A Multi-Dimensional Constraint Framework for Evaluating and Improving Instruction Following in Large Language Models

Junjie Ye[1*], **Caishuang Huang**[1*], **Zhuohan Chen**[1], **Wenjie Fu**[1], **Chenyuan Yang**[1], **Leyi Yang**[1], **Yilong Wu**[1], **Peng Wang**[3], **Meng Zhou**[4], **Xiaolong Yang**[4], **Tao Gui**[2], **Qi Zhang**[1], **Zhongchao Shi**[3], **Jianping Fan**[3], **Xuanjing Huang**[1]

[1] School of Computer Science, Fudan University

[2] Institute of Modern Languages and Linguistics, Fudan University

[3] Lenovo Research [4] Tencent

jjye23@m.fudan.edu.cn, {qz, tgui}@fudan.edu.cn

Abstract

Instruction following evaluates large language models (LLMs) on their ability to generate outputs that adhere to user-defined constraints. However, existing benchmarks often rely on templated constraint prompts, which lack the diversity of real-world usage and limit fine-grained performance assessment. To fill this gap, we propose a multi-dimensional constraint framework encompassing three constraint patterns, four constraint categories, and four difficulty levels. Building on this framework, we develop an automated instruction generation pipeline that performs constraint expansion, conflict detection, and instruction rewriting, yielding 1,200 code-verifiable instruction-following test samples. We evaluate 19 LLMs across seven model families and uncover substantial variation in performance across constraint forms. For instance, average performance drops from 77.67% at Level I to 32.96% at Level IV. Furthermore, we demonstrate the utility of our approach by using it to generate data for reinforcement learning, achieving substantial gains in instruction following without degrading general performance. In-depth analysis indicates that these gains stem primarily from modifications in the model's attention modules parameters, which enhance constraint recognition and adherence. Code and data are available in https://github.com/Junjie-Ye/MulDimIF.

1 Introduction

Instruction following is a fundamental capability of large language models (LLMs) (Bai et al., 2022; OpenAI, 2023; Reid et al., 2024; Yang et al., 2024), allowing them to generate responses that adhere to user-specified constraints (Zhou et al., 2023; Wen et al., 2024a; Dong et al., 2025). This skill is critical in real-world applications, particularly in agentic and tool-assisted workflows, where outputs must conform to strict formats such as JSON (Xi et al., 2023; Deng et al., 2024; Ye et al., 2024). Even minor deviations can lead to parsing failures and system breakdowns (Ye et al., 2025).

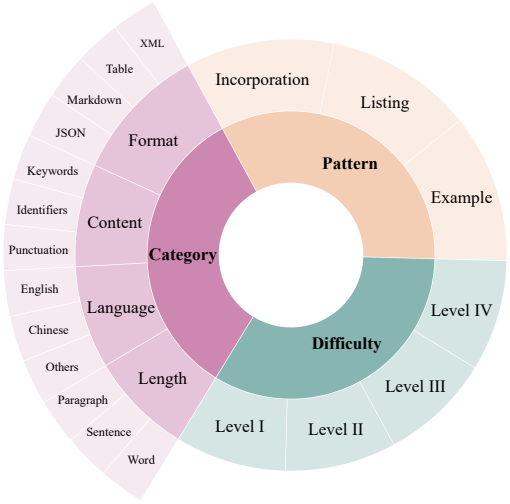


Figure 1: The hierarchical structure of the multi-dimensional constraint framework, which includes three constraint patterns, four constraint categories (subdivided into thirteen subcategories), and four levels of constraint difficulty.

Existing studies have investigated instruction following in LLMs by categorizing constraints (Zhou et al., 2023), crafting targeted prompts (He et al., 2024b), and evaluating model outputs using both code-based and model-based metrics (Jiang et al., 2024). Techniques such as tree search (Cheng et al., 2024) and reinforcement learning (RL) (He et al., 2024a) have also been used to improve instruction-following ability.

Despite these advances, current approaches suffer from several notable limitations. Most prominently, these benchmarks rely on rigid, predefined templates (Zhou et al., 2023; Jing et al., 2023; Wen et al., 2024b) that fail to capture the natural variability in how users express constraints. In addition, many evaluations use LLMs as

[*] Equal contributions.

judges, introducing model-induced biases (Jiang et al., 2024; Sun et al., 2024; Qin et al., 2024). Furthermore, while advanced techniques can boost instruction-following performance, there is limited analysis of why these improvements occur, limiting both interpretability and generalization (Zhang et al., 2024; Cheng et al., 2024; Dong et al., 2024).

To address this gap, we propose a multi-dimensional constraint framework that captures the diverse ways users specify constraints when interacting with LLMs. Illustrated in Figure 1, our framework enables fine-grained analysis by introducing three distinct constraint patterns including example, listing, and incorporation. It organizes constraints into four primary categories which are content, language, format, and length. These categories are further divided into thirteen specific subcategories. Additionally, the framework defines a four-level difficulty scale based on the number of constraints per instruction. Building on this framework, we develop an automated pipeline. The pipeline includes steps for constraint expansion, conflict detection, and instruction rewriting, transforming any initial instruction into one with code-verifiable constraints. Using this pipeline, we construct a dataset of 1,200 diverse instruction-following cases based on ShareGPT[1], allowing controlled model evaluation.

We evaluate 14 LLMs across seven model families and uncover significant variation in their ability to follow different forms of constraints. Notably, while models perform well on example pattern, they struggle with listing and incorporation patterns, emphasizing the challenges posed by complex instructions and the benefits of few-shot prompting (Brown et al., 2020). On average, performance declines from 77.67% at Level I to 32.96% at Level IV, with even the best model scoring only 67.50% overall.

To improve this, we reuse our pipeline to generate a new batch of constraint-based instructions and train the models using the GRPO (Shao et al., 2024) algorithm. Post-training, models demonstrate significantly enhanced instruction-following ability without sacrificing general performance. Parameter-level analysis and case studies reveal that most improvements stem from updates in attention modules, which appear to better align the model's focus with the given constraints.

Our contributions are summarized as follows: 1) We propose a multi-dimensional constraint framework that captures diverse constraint forms and enables fine-grained evaluation; 2) We design an automated instruction generation pipeline that transforms raw instructions into constraint-rich prompts; 3) We construct a diverse benchmark dataset containing 1,200 test cases for evaluating instruction following in 14 LLMs, and improve model performance via targeted training; and 4) We perform in-depth parameter-level analysis and show that instruction following improvements primarily arise from changes in attention mechanisms.

[1] https://sharegpt.com/

2 Related Works

Evaluation of Instruction Following Evaluating the instruction-following capabilities of LLMs has become a central focus in recent research. Benchmarks such as IFEval (Zhou et al., 2023), FollowEval (Jing et al., 2023), and FollowBench (Jiang et al., 2024) assess models on dimensions like logical reasoning and stylistic consistency, using either code-based or LLM-based evaluations. Multi-IF (He et al., 2024b) extends this to multilingual, multi-turn dialogue settings, while InfoBench (Qin et al., 2024) decomposes complex instructions into simpler subtasks to evaluate execution accuracy. CIF-Bench (Li et al., 2024) focuses on the generalization abilities of Chinese LLMs under zero-shot scenarios. Despite their breadth, many of these benchmarks rely on templated or highly constrained prompts, which limits their ability to capture real-world instruction diversity and support fine-grained evaluation. Our work addresses these limitations by introducing a multi-dimensional constraint framework comprising three constraint patterns, four constraint categories, and four difficulty levels. Built on this framework, we develop an automated instruction generation pipeline that enhances diversity and complexity through constraint expansion, conflict detection, and instruction rewriting.

Training of Instruction Following A range of algorithms have been proposed to improve the instruction-following performance of LLMs. Reinforcement learning approaches such as PPO (Schulman et al., 2017) and DPO (Rafailov et al., 2023) optimize model behavior based on user preferences. IOPO (Zhang et al., 2024) augments this by aggregating question-answer pairs across datasets to enrich preference signals and refine the optimization objective. Coninfer (Sun et al.,

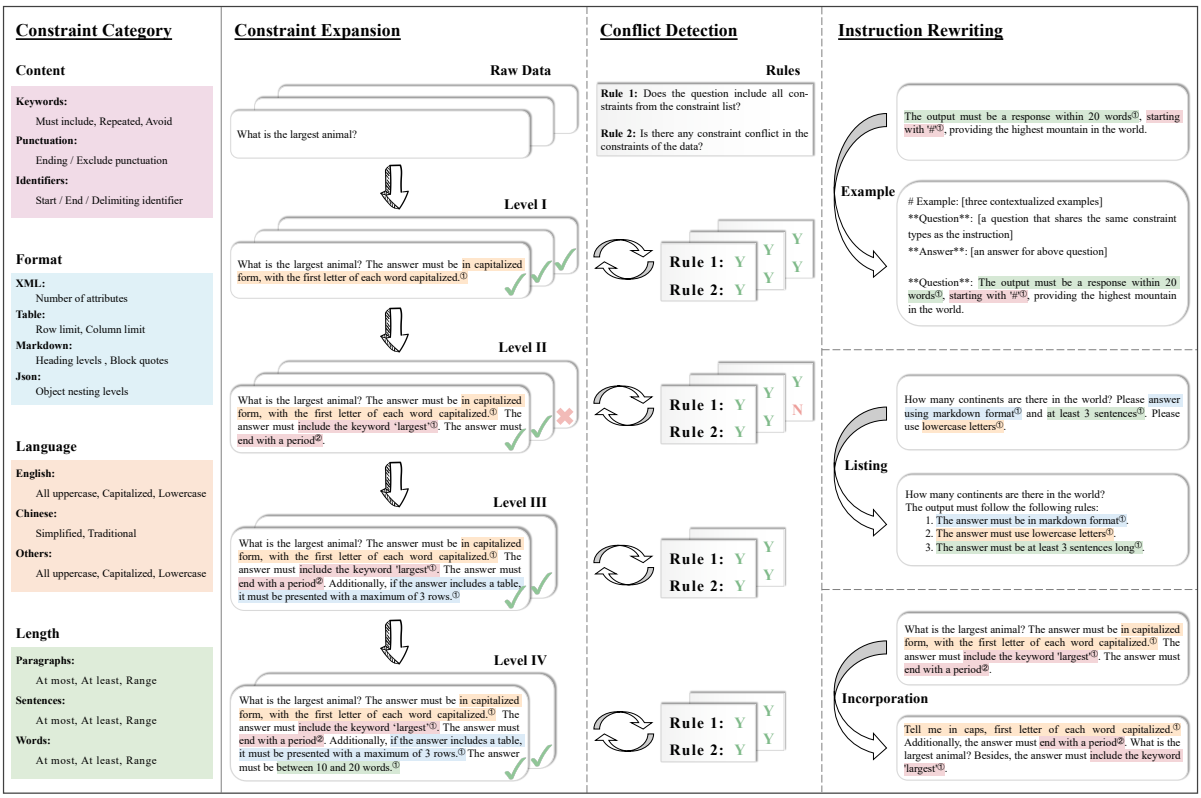


Figure 2: Illustration of the automated instruction generation pipeline. **Constraint Expansion**: Randomly selects a constraint category not yet included in the instruction and adds 1–2 specific constraints. **Conflict Detection**: Identifies whether the new instruction introduces redundant constraints or conflicts, and discards conflicting instructions. **Instruction Rewriting**: Rewrites the remaining instructions based on different constraint patterns.

2024) adopts a curriculum learning approach, incrementally increasing task difficulty during fine-tuning to improve constraint handling. While these methods yield measurable gains, they often lack in-depth analysis of the model characteristics driving these improvements, which limits their interpretability and generalizability. To fill this gap, we introduce a comprehensive data construction pipeline, enabling the creation of high-quality instruction-following datasets. Our parameter-level analysis suggests that a significant portion of the performance improvement arises from tuning the model's attention mechanisms, which enhances its ability to recognize and comply with constraints.

3 Approaches

3.1 Multi-Dimensional Constraint Framework

Existing benchmarks (Zhou et al., 2023; Jing et al., 2023; Jiang et al., 2024) often suffer from a lack of constraint diversity, limiting the breadth of instruction-following capabilities they can evaluate. To address this gap, we propose a multi-dimensional constraint framework, as illustrated in Figure 1.[2]

3.1.1 Constraint Pattern

Drawing inspiration from publicly available guidelines for writing instructions (Saravia, 2022), we identify three common patterns used to introduce constraints during interactions with LLMs.

Example The example pattern involves adding several question-answer pairs that share the same constraint type as the instruction to be followed. This method enhances the model's ability to comply with constraints through contextualized examples, a technique commonly known as in-context learning (Brown et al., 2020).

Listing The listing pattern presents constraints in a clearly structured, point-by-point format. This approach provides explicit communication of constraint requirements, making it especially effective in zero-shot scenarios.

Incorporation The incorporation pattern integrates constraints directly into the instruction, rather than listing them separately. While this

[2]Examples are available in Appendix A.

Data	Constraint Pattern			Constraint Category				Constraint Difficulty				Total
	Example	Listing	Incorporation	Content	Format	Language	Length	Level I	Level II	Level III	Level IV	
Training	1225	3308	3373	8888	9850	6168	9541	610	1614	2522	3160	7906
Test	400	400	400	1175	1210	694	1101	300	300	300	300	1200

Table 1: Distribution of training and test data constructed on the automated instruction generation pipeline.

approach maintains fluency, it may make it more difficult for LLMs to clearly interpret individual constraint requirements.

3.1.2 Constraint Category

Beyond the method of presentation, the types of constraints can vary widely. To enable fine-grained analysis, we categorize constraints into four main categories with thirteen subcategories.

Content Content constraints restrict the elements present in the model's output. These can be further divided into three subcategories: ensuring the inclusion of specific keywords, adhering to particular punctuation requirements, or referencing predefined identifiers.

Format Format constraints require the output to follow specific structural rules, often necessary for post-processing tasks. Common examples include outputs in XML, Table, Markdown, or JSON.

Language Language constraints specify the language to be used in the output, which is fundamental in translation tasks (Ye et al., 2023). Based on the language type, constraints are classified into English, Chinese, or other languages.

Length Length constraints enforce limits on the output's size. Depending on granularity, these constraints can apply at the paragraph level, sentence level, or word level.

3.1.3 Constraint Difficulty

In addition to type and presentation, the number of constraints also impacts task difficulty. We define four difficulty levels based on the number and variety of constraints present in the instruction.

Level I Level I includes an instruction with a single type of constraint, containing one or two individual constraint elements.

Level II Level II includes an instruction with two types of constraints, comprising a total of two to four individual constraint elements.

Level III Level III includes an instruction with three types of constraints, comprising a total of three to six individual constraint elements.

Level IV Level IV includes an instruction with four types of constraints, comprising a total of four to eight individual constraint elements.

3.2 Automated Instruction Generation Pipeline

Building upon the multi-dimensional constraint framework, we introduce an automated pipeline that transforms raw instructions into constrained versions that can be verified through code, as illustrated in Figure 2.

Constraint Expansion Constraint expansion involves adding new constraints to a given instruction. Specifically, we randomly select a constraint category not yet covered and add one or two specific constraints from that category. This process progressively generates instructions across varying levels of constraint difficulty.

Conflict Detection Conflict detection ensures the soundness of instructions after constraint expansion. It consists of two checks: first, verifying that the newly specified constraints have been correctly incorporated; second, ensuring that the constraints do not conflict with each other (e.g., requiring a sentence to be entirely lowercase while also demanding the presence of uppercase words). If either check fails, the instruction is discarded. Otherwise, it is retained, and constraint expansion continues until difficulty Level IV is reached.

Instruction Rewriting Instruction rewriting enhances instruction diversity by transforming a given instruction to match a specified pattern. Specifically, we randomly select a constraint pattern and rewrite the instruction accordingly. When handling example-based constraints, we uniformly select three question-answer pairs that share the same constraint subcategories as the original instruction to serve as contextual examples.

Using this pipeline, we generate 1,200 distinct test cases and manually write validation code for each data point. This code is used for a detailed analysis of the model's instruction-following capability. The statistical details of the data are presented in Table 1.

Family	Version	Constraint Pattern			Constraint Category				Constraint Difficulty				Overall
		Example	Listing	Incorporation	Content	Format	Language	Length	Level I	Level II	Level III	Level IV	
LLaMA3.1	Instruct-8B	40.25	36.00	32.25	80.13	64.74	44.28	49.60	64.67	40.67	27.33	12.00	36.17
	Instruct-70B	68.00	54.25	48.25	86.62	73.65	80.90	65.73	78.00	61.33	51.33	36.67	56.83
Qwen2.5	Instruct-7B	59.25	45.00	43.75	83.77	54.45	90.30	64.25	82.00	55.00	39.00	21.33	49.33
	Instruct-14B	66.75	52.50	51.25	84.42	62.74	84.95	74.60	82.00	63.67	49.33	32.33	56.83
	Instruct-32B	67.75	57.50	54.25	85.58	69.26	90.16	70.56	84.33	66.67	53.67	34.67	59.83
	Instruct-72B	70.25	60.25	55.25	87.01	70.89	**92.19**	72.98	84.33	66.67	53.67	43.00	61.92
DeepSeek-R1-Distill-LLaMA	Instruct-8B	42.00	28.00	28.00	80.00	50.44	44.72	53.09	59.67	40.33	18.33	12.33	32.67
	Instruct-70B	59.50	64.00	57.50	84.55	83.69	84.80	65.32	77.00	63.00	57.00	44.33	60.33
DeepSeek-R1-Distill-Qwen	Instruct-7B	31.25	26.50	27.50	78.70	50.19	24.60	51.08	61.33	33.33	14.33	4.67	28.42
	Instruct-14B	52.75	38.00	34.25	82.99	66.75	44.86	59.41	70.67	49.67	29.33	17.00	41.67
	Instruct-32B	57.00	51.50	50.50	85.32	78.29	69.18	63.71	76.33	59.33	43.33	33.00	53.00
Gemini1.5	Flash	71.50	61.00	64.50	88.70	80.68	87.84	74.06	86.00	68.00	57.67	51.00	65.67
	Pro	**73.50**	61.75	**65.25**	87.40	81.81	88.28	75.67	**86.67**	70.00	59.00	51.67	66.83
Claude3.5	Haiku	44.25	53.50	52.00	84.29	82.81	49.06	68.01	71.33	51.67	41.67	35.00	49.92
	Sonnet	72.50	**69.00**	61.00	86.62	83.44	84.23	**76.21**	82.67	**71.67**	**60.67**	**55.00**	**67.50**
GPT	3.5-Turbo	49.00	41.25	38.00	84.94	70.39	42.98	66.26	77.33	48.00	30.33	15.33	42.75
	4-Turbo	70.75	59.25	52.25	**89.09**	79.17	77.57	73.25	82.67	68.00	55.00	37.33	60.75
	4o-Mini	67.50	67.00	59.00	85.71	**84.57**	84.37	72.04	84.33	70.00	59.00	44.67	64.50
	4o	70.50	62.50	59.00	87.92	84.44	82.78	69.35	84.33	69.33	57.33	45.00	64.00

Table 2: Results of the evaluation of LLMs' instruction-following ability across different dimensions. 'Overall' denotes the overall score. The best results in each dimension are highlighted in **bold**.

4 Evaluations of LLMs

4.1 Models

We conduct an evaluation of 19 LLMs from seven model families, including four open-source and three closed-source, which collectively represent the current state of LLM capabilities. Among the open-source LLMs, we select *LLaMA3.1-Instruct-8B* and *LLaMA3.1-Instruct-70B* from the **LLaMA3.1 family** (Team, 2024); *Qwen2.5-Instruct-7B*, *Qwen2.5-Instruct-14B*, *Qwen2.5-Instruct-32B*, and *Qwen2.5-Instruct-72B* from the **Qwen2.5 family** (Yang et al., 2024); *DeepSeek-R1-Distill-LLaMA-8B* and *DeepSeek-R1-Distill-LLaMA-70B* from the **DeepSeek-R1-Distill-LLaMA family** (DeepSeek-AI et al., 2025); and *DeepSeek-R1-Distill-Qwen-7B*, *DeepSeek-R1-Distill-Qwen-14B*, *DeepSeek-R1-Distill-Qwen-32B*, and *DeepSeek-R1-Distill-Qwen2.5-Instruct-32B* from the **DeepSeek-R1-Distill-Qwen family** (DeepSeek-AI et al., 2025). For the closed-source LLMs, we include *Gemini1.5-Flash* and *Gemini1.5-Pro* from the **Gemini1.5 family** (Reid et al., 2024); *Claude3.5-Haiku* and *Claude3.5-Sonnet* from the **Claude3.5 family** (Bai et al., 2022); and *GPT-3.5-Turbo*, *GPT-4-Turbo*, *GPT-4o-Mini*, and *GPT-4o* from the **GPT family** (OpenAI, 2023).[3]

4.2 Experimental Setup

For generating instructions, we use GPT-4o for help.[4] To ensure that each model's capabilities are accurately represented, we use the built-in chat template for open-source models and the official API interface for closed-source models.[5] For consistency and reproducibility, we apply greedy decoding across all evaluations.

4.3 Main Results

Table 2 summarizes the results of our multi-dimensional evaluation of various LLMs. Several key findings emerge from this analysis.

Current LLMs vary substantially in their ability to follow different constraint forms. Most models perform best on the Example pattern, underscoring the efficacy of in-context learning (Min et al., 2022). In contrast, performance consistently declines on the Incorporation pattern, suggesting that following constraints in free-form remains a significant challenge. Additionally, average accuracy drops sharply from 77.67% at Level I to just 32.96% at Level IV, indicating that current models struggle to handle scenarios involving multiple or more complex constraints.

Instruction-following performance generally improves with increasing model size. Within most families, larger models exhibit better adherence to instructions, particularly in more demanding settings such as length-constrained or Level IV scenarios. This trend aligns with existing findings on scaling laws in LLMs (Kaplan et al., 2020; Chung et al., 2022). However, the GPT family deviates from this pattern. In certain settings, GPT-4o underperforms GPT-4o-Mini. This anomaly may reflect an alignment tax (Ouyang et al., 2022),

[3]More information can be found in Appendix C.

[4]The prompts can be found in Appendix B.

[5]Chat template for each LLM can be found in Appendix D.

Dataset	Capability	# Number
Training		
Ours	Instruction Following	7906
Test		
Ours	Instruction Following	1200
IFEval	Instruction Following	541
Multi-IF	Instruction Following	13447
MMLU	Knowledge	14042
GSM8K	Reasoning	1319
MATH	Reasoning	5000
HumanEval	Coding	164
MBPP	Coding	257

Table 3: Overview of the training and test datasets used when improving instruction-following capabilities.

where optimization for broader capabilities comes at the cost of precise instruction-following.

Stronger reasoning ability does not guarantee better instruction following. Although DeepSeek-R1-Distill-LLaMA and DeepSeek-R1-Distill-Qwen outperform LLaMA3.1 and Qwen2.5 in reasoning-focused benchmarks (DeepSeek-AI et al., 2025), their instruction-following performance is notably weaker. Closer examination reveals a disconnect between reasoning and answer: these models often identify the correct constraints in the reasoning processes but fail to implement them in answers. This gap highlights a pressing need for training methods that better integrate reasoning processes with instruction execution.

5 Improvements of LLMs

Building on the framework described in Section 3.1, we construct training data tailored for RL to improve the instruction-following capabilities of LLMs. Our approach enhances these capabilities while preserving general performance. Analysis suggests that performance gains stem primarily from updates to attention modules, increasing the model's sensitivity to task-specific constraints.

5.1 Dataset

Training We generate 7,906 single-turn, constraint-based instructions paired with verifiable code using the data generation pipeline described in Section 3.2. Due to the difficulty of reliably constructing golden answers, this dataset is used exclusively for RL.[6]

Test To evaluate model performance post-RL, we focus on four capabilities: 1) **Instruction Following.** We evaluate on our test set, along with two out-of-domain datasets: IFEval and Multi-IF, which includes multi-turn dialogue-based instructions. 2) **Knowledge.** General knowledge is assessed using MMLU (Hendrycks et al., 2021a). 3) **Reasoning.** We use GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b) to measure logical and mathematical reasoning. and 4) **Coding.** Programming ability is evaluated with HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). [7]

5.2 Experimental Setup

We conduct experiments on six LLMs without more than 14 billion parameters, applying the GRPO algorithm for RL. The setup includes a batch size of 1,024, mini-batch size of 512, 32 rollouts per update, and a learning rate of 1e-6. We use a sampling temperature of 0.7 and set the maximum output length to 8,192 tokens. The reward function is defined as the number of constraints satisfied in the output. Training is performed for one epoch on eight A800 GPUs. For evaluation, we apply each model's official chat template and use greedy decoding for consistency and reproducibility.

5.3 Results and Analysis

Performance Improvements Figure 3 presents the performance of individual LLMs before and after applying GRPO.[8] The results demonstrate substantial performance gains on our custom test set, with LLaMA3.1-Instruct-8B notably outperforming other models. Importantly, these improvements extend to out-of-domain instruction-following benchmarks and significantly enhance performance in multi-turn dialogue scenarios (i.e., Multi-IF), despite training being conducted solely on single-turn data. This suggests that the data generated by our multi-dimensional constraint framework exhibits strong generalization ability. Furthermore, although our training is focused on improving instruction-following capabilities, it does not degrade general-purpose performance. On general benchmarks, post-GRPO LLMs maintain parity with their original counterparts and, in some cases (e.g., MBPP), show clear improvements. These findings indicate that our pipeline produces data that is both compatible with and complementary to existing training corpora, enabling straightforward performance enhancements when integrated into current LLMs.

[6]The full distribution of training data is shown in Table 1.

[7]The information of the data is shown in Table 3.

[8]Detailed results are provided in Appendix E.

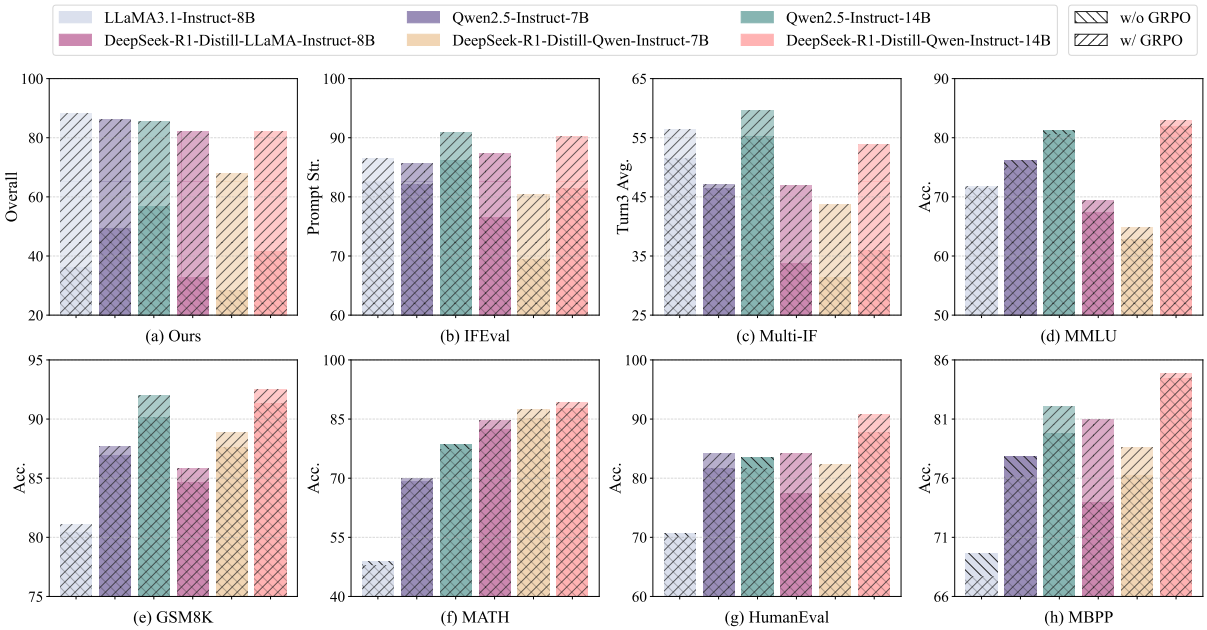
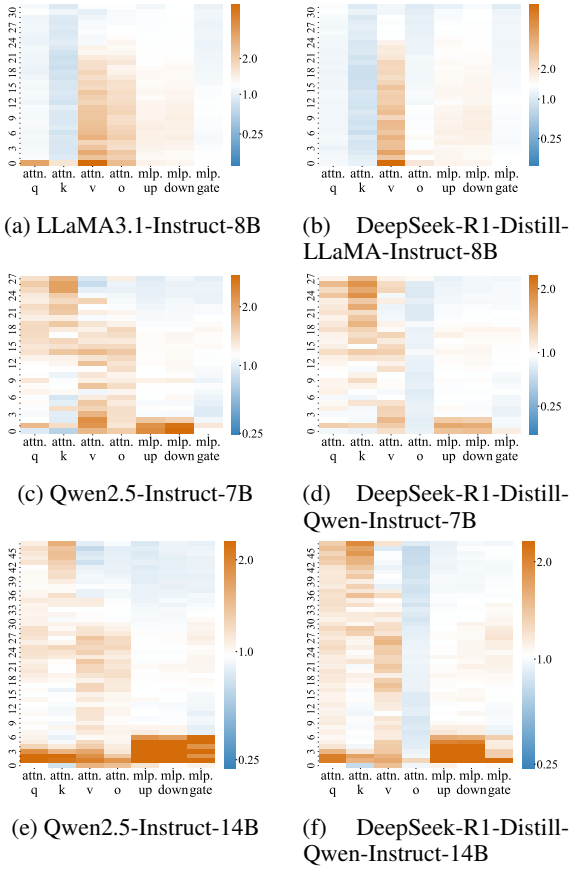


Figure 3: Performance comparison of each LLM on the test sets before and after applying GRPO.



(a) LLaMA3.1-Instruct-8B (b) DeepSeek-R1-Distill-LLaMA-Instruct-8B

(c) Qwen2.5-Instruct-7B (d) DeepSeek-R1-Distill-Qwen-Instruct-7B

(e) Qwen2.5-Instruct-14B (f) DeepSeek-R1-Distill-Qwen-Instruct-14B

Figure 4: Parameter change rates of LLMs after GRPO relative to the original ones across different modules.

Parameter-Level Analysis To better understand the sources of performance improvement, we conduct a parameter-level analysis. We compute the relative change rates of model parameters following GRPO, broken down by model modules, and summarize the results in Figure 4. Notably, the most substantial updates occur in the attention modules, suggesting that GRPO primarily refines the model's attention mechanisms. These changes are distributed relatively uniformly across layers, indicating a global rather than localized adjustment. Overall, applying GRPO with our data effectively tunes the model's attention distribution, enhancing its ability to identify and focus on critical input information and thereby improving its instruction-following and general performance.

Case Studies To visualize how modifications in attention mechanisms affect model behavior, we adopt the information flow analysis method proposed by Wang et al. (2023). We compute the importance of each input token with respect to the model's output and present representative visualizations in Table 4. After applying GRPO, the importance of constraint-related tokens increases, while the influence of irrelevant tokens diminishes; the relevance of core problem components remains largely unchanged. This suggests that the model has improved its ability to identify and prioritize constraint-related information without sacrificing its understanding of the overall input. Additionally, the reduced attention to irrelevant tokens may help minimize distraction from non-essential elements, which could explain the observed stability or gains in general task performance. These case studies further validate the effectiveness of our proposed framework and the utility of the data it produces.

Table 4: A visualization showing the importance of each input token to the output, with darker colors indicating greater significance.

6 Conclusion

In this paper, we propose a multi-dimensional constraint framework that categorizes instructions based on constraint pattern, constraint category, and constraint difficulty. Building on this framework, we design an automated instruction generation pipeline that transforms any instruction into a constraint-based one through the processes of constraint expansion, conflict detection, and instruction rewriting. Using this pipeline, we generate 1,200 test cases and conduct a comprehensive evaluation of 19 LLMs across seven different model families. We also construct a constraint-focused training dataset and apply the GRPO algorithm to enhance instruction-following capabilities in LLMs. The results show that models trained with our data achieve notable improvements while maintaining general capabilities. Parameter-level analysis and case studies indicate that these gains primarily stem from increased model sensitivity to constraint-relevant information.

Limitations

Although we evaluate and enhance the instruction-following ability of current LLMs using constructive data and analyze the underlying reasons for their improvement, our work still has two key limitations. On one hand, due to the complexity of answer construction, we do not train models from a pre-trained version, but instead apply GRPO directly to instruction-tuned models. Nevertheless, our results show that GRPO-trained models do not suffer any loss in general capability, and in some cases even demonstrate improvements, highlighting the compatibility of our constructed data with original training data. On the other hand, since our focus is primarily on enhancing instruction-following capabilities, we do not explore the effects of applying our method to domain-specific datasets. However, because our approach can convert any instruction into a constraint-based version, and case studies confirm that the model retains its focus on the core problem components, we believe that applying this method to other domains (e.g., reasoning, coding) can also yield additional performance gains.

References

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. 2021. Program synthesis with large language models. *CoRR*, abs/2108.07732.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, and 32 others. 2022. Constitutional AI: harmlessness from AI feedback. *CoRR*, abs/2212.08073.

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, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Jiale Cheng, Xiao Liu, Cunxiang Wang, Xiaotao Gu, Yida Lu, Dan Zhang, Yuxiao Dong, Jie Tang, Hongning Wang, and Minlie Huang. 2024. Spar: Self-play with tree-search refinement to improve instruction-following in large language models. *CoRR*, abs/2412.11605.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, and 12 others. 2022. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 81 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

Yang Deng, Xuan Zhang, Wenxuan Zhang, Yifei Yuan, See-Kiong Ng, and Tat-Seng Chua. 2024. On the multi-turn instruction following for conversational web agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 8795–8812. Association for Computational Linguistics.

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. *CoRR*, abs/2406.13542.

Guanting Dong, Xiaoshuai Song, Yutao Zhu, Runqi Qiao, Zhicheng Dou, and Ji-Rong Wen. 2025. Toward verifiable instruction-following alignment for retrieval augmented generation. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 23796–23804. AAAI Press.

Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. 2024a. 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, Miami, Florida, USA, November 12-16, 2024*, pages 10864–10882. Association for Computational Linguistics.

Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, Shruti Bhosale, Chenguang Zhu, Karthik Abinav Sankararaman, Eryk Helenowski, Melanie Kambadur, Aditya Tayade, Hao Ma, Han Fang, and Sinong Wang. 2024b. Multi-if: Benchmarking llms on multi-turn and multilingual instructions following. *CoRR*, abs/2410.15553.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Comput.*, 3(1):79–87.

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), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 4667–4688. Association for Computational Linguistics.

Yimin Jing, Renren Jin, Jiahao Hu, Huishi Qiu, Xiaohua Wang, Peng Wang, and Deyi Xiong. 2023. Followeval: A multi-dimensional benchmark for assessing the instruction-following capability of large language models. *CoRR*, abs/2311.09829.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Yizhi Li, Ge Zhang, Xingwei Qu, Jiali Li, Zhaoqun Li, Noah Wang, Hao Li, Ruibin Yuan, Yinghao Ma, Kai Zhang, Wangchunshu Zhou, Yiming Liang, Lei Zhang, Lei Ma, Jiajun Zhang, Zuowen Li, Wenhao Huang, Chenghua Lin, and Jie Fu. 2024. Cif-bench: A chinese instruction-following benchmark for evaluating the generalizability of large language models. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 12431–12446. Association for Computational Linguistics.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11048–11064. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

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, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 13025–13048. Association for Computational Linguistics.

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 *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy P. Lillicrap, Jean-Baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, Ioannis Antonoglou, Rohan Anil, Sebastian Borgeaud, Andrew M. Dai, Katie Millican, Ethan Dyer, Mia Glaese, Thibault Sottiaux, Benjamin Lee, and 34 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *CoRR*, abs/2403.05530.

Elvis Saravia. 2022. Prompt Engineering Guide. *https://github.com/dair-ai/Prompt-Engineering-Guide*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300.

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. *CoRR*, abs/2404.02823.

Meta Team. 2024. Introducing llama 3.1: Our most capable models to date.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Label words are anchors: An information flow perspective for understanding in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9840–9855. Association for Computational Linguistics.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxing Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. 2024a. Benchmarking complex instruction-following with multiple constraints composition. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxing Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. 2024b. Benchmarking complex instruction-following with multiple constraints composition. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, and 10 others. 2023. The rise and potential of large language model based agents: A survey. *CoRR*, abs/2309.07864.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, Jie Zhou, Siming Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. A comprehensive capability analysis of GPT-3 and GPT-3.5 series models. *CoRR*, abs/2303.10420.

Junjie Ye, Guanyu Li, Songyang Gao, Caishuang Huang, Yilong Wu, Sixian Li, Xiaoran Fan, Shihan Dou, Tao Ji, Qi Zhang, Tao Gui, and Xuanjing Huang. 2025. Tooleyes: Fine-grained evaluation for tool learning capabilities of large language models in real-world scenarios. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pages 156–187. Association for Computational Linguistics.

Junjie Ye, Yilong Wu, Sixian Li, Yuming Yang, Tao Gui, Qi Zhang, Xuanjing Huang, Peng Wang, Zhongchao Shi, Jianping Fan, and Zhengyin Du. 2024. Tl-training: A task-feature-based framework for training large language models in tool use. *CoRR*, abs/2412.15495.

Xinghua Zhang, Haiyang Yu, Cheng Fu, Fei Huang, and Yongbin Li. 2024. IOPO: empowering llms with complex instruction following via input-output preference optimization. *CoRR*, abs/2411.06208.

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

A Examples for Different Forms of Constraints

To illustrate the instructions generated by our method, we present a selection in Table 5, annotated with their corresponding dimension information. These examples demonstrate that our approach overcomes the templating limitations of prior methods and better captures the diversity of real-world needs. Additionally, the multi-dimensional categorization enables a more fine-grained performance analysis.

Instruction	Pattern	Category	Difficulty
# Example 1: **Question**: What Is The Room? The answer must use capitalized letters for each word. **Answer**: The Room Is A Living Room. # Example 2: **Question**: Describe life in 1980. The answer must use capitalized letters for each word. **Answer**: Life In 1980 Was Characterized By The Rise Of Personal Computers, Vibrant Music Scenes With Genres Like Disco And Rock, And Fashion Trends Featuring Bold Colors And Styles. People Enjoyed Watching Movies On VHS Tapes And Listening To Music On Cassette Players. # Example 3: **Question**: Can I insert PDFs or documents? The answer must use capitalized letters for each word. **Answer**: Yes, You Can Insert PDFs Or Documents. **Question**: Write a script about Matt Rhule's 0-3 record so far. The answer must use capitalized letters for each word.	Example	Language	Level I
# Example 1: **Question**: What are the best books to learn Cybersecurity in 2023? The answer must use capitalized letters for each word and must include the keyword 'Cybersecurity'. **Answer**: "Cybersecurity For Beginners" And "The Art Of Invisibility" Are Among The Best Books To Learn Cybersecurity In 2023. # Example 2: **Question**: Hi chat GPT4, how many questions can I ask from you when I have a free account here? The answer must be in all uppercase letters and must include the keyword 'LIMIT'. **Answer**: THE LIMIT IS 50 QUESTIONS PER MONTH. # Example 3: **Question**: I've read about a female wrestler using a move called "Kegel Crush". What kind of move could this be? It sounds like a creative name for a submission move. The answer must use capitalized letters for each word and must include the keyword 'submission'. **Answer**: The Kegel Crush Could Be A Unique Submission Move Where The Wrestler Applies Pressure To The Opponent's Core Muscles, Forcing A Submission Through Intense Abdominal Constriction. **Question**: I want you to create a detailed curriculum for mastering each of the skills listed below. Divide each skill into multiple sub-topics and list the skills required for each sub-topic and suggest the best online courses and books for each sub-topic. Make sure it should be free. Additionally, the answer must use capitalized letters for each word and must include the keyword 'Curriculum Design'.	Example	Content Language	Level II

Instruction	Pattern	Category	Difficulty
# Example 1: **Question**: What are the best books to learn Java in 2023? Please ensure that the answer uses capitalized letters for each word, is at most 50 words long, and includes the keyword 'Java Programming'. **Answer**: Effective Java By Joshua Bloch, Java: The Complete Reference By Herbert Schildt, And Head First Java By Kathy Sierra Are Excellent Books For Learning Java Programming In 2023. # Example 2: **Question**: What is the ice level in Plants vs Zombies mini-games? The answer must use capitalized letters for each word, be at most 20 words, and include the keyword 'Plants vs Zombies'. **Answer**: The Ice Level In Plants Vs Zombies Mini-Games Is Called Icebound Battleground. # Example 3: **Question**: Please provide a detailed explanation of performance evaluation by competencies and the S20/23 evaluation cited in APA standards that is specific to Ecuador. The response must be at least 150 words to ensure a comprehensive understanding of the topic. Additionally, the answer must use capitalized letters for each word. Furthermore, the explanation must include the keyword 'Ecuadorian Educational Standards' to ensure relevance to the local context. **Answer**: Performance Evaluation By Competencies In Ecuador Focuses On Assessing Individuals Based On Specific Skills And Abilities Relevant To Their Roles. This Approach Aligns With The Ecuadorian Educational Standards, Which Emphasize Competency-Based Learning To Ensure Students Acquire Practical Skills. The S20/23 Evaluation, Cited In APA Standards, Is A Framework Used To Measure These Competencies Through A Structured Assessment Process. In Ecuador, This Evaluation Is Adapted To Reflect Local Educational Goals And Cultural Contexts, Ensuring That Students Meet The National Standards. The Ecuadorian Educational Standards Play A Crucial Role In Shaping The Criteria For These Evaluations, Ensuring That They Are Relevant And Comprehensive. By Focusing On Competencies, The Evaluation Aims To Provide A Holistic View Of A Student's Abilities, Preparing Them For Future Challenges. This Methodology Is Integral To The Educational System In Ecuador, As It Promotes Continuous Improvement And Alignment With Global Educational Practices. **Question**: Can you provide a list of airlines that have branded fares? Please ensure that the answer is at most 50 words long, making it concise and to the point. Additionally, the answer must use capitalized letters for each word. The answer must also include the keyword 'branded fares.'	Example	Content Language Length	Level III

Instruction	Pattern	Category	Difficulty
# Example 1: **Question**: Provide a definition for the name “Patrick.” The answer should be concise, using at most 50 words, and must use capitalized letters for each word. Additionally, the answer must include the keyword ‘Saint’ to reflect the historical and cultural significance associated with the name. Furthermore, the answer must be formatted with a level 2 heading in Markdown. **Answer**: ## Patrick: A Name Historically Linked To Saint Patrick, The Patron Saint Of Ireland, Known For Spreading Christianity And Celebrated On Saint Patrick’s Day. # Example 2: **Question**: What type of energy do herbivores and omnivores gain when they eat plants? The answer must be at most 5 words long, use capitalized letters for each word, include the keyword ‘Energy’, and be formatted as a level 2 heading in Markdown. **Answer**: ## Chemical Energy From Plants # Example 3: **Question**: What is choline? Please provide your answer in at most 3 sentences, and ensure that each word in your response is capitalized. Additionally, the answer must include the keyword ‘nutrient’. Furthermore, the answer must include a level 2 heading formatted in Markdown. **Answer**: ## What Is Choline? Choline Is An Essential Nutrient That Supports Various Bodily Functions, Including Brain Development And Liver Function. It Plays A Crucial Role In The Synthesis Of Phospholipids, Which Are Vital Components Of Cell Membranes. **Question**: Format your response using markdown, ensuring the use of at least two heading levels, subheadings, bullet points, and bold to organize the information. List some conjugations for “to read” in Mandarin Chinese. The response must be concise, with a maximum of 150 words, and must include the conjugations in Traditional Chinese characters. Additionally, ensure that the response ends with a period.	Example	Content Format Language Length	Level IV

Instruction	Pattern	Category	Difficulty
Which 10 countries have the highest number of golf courses relative to their population size? The output must follow the following rules: 1. Use at most 50 words. 2. Provide the information in at least 3 sentences.	Listing	Length	Level I
Can you provide the schedule for DEF CON 11? The output must follow the following rules: 1. The answer must contain at most 50 words. 2. The answer must be presented in a table format with a maximum of 5 rows.	Listing	Format Length	Level II
Could you provide information on some tourist attractions in Norway and suggest the duration of visits for each? The output must follow the following rules: 1. The answer must use at least two heading levels to organize the information. 2. Any table included in the answer must not exceed five rows. 3. The answer must be between 200 and 300 words. 4. The answer must include at least three paragraphs. 5. The answer must use capitalized letters for each word.	Listing	Format Language Length	Level III
Can The Window Size Of A Microsoft Form Be Adjusted To Fit A Mobile Screen? The output must follow the following rules: 1. The answer must use capitalized letters for each word. 2. The answer must be no more than two sentences. 3. The answer must include the keyword 'responsive design.' 4. The answer must be formatted using a level 2 heading in Markdown.	Listing	Content Format Language Length	Level IV
Where can I find GPT-4 for free? When answering, it is essential to include the keyword 'GPT-4 access' as part of the response.	Incorporation	Content	Level I
How can I reverse engineer Geometry Dash? When answering, it is essential to include a JSON object that is structured with at least three levels of nesting. This requirement ensures that the reverse engineering process is detailed and comprehensive. Additionally, the explanation must incorporate the keyword 'decompilation' to emphasize this critical aspect of the reverse engineering process.	Incorporation	Content Format	Level II
Please list all Roman Emperors by length of reign, ensuring that the context is clear by including the keyword 'Roman Empire' in your answer. The response should be in English, with the first letter of each word capitalized. Additionally, present the information in a table format, adhering to a column limit of 3. The table should include columns for 'Emperor Name', 'Length of Reign', and 'Additional Information'.	Incorporation	Content Format Language	Level III
Who Was Ida B. Wells? In Your Response, It Is Important To Use Block Quotes To Highlight Key Quotes Or Important Points About Her Life And Contributions, As This Will Help Emphasize Her Impact. Additionally, Ensure That Your Response Contains At Least 150 Words To Provide A Comprehensive Overview Of Her Achievements And Influence. The Response Must Also Include The Keyword 'Civil Rights' To Highlight Her Significant Role In The Movement. Furthermore, The Response Should Use Capitalized Letters For Each Word To Maintain A Consistent Style Throughout The Text.	Incorporation	Content Format Language Length	Level IV

Table 5: Examples of instructions generated by the proposed approach.

B Prompts for Instruction Generation

Our automated instruction generation pipeline includes constraint expansion, conflict detection, and instruction rewriting, with several steps utilizing GPT-4o. The corresponding prompts are listed below. Some of the information used in the above prompts is presented in Table 6 and Table 7.

```
constraint_expand_prompt = '''
You are an expert in instruction-following data construction. Your task is to
↪  generate corresponding data as required.

You must carefully analyze and select specific constraints from the [New Constraint
↪  List]. Then, based on the original question in the provided [Data], generate
↪  new data that adheres to the [Data Generation Requirements]. Finally, respond
↪  in the format specified in the [Response Format].

[New Constraint List]: {new_constrint_list}

[Data Generation Requirements]:

    [Core Requirements]:

        1. Ensure only {c1} added, that is, {c2}. The word following [Main Category]
        ↪  should be the main category.

        2. Based on this analysis, select {c3} from the [New Constraint List] and
        ↪  construct an appropriate "Specific Constraint Content". Add it to the
        ↪  [Original Constraint List] in the provided data, and return the
        ↪  [Updated Constraint List].

        3. Modify the content of the [Original Question] in the provided data to
        ↪  **explicitly and clearly specify all the constraints** in the [Updated
        ↪  Constraint List]. The modified question must clearly describe each
        ↪  constraint in natural language, ensuring that the constraints are fully
        ↪  integrated into the question text. For example:
           - Original Question: "Tell me about machine learning."
           - Constraint: "The answer must use capitalized letters for each word."
           - Modified Question: "Tell me about machine learning. The answer must use
           ↪  capitalized letters for each word."

        4. Ensure that the Specific Constraint in each constraint triplet is
        ↪  detailed and specific, containing concrete information or examples
        ↪  (e.g., instead of "Must include", specify "Must include the keyword
        ↪  'machine learning'").

    [Notes]:

        1. The new constraint cannot conflict with the constraints in the [Original
        ↪  Constraint List].
```

```
        2. The modified [Question with the New Constraint] must **explicitly
        ↪  describe all the constraints** in natural language, ensuring that the
        ↪  constraints are fully integrated into the question text. Constraints
        ↪  should not be implicitly applied to the answer without being explicitly
        ↪  stated in the question.

        3. Make sure the Specific Constraint in each constraint triplet is as
        ↪  specific as possible, including concrete details or examples.

        4. **Important**: The response must strictly follow the [Response Format]
        ↪  exactly as specified. Do not include any numbering, bullet points, or
        ↪  additional formatting. The [Updated Constraint List] must be outputted
        ↪  as a single list of tuples in the exact format shown, without any
        ↪  additional characters or line breaks between the tuples.

        5. When generating the modified [Question with the New Constraint], ensure
        ↪  that the language is natural and well-polished. Enrich the phrasing of
        ↪  constraints to avoid redundancy and monotony.

[Response Format]:

    [Thinking Process]: xxx

    [Updated Constraint List]: [(Main Category, Subcategory, Specific Constraint),
    ↪  (Main Category, Subcategory, Specific Constraint), ...]  (The main category
    ↪  is the word after [Main Category], and the constraints we provide are just
    ↪  broad scopes. You need to find suitable specific constraints based on the
    ↪  question and its answers. The Specific Constraint should be detailed and
    ↪  specific.)

    [Question with the New Constraint]: xxx

[Data]:

    [Original Constraint List]: [{original_constraint_list}]

    [Original Question]: {original_question}
'''


conflict_detection_prompt = '''
You are an expert in data structure following instructions. You need to perform a
↪  series of checks on the given [Data] according to the [Check Requirements] and
↪  finally respond in the format specified in the [Response Format].

[Check Requirements]:
    1. Check if there is any constraint conflict in the "Constraint List" in the
    ↪  provided data. Explain first and then conclude.
    2. Check if the "Question" in the provided data clearly specifies all the
    ↪  constraint requirements in the "Constraint List". Explain first and then
    ↪  conclude.
    3. The response format should follow the requirements specified in the [Response
    ↪  Format] below.
```

```
[Response Format]:
    # Constraint Conflict Check #
    [Specific Explanation]:
    [Is there any constraint conflict in the constraints of the data]: [Yes/No]

    # Does the Question clearly specify all constraints in the Constraint List Check
    ↪  #
    [Specific Explanation]: [Explanation]
    [Does the question include all constraints from the constraint list]: [Yes/No]

[Data]:
    [Constraint List]: [{constraint_list}]
    [Question]: {quetsion}
'''

instruction_rewriting_listing = '''
You are an expert in constructing data based on instructions. You need to generate
↪  the corresponding data as required.
You should modify the given [Original Question] according to the [Core
↪  Requirements] without changing the original meaning of the question. Then,
↪  respond in the format specified in the [Reply Format].

[Core Requirements]:
    1. Fully understand the [Original Question] and the constraints listed in the
    ↪  [Constraint List].
    2. Change the expression of the [Original Question]. First, extract the core
    ↪  question from the [Original Question] that is not bound by constraints,
    ↪  then list the constraints corresponding to the [Constraint List] at the end
    ↪  of the sentence. Start with "The output must follow the following rules:"
    ↪  and list the constraints from the [Original Question] clearly after
    ↪  understanding the constraints.
    3. The modified question must remain consistent with the [Original Question] in
    ↪  terms of meaning and constraints.

[Reply Format]:
    [Constraint List Data]: Core question (does not include constraint descriptions
    ↪  in the constraint list),
    The output must follow the following rules:
    1.xxx 2.xxx

[Data]:
    [Original Question]:{original_question}
    [Constraint List]:{constraint_list}
'''

instruction_rewriting_incorporation = '''
You are an expert in data construction based on instructions. You need to generate
↪  the corresponding data as required.
You should modify the given [Data] according to the [Core Requirements] without
↪  changing the original meaning of the question. Then, respond in the format
↪  specified in the [Reply Format].
```

```
[Core Requirements]:
    1. Do not alter the question to directly satisfy the constraints.
    2. Fully understand the [Original Question] and the constraints within it.
    3. Modify the expression of the constraints in the [Original Question] by
    ↪  clearly describing them in the question, so that the question explicitly
    ↪  indicates the constraints, without changing its structure to meet those
    ↪  constraints directly.
    4. The modified question should keep the original meaning and intent, while the
    ↪  constraints are introduced as descriptive explanations or clarifications in
    ↪  the question.
    5. Ensure that the constraints are explicitly described in the question, making
    ↪  it clear that they need to be considered when answering, without altering
    ↪  the question to directly satisfy them.

[Reply Format]:
    [Constraint Integration Format Data]: xxx

[Data]:
    [Original Question]:{original_question}
    [Constraint List]:{constraint_list}
'''
```

Category	New Constraint List

Content	Main Category : Content
	Subcategory : {
	Keywords: Must include, Repeated, Avoid
	Identifiers: Start identifier, End identifier, Delimiting identifier
	Punctuation: Ending punctuation, Exclude punctuation
	}

Format	Main Category : Format
	Subcategory : {
	Markdown: Heading levels, Block quotes
	Json: Object nesting levels
	XML: Number of attributes
	Table: Row limit, Column limit
	}

Language	Main Category : Language
	Subcategory : {
	Chinese: Simpfied, Traditional
	English: All Uppercase, Capitalized, Lowercase
	}

Length	Main Category : Length
	Subcategory : {
	Words: At most, At least, Range
	Sentences: At most, At least, Range
	Paragraphs: At most, At least, Range
	}

Table 6: New constraint list for different constraint categories.

c1	c2	c3
one new constraint is	a single (Primary category, secondary category, specific constraint) triplet	one constraint
two new constraints are	two (Primary category, secondary category, specific constraint) triplets	two constraints

Table 7: ‘c1,’ ‘c2,’ and ‘c3’ for the prompt of contraint expansion.

C Detailed Information of Models

We conduct an evaluation of 19 LLMs from seven families, including four open-source and three closed-source, that collectively represent the capabilities of current LLMs.

Open-Source LLMs We evaluate eleven open-source LLMs across four model families:

- **LLaMA3.1 family**, developed by Meta, comprises open-source models that demonstrate strong performance in general knowledge and multilingual translation. We evaluate *LLaMA3.1-Instruct-8B* and *LLaMA3.1-Instruct-70B*.
- **Qwen2.5 family**, released by Alibaba, includes open-source models pre-trained on up to 18 trillion tokens. These models show substantial improvements in programming and mathematical reasoning. We evaluate *Qwen2.5-Instruct-7B*, *Qwen2.5-Instruct-14B*, *Qwen2.5-Instruct-32B*, and *Qwen2.5-Instruct-72B*.
- **DeepSeek-R1-Distill-LLaMA family** consists of models distilled from the LLaMA family using data generated by DeepSeek-R1 (DeepSeek-AI et al., 2025). These models exhibit notable gains in mathematical performance. We evaluate *DeepSeek-R1-Distill-LLaMA-8B* and *DeepSeek-R1-Distill-LLaMA-70B*.
- **DeepSeek-R1-Distill-Qwen family** is based on the Qwen2.5 family and similarly distilled using data generated by DeepSeek-R1. We evaluate *DeepSeek-R1-Distill-Qwen-7B*, *DeepSeek-R1-Distill-Qwen-14B*, and *DeepSeek-R1-Distill-Qwen-32B*.

Closed-Source LLMs We evaluate eight closed-source LLMs across three model families:

- **Gemini1.5 family**, developed by Google, represents a new generation of models built on the Mixture-of-Experts (Jacobs et al., 1991) architecture. These models demonstrate enhanced performance, particularly in long-context understanding across multiple modalities. We evaluate *Gemini1.5-Flash* and *Gemini1.5-Pro*.
- **Claude3.5 family**, developed by Anthropic, features models with strong general-purpose capabilities and coding capabilities. We evaluate *Claude3.5-Haiku* and *Claude3.5-Sonnet*.
- **GPT family**, released by OpenAI, includes models that represent the current frontier in LLM development. We evaluate *GPT-3.5-Turbo*, *GPT-4-Turbo*, *GPT-4o-Mini*, and *GPT-4o*.

D Chat Template for Different LLMs

For the open-source LLMs, we use their built-in chat templates for both training and evaluation. The chat templates for each model family are summarized below. Notably, DeepSeek-R1-Distill-LLaMA and DeepSeek-R1-Distill-Qwen share the same template.

```
LLaMA3.1-Instruct_template = '''
{{- bos_token }}
{%- if custom_tools is defined %}
    {%- set tools = custom_tools %}
{%- endif %}
{%- if not tools_in_user_message is defined %}
    {%- set tools_in_user_message = true %}
{%- endif %}
{%- if not date_string is defined %}
    {%- set date_string = "26 Jul 2024" %}
{%- endif %}
{%- if not tools is defined %}
    {%- set tools = none %}
{%- endif %}

{#- This block extracts the system message, so we can slot it into the right place.
↪  #}
{%- if messages[0]['role'] == 'system' %}
    {%- set system_message = messages[0]['content']|trim %}
    {%- set messages = messages[1:] %}
{%- else %}
    {%- set system_message = "" %}
{%- endif %}

{#- System message + builtin tools #}
{{- "<|start_header_id|>system<|end_header_id|>\n\n" }}
{%- if builtin_tools is defined or tools is not none %}
    {{- "Environment: ipython\n" }}
{%- endif %}
{%- if builtin_tools is defined %}
    {{- "Tools: " + builtin_tools | reject('equalto', 'code_interpreter') | join(",
    ↪  ") + "\n\n"}}
{%- endif %}
{{- "Cutting Knowledge Date: December 2023\n" }}
{{- "Today Date: " + date_string + "\n\n" }}
{%- if tools is not none and not tools_in_user_message %}
    {{- "You have access to the following functions. To call a function, please
    ↪  respond with JSON for a function call." }}
    {{- 'Respond in the format {"name": function name, "parameters": dictionary of
    ↪  argument name and its value}.' }}
    {{- "Do not use variables.\n\n" }}
    {%- for t in tools %}
        {{- t | tojson(indent=4) }}
        {{- "\n\n" }}
    {%- endfor %}
{%- endif %}
{{- system_message }}
```

```
{{- "<|eot_id|>" }}

{#- Custom tools are passed in a user message with some extra guidance #}
{%- if tools_in_user_message and not tools is none %}
    {#- Extract the first user message so we can plug it in here #}
    {%- if messages | length != 0 %}
        {%- set first_user_message = messages[0]['content']|trim %}
        {%- set messages = messages[1:] %}
    {%- else %}
        {{- raise_exception("Cannot put tools in the first user message when there's
        ↪  no first user message!") }}
{%- endif %}
    {{- '<|start_header_id|>user<|end_header_id|>\n\n' -}}
    {{- "Given the following functions, please respond with a JSON for a function
    ↪  call " }}
    {{- "with its proper arguments that best answers the given prompt.\n\n" }}
    {{- 'Respond in the format {"name": function name, "parameters": dictionary of
    ↪  argument name and its value}.' }}
    {{- "Do not use variables.\n\n" }}
    {%- for t in tools %}
        {{- t | tojson(indent=4) }}
        {{- "\n\n" }}
    {%- endfor %}
    {{- first_user_message + "<|eot_id|>"}}
{%- endif %}

{%- for message in messages %}
    {%- if not (message.role == 'ipython' or message.role == 'tool' or 'tool_calls'
    ↪  in message) %}
        {{- '<|start_header_id|>' + message['role'] + '<|end_header_id|>\n\n'+
        ↪  message['content'] | trim + '<|eot_id|>' }}
    {%- elif 'tool_calls' in message %}
        {%- if not message.tool_calls|length == 1 %}
            {{- raise_exception("This model only supports single tool-calls at
            ↪  once!") }}
        {%- endif %}
        {%- set tool_call = message.tool_calls[0].function %}
        {%- if builtin_tools is defined and tool_call.name in builtin_tools %}
            {{- '<|start_header_id|>assistant<|end_header_id|>\n\n' -}}
            {{- "<|python_tag|>" + tool_call.name + ".call(" }}
            {%- for arg_name, arg_val in tool_call.arguments | items %}
                {{- arg_name + '="' + arg_val + '"' }}
                {%- if not loop.last %}
                    {{- ", " }}
                {%- endif %}
                {%- endfor %}
            {{- ")" }}
        {%- else  %}
            {{- '<|start_header_id|>assistant<|end_header_id|>\n\n' -}}
            {{- '{"name": "' + tool_call.name + '", ' }}
            {{- '"parameters": ' }}
            {{- tool_call.arguments | tojson }}
```

```
            {{- "}" }}
        {%- endif %}
        {%- if builtin_tools is defined %}
            {#- This means we're in ipython mode #}
            {{- "<|eom_id|>" }}
        {%- else %}
            {{- "<|eot_id|>" }}
        {%- endif %}
    {%- elif message.role == "tool" or message.role == "ipython" %}
        {{- "<|start_header_id|>ipython<|end_header_id|>\n\n" }}
        {%- if message.content is mapping or message.content is iterable %}
            {{- message.content | tojson }}
        {%- else %}
            {{- message.content }}
        {%- endif %}
        {{- "<|eot_id|>" }}
    {%- endif %}
{%- endfor %}
{%- if add_generation_prompt %}
    {{- '<|start_header_id|>assistant<|end_header_id|>\n\n' }}
{%- endif %}
'''


Qwen2.5-Instruct_template = '''
{%- if tools %}
    {{- '<|im_start|>system\n' }}
    {%- if messages[0]['role'] == 'system' %}
        {{- messages[0]['content'] }}
    {%- else %}
       {{- 'You are Qwen, created by Alibaba Cloud. You are a helpful assistant.' }}
    {%- endif %}
    {{- "\n\n# Tools\n\nYou may call one or more functions to assist with the user
    ↪  query.\n\nYou are provided with function signatures within <tools></tools>
    ↪  XML tags:\n<tools>" }}
    {%- for tool in tools %}
        {{- "\n" }}
        {{- tool | tojson }}
    {%- endfor %}
    {{- "\n</tools>\n\nFor each function call, return a json object with function
    ↪  name and arguments within <tool_call></tool_call> XML
    ↪  tags:\n<tool_call>\n{\"name\": <function-name>, \"arguments\":
    ↪  <args-json-object>}\n</tool_call><|im_end|>\n" }}
{%- else %}
    {%- if messages[0]['role'] == 'system' %}
        {{- '<|im_start|>system\n' + messages[0]['content'] + '<|im_end|>\n' }}
    {%- else %}
        {{- '<|im_start|>system\nYou are Qwen, created by Alibaba Cloud. You are a
        ↪  helpful assistant.<|im_end|>\n' }}
    {%- endif %}
{%- endif %}
{%- for message in messages %}
```

```
    {%- if (message.role == "user") or (message.role == "system" and not loop.first)
    ↪  or (message.role == "assistant" and not message.tool_calls) %}
        {{- '<|im_start|>' + message.role + '\n' + message.content + '<|im_end|>' +
        ↪  '\n' }}
    {%- elif message.role == "assistant" %}
        {{- '<|im_start|>' + message.role }}
        {%- if message.content %}
            {{- '\n' + message.content }}
        {%- endif %}
        {%- for tool_call in message.tool_calls %}
            {%- if tool_call.function is defined %}
                {%- set tool_call = tool_call.function %}
            {%- endif %}
            {{- '\n<tool_call>\n{"name": "' }}
            {{- tool_call.name }}
            {{- '", "arguments": ' }}
            {{- tool_call.arguments | tojson }}
            {{- '}\n</tool_call>' }}
        {%- endfor %}
        {{- '<|im_end|>\n' }}
    {%- elif message.role == "tool" %}
        {%- if (loop.index0 == 0) or (messages[loop.index0 - 1].role != "tool") %}
            {{- '<|im_start|>user' }}
        {%- endif %}
        {{- '\n<tool_response>\n' }}
        {{- message.content }}
        {{- '\n</tool_response>' }}
        {%- if loop.last or (messages[loop.index0 + 1].role != "tool") %}
            {{- '<|im_end|>\n' }}
        {%- endif %}
    {%- endif %}
{%- endfor %}
{%- if add_generation_prompt %}
    {{- '<|im_start|>assistant\n' }}
{%- endif %}
'''

DeepSeek-R1-Distill = '''
{% if not add_generation_prompt is defined %}
    {% set add_generation_prompt = false %}
{% endif %}
{% set ns = namespace(is_first=false, is_tool=false, is_output_first=true,
↪  system_prompt='') %}
{%- for message in messages %}
    {%- if message['role'] == 'system' %}
        {% set ns.system_prompt = message['content'] %}
    {%- endif %}
{%- endfor %}
{{bos_token}}
{{ns.system_prompt}}
{%- for message in messages %}
    {%- if message['role'] == 'user' %}
```

```
        {%- set ns.is_tool = false -%}
        {{'<|User|>' + message['content']}}
    {%- endif %}
    {%- if message['role'] == 'assistant' and message['content'] is none %}
        {%- set ns.is_tool = false -%}
        {%- for tool in message['tool_calls']%}
            {%- if not ns.is_first %}
                {{'<|Assistant|><|tool_calls_begin|><|tool_call_begin|>' +
                ↪  tool['type'] + '<|tool_sep|>' + tool['function']['name'] +
                ↪  '\\n' + '```json' + '\\n' + tool['function']['arguments'] +
                ↪  '\\n' + '```' + '<|tool_call_end|>'}}
                {%- set ns.is_first = true -%}
            {%- else %}
                {{'\\n' + '<|tool_call_begin|>' + tool['type'] + '<|tool_sep|>' +
                ↪  tool['function']['name'] + '\\n' + '```json' + '\\n' +
                ↪  tool['function']['arguments'] + '\\n' + '```' +
                ↪  '<|tool_call_end|>'}}
                {{'<|tool_calls_end|><|end_of_sentence|>'}}
            {%- endif %}
        {%- endfor %}
    {%- endif %}
    {%- if message['role'] == 'assistant' and message['content'] is not none %}
        {%- if ns.is_tool %}
            {{'<|tool_outputs_end|>' + message['content'] + '<|end_of_sentence|>'}}
            {%- set ns.is_tool = false -%}
        {%- else %}
            {% set content = message['content'] %}
            {% if '</think>' in content %}
                {% set content = content.split('</think>')[-1] %}
            {% endif %}
            {{'<|Assistant|>' + content + '<|end_of_sentence|>'}}
        {%- endif %}
    {%- endif %}
    {%- if message['role'] == 'tool' %}
        {%- set ns.is_tool = true -%}
        {%- if ns.is_output_first %}
            {{'<|tool_outputs_begin|><|tool_output_begin|>' + message['content'] +
            ↪  '<|tool_output_end|>'}}
            {%- set ns.is_output_first = false %}
        {%- else %}
            {{'\\n<|tool_output_begin|>' + message['content'] +
            ↪  '<|tool_output_end|>'}}
        {%- endif %}
{%- endif %}
{%- endfor -%}
{% if ns.is_tool %}
    {{'<|tool_outputs_end|>'}}
{% endif %}
{% if add_generation_prompt and not ns.is_tool %}
    {{'<|Assistant|><think>\\n'}}
{% endif %}
'''
```

E Detailed Results

Table 8 to Table 13 present the detailed performance of the LLMs on each test set before and after GRPO. In these tables, Δ denotes the performance difference between the post-GRPO and pre-GRPO models, with positive values highlighted in green and negative values in red. The results demonstrate that applying our data for GRPO substantially enhances the models' capabilities across all aspects, highlighting the effectiveness of our data.

Models	Constraint Pattern			Constraint Category				Constraint Difficulty				Overall
	Example	Listing	Incorporation	Content	Format	Language	Length	Level I	Level II	Level III	Level IV	
LLaMA3.1-Instruct-8B												
w/o GRPO	40.25	36.00	32.25	80.13	64.74	44.28	49.60	64.67	40.67	27.33	12.00	36.17
w/ GRPO	89.00	91.00	84.25	95.23	92.20	98.70	93.38	94.33	88.67	85.33	84.00	88.08
Δ	48.75	55.00	52.00	15.10	27.46	54.42	43.78	29.66	48.00	58.00	72.00	51.91
Qwen2.5-Instruct-7B												
w/o GRPO	59.25	45.00	43.75	83.77	54.45	90.30	64.25	82.00	55.00	39.00	21.33	49.33
w/ GRPO	87.50	89.25	81.50	94.10	88.84	98.26	93.77	94.67	87.67	81.00	81.00	86.08
Δ	28.25	44.25	37.75	10.33	34.39	7.96	29.52	12.67	32.67	42.00	59.67	36.75
Qwen2.5-Instruct-14B												
w/o GRPO	66.75	52.50	51.25	84.42	62.74	84.95	74.60	82.00	63.67	49.33	32.33	56.83
w/ GRPO	81.50	86.75	77.75	92.85	85.08	97.83	91.30	91.33	85.00	79.67	72.00	82.00
Δ	14.75	34.25	26.50	8.43	22.34	12.88	16.70	9.33	21.33	30.34	39.67	25.17
DeepSeek-R1-Distill-LLaMA-Instruct-8B												
w/o GRPO	42.00	28.00	28.00	80.00	50.44	44.72	53.09	59.67	40.33	18.33	12.33	32.67
w/ GRPO	83.75	85.50	77.00	93.48	83.60	98.55	92.60	91.33	85.00	78.00	74.00	82.08
Δ	41.75	57.50	49.00	13.48	33.16	53.83	39.51	31.66	44.67	59.67	61.67	49.41
DeepSeek-R1-Distill-Qwen-Instruct-7B												
w/o GRPO	31.25	26.50	27.50	78.70	50.19	24.60	51.08	61.33	33.33	14.33	4.67	28.42
w/ GRPO	66.75	71.50	65.00	87.83	72.18	90.01	89.48	89.00	73.67	57.33	51.00	67.75
Δ	35.50	45.00	37.50	9.13	21.99	65.41	38.40	27.67	40.34	43.00	46.33	39.33
DeepSeek-R1-Distill-Qwen-Instruct-14B												
w/o GRPO	52.75	38.00	34.25	82.99	66.75	44.86	59.41	70.67	49.67	29.33	17.00	41.67
w/ GRPO	81.50	86.75	77.75	92.85	85.08	97.83	91.30	91.33	85.00	79.67	72.00	82.00
Δ	28.75	48.75	43.50	9.86	18.33	52.97	31.89	20.66	35.33	50.34	55.00	40.33

Table 8: Evaluation results on our custom test set.

Models	Prompt Str.	Instruction Str.	Prompt Loo.	Instruction Loo.
LLaMA3.1-Instruct-8B				
w/o GRPO	70.79	79.02	74.49	82.49
w/ GRPO	79.11	85.73	80.22	86.57
Δ	8.32	6.71	5.73	4.08
Qwen2.5-Instruct-7B				
w/o GRPO	72.83	80.34	74.86	82.01
w/ GRPO	75.97	82.97	79.30	85.61
Δ	3.14	2.63	4.44	3.60
Qwen2.5-Instruct-14B				
w/o GRPO	79.67	84.53	81.52	86.21
w/ GRPO	83.55	88.61	86.51	90.89
Δ	3.88	4.08	4.99	4.68
DeepSeek-R1-Distill-LLaMA-Instruct-8B				
w/o GRPO	61.55	71.82	67.65	76.50
w/ GRPO	79.30	85.13	82.07	87.41
Δ	17.75	13.31	14.42	10.91
DeepSeek-R1-Distill-Qwen-Instruct-7B				
w/o GRPO	53.79	65.59	58.04	69.42
w/ GRPO	68.76	77.22	72.09	80.34
Δ	14.97	11.63	14.05	10.92
DeepSeek-R1-Distill-Qwen-Instruct-14B				
w/o GRPO	70.24	78.42	74.12	81.41
w/ GRPO	84.47	89.45	86.32	90.29
Δ	14.23	11.03	12.20	8.88

Table 9: Evaluation results on IFEval.

Models	Italian	Spanish	Hindi	Portuguese	English	French	Chinese	Russian	Avg.
LLaMA3.1-Instruct-8B									
w/o GRPO	68.30	72.70	58.50	69.50	79.29	67.83	62.38	60.52	68.60
w/ GRPO	83.53	83.58	69.87	81.65	81.99	77.39	75.04	71.22	78.41
Δ	15.23	10.88	11.37	12.15	2.70	9.56	12.66	10.70	9.81
Qwen2.5-Instruct-7B									
w/o GRPO	78.78	79.01	59.25	75.64	76.66	75.99	72.56	70.76	73.83
w/ GRPO	79.91	81.96	62.67	78.27	78.72	79.16	78.35	72.89	76.62
Δ	1.13	2.95	3.42	2.63	2.06	3.17	5.79	2.13	2.79
Qwen2.5-Instruct-14B									
w/o GRPO	78.98	83.08	65.79	79.43	83.95	79.43	76.03	72.89	78.05
w/ GRPO	84.55	87.23	69.47	86.42	86.96	84.51	81.48	79.02	82.85
Δ	5.57	4.15	3.68	6.99	3.01	5.08	5.45	6.13	4.80
DeepSeek-R1-Distill-LLaMA-Instruct-8B									
w/o GRPO	53.82	56.23	43.15	53.22	60.44	51.44	64.01	49.24	54.37
w/ GRPO	76.08	72.62	50.86	75.44	79.92	67.61	70.49	61.55	70.21
Δ	22.26	16.39	7.71	22.22	19.48	16.17	6.48	12.31	15.84
DeepSeek-R1-Distill-Qwen-Instruct-7B									
w/o GRPO	54.81	57.78	41.32	55.97	60.00	54.80	65.35	50.66	55.36
w/ GRPO	70.97	73.99	52.81	69.20	72.52	70.67	70.49	60.98	68.09
Δ	16.16	16.21	11.49	13.23	12.52	15.87	5.14	10.32	12.73
DeepSeek-R1-Distill-Qwen-Instruct-14B									
w/o GRPO	68.39	67.75	48.09	62.08	72.28	66.74	73.54	62.73	65.68
w/ GRPO	81.82	83.13	63.25	81.03	83.89	80.84	82.41	77.20	79.54
Δ	13.43	15.38	15.16	18.95	11.61	14.10	8.87	14.47	13.86

Table 10: Evaluation results on Multi-IF for turn 1.

Models	Italian	Spanish	Hindi	Portuguese	English	French	Chinese	Russian	Avg.
LLaMA3.1-Instruct-8B									
w/o GRPO	59.62	64.31	50.71	62.76	71.64	62.92	54.81	41.97	59.82
w/ GRPO	70.76	73.49	55.87	67.89	72.79	69.49	66.35	54.57	66.94
Δ	11.14	9.18	5.16	5.13	1.15	6.57	11.54	12.60	7.12
Qwen2.5-Instruct-7B									
w/o GRPO	60.33	62.60	44.81	55.74	62.39	61.74	56.27	50.59	57.30
w/ GRPO	61.78	61.58	47.13	59.72	64.78	60.89	57.85	52.45	58.81
Δ	1.45	-1.02	2.32	3.98	2.39	-0.85	1.58	1.86	1.51
Qwen2.5-Instruct-14B									
w/o GRPO	68.64	67.34	54.27	66.05	71.21	67.13	62.72	55.00	64.66
w/ GRPO	72.35	72.09	61.04	72.07	74.25	72.15	67.76	61.56	69.61
Δ	3.71	4.75	6.77	6.02	3.04	5.02	5.04	6.56	4.95
DeepSeek-R1-Distill-LLaMA-Instruct-8B									
w/o GRPO	44.92	45.01	28.29	44.90	55.15	41.90	54.69	30.67	44.04
w/ GRPO	63.35	63.59	43.48	63.13	69.34	59.72	61.08	38.14	58.66
Δ	18.43	18.58	15.19	18.23	14.19	17.82	6.39	7.47	14.62
DeepSeek-R1-Distill-Qwen-Instruct-7B									
w/o GRPO	39.96	48.02	30.50	46.03	51.25	42.87	52.06	31.82	43.41
w/ GRPO	57.43	59.16	44.02	54.55	61.98	57.56	59.39	38.69	54.72
Δ	17.47	11.14	13.52	8.52	10.73	14.69	7.33	6.87	11.31
DeepSeek-R1-Distill-Qwen-Instruct-14B									
w/o GRPO	52.39	51.48	29.95	43.09	57.29	49.98	62.38	36.97	48.50
w/ GRPO	69.54	69.60	46.44	67.33	73.39	71.96	69.69	56.38	66.12
Δ	17.15	18.12	16.49	24.24	16.10	21.98	7.31	19.41	17.62

Table 11: Evaluation results on Multi-IF for turn 2.

Models	Italian	Spanish	Hindi	Portuguese	English	French	Chinese	Russian	Avg.
LLaMA3.1-Instruct-8B									
w/o GRPO	51.27	54.55	41.78	54.54	63.75	54.91	45.53	35.31	51.46
w/ GRPO	59.08	61.03	46.79	59.75	64.72	58.53	53.71	41.47	56.43
Δ	7.81	6.48	5.01	5.21	0.97	3.62	8.18	6.16	4.97
Qwen2.5-Instruct-7B									
w/o GRPO	48.61	49.02	35.92	46.71	52.55	49.22	46.74	38.74	46.48
w/ GRPO	50.11	47.94	36.70	48.30	53.85	48.50	47.02	40.02	47.15
Δ	1.50	-1.08	0.78	1.59	1.30	-0.72	0.28	1.28	0.67
Qwen2.5-Instruct-14B									
w/o GRPO	59.33	56.80	45.65	56.67	62.40	58.83	51.82	44.59	55.19
w/ GRPO	62.19	61.61	52.41	61.42	66.33	61.63	56.03	50.04	59.62
Δ	2.86	4.81	6.76	4.75	3.93	2.80	4.21	5.45	4.43
DeepSeek-R1-Distill-LLaMA-Instruct-8B									
w/o GRPO	34.95	32.51	19.85	35.67	41.91	32.33	44.42	23.06	33.67
w/ GRPO	50.37	48.43	29.32	52.51	58.75	46.58	50.12	31.16	46.94
Δ	15.42	15.92	9.47	16.84	16.84	14.25	5.70	8.10	13.27
DeepSeek-R1-Distill-Qwen-Instruct-7B									
w/o GRPO	24.85	32.60	19.90	34.33	36.95	33.71	42.58	22.86	31.35
w/ GRPO	43.45	44.81	31.23	45.55	53.39	44.70	49.72	30.80	43.76
Δ	18.6	12.21	11.33	11.22	16.44	10.99	7.14	7.94	12.41
DeepSeek-R1-Distill-Qwen-Instruct-14B									
w/o GRPO	35.43	37.45	21.55	34.30	41.11	36.01	51.34	27.86	35.86
w/ GRPO	56.74	56.19	35.45	54.27	63.12	58.47	58.66	42.65	53.92
Δ	21.31	18.74	13.90	19.97	22.01	22.46	7.32	14.79	18.06

Table 12: Evaluation results on Multi-IF for turn 3.

Models	MMLU	GSM8K	MATH	HumanEval	MBPP
LLaMA-3.1-8B-Instruct					
w/o GRPO	71.4	80.44	48.94	70.73	69.65
w/ GRPO	71.72	81.12	47.74	70.12	67.7
Δ	0.32	0.68	1.20	0.61	1.95
Qwen2.5-7B-Instruct					
w/o GRPO	76.21	86.88	69.12	81.71	77.82
w/ GRPO	75.07	87.72	69.82	84.15	76.26
Δ	1.14	0.84	0.70	2.44	1.56
Qwen2.5-14B-Instruct					
w/o GRPO	81.21	90.14	78.44	83.54	79.77
w/ GRPO	80.62	91.96	77.18	81.71	82.1
Δ	0.59	1.82	1.26	1.83	2.33
DeepSeek-R1-Distill-Llama-8B					
w/o GRPO	67.37	84.61	82.3	77.44	73.93
w/ GRPO	69.39	85.82	84.54	84.15	80.93
Δ	2.02	1.21	2.24	6.71	6.99
DeepSeek-R1-Distill-Qwen-7B					
w/o GRPO	62.77	87.56	85.1	77.44	76.26
w/ GRPO	64.74	88.86	87.42	82.32	78.6
Δ	1.97	1.30	2.32	4.88	2.34
DeepSeek-R1-Distill-Qwen-14B					
w/o GRPO	81.91	91.28	87.68	87.8	84.44
w/ GRPO	83	92.49	89.18	90.85	84.82
Δ	1.09	1.21	1.50	3.05	0.38

Table 13: Evaluation results on general domains.