104.08773.

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. 2023. Scaling data-constrained language models. *Preprint*, arXiv:2305.16264.

OpenAI. 2023. Gpt-4 system card.

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. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13025–13048, Bangkok, Thailand. Association for Computational Linguistics.

Team Qwen. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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. *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`.

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

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. *Preprint*, arXiv:2109.01652.

Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew E. Peters. 2020. Learning from task descriptions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1361–1375, Online. Association for Computational Linguistics.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. 2023. Rethinking benchmark and contamination for language models with rephrased samples. *Preprint*, arXiv:2311.04850.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2024a. Instruction tuning for large language models: A survey. *Preprint*, arXiv:2308.10792.

Xinghua Zhang, Haiyang Yu, Cheng Fu, Fei Huang, and Yongbin Li. 2024b. Iopo: Empowering llms with complex instruction following via input-output preference optimization. *Preprint*, arXiv:2411.06208.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2024a. A survey of large language models. *Preprint*, arXiv:2303.18223.

Yingxiu Zhao, Bowen Yu, Binyuan Hui, Haiyang Yu, Minghao Li, Fei Huang, Nevin L. Zhang, and Yongbin Li. 2024b. Tree-instruct: A preliminary study of the intrinsic relationship between complexity and alignment. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16776–16789, Torino, Italia. ELRA and ICCL.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. LIMA: Less is more for alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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. *Preprint*, arXiv:2311.07911.

A Dataset Distribution

# of Constraints	Count	Percentage (%)
≤ 3	4272	15.2
4	7774	27.6
5	7596	27.1
6	5321	18.9
≥ 7	3161	11.2

Table 7: Distribution of constraint numbers in the instructions of the dataset

Table 7 presents the distribution of the number of constraints within our synthesized instructions, which comprise a total of 28K instances with an average of 4.81 constraints per instruction. Of these, an average of 2.56 constraints are evaluated solely by the LLM supervisor, while 2.25 constraints are jointly evaluated by the LLM supervisor and the LLM-generated function."

B Model Training

For model training, we utilize LLaMA-Factory (Zheng et al., 2024) for all stages. For training Qwen2-7B, LLaMA3-8B, and GLM-4-9B, we use 8 × A100 GPUs. For Qwen2-72B and LLaMA3-70B, we scale up to 32 × A100 GPUs.

In the SFT phase, we perform full supervised fine-tuning on Qwen2-7B, LLaMA3-8B, and GLM-4-9B with a learning rate of 2×10^{-6}, using a cosine scheduler and a warm-up ratio of 0.1. The global batch size is set to 128, and the models are trained for 3 epochs. The maximum context length is 8192 tokens. For Qwen2-72B and LLaMA3-70B, the global batch size is increased to 512.

In the DPO phase, the learning rate is set to 1×10^{-6}, with a cosine scheduler and a warm-up ratio of 0.1. The global batch size is 64, and training is performed for 2 epoch with a preference beta value of 0.1. The maximum context length remains 8192 tokens.

C Prompt

Prompt for fine-grained constraints generation

As an expert in contextual language constraints, you will create **{Number}** constraints and combine them with the original instruction to generate a new, more complex instruction.
When creating these constraints, you should first identify a general category that encompasses the overall restrictions you wish to impose. Also, be mindful that constraints should not be mistaken for additional information or descriptions; they are merely to narrow the potential response scope. Furthermore, you need to consider whether the added constraints align with the original instruction, whether the instruction with added constraints is reasonable and likely to be a real instruction that a user might issue, and whether it is excessively rigid.
These are the categories of constraints that have been provided for you to choose from, if they are not suitable, you can also create your own constraints:
{Random Part of Constraints Categories}
Please note that your response should only return the new instruction without any additional information (such as the added constraints and the justification for the instruction's reasonableness)
Here is my original instruction: **{Original Instruction}**.
The new instruction is:

Prompt for instruction verification

You are a linguistics expert. I will provide you with an original instruction and an revised instruction with added format constraints.
You need to extract the newly added constraints by comparing the original and new instructions, list them in the form of [Constraint N], and then determine if the original and new instructions meet the following conditions:
1. The revised instruction should contain all the content of the original instruction.
2. The constraints added on the new instruction should be reasonable should not conflict with each other.
3. The revised instruction should be a reason-

able and meaningful question likely to be a real question a user might ask, and contain enough context for answering, and it should be an instruction rather than a statement.
The input format is:
[Original instruction]: Original instruction
[Revised Instruction]: Revised instruction with added format constraints
The output format is:
[Constraints Indentified]:
Constraint 1: Your first extracted constraint
Constraint 2: Your second extracted constraint
...
Constraint N: Your Nth extracted constraint
[Analysis]: Here, you need to analyze each condition one by one to see if they are met.
[Final Result]: Output YES or NO here. If all the 3 conditions are met you should output YES, otherwise output NO. Do not include any other information.
Now please evaluate the following original instruction and revised instruction and provide your judgment:
[Original instruction]: **{Original Instruction}**
[Revised Instruction]: **{Revised Instruction}**
Please provide your judgment:

Prompt for constraints classification

You are a linguistics expert. I will provide you with an original instruction, and a revised instruction that includes additional constraints. Your task is to identify the constraints added in the revised instruction compared with the original instruction and determine which of these constraints relate to keywords, length, or changing case.
To be more specific:
Keyword Usage may include requirements about the presence of specific keywords, the frequency of these keywords, and letter frequency in keywords. Note that only keywords with specific definitions or requirements are considered, instead of general keywords like transition phrases or third-person perspectives.
Length Requirements may include limits on the number of words, number of characters, or the length of each sentence or the whole response.
Case Constraints may involve requirements about the use of capital words or lowercase words in the prompt.
You also need to state why the constraints can be checked by pure Python code without searching for outside resources and assuming some certain prerequisites.
Input format:
Original Instruction: What is oyster sauce?
Revised Instruction: Describe oyster sauce, use only one-sentence responses, begin with "Oyster sauce is", and incorporate an idiomatic expression that illustrates its flavor profile and do not exceed 200 words. Do not use any contractions in your response.
Output format:

```
{
    "Constraints_extracted": {
       "Constraint 1": "Use only one-
        sentence responses.",
        "Constraint 2": "Begin with
        'Oyster sauce is.'",
      "Constraint 3": "Incorporate an
        idiomatic expression that
      illustrates its flavor profile.",
       "Constraint 4": "Do not exceed
        200 words.",
      "Constraint 5": "Do not use any
        contractions."
    },
    "Analysis": "Constraint 2 is
    related to keywords constraints
    and can be checked by
    python code using startwith()
    function. Constraint 4 is related
    to length constraints and can
    be checked by python code
    using len() and split() function
    to count how many words.
    Constraints 5 is related to
    keywords constraints but can
    not be checked by python code
    since the variety of contractions
    is too large.",
    "Final_result": ["Constraint 2",
    "Constraint 4"]
}
```

The value of "Constraints_extracted" should be a dictionary containing the constraints extracted from the revised instruction. The value of "Analysis" should be a string explaining which constraints relate to keywords,

length, or changing case and why they can be checked by pure Python code. The value of "Final_result" should be a python list containing the constraints that relate to keywords, length, or changing case and can be checked by pure Python code.
Provide your judgment result below, Please note that you should only return a json object with the format we discussed above:
Original Instruction: **{Original Instruction}**
Revised Instruction: **{Revised Instruction}**
Output:

Prompt for generating verification function

You are an expert for writing evaluation functions in Python to evaluate whether a response strictly follows a format constraint in the user instruction.
Input Format: A format constraint in the user instruction.
Output Format: A single JSON includes the evaluation function in the key ‘func‘, and a list of three test cases in the key ‘cases‘, which includes an input in the key ‘input‘ and an expected output in the key ‘output‘ in (true, false). Here is an example of output JSON format:

```
{{"func": JSON_STR(use only
\\n instead of \n),
"cases": [{{"input": bool,
"output": bool}}]}}.
```

Other Requirements:
1. Please write a Python function named ‘evaluate‘ to evaluate whether an input string ‘response‘ follows this format constraint. If it follows, simply return True, otherwise return False.
2. If your function requires any external libraries, ensure to include the import statements within the evaluate function.
Here is the constraint: **{Constraint}**
Please output your json here:

Prompt for constraints-following evaluation

You are a linguistics expert. I will provide you with a instruction and a response to this instruction. I will also give your a list of constraints that the response should follow. Your task is to determine whether the response adheres to these constraints.
Please follow the input and output formats provided below:
Input format:
[Instruction]: Provide a summary of the benefits of learning a second language in three bullet points. Each bullet point should be one sentence long and include the word "advantage." Avoid using technical jargon and ensure the summary is suitable for a general audience.
[Response]:
- One advantage of learning a second language is enhanced cognitive abilities.
- Another one is the increased cultural awareness and appreciation.
- A third advantage is the improved employment opportunities.
[Constraints]: ["The summary should be in three bullet points.", "Each bullet point should be one sentence long.", "Each bullet point should include the word ’advantage’.", "Avoid using technical jargon.", "Ensure the summary is suitable for a general audience."]
Output format:

```
{{
  "Analysis": {{
    "Constraint 1": "Constraint 1 is
    met, the response contains three
       bullet points.",
    "Constraint 2": "Constraint 2 is
       met, each bullet point
       is one sentence long.",
    "Constraint 3": "Constraint 3 is
       not met, the setence after
       the second bullet point
       does not include the word
       'advantage'.",
    "Constraint 4": "Constraint 4 is
       met, the response avoids
       technical jargon.",
    "Constraint 5": "Constraint 5 is
       met, the summary is suitable
       for a general audience."
  }},
 "Final_result": [true, true, false,
  true, true]
}}
```

The value of "Final_result" should be a python

list of boolean values indicating whether each constraint is met.
Provide your judgment result below, Please note that you should only return a json object with the format we discussed above:
[Instruction]: **{Instruction}**
[Response]: **{Response}**
[Constraints]: **{Constraints}**
[Output]:

Constraints Categories

Keyword Usage:
Description: Ensuring the use of specific keywords or avoiding certain forbidden words in the text. This includes requirements for the number, frequency, occurrence of specific letters, and placement of keywords.
Example:

- Keywords existence
- Forbidden words
- Keywords frequency
- Letter frequency in keywords
- Keywords in specific positions

Language Style:
Description: Adhering to specific language style or tone in the response, such as using a particular dialect or regional language, adopting a formal or informal tone, using gender-specific or gender-neutral language, or employing idioms or colloquial expressions.
Example:

- Constraints on what kinds of Language should be used in response
- Specific dialects or regional language constraints
- Formal or informal tone
- Gender-specific / Gender-neutral language
- Use of idioms or colloquial expressions

Length Requirements:
Description: Specifying concrete limits on text length including the number of paragraphs, sentences, words, initial words in paragraphs, or length of each sentence in terms of words or characters.
Example:

- Number of Paragraphs
- Number of Sentences
- Number of Words
- First Word in i-th Paragraph should be ...
- Number of characters
- Length of each sentence in terms of words or characters

Content Structure:
Description: Organizing content according to specific requirements, including the number of placeholders, inclusion of postscripts, presence of specific phrases or idioms, use of specific tags or markers, and the number of references or citations.
Example:

- Number of placeholders
- Postscript
- Specific phrases or idioms
- Presence of specific tags or markers
- Number of references or citations

Case Constraints:
Description: Imposing constraints on the use of upper or lower case letters in the text, including overall frequency, use of title case for headings, consistency within paragraphs, and consistency in the use of abbreviations or acronyms.
Example:

- Capital words or Lowercase words
- Frequency of capital/lower words
- Title case for headings
- Case consistency within a paragraph
- Consistency in the use of abbreviations or acronyms

Formatting Rules:
Description: Specifying concrete formatting requirements for the text, including multiple sections, the number of bullet lists, highlighted sections, the name of the title, and specific alignment (left, right, center).
Example:

- Multiple sections
- Number of bullet lists
- Number of highlighted sections
- Name of the title
- Specific alignment (left, right, center)

Mixed Approaches:
Description: Combining various methods in the text response, such as repeating user prompts before answering, providing multiple

responses for a single prompt, writing from different perspectives, and integrating questions and answers in the response.
Example:

- Repeat the user prompts before answering the question
- Give multiple responses for a single prompt
- Use of different perspectives in the response
- Integrating questions and answers in the response

Punctuation Usage:
Description: Imposing specific rules on the use of punctuation marks, such as avoiding commas or colons, using specific punctuation marks at certain positions, the frequency of semicolons or ellipses, and the use of exclamation marks or question marks.
Example:

- No use of comma/colons
- Specific punctuation marks at certain positions
- Frequency of semicolons or ellipses
- Use of exclamation marks or question marks

Opening and Closing Rules:
Description: Specifying concrete requirements for the opening and closing of the text, such as starting or ending with specific words, punctuation, or quotations, including a famous quote, or beginning or ending with a summary statement.
Example:

- Start/end with specific words
- Start/end with specific punctuation or quotation
- Start/end with a famous quote
- Start/end with a summary statement

Literary Techniques:
Description: Using specific literary techniques to enhance the text, including metaphors or similes, alliteration or assonance, hyperbole or understatement, irony or sarcasm, and personification or onomatopoeia.
Example:

- Use of metaphors or similes
- Use of alliteration or assonance
- Use of hyperbole or understatement
- Use of irony or sarcasm
- Use of personification or onomatopoeia

Output Formatting:
Description: Ensuring the text is output in a specified format, such as a table or list, using a specific font or color, in a specific file format (e.g., PDF, CSV), in a certain structure (e.g., JSON, XML), or in a particular layout (e.g., grid, list).
Example:

- Output in a specific format (e.g., table, list)
- Output in a specific font or color
- Output in a specific file format (e.g., PDF, CSV)
- Output in a specific structure (e.g., JSON, XML)
- Output in a specific layout (e.g., grid, list)

Perspective Constraints:
Description: Ensuring the text is written from a specific narrative perspective, such as strictly first-person, second-person, or third-person, alternating perspectives in different sections, using an omniscient or limited viewpoint, and avoiding shifts in perspective mid-paragraph.
Example:

- Write strictly from a first-person, second-person, or third-person perspective
- Alternate perspectives in different sections
- Use an omniscient or limited viewpoint
- Avoid shifting perspectives mid-paragraph

D Detailed Experimental Results

Table 8 shows the detailed experimental results across IFEval, FollowBench and InFoBench.

Table 8: Complete results on two instruction-following benchmarks: IFEval, FollowBench and InFoBench.

Model	IFEval				FollowBench							InFoBench		
	Pr. (S)	Ins. (S)	Pr. (L)	Ins. (L)	L1	L2	L3	L4	L5	HSR-Avg	SSR-Avg	Easy	Hard	Overall
GPT-3.5-Turbo-1106	60.4	69.5	63.8	72.8	80.3	68.0	68.6	61.1	53.2	66.2	72.5	90.4	85.1	86.7
GPT-4-1106-Preview	76.9	83.6	79.3	85.3	84.7	75.6	70.8	73.9	61.9	73.4	77.2	90.1	89.1	89.4
GPT-4o-2024-0513	81.1	86.7	85.4	89.6	87.2	77.8	73.4	74.9	70.2	76.7	79.4	89.2	92.1	90.7
GLM-4-0520	79.1	85.0	83.7	88.7	82.1	73.7	70.5	65.7	60.5	70.5	75.3	85.7	87.8	87.1
Qwen2-7B(LMSYS-Chat)	37.9	48.8	39.2	50.2	61.2	53.9	37.6	27.8	26.0	41.3	54.3	77.5	75.7	76.3
Qwen2-7B-Instruct	50.8	60.9	55.3	64.6	76.5	63.3	58.2	42.0	37.7	55.5	63.7	83.3	81.0	81.8
AutoIF-Qwen2-7B-DPO	44.0	55.0	46.6	57.9	-	-	-	-	-	-	56.6	-	-	-
FiGV-Qwen2-7B-SFT	64.9	74.3	69.9	78.7	73.1	65.4	57.3	42.1	40.6	55.7	63.2	84.3	82.0	82.7
FiGV-Qwen2-7B-DPO	67.5	77.0	71.7	80.5	72.2	70.8	53.2	47.8	41.0	57.0	65.1	84.6	83.7	84.0
LLaMA3-8B(LMSYS-Chat)	42.9	52.2	44.0	53.3	62.1	52.0	39.6	29.0	24.8	41.5	56.1	78.9	74.3	75.7
LLaMA3-8B-Instruct	69.9	78.2	77.6	84.4	75.9	69.1	59.5	49.8	42.6	59.4	67.3	83.4	84.0	83.8
AutoIF-LLaMA3-8B-DPO	28.8	42.4	43.1	56.0	-	-	-	-	-	-	49.9	-	-	-
FiGV-LLaMA3-8B-SFT	67.7	76.7	72.6	80.5	72.4	70.4	59.2	44.1	42.8	57.8	67.0	80.5	80.0	80.2
FiGV-LLaMA3-8B-DPO	74.1	81.5	77.1	84.1	75.5	72.1	59.9	49.2	45.7	60.5	67.4	82.5	81.9	82.3
GLM4-9B(LMSYS-Chat)	41.3	52.2	42.3	53.1	62.1	54.8	42.9	32.8	25.1	43.5	57.9	76.4	74.8	75.3
GLM4-9B-Chat	69.7	77.8	71.0	79.1	76.2	67.8	56.8	51.4	45.3	59.5	66.9	82.3	81.7	81.9
FiGV-GLM4-9B-SFT	67.1	76.3	70.4	79.0	74.9	69.1	61.0	49.8	37.5	58.5	66.7	83.8	81.7	82.2
FiGV-GLM4-9B-DPO	73.9	81.2	77.3	83.8	74.5	73.2	62.5	51.1	46.1	61.5	69.3	85.4	84.1	84.5
Qwen2-72B-Instruct	77.1	80.5	84.3	86.9	84.3	73.7	67.8	61.8	57.2	68.9	73.2	85.2	85.0	85.0
AutoIF-Qwen2-72B-Instruct-DPO	80.2	86.1	82.3	88.0	-	-	-	-	-	-	67.5	-	-	-
FiGV-Qwen2-72B-SFT	78.6	84.7	82.6	87.9	80.3	69.5	62.5	57.1	55.1	64.9	69.8	87.4	87.3	87.4
FiGV-Qwen2-72B-DPO	81.0	85.4	84.5	88.3	82.3	71.0	67.5	58.7	56.0	67.1	72.5	89.6	89.0	89.4
LLaMA3-70B-Instruct	77.6	84.4	84.8	89.6	75.7	71.4	60.4	61.9	54.3	64.7	69.0	87.5	88.1	88.0
AutoIF-LLaMA3-70B-Instruct-DPO	80.2	86.7	85.6	90.4	-	-	-	-	-	-	66.5	-	-	-
FiGV-LLaMA3-70B-SFT	77.3	83.6	82.7	86.3	74.6	72.0	66.3	49.6	53.3	63.2	68.9	85.2	85.8	85.6
FiGV-LLaMA3-70B-DPO	81.4	86.2	85.9	90.7	76.0	71.2	60.8	55.4	61.1	64.9	69.1	89.2	88.9	89.0