EIFBENCH: Extremely Complex Instruction Following Benchmark for Large Language Models

Anonymous ACL submission

Abstract

With the development and widespread application of large language models (LLMs), the new paradigm of "*Model as Product*" is rapidly evolving, and demands higher capabilities to address complex user needs, often requiring precise workflow execution which involves the accurate understanding of multiple tasks. However, existing benchmarks focusing on single-task environments with limited constraints lack the complexity required to fully reflect real-world scenarios. To bridge this gap, we present the **E**xtremely Complex **I**nstruction **F**ollowing **Bench**mark (**EIFBENCH**), meticulously crafted to facilitate a more realistic and robust evaluation of LLMs. EIFBENCH not only includes multi-task scenarios that enable comprehensive assessment across diverse task types concurrently, but also integrates a variety of constraints, replicating complex operational environments. Furthermore, we propose the **Seg**ment **P**olicy **O**ptimization (**SegPO**) algorithm to enhance the LLM's ability to accurately fulfill multi-task workflow. Evaluations on EIFBENCH have unveiled considerable performance discrepancies in existing LLMs when challenged with these extremely complex instructions. This finding underscores the necessity for ongoing optimization to navigate the intricate challenges posed by LLM applications.

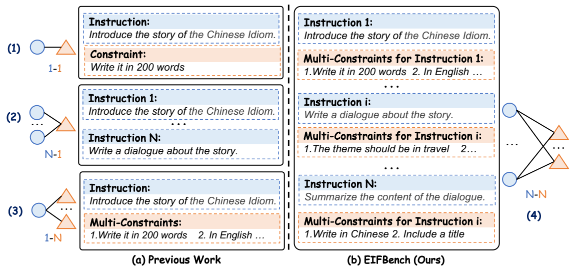

Figure 1: Existing benchmarks, represented on the left, either focus on completing a single instruction or handling multiple instructions with only one constraint each. In contrast, EIFBENCH presents a multi-instruction, multi-constraint benchmark, designed to more closely align with real-world complexities and demands.

1 Introduction

The advent of large language models (LLMs) has transformed real-world applications by improving models' ability to comprehend a diverse range of human instructions, from simple conversations to complex problem solving (Sanh et al., 2022; Dubois et al., 2023). Thus, instructions have become central to effective human-machine interaction in this new landscape (Zhong et al., 2021; Mishra et al., 2022; Gao et al., 2024), especially the paradigm of "*Model as Product*" has deeply entered the collective consciousness where LLM agents need to accurately complete a series of tasks to meet user demands (Xiong et al., 2025; Hu et al., 2024; Alakuijala et al., 2025). However, as user demands grow more sophisticated, traditional benchmarks (Zhong et al., 2024; Chia et al., 2023), which focus on specific tasks, are insufficient to evaluate models' comprehensive ability to handle multifaceted instructions. This shortfall underscores the need for innovative evaluation frameworks capable of accurately assessing how models understand and execute complex instructions (Zhou et al., 2023; Wang et al., 2023; Xu et al., 2024).

To evaluate the instruction following abilities of LLMs, several benchmarks (Zhou et al., 2023; Qin et al., 2024; Li et al., 2024) have been proposed, which can be categorized into three main types as shown in Fig. 1: (1) ***Single-Instruction Single-Constraint*** benchmarks, such as IFEval (Zhou et al., 2023) and INFOBENCH (Qin et al., 2024), focus on tasks governed by a single constraint, providing insights into basic instruction following abilities. (2) ***Single-Instruction Multi-Constraint*** benchmarks, like CFBench (He et al., 2024b), evaluate how models handle a single instruction with multiple constraints across content, numerical, and other dimensions simultaneously.

(3) ***Multi-Instruction Single-Constraint*** scenarios, such as those explored by SIFo (Chen et al., 2024), test models' adherence to sequences of instructions, assessing their adaptability and versatility while maintaining focus on a single constraint. Nonetheless, research still lacks in addressing **multi-instruction multi-constraint** scenarios, which more accurately reflect real-world complexities, especially in the era of LLMs serving as agents with workflow execution involving multiple tasks.

Multi-instruction multi-constraint (MIMC) scenarios are ubiquitous in real-world applications, such as workflow automation (Zhang et al., 2022; Taylor et al., 2023) and healthcare scheduling (Bakhshandeh and Al-e-hashem, 2024; Li et al., 2021). For example, in cloud-based workflow automation, orchestrating computational tasks such as data preprocessing, model inference, and report generation requires balancing resource allocation, execution time, and task dependencies (Xiong et al., 2016). However, existing LLMs struggle with such complexity, with performance dropping by over 30% with over 5 constraints (He et al., 2024b). Bridging this gap necessitates benchmarks that mirror real-world MIMC dynamics, integrating both task interdependence and constraint scalability to foster robust and adaptable LLMs.

In response to these challenges, we introduce the **E**xtremely Complex **I**nstruction **F**ollowing **Bench**mark (**EIFBENCH**), specifically designed to address the shortcomings of current benchmarks by providing a comprehensive framework that mirrors the complexities of real-world task environments. As shown in Fig. 1, EIFBENCH is unique in its inclusion of multi-task scenarios, drawn from diverse sources and integrated with multifaceted constraints[1]. This design allows for an in-depth assessment of a model's ability to manage complex demands. In addition, we introduce the **Segment Policy Optimization (SegPO)** algorithm, which features advantage estimation for outputs corresponding to each instruction within multi-instruction inputs. The main contributions of this paper are summarized as follows:

- We first develop the extremely complex instruction following benchmark (EIFBENCH), simulating real-world applications with multiple instructions and constraints.
- We propose the segment policy optimization (SegPO) algorithm by calculating the advantages separately for each output that responds to the corresponding instruction within the input, encouraging more nuanced feedback in following multiple instructions.
- We conduct a detailed analysis of 17 LLMs, encompassing both open-source and closed-source models, uncovering their limitations in processing complex instructions and pinpointing areas for enhancement to better adapt to real-world complex scenarios. The SegPO algorithm demonstrates significant improvements, achieving increases of 14.85% compared to the base LLM and 3.40% compared to GRPO models on EIFBENCH, respectively.

[1]In this work, plain text datasets refer to non-conversational plain text datasets.

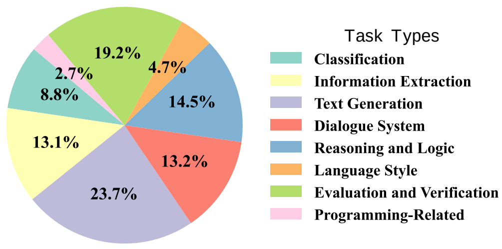

Figure 2: Task type distribution in EIFBENCH.

2 EIFBENCH

2.1 Task and Constraint Taxonomy

To thoroughly assess the capability of large language models (LLMs) in adhering to complex instructions, we introduce an exceptionally challenging instruction following benchmark. Specifically, we categorize both tasks and constraints to structure the evaluation. For tasks, we identify and compile 8 types of tasks based on traditional NLP tasks. Regarding constraints, we establish a two-level hierarchical taxonomy for the organization.

2.1.1 Task Categories

In line with instruction following existing works (Zhang et al., 2024a; Li et al., 2024), we categorize the tasks in EIFBENCH into eight primary types. [2] These categories provide a comprehensive framework for systematically evaluating model performance across diverse task settings. The distribution of these task categories is shown in Fig. 2.

Classification involves sentiment analysis, text and toxic content classification, empathy detection, and

[2]In this work, following an instruction refers to completing one specific task and producing the corresponding response.

Benchmark	Multi-Constraint	Multi-Instruction	Multi-Type	Average Constraint	Average Instruction
CIF-Bench (Li et al., 2024)	✗	✗	✗	1.00	1.00
FollowBench (Jiang et al., 2024)	✓	✗	✗	3.00	1.00
ComplexBench (Wen et al., 2024)	✓	✗	✗	4.19	1.00
CFBench (He et al., 2024b)	✓	✗	✗	4.24	1.00
SIFo (Chen et al., 2024)	✗	✓	✗	1.00	4.17
EIFBENCH (Ours)	✓	✓	✓	**74.01**	**8.24**

Table 1: EIFBENCH encompasses multi-instruction multi-constraint samples across multiple data types. "Multi-type" refers to the inclusion of data from various formats, such as plain text, dialogue, and multi-party dialogue, highlighting diverse communication styles and structures.

social norm judgment.

Information Extraction focuses on extracting key information such as named entity recognition, keyword annotation, and entity relationships.

Text Generation tasks cover creative and practical outputs, including story generation, text expansion, and headline content generation.

Dialogue System tasks are designed for developing interactive agents through dialogue generation, intent recognition, and state information tracking.

Reasoning and Logic tasks require logical inference and critical thinking, including commonsense and multi-hop reasoning question answering.

Language Style tasks involve style manipulation and analysis, such as style transfer, sarcasm detection, and dialect variation recognition.

Evaluation and Verification tasks concentrate on verifying information and assessing text quality, including fact consistency verification.

Programming-Related tasks evaluate programming understanding through code generation, debugging, and explanation capabilities.

In addition, tasks are structured into distinct modes: parallel for simultaneous dimension consideration, serial for chain dependencies, conditional for adaptability to varying conditions, and nested for hierarchical structures. These categories provide a systematic evaluation of model capabilities in the benchmark.

2.1.2 Constraint Categories

Following established research on instruction following (Zhang et al., 2024b), we have developed a comprehensive constraint system for EIFBENCH. This system categorizes constraints into four primary types: Content Constraints, Situation Constraints, Style Constraints, and Format Constraints. These categories provide a structured framework to systematically evaluate the capabilities of language models across a wide range of instructional scenarios. The distribution is shown in Fig. 3. Detailed

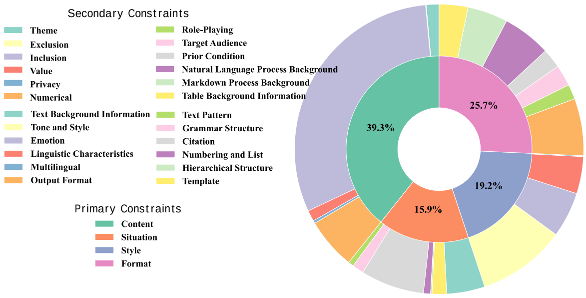

Figure 3: Constraint type distribution in EIFBENCH.

descriptions of the specific constraint dimensions within each category are provided in Appendix A.

Content Constraints. These ensure the text follows specific thematic topics, inclusion/exclusion criteria, values, tone, style, privacy considerations, and numerical precision.

Situation Constraints. These emphasize contextual elements like audience specifications, preconditions, and incorporate various knowledge and background information formats.

Style Constraints. These govern tone, emotion, style, and multilingual features to suit the required stylistic and emotional text aspects.

Format Constraints. These ensure adherence to essential structural requirements such as output formats, text patterns, grammar, accurate sentence structure, and hierarchical organization.

2.2 Construction Workflow

The overall construction process includes several key stages: 1) Taxonomy of Constraints and Tasks, 2) Multi-scenario Data Collection, 3) Task Expansion, 4) Constraint Expansion, 5) Quality Control, and 6) Response Generation & Evaluation.

1) Taxonomy of Constraints and Tasks. We establish two taxonomies for constraints and tasks, as presented in Section 2.

2) Multi-scenario Data Collection. Our collection process involves three types of datasets: plain text, dyadic dialogue, and multi-party dia-

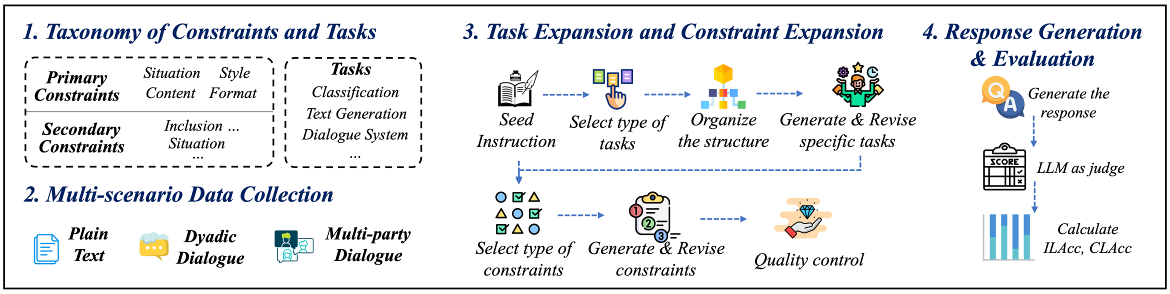

Figure 4: Pipeline for constructing the benchmark.

logue. Plain text samples are drawn from existing works (Wen et al., 2024; Li et al., 2024). For dyadic dialogues, we gather real-life interactions, undergo cleaning and noise reduction, and use large language models (LLMs) to condense conversations while preserving key information. Multi-party dialogue data is synthesized with LLMs, crafting diverse scenarios and participant numbers. Specific prompts guide LLMs to produce varied and representative dialogue content, enhancing the depth and applicability of the dataset.

3) Task Expansion. Tasks are expanded into series in the plain text scenario (see Section 2.1.1). Using LLMs, we develop complex task sets with dependencies and parallelism. We also conduct rigorous quality assessments, removing redundant, infeasible, and contradictory tasks, thus ensuring the quality and consistency of the generated data. In dyadic and multi-party dialogue scenarios, we directly generate multiple new tasks, ensuring each reflects the complexity of real-world interactions.

4) Constraint Expansion. In the constraint expansion process, we refine simple instructions using a predefined taxonomy (see Section 2.1.2). Utilizing LLMs, complexity is incrementally added, ensuring tasks encompass a broad spectrum of requirements and constraints. This iterative review targets and clarifies ambiguous semantics to ensure constraints are objectively evaluated and quantified. This method not only adds complexity and challenge but also enhances the realism and comprehensiveness of the data generated.

5) Quality Assessment. Our quality assessment covers instruction-level and constraint-level validation. For instruction-level validation, we ensure logical consistency and feasibility for LLMs, removing contradictory, redundant, or infeasible tasks while maintaining a diverse, moderate difficulty task set of 6 to 12 instructions. In constraint-level validation, constraints are iteratively refined

Category	#N	Min.	Max.	Avg.
Plain Text	450	41	107	73.27
Dyadic Dialogue	450	47	107	73.38
Multi-party Dialogue	100	63	116	80.26

Table 2: Statistics of EIFBENCH. #N denotes data instances; Min., Max., and Avg. mean the minimum, maximum, and average number of constraints per instance.

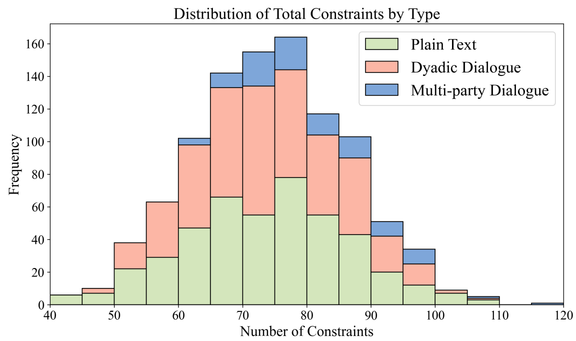

Figure 5: Distributions of total constraints for different text categories.

using predefined taxonomies, ensuring they are objectively quantified and within model capabilities, addressing any ambiguity or infeasibility.

6) Response Generation & Evaluation. First, using the instruction data, we employ various language models to generate the corresponding outputs. To verify their compliance, we then prompt large language models to assess each constraint satisfaction for the outputs, generating a binary outcome (0/1) that indicates whether the generated output satisfies the respective constraints.

As shown in Table 2, EIFBENCH comprises 1,000 instances. Across three subsets, the minimum, maximum, and average numbers of constraints per instance are reported. Fig. 5 and Table 3 illustrate the distribution of constraint numbers and instruction numbers within EIFBENCH.

Scenario	6	7	8	9	10	11	12
Plain Text	15	76	136	139	76	7	1
Dyadic Dialogue	42	113	152	108	33	2	-
Multi-party Dialogue	-	11	47	27	13	1	1

Table 3: Distributions of instructions with different numbers of constraints.

2.3 Evaluation Protocol

We employ GPT-4o (OpenAI, 2023) as the evaluation model to assess constraint adherence in generated responses. Following established practices (Wen et al., 2024), the k-th constraint in the j-th instruction of i-th instance is given a binary compliance score $S_{i,j,k} \in \{0, 1\}$, with 1 signifying full compliance and 0 indicating non-compliance.

Instruction-Level Accuracy (ILA) measures the success rate of individual instructions by averaging compliance across all instructions within a single instance. For the i-th instance, m_i denotes the number of instructions and $c_{i,j}$ is the number of constraints in the j-th instruction. We calculate the average score for n instances as the final metric.

$$\text{ILA}_i = \frac{1}{m_i}\sum_{j=1}^{m_i}\left(\frac{1}{c_{i,j}}\prod_{k=1}^{c_{i,j}} S_{i,j,k}\right) \tag{1}$$

$$\text{ILA} = \frac{1}{n}\sum_{i=1}^{n}\text{ILA}_i \tag{2}$$

Constraint-Level Accuracy (CLA) assesses the fulfillment of individual constraints, making it crucial for identifying specific requirement violations.

$$\text{CLA}_i = \frac{1}{\sum_{j=1}^{m_i} c_{i,j}}\sum_{j=1}^{m_i}\sum_{k=1}^{c_{i,j}} S_{i,j,k} \tag{3}$$

$$\text{CLA} = \frac{1}{n}\sum_{i=1}^{n}\text{CLA}_i \tag{4}$$

These metrics progressively assess compliance at different granularities: from strict instruction-level compliance (ILA) to fine-grained constraint-level analysis (CLA).

2.4 Quality Control

To ensure high-quality evaluation data, we implement a post-inspection protocol following initial generation. First, we leverage `Qwen2.5-72B-Instruct` to systematically verify instruction-clarity alignment, logical consistency of constraints, and overall task feasibility, while automatically detecting and correcting identifiable errors through iterative self-refinement. Subsequently, three certified annotation specialists perform manual review to remove redundant constraints and instructions, revise infeasible tasks, and resolve ambiguous phrasing, ensuring both technical rigor and practical usability.

3 Segment Policy Optimization

Reasoning LLMs (Reid et al., 2024; Jaech et al., 2024) have demonstrated improved performance on complex tasks via structured reasoning. However, models primarily designed for single-task domains such as mathematical problem solving and code generation often struggle in settings requiring the concurrent execution of multiple instructions. To bridge this gap, we propose segment policy optimization (SegPO), which integrates reasoning mechanisms and instruction-level evaluation into the advantage design, thereby enhancing task alignment and improving both the accuracy and robustness of model outputs.

For each query q, SegPO involves sampling a group of outputs $\{y_1, y_2, \ldots, y_G\}$ from the old policy $\pi_{\theta_{\text{old}}}$. The policy model is optimized by maximizing the following objective:

$$\begin{aligned}
&\mathcal{J}_{\text{SegPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(y|q)} \\
&\quad\left\{\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\Big\{\min\Big[r_{i,t}(\theta)(A^{o}_{i,t}+A^{\phi}_{i,t}),\right. \\
&\quad\text{clip}\big(r_{i,t}(\theta),1-\epsilon,1+\epsilon\big)(A^{o}_{i,t}+A^{\phi}_{i,t})\Big] \\
&\quad\left.-\beta\mathbb{D}_{\text{KL}}\left[\pi_\theta||\pi_{\text{ref}}\right]\Big\}\right\}
\end{aligned} \tag{5}$$

where ϵ and β are hyper-parameters, $A^{o}_{i,t}$ and $A^{\phi}_{i,t}$ are the advantages, $r_{i,t}(\theta)$ represents the probability ratio or importance sampling weight between the new policy π_θ and the old policy $\pi_{\theta_{\text{old}}}$ and $\mathbb{D}_{\text{KL}}\left[\pi_\theta||\pi_{\text{ref}}\right]$ denotes the KL divergence between the trained policy and the reference policy. Detailed information is shown in Appendix B.

In SegPO, the advantage for the t-th token in the response y_i consists of two parts: the global advantage $A^{o}_{i,t}$ and the segment advantage $A^{\phi}_{i,t}$. For the global advantage, we use a group of rewards $\{r^{o}_{1}, \cdots, r^{o}_{G}\}$ corresponding to the outputs within each group for computation. For the segment advantage $A^{\phi}_{i,t}$, we select the group of rewards $\{r^{\phi}_{1,I^i_t}, \cdots, r^{\phi}_{G,I^i_t}\}$ for the corresponding I^i_t-th instruction in outputs for computation. The process is as follows:

$$A_{i,t}^{o} = \frac{r_i^o - \text{mean}(\{r_1^o, \cdots, r_G^o\})}{\text{std}(\{r_1^o, \cdots, r_G^o\})}, \tag{6}$$

$$A_{i,t}^{\phi} = \frac{r_{i,I_t^i}^{\phi} - \text{mean}(\{r_{1,I_t^i}^{\phi}, \cdots, r_{G,I_t^i}^{\phi}\})}{\text{std}(\{r_{1,I_t^i}^{\phi}, \cdots, r_{G,I_t^i}^{\phi}\})}. \tag{7}$$

Specifically, we employ both LLM-based and rule-based systems to determine the rewards. For each response y_i to the query q, r_i^o captures accuracy and format compliance. Our rule-based system mandates that reasoning is enclosed between 'start_think' and 'end_think' tags, and answers between 'start_answer' and 'end_answer'. The format score, r_i^f, is one if the format is adhered to, otherwise zero. We assess ILA_i and CLA_i metrics using state-of-the-art LLMs (i.e., `Qwen2.5-72B-Instruct`), with scores increased if all instructions are correctly executed. Furthermore, for the t token in the response y_i associated with the I_t^i-th instruction, we define the segment reward $r_{i,I_t^i}^{\phi}$ as 1 if all the constraints in the I_t^i-th instruction are satisfied, else 0. Details of the training template are provided in the Appendix. The reward process is summarized as follows:

$$r_i^o = \text{ILA}_i + \text{CLA}_i + \prod_{j=1}^{m_i} \prod_{k=1}^{c_{i,j}} S_{i,j,k} + r_i^f, \tag{8}$$

$$r_{i,I_t^i}^{\phi} = \prod_{k=1}^{c_{i,I_t^i}} S_{i,I_t^i,k}. \tag{9}$$

4 Experiments

4.1 Baselines

We compare the performance of both proprietary and open-source LLMs trained on diverse corpora. In the proprietary category, we evaluate models such as GPT-4o (OpenAI, 2023), GPT-4o-mini (OpenAI, 2023), Claude3.5-Sonnet (Anthropic, 2024b), Claude3.5-Haiku (Anthropic, 2024a) and gemini-1.5-Pro (Reid et al., 2024). Among open-source models, we assess LLaMA3.1 (Dubey et al., 2024), Qwen2 (Yang et al., 2024a), Qwen2.5, DeepSeek-R1 (Reid et al., 2024), QwQ-32B (Yang et al., 2024b), and Qwen3 (Yang et al., 2025) to explore their efficiency.

4.2 Settings

For inference, we efficiently process proprietary models through their APIs. For open-source models, we employ a robust setup consisting of four Nvidia A100 GPUs, each equipped with 80GB of VRAM, utilizing the vLLM framework on EIFBENCH where applicable. This configuration enables the completion of all tasks in roughly 30 minutes. During evaluation, the GPT-4o model serves as the evaluator, with assessment durations ranging from 4 to 10 hours based on task complexity. Our code and dataset are available at the Repository.

4.3 Results Analysis

4.3.1 How do existing LLMs perform?

The EIFBENCH evaluation, detailed in Tables 4, challenges language models by simulating real-world scenarios across three datasets: plain text tasks, dialogue tasks, and multi-party dialogue tasks. These datasets reflect diverse practical applications, with plain text focusing on simple information processing, dyadic dialogues examining conversational dynamics, and multi-party dialogues showcasing collaborative discussions.

Our evaluation uses two key metrics: Instruction-Level Accuracy (ILA) and Constraint-Level Accuracy (CLA). Recent studies (Zhang et al., 2024a,b; Li et al., 2024) emphasize CLA, which measures models' effectiveness in meeting individual constraints with high accuracy. Yet, ILA reveals challenges, as models often fail to satisfy all constraints of a single instruction, resulting in a low probability of executing all instructions in an instance. This highlights the need to enhance multi-task capabilities for adhering to comprehensive instructions in the challenging contexts of the EIFBENCH dataset.

Model performance varies notably across categories, revealing task-type dependencies. In closed-source models, GPT-4o excels in ILA with relatively lower CLA. This indicates its capability to focus and complete individual sub-tasks effectively, albeit less so on fulfilling comprehensive constraints. In the realm of open-source models, Qwen2.5-72B-Instruct performs exceptionally well, balancing both instruction completion and constraint adherence. Among reasoning models, DeepSeek-R1 demonstrates robust competitiveness, effectively handling complex reasoning tasks. These findings emphasize the varying strengths of models across different task types and their alignment with specific task demands.

4.3.2 Effectiveness of SegPO

We implement the Group Relative Policy Optimization (GRPO) (Shao et al., 2024) framework, employing the overall reward r_o as the advantage

Model	Plain Text		Dyadic Dialogue		Multi-party Dialogue	
	ILA ↑	CLA ↑	ILA ↑	CLA ↑	ILA ↑	CLA ↑
Closed-Source LLMs						
GPT-4o	**0.2480**	0.6518	0.2166	0.5631	0.2226	0.5786
Claude-3.5-Sonnet	0.0896	0.3951	0.0919	0.4142	0.0663	0.3865
GPT-4o-mini	0.0826	0.5299	0.0930	0.4892	0.0952	0.6001
Claude-3.5-Haiku	0.0332	0.2214	0.0251	0.1613	0.0142	0.1081
gemini-1.5-Pro	0.1669	0.6705	0.2717	0.7461	0.1972	0.7693
Open-Source LLMs						
LLaMA3.1-8B-Instruct	0.0127	0.2918	0.0069	0.1845	0.0024	0.2898
LLaMA3.1-70B-Instruct	0.0222	0.3696	0.0250	0.3297	0.0156	0.3774
Qwen2-7B-Instruct	0.0261	0.3531	0.0269	0.2954	0.0136	0.3666
Qwen2-72B-Instruct	0.0823	0.5924	0.1336	0.6458	0.0878	0.6345
Qwen2.5-7B-Instruct	0.0503	0.5051	0.0742	0.5526	0.0572	0.5878
Qwen2.5-72B-Instruct	0.1983	0.7565	<u>0.2787</u>	0.7657	**0.2636**	**0.8308**
QwQ-32B	0.0884	0.4724	0.0909	0.4220	0.0820	0.5439
DeepSeek-R1	<u>0.2219</u>	0.6860	**0.3486**	**0.7906**	0.2251	0.7465
Qwen3-32B	0.2050	<u>0.7694</u>	0.2513	<u>0.7799</u>	<u>0.2299</u>	<u>0.8078</u>
Qwen3-32B w/o thinking	0.2073	**0.7703**	0.2396	0.7794	0.2119	0.7445
Qwen3-235B-A22B	0.1700	0.6712	0.2328	0.7296	0.2120	0.7462
Qwen3-235B-A22B w/o thinking	0.1775	0.6692	0.2252	0.7282	0.2011	0.7444

Table 4: Performance metrics across different task categories: Plain Text, Dyadic Dialogue, and Multi-party Dialogues. The best and second-best results are highlighted in bold and underlined.

Model	Plain Text		Dyadic Dialogue		Multi-party Dialogue	
	ILA ↑	CLA ↑	ILA ↑	CLA ↑	ILA ↑	CLA ↑
Qwen2.5-7B-Instruct	0.0503	0.5051	0.0742	0.5526	0.0572	0.5878
Qwen2.5-7B-Instruct w/ GRPO	0.1345	0.6237	0.1591	0.6393	0.2183	0.7392
Qwen2.5-7B-Instruct w/ SegPO	**0.1460**	**0.6693**	**0.1797**	**0.6791**	**0.2713**	**0.7727**

Table 5: SegPO Performance across different task categories compared to GRPO.

value to enhance model capabilities. As illustrated in Table 5, SegPO achieves significant improvements compared to base model and GRPO with respective 14.85% and 3.40% average increases, which confirms the effectiveness and necessity of segment-level advantage computation for accurate understanding of multiple task. The reason is that respectively calculating advantages for the response corresponding to each instruction in the input may result in a more precise reward, effectively steering the model's learning.

4.3.3 Full Constraint Satisfaction Analysis

In real-world scenarios, fully satisfying *all* constraints across *all* instructions is crucial especially for LLM agents with long-horizon decision-making involving multiple tasks, aside from ILA and CLA metrics. Our analysis revealed that the leading performance in dyadic dialogue was achieved by gemini-1.5-Pro and DeepSeek-R1, both scoring 0.0044, with GPT-4o following as the second-best at 0.0022. All other models recorded a performance score of zero. This relatively low performance highlights the increased difficulty posed by our benchmark, which, unlike previous datasets with limited constraints, is crafted to simulate realistic tasks such as smart home operations. These scenarios require handling multiple interdependent constraints simultaneously. The results indicate the current models' limitations in reasoning and executing complex, constraint-rich instructions, emphasizing the need for further advancements in their

Type	Human 1	Human 2	Human 3
Plain Text	0.9234	0.9342	0.9083
Dyadic Dialogue	0.9341	0.9268	0.9326
Multi-Party Dialogue	0.9118	0.9021	0.9164
Average	0.9231	0.9210	0.9191

Table 6: PCC Between `Qwen2.5-72B-Instruct` and expert evaluations on quality assessment.

Type	Human 1	Human 2	Human 3
Plain Text	0.7123	0.7236	0.7172
Dyadic Dialogue	0.7438	0.7632	0.7524
Multi-Party Dialogue	0.7551	0.7459	0.7376
Average	0.7371	0.7442	0.7357

Table 7: The kappa coefficient between expert evaluations and GPT-4o-as-Judge in the evaluation process.

capabilities.

4.4 Quality Assessment

We validated the benchmark's quality through both data generation and evaluation processes. First, we assessed the dataset from `Qwen2.5-72B-Instruct` by randomly selecting 50 instances, comparing model scores with evaluations from three experts for contradictions, redundancy, and infeasibility within instructions and constraints. The Pearson Correlation Coefficient (PCC) in Table 6 shows strong consistency, supporting benchmark credibility. Additionally, we validated LLM-judge evaluations by comparing them with human assessments across three datasets. We randomly selected 500 constraints per dataset based on LLM-generated responses and calculated Fleiss' Kappa scores (Fleiss, 1971) between the results from GPT-4o-as-judge and human evaluators. High consistency in Table 7 confirms the reliability of our evaluation process.

5 Related work

5.1 Instruction Following

Recent advancements in fine-tuning large language models (LLMs) show that annotated instructional data significantly enhances models' ability to comprehend and execute diverse language instructions (Weller et al., 2020; Ye and Ren, 2021; Mishra et al., 2022). Building on this, incorporating more detailed and sophisticated instructions has been shown to further improve model capabilities (Lou et al., 2023). For instance, (Xu et al., 2024) presents a method of incrementally generating complex instructions from seed instructions using LLMs, enabling LLaMA to surpass 90% of ChatGPT's performance in 17 out of 29 skills. Additionally, research is increasingly focusing on constrained instructions (Sun et al., 2024; Dong et al., 2024; He et al., 2024a), a subset of complex instructions, aimed at enhancing models' ability to handle intricate challenges by increasing instructional constraints.

5.2 Evaluation of Instruction Following

Instruction following significantly impacts the effectiveness of large language models (LLMs) (Liu et al., 2023). Early work focused on evaluating compliance with simple directives, often involving single constraints like semantic (Zheng et al., 2023; Liu et al., 2024) or formatting (Xia et al., 2024; Tang et al., 2024) requirements. As LLMs find their way into more complex real-world applications, the need to assess their capacity to handle sophisticated instructions has grown (Qin et al., 2024; Jiang et al., 2024). For example, (Sun et al., 2024) introduced the Conifer dataset to enhance LLMs' handling of multi-level instructions with complex constraints, while (Qin et al., 2024) designed a method for decomposing single instructions into multiple constraints. Moreover, (He et al., 2024b) created benchmarks using real-world constraints, and (Wen et al., 2024) further innovated by integrating diverse constraint types. Despite these advancements, current datasets often lack the extensive constraints seen in multi-instruction, multi-constraint real-world scenarios.

6 Conclusion

In conclusion, this study introduces the **E**xtremely Complex **I**nstruction **F**ollowing **Bench**mark (**EIFBENCH**), addressing existing single-task dataset limitations by incorporating multi-task scenarios and constraints for realistic evaluation of large language models (LLMs). We also propose the **Seg**ment **P**olicy **O**ptimization (**SegPO**) algorithm algorithm, which enhances LLMs' multi-task workflow execution, showing a 14.85% improvement on EIFBENCH over `Qwen2.5-7B-Instruct`. Evaluations reveal significant performance gaps, highlighting the need for models capable of tackling real-world complexities. This benchmark sets a new standard, steering future research toward developing robust and adaptable systems for practical applications.

7 Limitations

While EIFBENCH provides a robust evaluation framework for plain text, dyadic dialogue, and multi-party tasks, it has two limitations that could be addressed in future work. First, the inter-task relationships could be further enhanced to reflect more complex, real-world dependencies, such as multi-step reasoning or conditional task execution. Second, the dataset currently focuses primarily on Chinese instructions, which limits its applicability to multilingual scenarios. Expanding to include more languages would improve its global relevance and enable evaluation of LLMs' cross-lingual capabilities. Addressing these limitations would make EIFBENCH even more comprehensive and aligned with practical applications.

References

Minttu Alakuijala, Ya Gao, Georgy Ananov, Samuel Kaski, Pekka Marttinen, Alexander Ilin, and Harri Valpola. 2025. Memento no more: Coaching ai agents to master multiple tasks via hints internalization. *arXiv preprint arXiv:2502.01562*.

AI Anthropic. 2024a. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1.

AI Anthropic. 2024b. Claude 3.5 sonnet model card addendum. *Claude-3.5 Model Card*, 3(6).

Azam Bakhshandeh and Seyed Mohammad Javad Mirzapour Al-e-hashem. 2024. A multi-objective scheduling model in medical tourism centers considering multi-task staff training. *Eng. Appl. Artif. Intell.*, 131:107808.

Xinyi Chen, Baohao Liao, Jirui Qi, Panagiotis Eustratiadis, Christof Monz, Arianna Bisazza, and Maarten de Rijke. 2024. The sifo benchmark: Investigating the sequential 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 1691–1706. Association for Computational Linguistics.

Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. 2023. INSTRUCTEVAL: towards holistic evaluation of instruction-tuned large language models. *CoRR*, abs/2306.04757.

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.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. 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*.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Jie Gao, Simret Araya Gebreegziabher, Kenny Tsu Wei Choo, Toby Jia-Jun Li, Simon Tangi Perrault, and Thomas W. Malone. 2024. A taxonomy for human-llm interaction modes: An initial exploration. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems, CHI EA 2024, Honolulu, HI, USA, May 11-16, 2024*, pages 24:1–24:11. ACM.

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.

Qianyu He, Jie Zeng, Wenhao Huang, Lina Chen, Jin Xiao, Qianxi He, Xunzhe Zhou, Jiaqing Liang, and

Yanghua Xiao. 2024b. Can large language models understand real-world complex instructions? In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 18188–18196. AAAI Press.

Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Qianli Ma, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, Jianbo Yuan, Jiwei Li, Kun Kuang, Yang Yang, Hongxia Yang, and Fei Wu. 2024. InfiAgent-DABench: Evaluating agents on data analysis tasks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 19544–19572. PMLR.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

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.

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.

Yong Li, Xuan-Yu Jiao, Bai-Qing Sun, Qiu-Hao Zhang, and Junyou Yang. 2021. Multi-welfare-robot cooperation framework for multi-task assignment in healthcare facilities based on multi-agent system. In *IEEE International Conference on Intelligence and Safety for Robotics, ISR 2021, Tokoname, Japan, March 4-6, 2021*, pages 413–416. IEEE.

Xiao Liu, Xuanyu Lei, Shengyuan Wang, Yue Huang, Andrew Feng, Bosi Wen, Jiale Cheng, Pei Ke, Yifan Xu, Weng Lam Tam, Xiaohan Zhang, Lichao Sun, Xiaotao Gu, Hongning Wang, Jing Zhang, Minlie Huang, Yuxiao Dong, and Jie Tang. 2024. Alignbench: Benchmarking chinese alignment of 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 11621–11640. Association for Computational Linguistics.

Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo, Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. 2023. Trustworthy llms: a survey and guideline for evaluating large language models' alignment. *CoRR*, abs/2308.05374.

Renze Lou, Kai Zhang, and Wenpeng Yin. 2023. A comprehensive survey on instruction following. *arXiv preprint arXiv:2303.10475*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3470–3487. Association for Computational Linguistics.

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

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.

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, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, James Molloy, Jilin Chen, Michael Isard, Paul Barham, Tom Hennigan, Ross McIlroy, Melvin Johnson, Johan Schalkwyk, Eli Collins, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Clemens Meyer, Gregory Thornton, Zhen Yang, Henryk Michalewski, Zaheer Abbas, Nathan Schucher, Ankesh Anand, Richard Ives, James Keeling, Karel Lenc, Salem Haykal, Siamak Shakeri, Pranav Shyam, Aakanksha Chowdhery, Roman Ring, Stephen Spencer, Eren Sezener, and et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *CoRR*, abs/2403.05530.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon

Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv: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.

Xiangru Tang, Yiming Zong, Jason Phang, Yilun Zhao, Wangchunshu Zhou, Arman Cohan, and Mark Gerstein. 2024. Struc-bench: Are large language models good at generating complex structured tabular data? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Short Papers, NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 12–34. Association for Computational Linguistics.

Connor J Taylor, Kobi C Felton, Daniel Wigh, Mohammed I Jeraal, Rachel Grainger, Gianni Chessari, Christopher N Johnson, and Alexei A Lapkin. 2023. Accelerated chemical reaction optimization using multi-task learning. *ACS Central Science*, 9(5):957–968.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

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 2020, Online, November 16-20, 2020*, pages 1361–1375. 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. 2024. Benchmarking complex instruction-following with multiple constraints composition.

Congying Xia, Chen Xing, Jiangshu Du, Xinyi Yang, Yihao Feng, Ran Xu, Wenpeng Yin, and Caiming Xiong. 2024. FOFO: A benchmark to evaluate llms' format-following capability. 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 680–699. Association for Computational Linguistics.

Fu Xiong, Cang Yeliang, Zhu Lipeng, Hu Bin, Deng Song, and Wang Dong. 2016. Deadline based scheduling for data-intensive applications in clouds. *The Journal of China Universities of Posts and Telecommunications*, 23(6):8–15.

Guangzhi Xiong, Qiao Jin, Xiao Wang, Yin Fang, Haolin Liu, Yifan Yang, Fangyuan Chen, Zhixing Song, Dengyu Wang, Minjia Zhang, et al. 2025. Rag-gym: Optimizing reasoning and search agents with process supervision. *arXiv preprint arXiv:2502.13957*.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024a. Qwen2 technical report. *CoRR*, abs/2407.10671.

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, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024b. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Qinyuan Ye and Xiang Ren. 2021. Learning to generate task-specific adapters from task description. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021*, pages 646–653. Association for Computational Linguistics.

Lijun Zhang, Xiao Liu, and Hui Guan. 2022. Automtl: A programming framework for automating efficient multi-task learning. 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*.

Tao Zhang, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Tao Zhang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, Bin Cui, Wentao Zhang, and Zenan Zhou. 2024a. Cfbench: A comprehensive constraints-following benchmark for llms. *CoRR*, abs/2408.01122.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. 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*.

Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. 2021. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 2856–2878. Association for Computational Linguistics.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. Agieval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 2299–2314. Association for Computational Linguistics.

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 Taxonomy of Constraint

We present the taxonomy of constraint in Table 8.

B Detailed Information on SegPO

The ratio $r_{i,t}(\theta)$ represents the probability ratio or importance sampling weight between the new policy π_θ and the old policy $\pi_{\theta_{\text{old}}}$:

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})}, \tag{10}$$

and SegPO estimates the KL divergence with the following unbiased estimator:

$$\mathbb{D}_{\text{KL}}\left[\pi_\theta || \pi_{\text{ref}}\right] = \frac{\pi_{\text{ref}}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} - \log \frac{\pi_{\text{ref}}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} - 1. \tag{11}$$

C Experiment Analysis

C.1 Factors Influencing Instruction Following

To conduct our investigation, we sampled both open-source and closed-source models with varying performance levels—some exemplary and others average—and visualized their results. Our investigation identifies two critical dimensions influencing instruction adherence in language models: (1) the number of instructions per instance and (2) the number of constraints per instruction. As illustrated in Fig. 6, performance degrades progressively as these variables increase, though with minor patterns. This decline is particularly pronounced with an increase in constraints, likely because each additional constraint raises the complexity of completing a task, making it more challenging for the model to meet all requirements. Conversely, the interdependence between instructions is generally low, meaning that an increase in the number of instructions does not lead to as steep a performance decline. This is primarily because the difficulty lies in managing multiple tasks simultaneously, rather than the instructions themselves being interrelated. In some instances, especially where there are larger numbers of instructions and constraints, performance may inexplicably improve. This can be attributed to the smaller sample sizes in these scenarios, leading to greater variability in performance outcomes. Overall, this analysis underscores the intricacies of maintaining consistent instruction adherence across diverse scenarios.

C.2 Performance from state-of-art LLMs

We employ other state-of-the-art large language models, i.e., gemini-2.0-Flash, as external evaluators to assess performance. The results are shown in Table 9. Their evaluations generally align with those of GPT-4o-as-judge, demonstrating consistent outcomes.

D Data Instance

In this section, we present an example instance to illustrate the application and analysis of the idiom.

Instruction_0: "Explain the origin and significance of the Chinese idiom 'drawing legs on a snake' (huà shé tiān zú)"

Constraints:

- Must provide a detailed account of the idiom's historical background and origin.
- Avoid using the words "meaning" or "explanation" to describe its significance.
- Follow the context → story → implications structure.

Instruction_1: "Create a sentence containing the idiom 'drawing legs on a snake' "

Constraints:

- The sentence must be 20-30 Chinese characters long.
- The sentence must be non-declarative (e.g., rhetorical question, exclamation, or imperative).

Instruction_2: "Analyze the specific scenario of 'drawing legs on a snake' in your created sentence."

Constraints:

- Describe in detail the superfluous action within the scenario.
- Include a root-cause analysis of why this "unnecessary addition" leads to negative consequences.

Constraint Type	Constraint Dimension
Content Constraint	Theme Constraint Exclusion Constraint Inclusion Constraint Value Constraint Privacy Constraint Numerical Constraint
Situation Constraint	Role-Playing Constraint Target Audience Constraint Prior Condition Constraint Natural Language Process Background Information Constraint Markdown Process Background Information Constraint Table Background Information Constraint Text Background Information Constraint
Style Constraint	Tone and Style Constraint Emotion Constraint Linguistic Characteristics Constraint Multilingual Constraint
Format Constraint	Output Format Constraint Text Pattern Constraint Grammar Structure Constraint Citation Constraint Numbering and List Constraint Hierarchical Structure Constraint Template Constraint

Table 8: Constraints and Their Dimensions

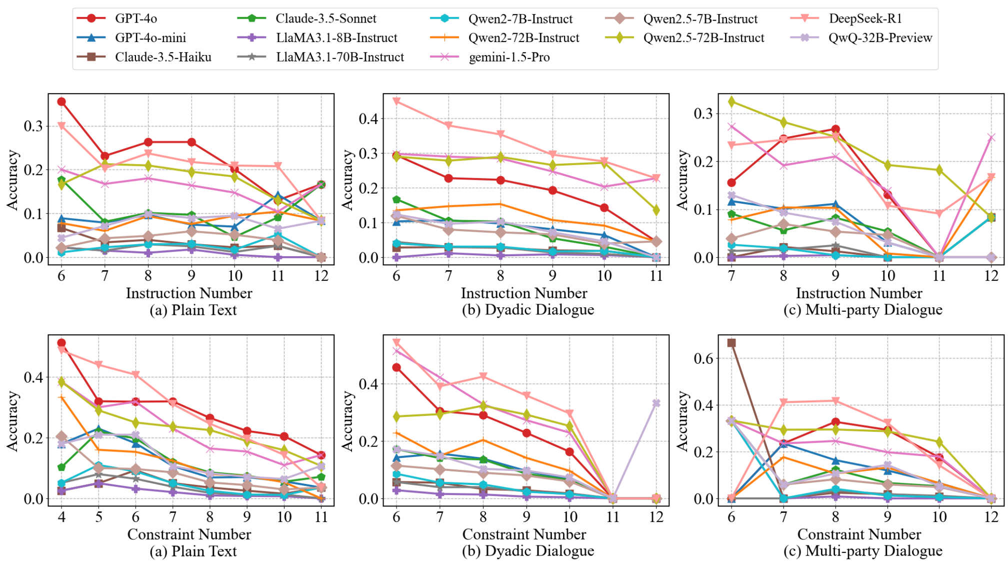

Figure 6: Performance on different numbers of instructions and constraints.

Model	Plain Text		Dyadic Dialogue		Multi-party Dialogues	
	ILA ↑	CLA ↑	ILA ↑	CLA ↑	ILA ↑	CLA ↑
Closed-Source LLMs						
GPT-4o	**0.3248**	0.6941	0.2273	0.6100	0.2641	0.6155
Claude-3.5-Sonnet	0.1283	0.4235	0.0800	0.4860	0.0988	0.4200
GPT-4o-mini	0.1263	0.5898	0.0892	0.5588	0.1197	0.6432
Claude-3.5-Haiku	0.0463	0.2425	0.0365	0.2379	0.0295	0.1288
gemini-1.5-Pro	0.2001	0.7018	0.1732	0.7349	0.1920	0.7747
Open-Source LLMs						
LLaMA3.1-8B-Instruct	0.0293	0.3954	0.0184	0.3357	0.0149	0.4151
LLaMA3.1-70B-Instruct	0.0502	0.4616	0.0439	0.5337	0.0451	0.5086
Qwen2-7B-Instruct	0.0475	0.4371	0.0346	0.4237	0.0321	0.4731
Qwen2-72B-Instruct	0.1300	0.6688	0.1272	0.6847	0.1173	0.6941
Qwen2.5-7B-Instruct	0.0767	0.5677	0.0654	0.6224	0.0570	0.6310
Qwen2.5-72B-Instruct	0.2182	0.7656	0.1732	0.7395	0.1678	0.7887
Qwen3-32B	<u>0.2826</u>	<u>0.8002</u>	<u>0.2702</u>	<u>0.7924</u>	**0.2894**	**0.8328**
Qwen3-32B w/o think	0.2724	**0.8035**	**0.2726**	**0.7929**	0.2559	<u>0.8246</u>
Qwen3-235B-A22B	0.2382	0.6949	0.2624	0.7527	0.2499	0.7518
Qwen3-235B-A22B w/o think	0.2435	0.6951	0.2571	0.7442	<u>0.2645</u>	0.7651
QwQ-32B-Preview	0.1017	0.5203	0.0640	0.4632	0.0840	0.5659
DeepSeek-R1	0.2449	0.6985	0.1797	0.7390	0.2060	0.7412

Table 9: Performance metrics across different task categories: Plain Text, Dyadic Dialogue, and Multi-party Dialogue with gemini-2.0-Flash as evaluator. The best and second-best results are highlighted in bold and underlined.

E Prompt

E.1 Prompt for task expansion

You are an assistant to help generate comprehensive multi-task tasks from basic tasks/basic texts/basic dialogues. Based on the given basic task, please design 5-10 different types of extended tasks, which must be reasonable and meet actual needs. The generated tasks should be placed after “-output-:”.

Please follow these rules when generating tasks:

1. Task design must be based on the input text content or the already designed task output content.

2. Task instructions should be clear and specific.

3. Each task should include explicit output format requirements.

5. Aim to increase task difficulty, selecting tasks that require multi-step reasoning and thinking.

6. Tasks should be related; specific task details can vary. The connections can be selective, sequential, parallel, etc. At least three types of connections are needed, including:
A. Parallel Task Mode: Analyzing multiple dimensions simultaneously
B. Sequential Task Mode: Task chain dependency
C. Conditional Selection Mode: Branch based on different situations, considering possible branches of the task, and design different tasks for different branches
D. Nested Task Mode: Hierarchical task structure
7. Task Design Principles:
- Clear goals
- Clear instructions
- Specific steps
- Standardized format
- Evaluability
8. Task types may repeat, but task content may not.

9. Note that expansion must be based on the given basic task, expanding into richer, more comprehensive, and varied integrated tasks. The text material in the given basic task must be retained, as subsequent tasks will all involve it!

10. Write the extended tasks after “-output-:”, and the thought process and analysis for generating the extended tasks after “-explanation-:”.

-input-:
{text}

-output-:

-explanation-:
...

Task-type examples should be chosen from the following categories. The specific task examples are listed in each task category. Note that you are only required to design tasks, not provide example outputs:

1. Classification Task
- Sentiment Analysis
- Text Classification
- Toxic Content Detection
- Empathy Detection
- Stereotype Detection
- Social Norm Judgment

2. Information Extraction
- Named Entity Recognition
- Keyphrase Annotation
- Coreference Resolution
- Entity Relationship Classification

3. Text Generation - Story Creation
- Poetry Generation
- Recipe Generation
- Outline Generation
- Text Expansion/Compression
- Title Generation
- Data Description Generation
- Text Rewriting/Simplification

4. Dialogue Systems
- Dialogue Generation
- Intent Recognition
- Question Generation/Rewriting
- Dialogue State Tracking
- Role-playing Dialogue

5. Reasoning and Logic
- Common Sense QA
- Multi-hop QA
- Critical Thinking Judgment
- Mathematical Reasoning
- Theory of Mind Reasoning

6. Language Style
- Style Transfer
- Language Detection
- Sarcasm Detection
- Spelling/Punctuation Error Detection

7. Evaluation and Verification
- Text Quality Evaluation
- Fact-checking
- Answer Verification
- Uncertainty Judgment

8. Programming-related
- Code Generation/Debugging

- Code Explanation
- Code Translation

Each example format is as follows:
Task x:
Type:
Specific Requirements:
- ...
- ...

E.2 Prompt for task revision

You are a task optimization expert. Please analyze and optimize the given task set.

Input text and tasks are as follows:

--input--:
query: {input_text}
task: {task}

First, output all optimized tasks (if there are no modifications, output the original tasks) in Chinese after “-output-”. Secondly, write the optimized rationale and analysis process after“-explanation-”. Please strictly follow this format.

The output format is as follows:

-output-:
Task 1: ...
Task 2: ...

-explanation-:

Please follow these steps for analysis and optimization:
1. Input Analysis
Input Type Judgment:
- Determine if it’s a complete text or a task requirement
- Check if it includes a creative/analytical directive
- Assess the amount of information provided by the text/task
- Preserve textual information if input text analysis is involved

2. Task Reasonability Check Analyze each task:
Reasonability of Task X:
A. Matching Degree with Input
- Does the task rely on the actual provided information, and is there excessive speculation or extension?

B. Executability of the Task
- Is there sufficient information to support it, and are the scoring criteria operable?

C. Existing Problems
- [List specific problems]

3. Modification Suggestions
If modification is required, provide specific modification directions and design of revised tasks, and modify the tasks according to this suggestion.

E.3 Prompt for task combination

You are an integration assistant for input and task requirements. Your goal is to combine the basic tasks (including tasks and reading materials) given in “–input–”: and the expanded tasks in “–task–”: to generate comprehensive tasks that include reading material information and task information. Please note that the integrated tasks will not fetch text information from elsewhere, so ensure that the generated comprehensive tasks include the text material. Please integrate all tasks from the expanded tasks into the comprehensive tasks. Identify all expanded tasks and ensure the number of sub_instructions matches the number of sub-tasks in the expanded tasks. Please ensure to extract and integrate the materials and text information involved in –input–:, and do not omit any details.

--input--:
{input_text}
--tasks--:
{task}

Please put the generated content in Chinese after “-output-:”, including: 1. First, place the comprehensive task and the text materials involved in the task after “INSTRUCTION:”. Please ensure to fully include the reading materials from the basic tasks. 2. Then, sequentially output all sub-instructions, each starting with “SUB_INSTRUCTION_X:”, including “instruction:” and “constraints:” parts. After “instruction:”, write the content of the sub-instruction, and after “constraints:”, write several constraint items. Each constraint should follow the format “- constraint content [constraint type]”. Lastly, provide the specific combination process after “-explanation-:”.

The output format is as follows:

-output-:
INSTRUCTION:
...
SUB_INSTRUCTION_x:
instruction: ...
constraints:
- ... [...]
- ... [...]

--explaination--:
...

Please follow these steps for analysis and combination:

1. Input Analysis
- Extract **original tasks** and corresponding **text materials** from −input−
- Extract specific text content and basic requirements
- Extract all specific requirements from the input text
- If the input involves text, it must be placed in the comprehensive task to avoid missing the input text

2. Task Expansion Analysis
- Extract **sub-task** information (information type, information volume, target) from −task−
- Retrieve related expanded tasks (such as information extraction, reasoning, etc.)
- Understand the relevance and progression relationship between tasks
- Identify all constraints and restrictions
- Record keywords and special conditions
- Note that all tasks are prefixed with "Task", be sure to identify all tasks, and the number of tasks should match the number of sub_instructions
- List all tasks and their sub-tasks

3. Combine into a New Comprehensive Task
- Expand **original tasks**, **text materials**, and all **sub-tasks** into a comprehensive analysis task, and output to INSTRUCTION
- Maintain logical connections between tasks
- Ensure the INSTRUCTION meets all sub-task requirements
- Make sure to integrate all tasks and incorporate the input
- Ensure all original requirements are covered

4. Integrated Output
- A unified main instruction, output the new comprehensive task (INSTRUCTION):
Use natural language connectors, the task requirements should be connected with natural language, such as "then", "next", "finally", to maintain fluency
Ensure the integration of input and task, the combined content should be output to INSTRUCTION, forming a coherent and smooth instructional language
Ensure all requirements are covered

- A series of sub-instructions (SUB_INSTRUCTION)
Each sub-instruction contains specific tasks (instruction)
Each sub-instruction includes specific constraints (constraints), generating 5-10 specific constraints
The purpose of the constraints is to complete the task as much as possible, the more detailed, the better, with difficulty ranging from simple to complex
Note that constraints should not be ambiguous or unclear
Each constraint and its type should be selected from the following 24 types:

- Theme Constraint
- Exclusion Constraint
- Inclusion Constraint
- Value Constraint
- Privacy Constraint
- Numerical Constraint
- Role-Playing Constraint
- Target Audience Constraint
- Prior Condition Constraint
- Natural Language Process Background
- Markdown Process Background
- Table Background Information
- Text Background Information
- Tone and Style Constraint
- Emotion Constraint
- Linguistic Characteristics
- Multilingual Constraint
- Output Format Constraint
- Text Pattern Constraint
- Grammar Structure Constraint
- Citation Constraint
- Numbering and List Constraint
- Hierarchical Structure Constraint
- Template Constraint

Precautions:
1. Sub-tasks must be clearly mentioned in the integrated instruction.
2. Do not change the wording and expressions of the original instructions.
3. Split according to the order in which the tasks appear in the instructions.
4. Each sub-task is equipped with 5-10 constraint items, with constraint types selected from the above 24 types.
5. When integrating, ensure that new tasks are organically combined with the original content, i.e., do not generate instructions based only on information from input.

E.4 Prompt for constraint expansion

You are an expert at generating constraints.
Please modify the original constraint information for each instruction.
For every SUB_INSTRUCTION, generate 6-10 high-quality constraints.
Each constraint must address key requirements of the task with measurable analysis rather than general statements.

Input information is as follows:

--input--:
{input_text}

When outputting content, please place the generated content after “-output-” first.
Begin with “INSTRUCTION”, followed by each sub-instruction sequentially, with each sub-instruction starting with “SUB_INSTRUCTION_X”, including both “instruction” and “constraints”.
After “instruction”, write the content of the sub-instruction, and after “constraints”, write several constraint items.
Each constraint should follow the format “- constraint content [constraint type]”.

Finally, place the analysis of the modification process after -explanation-.

The generated format is as follows:

```
-output-:
INSTRUCTION:
...

SUB_INSTRUCTION_0:
instruction: ...
constraints:
- ... [...]
- ... [...]
...

SUB_INSTRUCTION_1:
instruction: ...
constraints:
- ... [...]
...

--explaination--:
...
```

Specific modification requirements:
1. Each constraint must be specific and clear, avoiding vague expressions, and the constraint structure should use “and,” “or,” “not” types.
2. Each constraint must include measurable standards, such as specific numbers, clear criteria, etc.
Also, note that these constraints are for the model to follow, avoiding situations that are impossible to assess, such as “Please respond within 5 seconds after reading,” which cannot be evaluated for compliance.
3. Avoid generic vocabulary; examples below:
Avoid using generic words often found in constraints:

Quality Descriptors:
“appropriate,” “suitable,” “adequate,” “sufficient,” “complete,” “detailed,” “accurate,” “clear,” “varied”

Logical Descriptors:
“logicality,” “coherent,” “orderly,” “hierarchical,” “structured,” “systematic”

Effect Descriptors:
“comprehensive,” “practical,” “vivid,” “specific,” “pictorial,” “persuasive,” “effective,” “helpful”

Standard Descriptors:
“meets requirements,” “standard-compliant,” “sufficient,” “as stipulated,” “qualified,” “standard fit”

Feature Descriptors:
“characteristic,” “feature,” “prominent,” “obvious,” “outstanding”

These words should be replaced with specifically measurable standards, for example:
“suitable” -> “must include 3 specific examples”
“detailed” -> “no less than 100 words”
“vivid” -> “must use more than 3 figures of speech”
“logicality” -> “must follow [cause-process-result] order”
“persuasive” -> “must cite 1 authoritative data source”

4. Each constraint must be of one of the following types:

- Theme Constraint
- Exclusion Constraint
- Inclusion Constraint
- Value Constraint
- Privacy Constraint
- Numerical Constraint
- Role-Playing Constraint
- Target Audience Constraint
- Prior Condition Constraint
- Natural Language Process Background
- Markdown Process Background
- Table Background Information
- Text Background Information
- Tone and Style Constraint
- Emotion Constraint
- Linguistic Characteristics
- Multilingual Constraint
- Output Format Constraint
- Text Pattern Constraint
- Grammar Structure Constraint
- Citation Constraint
- Numbering and List Constraint
- Hierarchical Structure Constraint
- Template Constraint

E.5 Prompt for constraint revision

You are an assistant for modifying constraints.
Please analyze the original constraint information in the instruction for potential issues, and modify the constraints for each SUB_INSTRUCTION to generate 6-10 high-quality constraints.

Each constraint must address specific, measurable requirements for the task, rather than general statements.

--input--:
{input_text}

When outputting content, first combine the modification analysis process and output the modified content (if no modifications, output the original content) after “-output-”, including INSTRUCTION and the modified SUB_INSTRUCTION information, where each SUB_INSTRUCTION consists of instruction and constraints.
Each constraint should follow the format “- specific constraint content [constraint type]”.
Finally, provide the analysis of the modification process after -explanation-.

The generated format is as follows:

```
-output-:
INSTRUCTION:
...

SUB_INSTRUCTION_x:
instruction: ...
constraints:
- ... [...]
- ... [...]
...

-explanation-:
...
```

Please follow these steps for analysis and modification:

1. Examine each instruction for the following 8 types of issues and modify any issues found

1.1 Vague constraints/lack of specific evaluation metrics need to be detailed into evaluable metrics
Example:
Original constraint:
- The article structure must be reasonable

Modified to:
- The article must include introduction, analysis, and conclusion sections, with each section not less than 200 words

Below are frequently used vague words that should be avoided:
Quality Descriptors: “appropriate”, “suitable”, “adequate”, “sufficient”, “complete”, “detailed”, “accurate”, “clear”, “varied”
Logical Descriptors: “logicality”, “coherent”, “orderly”, “hierarchical”, “structured”, “systematic”
Effect Descriptors: “comprehensive”, “practical”, “vivid”, “specific”, “pictorial”, “persuasive”, “effective”, “helpful”
Standard Descriptors: “meets requirements”, “standard-compliant”, “sufficient”, “as stipulated”, “qualified”, “standard fit”
Feature Descriptors: “characteristic”, “feature”, “prominent”, “obvious”, “outstanding”

First, check if any vague words appear in the constraints, then refine the vague constraints into evaluable metrics based on the specific task context. Here are some examples:
“suitable” -> “must include 3 specific examples”
“detailed” -> “no less than 100 words”
“vivid” -> “must use more than 3 figures of speech”
“logicality” -> “must follow [cause-process-result] order”
“persuasive” -> “must cite 1 authoritative data source”
“rich emotional color” -> Use at least two rhetorical devices (parallelism, contrast, metaphor, personification, or exaggeration) to express emotions

1.2 Duplicate constraints need to be distinguished
Example:
Original constraint:
- Must use formal language
- Must use standard language
Problem: The two constraints are similar and lack distinction
Modification suggestion:
- Must use honorific words like “you, your, respectfully”
- Must avoid using interjectory words like “oh, ah, um”

1.3 Logical contradictions
Example:
Original constraint:
- Both “relaxed and gentle” and “professional terminology” require a remedy
Suggested modification:
- The tone must be friendly and professional, with easy-to-understand explanations provided for professional terminology

1.4 Lack of key constraints
Example:
E-commerce customer service scenario
Suggested modification:
- Must explain the shop’s specific compensation plan
- Must provide direct contact details for customer service
- Must specify the follow-up timeline

Original constraint:
- Modify according to the following format
Modification suggestion:

- Modify according to the table format

1.5 Contradictory constraints:
Original constraint:
- Requires classical Chinese style
- Requires vividness
Suggestion: Adjust to:
- Use classical vocabulary but ensure modern readers can understand
- Provide modern explanations for each term

1.6 Lack of key definitions:
Original constraint:
- The calculation of the number of "events" lacks a clear definition
Suggestion: Add:
- Clearly define "event" as "an independent action and its corresponding object"
- Provide specific examples for event judgment

1.7 Data source missing
Original constraint:
- "Must include specific data or factual references"
- "Must be based on specific data and facts"
But no instructions on how to obtain and verify data sources
Suggestion: Add data source requirements:
- "Must cite authoritative market research agencies or official publications, and specify the source"
- "Data must be from statistical results within the past 2 years"

1.8 Evaluation criteria unclear:
Original constraint:
- "Applicable scenarios must be reasonable and consistent with market reality"
- "Usage suggestions must be specific and feasible"
But no evaluation criteria provided
Suggestion: Set specific evaluation indicators:
- "Each suggestion must include usage scenarios, expected effects, and cost considerations"
- Add feasibility verification:
- "Each suggestion must be supported by actual cases"

2. Constraint types should be selected from the following 24 categories while varying the types as much as possible to enrich the diversity of the constraints:

- Theme Constraint
- Exclusion Constraint
- Inclusion Constraint
- Value Constraint
- Privacy Constraint
- Numerical Constraint
- Role-Playing Constraint
- Target Audience Constraint
- Prior Condition Constraint
- Natural Language Process Background
- Markdown Process Background
- Table Background Information
- Text Background Information
- Tone and Style Constraint
- Emotion Constraint
- Linguistic Characteristics
- Multilingual Constraint
- Output Format Constraint
- Text Pattern Constraint
- Grammar Structure Constraint
- Citation Constraint
- Numbering and List Constraint
- Hierarchical Structure Constraint
- Template Constraint

E.6 Prompt for constraint combination

Now you are an assistant in integrating tasks and constraints; please help me optimize this comprehensive task's instruction (INSTRUCTION), sub-instructions (SUB_INSTRUCTION), and constraints.
Please ensure that the input information/reading materials in the INSTRUCTION are retained; otherwise, subsequent tasks cannot be completed.

Input information is as follows:

--input--:
{input_text}

First, output the modified comprehensive task content (if no modifications, output the original content) after "-output-", including INSTRUCTION and the modified SUB_INSTRUCTION information. Each SUB_INSTRUCTION consists of instruction and constraints, with each constraint structured as follows: "- specific content [constraint type]".
Finally, put the specific modification analysis process after -explanation-.

The output format is as follows:

```
-output-:
INSTRUCTION:
...

SUB_INSTRUCTION_0:
instruction: ...
constraints:
- ... [...]
- ... [...]
...

SUB_INSTRUCTION_1:
instruction: ...
constraints:
```

- ... [...]
- ... [...]
...

-explanation-:
...

Please follow these steps for analysis and modification:

1. Analyze the existing instructions and constraints for issues:
- Check if the structure is reasonable
- Identify duplicate or contradictory requirements
- Discover vague or non-executable constraints
- Find missing key requirements

2. Provide update suggestions:
- Instruction update: Make it clearer and more targeted for the comprehensive task
- Sub-instruction update: Specify each atomic task
- Constraint update: Provide executable and verifiable constraints
- Start with -output-, output the modified instructions (INSTRUCTION), sub-instructions (SUB_INSTRUCTION), and constraints
- Each constraint includes two parts: content [type]

3. When modifying, be sure to keep the input information such as reading materials in the original INSTRUCTION. Do not delete specific query information, causing text errors.

4. Each constraint type must be one of the following, if it is not among these types, please modify the constraint type to one of the following types. If it cannot be modified, regenerate constraints that meet these types:

- Theme Constraint
- Exclusion Constraint
- Inclusion Constraint
- Value Constraint
- Privacy Constraint
- Numerical Constraint
- Role-Playing Constraint
- Target Audience Constraint
- Prior Condition Constraint
- Natural Language Process Background
- Markdown Process Background
- Table Background Information
- Text Background Information
- Tone and Style Constraint
- Emotion Constraint
- Linguistic Characteristics
- Multilingual Constraint
- Output Format Constraint
- Text Pattern Constraint
- Grammar Structure Constraint
- Citation Constraint
- Numbering and List Constraint
- Hierarchical Structure Constraint
- Template Constraint

E.7 Prompt for instruction-level validation

You are now an assistant to modify sub-tasks.
You need to modify the given sub-tasks according to the following steps:

1. Analyze the relationship between sub-tasks and evaluate their role in the comprehensive task, removing contradictory sub-tasks.
2. Note that sub-tasks are carried out by a large model, so remove tasks that the large model cannot complete, such as internet searches, finding related information, statistical data analysis, etc.
3. Delete tasks with weak logical connections. The relationships between sub-tasks can be: A. Parallel Task Mode: Analyzing multiple dimensions simultaneously
B. Serial Task Mode: Chain-dependent tasks
C. Conditional Selection Mode: Branching based on different situations, considering possible branches of a task, and designing different tasks for different branches
D. Nested Task Mode: Hierarchical task structure
4. Sub-task selection criteria: - Remove tasks that an AI model cannot accomplish (such as network searches, finding information)
- Remove tasks with weak logical connections
- Remove redundant, contradictory, or unreasonable tasks
- Optimize sub-task content to be of moderate difficulty and meet practical needs
- It is acceptable to propose some creative tasks
- Ensure that the number of generated sub-tasks is between 6 and 14
- Try to diversify task types, with at least 3 different styles of tasks
- Remove tasks that require an AI model to use tools, such as Named Entity tools, etc.

5. Select the main task categories from the following, and the directions under each category can be diversified: 1. Classification Task
- Sentiment Analysis
- Text Classification
- Toxic Content Detection
- Empathy Detection
- Stereotype Detection
- Social Norm Judgment

2. Information Extraction
- Named Entity Recognition

- Keyphrase Annotation
- Coreference Resolution
- Entity Relationship Classification

3. Text Generation - Story Creation
- Poetry Generation
- Recipe Generation
- Outline Generation
- Text Expansion/Compression
- Title Generation
- Data Description Generation
- Text Rewriting/Simplification

4. Dialogue Systems
- Dialogue Generation
- Intent Recognition
- Question Generation/Rewriting
- Dialogue State Tracking
- Role-playing Dialogue

5. Reasoning and Logic
- Common Sense QA
- Multi-hop QA
- Critical Thinking Judgment
- Mathematical Reasoning
- Theory of Mind Reasoning

6. Language Style
- Style Transfer
- Language Detection
- Sarcasm Detection
- Spelling/Punctuation Error Detection

7. Evaluation and Verification
- Text Quality Evaluation
- Fact-checking
- Answer Verification
- Uncertainty Judgment

8. Programming-related
- Code Generation/Debugging
- Code Explanation
- Code Translation

Input Total Task:
{input_text}

Input Sub-tasks:
{sub_instruction}

6. First, output the modified comprehensive task content (if no modifications, output the original content) after -output-, including the modified INSTRUCTION and SUB_INSTRUCTION_x, where SUB_INSTRUCTION_x is formatted as 'sub-task content [task type]'.
Finally, place the specific modification analysis process after -explanation-.

-output-:
INSTRUCTION:
...

SUB_INSTRUCTION_0:
... [task type]

SUB_INSTRUCTION_1:
... [task type]

-explanation-:
...

E.8 Prompt for constraint-level validation

You are a constraint evaluation assistant.
Your task is to determine whether the given constraints can be completed by a large model. Please evaluate according to the following rules:

1. **Input**:
- Constraint content: A segment of text describing the task requirements.
- Input dialogue: A segment of the user's conversation with the model.

2. **Evaluation Rules**:
- If the input dialogue **lacks the critical information needed to fulfill the constraint** (e.g., the constraint requires extracting person entities, but no person is mentioned in the dialogue), then output "No".
- If the constraint **goes beyond the model's capability** (e.g., needs real-time data or external resources), then output "No".
- If the input dialogue provides sufficient information and the constraint falls within the model's capability, then output "Yes".
- If the model outputs "No", minimally modify the constraint content to make it feasible for the model to complete it.

3. **Output**:
- If the output is "No", provide the modified constraint content to make it feasible for the model.
- If the output is "Yes", no modification is needed.

4. **Examples**:
- Example 1:
- Instruction content: Extract entities
- Constraint content: Extract person entities from the dialogue.
- Input dialogue: User says, "Yesterday I went to the park with Xiaoming."
- Output: Yes
- Example 2:
- Instruction content: Extract entities
- Constraint content: Extract person entities from the dialogue.
- Input dialogue: User says, "The weather was great yesterday, and I went for a walk in the park."
- Output: No
- Reason: No person entities in the dialogue
- Modified content: If any person

```
entities are present, extract them.
- Example 3:
- Instruction content: Generate text
- Constraint content: Generate a
100-word text describing the summer
scenery, using at least 3 similes.
- Input dialogue: User says, “Please
write a passage about summer.”
- Output: Yes
- Example 4:
- Instruction content: Generate text
- Constraint content: Generate a
100-word text describing the summer
scenery, and cite at least 2 academic
papers.
- Input dialogue: User says, “Please
write a passage about summer.”
- Output: No
- Reason: Unable to cite academic papers
- Modified constraint: Generate a
100-word text describing the summer
scenery, using at least 3 similes.

5. **Task**:

   - Instruction content: {instruction}
   - Constraint content: {constraint}
   - Input dialogue: {input}
   - Output:
   - Reason:
   - Modified constraint:
```

E.9 Prompt for training process

```
You are now an AI assistant responsible
for generating answers to specified
tasks. You need to generate answers
following these requirements:

1. Strictly generate answers based
on the given input material and
corresponding sub_instruction.
2. Generate answers for each
sub_instruction, ensuring consistency
among answers to different
sub_instructions.
3. Follow the constraints of each
sub_instruction strictly to generate
answers.
4. First, think through each sub-task
in detail using analytical skills to
deeply understand the issues, and then
provide answers. The thought process
for each sub-task should be detailed
between start_think and end_think, and
the answer should be fully presented
between start_answer and end_answer.
5. The thought process and answer
for each sub_instruction should be
placed between start_sub_instruction_x
and end_sub_instruction_x, where
sub_instruction_x is the specific
identifier for the sub-task. Ensure
there are no extra spaces, quotes, or
symbols before and after these markers.

Notes: 1. Strictly adhere to the
constraints.
2. Ensure the quality of answers.
3. Do not output the input content.
4. The format is as follows:

start_sub_instruction_0
start_think
Deeply analyze this sub-task, ...
end_think
start_answer
Based on the above analysis, the detailed
answer to sub-task 0 is ...
end_answer
end_sub_instruction_0

start_sub_instruction_1
start_think
In this sub-task, consider various
factors, ...
end_think
start_answer
Based on the above analysis, the answer
to sub-task 1 is ...
end_answer
end_sub_instruction_1
...

Referring to the above format and
generation requirements, please think
through and generate specific answers
for the following task:

--input--:
{input_text}
--output--:
```