FollowBench: A Multi-level Fine-grained Constraints Following Benchmark for Large Language Models

Yuxin Jiang\textsuperscript{1,2}*, Yufei Wang\textsuperscript{3}, Xingshan Zeng\textsuperscript{2}, Wanjun Zhong\textsuperscript{2}, Liangyou Li\textsuperscript{3}, Fei Mi\textsuperscript{3}, Lifeng Shang\textsuperscript{2}, Xin Jiang\textsuperscript{3}, Qun Liu\textsuperscript{3}, Wei Wang\textsuperscript{1,2}

The Hong Kong University of Science and Technology (Guangzhou)\textsuperscript{1}, The Hong Kong University of Science and Technology\textsuperscript{2}, Huawei Noah’s Ark Lab\textsuperscript{3}
yjiangcm@connect.ust.hk, wang.yufei1@huawei.com, weiwcs@ust.hk

Abstract

The ability to follow instructions is crucial for Large Language Models (LLMs) to handle various real-world applications. Existing benchmarks primarily focus on evaluating pure response quality, rather than assessing whether the response follows constraints stated in the instruction. To fill this research gap, in this paper, we propose FollowBench, a Multi-level Fine-grained Constraints Following Benchmark for LLMs. FollowBench comprehensively includes five different types (i.e., Content, Situation, Style, Format, and Example) of fine-grained constraints. To enable a precise constraint following estimation on diverse difficulties, we introduce a Multi-level mechanism that incrementally adds a single constraint to the initial instruction at each increased level. To assess whether LLMs’ outputs have satisfied every individual constraint, we propose to prompt strong LLMs with constraint-evolution paths to handle challenging open-ended instructions. By evaluating 13 closed-source and open-source popular LLMs on FollowBench, we highlight the weaknesses of LLMs in instruction following and point towards potential avenues for future work. The data and code are publicly available at https://github.com/YJiangcm/FollowBench.

1 Introduction

Large Language Models (LLMs) (Brown et al., 2020; OpenAI, 2022) pre-trained on web-scale corpora have showcased proficiency in generating fluent and realistic text. Yet, human instructions in real-life cases require the model to generate text that not only possesses a high degree of naturalness but adheres to specific constraints (Yang et al., 2023). For instance, the model may be required to recommend ten books that are specifically written in Chinese (Figure 1), or it might be expected to generate responses that have a certain tone.

The dominant paradigm for assessing if a model can follow instructions involves using human annotators or strongly aligned LLMs to judge its response quality, in terms of helpfulness, relevance, accuracy, depth, creativity, and level of detail (Wang et al., 2023a; Li et al., 2023; Zheng et al., 2023; Xu et al., 2023). However, prior work still has two limitations. Firstly, they ignore the fine-grained constraints inside instructions, which are essential and objective standards for evaluating the instruction-following capability. While several benchmarks have rigorously explored individual constraint types, including semantic restrictions (Chen et al., 2022) and complex formatting (Tang et al., 2023), there exists a lack of comprehensive analysis across the diverse spectrum of constraint categories. Secondly, few benchmarks consider the varying difficulty of instructions, which is controlled by the number of imposed constraints. This makes it challenging to precisely assess the degree to which LLMs can follow instructions. Towards this end, our research question is: how can we systemically and precisely evalu-
uate the instruction-following capability of LLMs?

In this paper, we construct FollowBench, a Multi-level Fine-grained Constraints Following Benchmark. FollowBench comprehensively includes five different types of constraints from real-world scenarios, namely Content (i.e., explicit restrictions on the response content), Situation (i.e., specific situation/background information added to the question), Style (i.e., response style requirements), Format (i.e., response format requirements), and Example (i.e., example pattern recognition and following). To precisely estimate the difficulty degree to which LLMs can follow instructions, as shown in Figure 1, we propose a novel Multi-level mechanism that incrementally adds a single constraint to straightforward instructions at each increased level. The multi-level mechanism enables us to pinpoint the difficulty level at which LLMs fail to follow instructions, thereby estimating the upper limit of instruction-following capability in LLMs more precisely. Overall, FollowBench consists of 820 meticulously curated instructions from over 50 NLP tasks, including both closed- and open-ended questions. For evaluation purposes, we propose a hybrid evaluation method comprising rule-based and model-based solutions. Given LLMs’ outputs, both solutions judge whether the outputs satisfy each of the constraints in the instructions. The rule-based solutions focus on closed-ended instructions while the model-based solutions are applied to opened-ended instructions. For model-based solutions, instead of merely using current instructions and responses as input, we additionally provide the evolution process of the instructions in the input prompts to LLM judges to better understand each individual constraint. Both the data construction and the evaluation undergo human verification.

In our experiments, we propose three metrics to assess the instruction-following ability of 13 prominent closed-source and open-source LLMs on FollowBench. Our principal observations are: (1) the performance of all tested models declines substantially with an increase in difficulty level (the number of constraints in an instruction); (2) although closed-source models such as GPT-4 and GPT-3.5 only consecutively satisfy around three constraints on average, they still markedly surpass all open-source models; (3) certain specific constraint categories, such as Situation and Example, prove to be more challenging for LLMs than others; (4) beyond capabilities such as knowledge and reasoning, instruction following can offer an additional lens for comprehensively assessing the proficiency of LLMs.

2 Related Work

2.1 Instruction-Following Language Models

Prior research has found that LLMs fine-tuned with annotated “instructional” data, which is composed of language instructional commands and their desired outcomes, can be effective at following general language instructions (Weller et al., 2020; Sanh et al., 2021; Mishra et al., 2022; Jiang et al., 2024). To enhance the understanding of LLMs regarding the intricate and varied intentions of users in real-world scenarios, works like ChatGPT (OpenAI, 2022) and GPT-4 (OpenAI, 2023) implement instruction tuning across a wide range of human-crafted instructions and task categories. Recent studies (Zheng et al., 2023; Xu et al., 2023; Jiang et al., 2023) have pivoted towards automatically generating high-quality data to enhance the instruction-following capability of LLMs, addressing the challenges posed by labor-intensive human annotation.

2.2 Evaluation for Instruction Following

There are several research efforts in evaluating LLMs’ following capability towards particular tasks. Tang et al. (2023) focuses on evaluating LLMs’ generation capability towards complex structured tabular data in text, HTML, and Latex. They first collect tables from existing NLP benchmarks and websites, then construct guiding instructions based on these data. Chen et al. (2022) evaluates whether LLMs can follow particular knowledge-intensive generation instructions. They first provide a list of examples (e.g., a list of sports stars in the UK), followed by a constraint that is contradicted by the examples (e.g., not mentioning any athletes). These benchmarks can only demonstrate particular types of instruction-following capability of LLMs. In contrast, FollowBench comprehensively includes instructions with five different types of fine-grained constraints in multi-level difficulty and FollowBench should provide a well-rounded and precise estimation of instruction-following capability for existing LLMs. For more details on LLM evaluation, we refer to recent surveys (Chang et al., 2023; Wang et al., 2023b).
FollowBench

As shown in Table 1, FollowBench encompasses five distinct fine-grained constraint categories: Content, Situation, Style, Format, and Example. Each category consists of instructions from various NLP tasks. Different from previous benchmarks, we introduce a Multi-level mechanism that incrementally adds constraints to an initial instruction (see examples in Figure 2), producing a set of instructions ranging from 1 to 5 constraints. In the following part of this paper, we use “level n” to denote an instruction containing n constraints. It is worth noticing that the way of adding constraints is meticulously designed for each task within its respective constraint category. The multi-level mechanism enables us to pinpoint the difficulty level at which LLMs fail to follow instructions, thereby estimating the upper bound of instruction-following capability in LLMs more precisely.

To encapsulate, we will introduce the data construction process of FollowBench, including fine-grained constraints and the Multi-level mechanism, in §3.1. In §3.2, we propose an evaluation protocol with three metrics that seamlessly integrate with the multi-level mechanism.

### 3.1 Data Construction

#### Content Constraints

Content constraints refer to explicit impositions of specific conditions that shape the depth or scope of the response content. An example is shown in Figure 2, which sets specific criteria for the retrieved object. Ensuring that LLMs adhere to content constraints has become a critical challenge in Controlled Text Generation (Zhang et al., 2022), as it demands models to understand specific guidelines and adapt responses to prescribed conditions (Chen et al., 2022). To this end, we first collect data from the following tasks: (1) Complex Information Extraction aims at retrieving specific information about specific objects from the given text; (2) Text Generation with Language Constraints requires to generate fluent on-topic content while respecting a specified constraint; (3) Open-ended Question Answering comes from real scenarios (e.g., open-source platforms) to prevent the risk of data leakage. Subsequently, we construct multi-level instructions by adding one content constraint to the collected instructions each time. The manners of introducing additional constraints depend on different tasks (see details in Appendix A.1). For Complex Information Extraction, we gradually narrow down the scope of the information to be extracted. For Text Generation with Language Constraints, we incorporate additional restrictions from WordNet (Miller, 1992) and Wikidata (Vrandečić and Krötzsch, 2014). For Open-ended Question Answering, we utilize advanced LLMs like GPT-4 to generate a new instruction with one more constraint based on the given instruction. While the output from the LLMs serves primarily as a reference, we handpick the most relevant and challenging synthesized instructions to ensure data quality.
Figure 2: FollowBench covers five fine-grained categories of constraints. Within each constraint type, we construct a range of Multi-level instructions by incrementally adding constraints (highlighted in red). There are five levels in total; however, we only display the first two levels from each category for demonstration purposes.

**Situation Constraints**  
Situation Constraints refer to impositions of specific situations or backgrounds that implicitly guide the appropriate answer of the response. For instance, it is necessary to illustrate the situation when asking for customized suggestions, as shown in Figure 2. Another example is to customize LLMs to simulate various characters under certain circumstances, namely Role-playing, which provides a more nuanced interaction for users (Shanahan et al., 2023; Wang et al., 2023c). Situation constraints push LLMs beyond mere factual retrieval or surface-level synthesis, demanding a nuanced understanding, a dynamic adaptation, and complicated reasoning to the situation (Yao et al., 2022; Liu et al., 2023). Besides real-life questions, we also consider Complex Situation Reasoning tasks including Math Word Problems, Time/Spatial Reasoning, and Code Generation. These tasks all require interpreting and solving problems within a given situation, thus matching the definition of situation constraints. We first collect initial instructions from these sources and then manually curate multi-level instructions by incrementally supplementing situation information inside (see Appendix A.2).

**Style Constraints**  
Style Constraints control the stylistic variations of output to accomplish specific stylistic goals, such as tone, sentiment, formality, and empathy (Tsai et al., 2021), as illustrated in Figure 2. The challenges of style constraints for LLMs are the intricate understanding and adaptation of language nuances, ensuring contextually appropriate and stylistically consistent outputs (Smith et al., 2020; Cheng and Li, 2022). Drawing from Open-ended Question Answering datasets and online platforms, we collect initial instructions and then leverage LLMs’ in-context learning capability to craft instructions with multi-level style constraints. The prompt template can be viewed in Figure 8. Human experts subsequently review and refine the outputs produced by LLMs.

**Format Constraints**  
Format Constraints refer to stipulations governing the structural, linguistic, or output presentation of generated content. An example is shown in Figure 2, which sets limits on word length and requires the format of the response to be a table. Format constraints necessitate a deep, nuanced understanding of language and structure, allowing them to flexibly adapt outputs according to diverse and often intricate specifications (Zhao et al., 2023). Recent work has pointed out that even the most superior LLMs may struggle with tasks that require generating complex, structured outputs such as tables, JSON, HTML, or LaTeX (Tang et al., 2020).
To include a variety of format constraints, we first collect instructions from broader domains, encompassing Text-to-Table Generation and Open-ended Question Answering, then we utilize powerful LLMs to sequentially add format constraints ranging from length and hierarchy to specialized linguistic features and output mediums. See Figure 9 for the prompt template. Finally, we ask human experts to carefully check and refine the synthesized instructions.

**Example Constraints** LLMs have demonstrated stunning few-shot learning ability (Brown et al., 2020), which enables them to adapt quickly to a new query by recognizing patterns from just a few examples provided in the prompt. However, the robustness of few-shot learning, which means whether LLMs can still follow correct patterns after introducing “noise” examples, has not been explored. Thus, we propose a novel constraint category named Example Constraints to evaluate the example pattern recognition and following capability of LLMs. We automatically craft instructions with multi-level example constraints based on PromptSource (Bach et al., 2022), where instructions at level $n$ have $n-1$ noise examples in the input. The details are illustrated in Appendix A.3.

**Mixed Constraints** For the above five constraint categories, we construct multi-level instructions by adding the same type of constraint sequentially. Nevertheless, real-world scenarios often require more than one type of constraint to be enforced in a singular instruction. Therefore, we define Mixed Constraints as the composition of varied constraint categories. For instance, in the Text Editing task, we may want to add some content as well as adjust the output format. Besides, we also consider several tasks that are naturally suitable for constructing mixed constraints, including Summarization, Machine Translation, and Story Generation (see Appendix A.4). Instructions with multi-level mixed constraints are produced by specifying the format of generating answers (Format Constraints), requiring the generated text to include or not include certain keywords (Content Constraints), etc.

**Data Quality Control** To ensure the data quality of FollowBench, we implement a dual-layer verification system for each instruction. Two annotators independently evaluate: (1) the appropriateness of the instruction for its designated constraint category, and (2) the validity of the added constraint within the instruction. In instances of divergent evaluations, a third annotator intervenes for a detailed review to ensure consensus.

We analyze the comprehensiveness and diversity of in FollowBench, which includes 820 instructions in total. To maintain data diversity, we strive to ensure that the ROUGE-L score between any two initial instructions is below 0.7. Figure 3 shows the verb-noun structure of FollowBench instructions, where the top 20 verbs (inner circle) and their top 4 direct noun objects (outer circle) are depicted.

### 3.2 Evaluation Protocol

Given that nearly half of instructions in FollowBench are open-ended without reference answers, devising a rule-based program to assess the outputs is extremely challenging. To overcome this, inspired by (Gilardi et al., 2023; Huang et al., 2023), we propose to develop a model-based approach by using strong LLMs\(^1\) as judges. Previous works leverage strong LLMs to determine the quality of a response, by prompting them to consider multiple factors such as usefulness, relevance, and level of detail (Li et al., 2023; Zheng et al., 2023). To effectively guide strong LLMs to judge the constraint following capability objectively and faithfully, we propose a Multi-level-aware prompt template, as shown in Figure 4. Rather than merely presenting the instruction and

\(^1\)We use GPT-4-Preview-1106 in our experiments.
Figure 4: Prompt template for model-based evaluation.

As described in §3, we construct FollowBench by incrementally adding five constraints to an initial instruction, enabling us to pinpoint the difficulty level at which LLMs fail to follow instructions. Therefore, we propose a metric called Consistent Satisfaction Levels (CSL) to estimate how many consecutive levels a model can satisfy, beginning from level 1:

\[
CSL = \frac{1}{g} \sum_{i=1}^{g} \arg\max_{l} \left( l \prod_{n=1}^{l} S_{n}^{i} \right) \tag{3}
\]

where \( g \) is the group number of initial instructions, \( S_{n}^{i} = 1 \) if all constraints of the \( i \)-th instruction at level-\( n \) are satisfied and \( S_{n}^{i} = 0 \) otherwise.

4 Experiments

This section first introduces experimental setup in §4.1, and then presents the main experiment results across two key dimensions: difficulty level in §4.2 and constraint category in §4.3.

4.1 Experimental Setup

We evaluate 13 popular LLMs including GPT-4-Preview-1106 (OpenAI, 2023), GPT-3.5-Turbo-1106 (OpenAI, 2022), Qwen-Chat-72B/14B/7B (Bai et al., 2023), LLaMA2-Chat-70B/13B/7B (Touvron et al., 2023), WizardLM-13B-V1.2 (Xu et al., 2023), Vicuna-13B/7B-V1.5 (Zheng et al., 2023), Baichuan2-Chat-7B (Baichuan, 2023), and ChatGLM3-6B (Du et al., 2022). We access GPT-4-Preview-1106 and GPT-3.5-Turbo-1106 via OpenAI API. We access other open-source LLMs from their official repositories. During the inference process, we set the temperature to 0 to ensure deterministic outputs. We set the maximum generation length to 2048. Other parameters use their default values. To facilitate the multilingual evaluation of LLM’s instruction-following ability, we also craft a Chinese version of FollowBench, namely FollowBench-zh, in Appendix D.

4.2 Level-categorized Results

Table 2 provides a comprehensive comparison of various models across five difficulty levels, denoted as L1 to L5. The detailed results for each constraint category are listed in Appendix B. From a bird’s-eye view, we can infer that the performance typically diminishes as we progress from L1 to L5 for almost all models. This trend coincides with the increasing complexity or stringent requirements associated with higher levels. Besides, models with larger architectures generally outperform their smaller counterparts. However, it’s worth noting
As depicted in Figure 5, we assess various models over different constraint categories to succinctly showcase the instruction-following capability of LLMs in a singular dimension. Notably, GPT-4 and GPT-3.5 surpass open-source models in every constraint category, with a pronounced advantage in Content, Situation, Example, and Mixed constraints. Furthermore, most models demonstrated commendable proficiency under the Style constraint. While GPT-4, GPT-3.5, and LLaMA2-Chat-70B were the frontrunners, the trend suggests that style adaptation is an area where many models excel, hinting at its utility in real-world applications. However, the Example and Mixed constraints posed a challenge to most models. While GPT-4 led the segment, even its scores were noticeably lower than in other categories. To illustrate, in the “Example” category, we evaluated the instruction-following capabilities of LLMs by introducing “noise examples” with varying natural language templates. The observed performance decline is primarily due to the LLMs’ limited training in processing such noisy inputs within context-based learning scenarios. Typically, LLMs are fine-tuned on clean and uniform datasets, which do not adequately prepare them to sift through and ignore

Table 2: Results across five difficulty levels. For each level, we compute the average score of all constraint categories. Proprietary LLMs, open-sourced LLMs (large), open-sourced LLMs (medium), and open-sourced LLMs (small) are distinguished by different colors.

<table>
<thead>
<tr>
<th>Model</th>
<th>HSR (%)</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>SSR (%)</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>CSL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>L1</td>
<td>L2</td>
<td>L3</td>
<td>L4</td>
<td>L5</td>
<td>Avg.</td>
<td>L1</td>
<td>L2</td>
<td>L3</td>
<td>L4</td>
<td>L5</td>
</tr>
<tr>
<td>GPT-4-Preview-1106</td>
<td>84.7</td>
<td>75.6</td>
<td>70.8</td>
<td>73.9</td>
<td>61.9</td>
<td>73.4</td>
<td>84.7</td>
<td>77.0</td>
<td>75.3</td>
<td>77.0</td>
<td>72.3</td>
</tr>
<tr>
<td>GPT-3.5-Turbo-1106</td>
<td>80.3</td>
<td>68.0</td>
<td>68.6</td>
<td>61.1</td>
<td>53.2</td>
<td>66.2</td>
<td>80.3</td>
<td>71.2</td>
<td>74.2</td>
<td>69.6</td>
<td>67.1</td>
</tr>
<tr>
<td>Qwen-Chat-72B</td>
<td>73.8</td>
<td>63.3</td>
<td>54.3</td>
<td>45.2</td>
<td>39.9</td>
<td>55.3</td>
<td>73.8</td>
<td>67.5</td>
<td>63.2</td>
<td>57.6</td>
<td>56.0</td>
</tr>
<tr>
<td>LLaMA2-Chat-70B</td>
<td>59.9</td>
<td>53.3</td>
<td>46.0</td>
<td>40.2</td>
<td>37.9</td>
<td>47.5</td>
<td>59.9</td>
<td>57.3</td>
<td>55.7</td>
<td>53.3</td>
<td>53.2</td>
</tr>
<tr>
<td>Qwen-Chat-14B</td>
<td>62.8</td>
<td>56.2</td>
<td>47.7</td>
<td>38.7</td>
<td>30.9</td>
<td>47.3</td>
<td>62.8</td>
<td>61.9</td>
<td>57.7</td>
<td>52.6</td>
<td>51.4</td>
</tr>
<tr>
<td>WizardLM-13B-V1.2</td>
<td>68.8</td>
<td>64.1</td>
<td>53.1</td>
<td>40.8</td>
<td>35.8</td>
<td>52.5</td>
<td>68.8</td>
<td>65.7</td>
<td>61.8</td>
<td>53.4</td>
<td>53.9</td>
</tr>
<tr>
<td>LLaMA2-Chat-13B</td>
<td>57.0</td>
<td>56.0</td>
<td>50.4</td>
<td>44.4</td>
<td>38.1</td>
<td>49.5</td>
<td>57.0</td>
<td>50.6</td>
<td>50.0</td>
<td>48.6</td>
<td>52.2</td>
</tr>
<tr>
<td>Vicuna-13B-V1.5</td>
<td>71.2</td>
<td>60.2</td>
<td>49.6</td>
<td>40.6</td>
<td>34.0</td>
<td>51.1</td>
<td>71.2</td>
<td>64.8</td>
<td>59.9</td>
<td>54.5</td>
<td>53.6</td>
</tr>
<tr>
<td>Qwen-Chat-7B</td>
<td>55.9</td>
<td>51.7</td>
<td>38.7</td>
<td>33.1</td>
<td>23.3</td>
<td>40.6</td>
<td>55.9</td>
<td>58.2</td>
<td>51.6</td>
<td>48.6</td>
<td>45.9</td>
</tr>
<tr>
<td>LLaMA2-Chat-7B</td>
<td>58.0</td>
<td>51.3</td>
<td>47.4</td>
<td>39.5</td>
<td>35.3</td>
<td>46.3</td>
<td>58.0</td>
<td>56.5</td>
<td>55.6</td>
<td>52.5</td>
<td>51.4</td>
</tr>
<tr>
<td>Vicuna-7B-V1.5</td>
<td>60.8</td>
<td>52.0</td>
<td>42.2</td>
<td>33.3</td>
<td>23.9</td>
<td>42.4</td>
<td>60.8</td>
<td>58.5</td>
<td>55.5</td>
<td>48.3</td>
<td>49.0</td>
</tr>
<tr>
<td>Baichuan2-Chat-7B</td>
<td>58.3</td>
<td>46.1</td>
<td>40.7</td>
<td>30.4</td>
<td>25.5</td>
<td>40.2</td>
<td>58.3</td>
<td>55.4</td>
<td>54.9</td>
<td>49.9</td>
<td>49.3</td>
</tr>
<tr>
<td>ChatGLM3-6B</td>
<td>60.9</td>
<td>46.6</td>
<td>36.7</td>
<td>27.8</td>
<td>21.4</td>
<td>38.7</td>
<td>60.9</td>
<td>55.3</td>
<td>51.2</td>
<td>47.9</td>
<td>45.0</td>
</tr>
</tbody>
</table>

that the scaling law does not apply as effectively to LLaMA2-Chat-70B. It can be observed from Appendix B that while LLaMA-2-Chat-70B does indeed outperform LLaMA-2-Chat-13B in Situation constraints, it shows a relative underperformance in Format and Mixed Constraints categories. More importantly, there’s a marked performance gap between closed-source models (i.e., GPT-4 and GPT-3.5) and open-source models. Regarding CSL, it can be deduced that the instruction-following upper bound for GPT-4 and GPT-3.5 is approximately 3 constraints (level 3) added to an initial instruction. In contrast, open-source models typically have an upper limit of about 2 constraints (level 2). This significant difference underscores the better instruction-following ability of proprietary models, possibly due to superior data quality or optimization strategies such as RLHF (Ouyang et al., 2022). Furthermore, even the most sophisticated models are limited to following instructions with about three constraints, suggesting significant potential for further improvement.

4.3 Constraint-categorized Results

As depicted in Figure 5, we assess various models over different constraint categories to succinctly showcase the instruction-following capability of LLMs in a singular dimension. Notably, GPT-4 and GPT-3.5 surpass open-source models in every constraint category, with a pronounced advantage in Content, Situation, Example, and Mixed constraints. Furthermore, most models demonstrated commendable proficiency under the Style constraint. While GPT-4, GPT-3.5, and LLaMA2-Chat-70B were the frontrunners, the trend suggests that style adaptation is an area where many models excel, hinting at its utility in real-world applications. However, the Example and Mixed constraints posed a challenge to most models. While GPT-4 led the segment, even its scores were noticeably lower than in other categories. To illustrate, in the “Example” category, we evaluated the instruction-following capabilities of LLMs by introducing “noise examples” with varying natural language templates. The observed performance decline is primarily due to the LLMs’ limited training in processing such noisy inputs within context-based learning scenarios. Typically, LLMs are fine-tuned on clean and uniform datasets, which do not adequately prepare them to sift through and ignore
irrelevant or misleading information. This limitation becomes apparent when faced with the intricacies of real-world data. Our findings underscore the complexity of these constraints and pinpoint an area for potential improvement.

5 Analysis

This section includes: an ablation study confirming our prompt template’s effectiveness for model-based evaluation (§5.1); a comparison of instruction following vs. other LLM’s abilities (§5.2); an examination of failure consistency (§5.3); and an investigation of various decoding strategies (§5.4). In addition, a case study is presented in Appendix C for further analysis.

5.1 Ablation Study of Model-based Evaluation

We randomly sample 100 cases that require LLM evaluation, encompassing five constraints, five distinct levels, and four diverse models to guarantee comprehensive representation. Then we ask three expert-level human labelers to assess whether the model’s response satisfies all the constraints in each case and use the majority voting as the final human annotations. As shown in Table 3, our prompt template (Figure 4) registers an impressive 88% agreement with expert human evaluations, surpassing even the internal agreement among human experts, which stands at 85%. Remarkably, when the evolution process of multi-level constraints is removed from our prompt template, the agreement rate dips by 9%. This underlines the instrumental role played by the detailed portrayal of the instruction’s evolution in enhancing LLM’s precision in discernment. In contrast, we also employ the prompt template from Vicuna (Zheng et al., 2023), a standard prompt for assessing the overall quality of response. This template prompts the LLM to assign a score from 0 to 10 for each response. We consider responses with a score above 5.0 to meet all the constraints of an instruction. This approach achieves 67% agreement with human evaluators. Such a disparity highlights the fundamental difference between assessing the instruction-following ability and the overall response quality.

5.2 Instruction Following vs. Other Abilities

Table 4 presents a comparison of representative LLMs across different abilities, not just instruction following (FollowBench). This includes overall response quality (AlpacaEval (Li et al., 2023)), knowledge (MMLU (Hendrycks et al., 2021)), and reasoning (BBH (Suzgun et al., 2022)). We can find that our FollowBench provides an additional perspective for a holistic LLM evaluation. As an illustration, while the performance of WizardLM-13B-V1.2 exceeds that of GPT-3.5 in terms of overall response quality, it notably lags behind in instruction-following ability. Similarly, Vicuna-V1.5 excels over LLaMA2-Chat in the realms of knowledge and reasoning but struggles with instruction-following tasks.

5.3 Does Failure at Lower Level Necessarily Lead to Failure at Higher Level?

For a set of instructions that has five difficulty levels, if a model’s response doesn’t satisfy the constraints at level $n$, where $n$ ranges from 1 to 4, we define the failure consistency as the percentage that the response will also not fulfill the constraints at any subsequent level greater than $n$. Combining Table 2 and Table 5, it can be seen that models with better instruction-following capability may exhibit lower failure consistency. One possible reason is that the instruction-following ability of more powerful models is less sensitive to the number of constraints in an instruction, thus they are better equipped to adapt and fulfill the requirements even as the constraints increase. This adaptability means that while they may falter at a lower difficulty level, they can still manage to meet the demands of higher difficulty levels, leading to a decrease in failure consistency.

<table>
<thead>
<tr>
<th>Prompt</th>
<th>Agreement with Human</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>88%</td>
</tr>
<tr>
<td>Ours w/o ML</td>
<td>79%</td>
</tr>
<tr>
<td>Vicuna-Single</td>
<td>67%</td>
</tr>
</tbody>
</table>

Table 3: Agreement between human and diverse prompt templates. We use ML to denote multi-level.

<table>
<thead>
<tr>
<th>Model</th>
<th>Following</th>
<th>Overall</th>
<th>Knowledge</th>
<th>Reasoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-4-Preview-1106</td>
<td>3.3</td>
<td>97.7</td>
<td>86.4</td>
<td>86.7</td>
</tr>
<tr>
<td>GPT-3.5-turbo-1106</td>
<td>2.9</td>
<td>86.3</td>
<td>70.0</td>
<td>70.1</td>
</tr>
<tr>
<td>LLaMA2-Chat-70B</td>
<td>2.1</td>
<td>92.7</td>
<td>63.0</td>
<td>60.8</td>
</tr>
<tr>
<td>WizardLM-13B-V1.2</td>
<td>2.2</td>
<td>89.2</td>
<td>52.7</td>
<td>–</td>
</tr>
<tr>
<td>LLaMA2-Chat-13B</td>
<td>2.2</td>
<td>81.1</td>
<td>53.6</td>
<td>40.2</td>
</tr>
<tr>
<td>Vicuna-13B-V1.5</td>
<td>2.1</td>
<td>55.8</td>
<td>51.5</td>
<td></td>
</tr>
<tr>
<td>LLaMA2-Chat-7B</td>
<td>1.9</td>
<td>71.4</td>
<td>45.8</td>
<td>35.6</td>
</tr>
<tr>
<td>Vicuna-7B-V1.5</td>
<td>1.7</td>
<td>–</td>
<td>49.8</td>
<td>43.4</td>
</tr>
</tbody>
</table>

Table 4: Model comparison on different abilities.
Table 5: Results on failure consistency.

<table>
<thead>
<tr>
<th>Model</th>
<th>Failure Consistency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-4-Preview-1106</td>
<td>42.2</td>
</tr>
<tr>
<td>WizardLM-13B-V1.2</td>
<td>57.3</td>
</tr>
<tr>
<td>Vicuna-7B-V1.5</td>
<td>61.8</td>
</tr>
<tr>
<td>ChatGLM3-6B</td>
<td>64.0</td>
</tr>
</tbody>
</table>

5.4 Does Different Decoding Strategies Affect the Instruction-following Ability?

In this section, we systematically investigate the impact of different decoding strategies, represented by the temperature parameter $\tau$, on LLM’s instruction-following ability. The temperature $\tau$ is a commonly used parameter that controls the sharpness of the distribution from which we sample the next token:

$$P(w) = \frac{\exp(z_w / \tau)}{\sum_{w' \in V} \exp(z_{w'} / \tau)}$$  \hspace{0.5cm} (4)$$

where $z_w$ is the logit for word $w$, $V$ is the vocabulary. Lower values for temperature result in more consistent outputs, while higher values generate more diverse and creative results. As illustrated in Figure 6, the temperature $\tau$ has a tangible influence on the instruction-following ability across all four models. The sweet spot seems to be somewhere in the middle where there’s enough variability to capture the nuances and intricacies of complex instructions, yet not so much that the model goes off tangent. This balanced behavior ensures that the model remains within the desired context, producing outputs that align closely with the given instructions while also allowing for a slight creative touch when needed.

6 Conclusion

In this paper, we introduce FollowBench, a Multi-level Fine-grained Constraints Following Benchmark tailored for gauging the instruction-following capability of LLMs. FollowBench covers five fine-grained constraint categories and over 50 NLP tasks, utilizes a novel Multi-level mechanism for precisely estimating the upper limit of instruction-following capability. Furthermore, we propose an evaluation protocol with three metrics that seamlessly integrate with the multi-level mechanism. Our extensive tests over 13 popular LLMs reveal a substantial performance advantage for GPT-4 and GPT-3.5 over their counterparts, and there is still significant room for improving the instruction-following ability of current LLMs.

Limitations

While our study contributes valuable insights, it is essential to acknowledge several limitations that warrant consideration.

Firstly, our current investigation is confined to single-round interactions, aiming to offer a controlled environment for evaluation. Future research may extend its scope to multi-round conversations to comprehensively assess the instruction-following proficiency of LLMs in more dynamic and extended dialogues (Kwan et al., 2024).

Secondly, the model-based evaluation framework employed in our experiments, while rigorous, relies on prompt engineering, introducing an inherent imperfection. Despite our meticulous selection of high-performing prompts, the potential for further optimization remains, which may impact the reported evaluation metrics.

Lastly, we refrain from proposing specific solutions to address identified weaknesses of LLMs in instruction following. A plausible avenue for future research involves fine-tuning LLMs using our proposed FollowBench as a benchmark, providing a potential roadmap for enhancing instruction adherence. We defer the exploration of these aspects to subsequent studies, recognizing the need for a comprehensive examination of LLM capabilities across varying interaction complexities.

Ethics Statement

Our paper aims to systemically and precisely evaluate the capability of LLMs to follow natural language instructions. However, it is essential to bear in mind that malicious instructions have the potential to prompt the model to generate harmful or inappropriate outputs. Therefore, ensuring safe and responsible practices when assessing
the instruction-following capability of LLMs is of paramount importance. In FollowBench, each piece of data undergoes a meticulous human review process to identify and eliminate any potentially harmful instructions or offensive content. This rigorous approach underscores our commitment to maintaining a secure and ethical evaluation framework.

Acknowledgments

W. Wang was supported by HKUST(GZ) Grant G0101000028, CCF-HuaweiDBC202302, Guangzhou Municipal Science and Technology Project (No. 2023A03J0003, 2023A03J0013 and 2024A03J0621).

References


Zhengxiao Du, Yujié Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, ZhiLin Yang, and Jie Tang. 2022. Glm:


Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiahuan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions.
A Data Generation Process

Here we outline the sources for our data and provide a detailed description of the data generation process for each constraint category.

A.1 Content Constraints

The data of content constraints is constructed from five tasks as follows:

- **Data-to-Text Generation** We create instructions with 1 to 5 constraints by adapting samples from E2E (Novikova et al., 2017). Different from the original task, we ask the model to extract the flat meaning representations according to the corresponding natural language texts. The number of constraints increases with the number of attributes and the number of restaurants. We use exact match as the evaluation metric.

- **Document-Level Event Argument Extraction** We create instructions by adapting samples from WIKIEVENTS (Li et al., 2021). Given a document, the model is required to extract $n$ events that satisfy a specific event template, where $n \in [1, 5]$ corresponds to the number of constraints. We use accuracy as the evaluation metric.

- **Document-Level Named Entity Recognition** We derive instructions from samples in the CONLL-2003 dataset (Tjong Kim Sang and De Meulder, 2003). We ask the model to extract a single named entity from a provided document. Notably, as the number of constraints rises, the requirements for the retrieved named entity correspondingly increase. For example, “extract one named entity that is a location” $\rightarrow$ “extract one named entity that is a location in east Asia”. We use accuracy as the evaluation metric.

- **Text Generation with Language Constraints** COGNAC (Chen et al., 2022) is a challenging benchmark wherein models are presented with a topic accompanied by example text and explicit constraints on the text to avoid. We curate data from COGNAC, formulating instructions with 1 to 5 constraints by integrating additional linguistic restrictions from WordNet (Miller, 1992) and WikiData (Vrandečić and Krötzsch, 2014).

- **Open-ended Question Answering** We first collect multi-level instructions from self-instruct evaluation set (Wang et al., 2023a), helpful evaluation released by Anthropic (Bai et al., 2022), Vicuna evaluation (Zheng et al., 2023), and Koala evaluation (Geng et al., 2023), as well as open-source platforms such as Quora, Reddit, and ShareGPT. Given the challenges associated with iteratively adding constraints to an initial instruction, we prompt GPT-4 with a specific prompt shown in Figure 7 to generate a new instruction with one more constraint based on the given instruction. The above process is repeated five times. Finally, we obtain a set of instructions ranging from 1 to 5 constraints.

A.2 Situation Constraints

The data of situation constraints is constructed from tasks as follows:

- **Suggestion Generation, Role-playing** We collect multi-level instructions that fit within

---

2https://www.quora.com
3https://www.reddit.com
4https://sharegpt.com
the paradigm of situation constraints from Open-ended Question Answering datasets and online platforms. Examples include asking the model to give suggestions under specific circumstances, asking the model to act as a terminal and output based on the given information, etc.

- **Math Word Problems** The initial instructions are collected from GSM8K (Cobbe et al., 2021) and AGIEval (Zhong et al., 2023). We then manually add constraints progressively by enhancing the situation descriptions, ensuring that the core question remains unaltered. We use accuracy as the evaluation metric.

- **Time/Spatial Reasoning** We generate data by refining samples from BIG-Bench Hard (Suzgun et al., 2022). For Time Reasoning, we increase the difficulty level by incorporating additional temporal concepts, such as weeks, months, and years. In the realm of Spatial Reasoning, we opt for a logical deduction task that necessitates deducing the order of a sequence of objects. Here, the number of constraints escalates by augmenting the task with detailed location descriptions for a new object. We use accuracy as the evaluation metric.

- **Code Generation** We sourced initial instructions from HumanEval (Chen et al., 2021) and enhanced the difficulty level by adding complexity to the function descriptions within the instructions. We use pass@1 (Kulal et al., 2019) as the evaluation metric.

### A.3 Example Constraints

Specifically, we choose 40 diverse NLP tasks from PromptSource (Bach et al., 2022), where each task has more than 5 question templates. Additionally, we create 29 answer templates (shown in Table 6) that regulate the format of the response. For instructions at difficulty level 1, we utilize the standard 5-shot prompting, where 5 shots are equipped with 1 sampled question template and 1 sampled answer template, and the model is required to respond to a query using the answer template. For instructions at difficulty level $n$ ($1 < n \leq 5$), the 5 shots are randomly paired with $n$ question templates and $n$ corresponding answer templates. Based on the question template of the query, the model is required to recognize the matched question template in the 5 shots and respond using the corresponding answer template. We use accuracy as the evaluation metric.

### A.4 Mixed Constraints

In this paper, we consider four below tasks which are naturally suitable for constructing mixed constraints:

- **Text Editing** We start by gathering text from different online sources, like sentences, letters, and emails. Next, we create instructions with multi-level mixed constraints by increasingly adding an editing requirement to the text at each level. For example, “swap the first and last words in the sentence” (Content Constraints), “response using ‘###’ at the beginning” (Format Constraints), etc. We write rule-based programs for individual instructions to assess the satisfaction of internal constraints, employing exact match as the evaluation metric.

- **Summarization** The initial instructions are sampled from CNN/Daily Mail (Nallapati et al., 2016), XSum (Narayan et al., 2018), SAMSum (Gliwa et al., 2019), English Gigaword (Graff et al., 2003), and arXiv (arXiv.org)
submitters, 2023). The instructions with multi-level mixed constraints are produced by specifying the format of generating answers (Format Constraints), requiring the generated text to include or not include certain keywords (Content Constraints), etc. We write rule-based programs for individual instructions to assess the satisfaction of internal constraints, employing accuracy as the evaluation metric.

• **Machine Translation** The initial instructions are sampled from OpenSubtitles (Lison and Tiedemann, 2016), TED Talks (Cettolo et al., 2012), and News-Commentary (Tiedemann, 2012). Then we construct instructions from level 1 to level 5 using a similar pipeline as that of Summarization. We write rule-based programs for individual instructions to assess the satisfaction of internal constraints, employing accuracy as the evaluation metric.

• **Story Generation** We collect initial instructions from ROCStories (Mostafazadeh et al., 2016) and WritingPrompts (Fan et al., 2018). Then we add 5 mixed constraints sequentially to the initial instructions based on the ground truth, such as the number of sentences in the generated story (Format Constraints), requiring the generated text to include certain keywords (Content Constraints), specifying the writing style (Style Constraints), etc.

### B Detailed Experimental Results

Here we list the experimental results across 5 difficulty levels for each constraint category, including Content Constraints in Table 7, Situation Constraints in Table 8, Style Constraints in Table 9, Format Constraints in Table 10, Example Constraints in Table 11, and Mixed Constraints in Table 12.

### C Case Study

Table 13 and Table 14 show the respective responses and evaluation results of GPT-4 and WizardLM-13B-V1.5 when tasked with a level-5 instruction under the category of Content Constraints. It can be observed that GPT-4 meets all five specified constraints, whereas WizardLM-13B-V1.5 fails to fulfill the third constraint, which mandates that the output animals must be able to swim. Besides, these two cases also validate the effectiveness of our model-based evaluation.

### D FollowBench-zh

To facilitate the multilingual evaluation of LLM’s instruction-following ability, we have additionally crafted a Chinese version of FollowBench, denoted as FollowBench-zh. This involved employing a data generation process analogous to that utilized in the development of the English version. Overall, FollowBench-zh consists of 790 meticulously curated instructions from over 50 NLP tasks, including both closed- and open-ended questions. The detailed data statistics are listed in Table 15.

Following §3.2 and §4.1, we evaluate 13 popular LLMs on FollowBench-zh. The prompt template for model-based evaluation of FollowBench-zh is shown in Figure 10. It is noticeable that although LLaMA2-Chat-70B/13B/7B, WizardLM-13B-V1.2, and Vicuna-13B/7B-V1.5 are not specifically trained on Chinese corpora, they can still understand and respond in Chinese. Table 16 provides a comprehensive comparison of various models across five difficulty levels, denoted as L1 to L5. Similar to FollowBench, the performance of nearly all models on FollowBench-zh typically diminishes as we progress from L1 to L5. Nevertheless, GPT-3.5 exhibits a notably diminished proficiency in following instructions on FollowBench-zh in comparison to GPT-4, showcasing a more pronounced performance gap than observed on FollowBench. Moreover, models such as Baichuan2-Chat-7B and ChatGLM3-6B, which are pre-trained on a combination of English and Chinese corpora, demonstrate comparable or even better performance compared to their open-source counterparts. This highlights the significance of incorporating diverse linguistic datasets in pre-training to enhance the multilingual instruction-following capability of LLMs. Figure 11 depicts the instruction-following capability of LLMs over different constraint categories, with GPT-4 standing out notably among its counterparts. In a nutshell, there is still a substantial opportunity for enhancing the instruction-following capabilities of existing LLMs.
### Prompt Template (Open-ended Question Answering in Content Constraints)

You are an Instruction Rewriting Expert. You need to rewrite #Given Instruction# based on #Rewriting Requirement#, in order to obtain a #Rewritten Instruction#. Basically, #Rewritten Instruction# should adhere to the following guidelines:

1. Your rewriting cannot omit the non-text parts such as the table and code in #Given Instruction#.
2. #Rewritten Instruction# must be reasonable and must be understood and responded by humans.
3. You should try your best not to make the #Rewritten Instruction# become verbose. #Rewritten Instruction# can only add 10 to 20 words into #Given Instruction#.

#Given Instruction#

(given_instruction)

#Rewritten Instruction#

Asking the AI to mimic a specific author's writing style.

- Add a Subtask or Another Related Question.
- Narrow Down the Topic: Instead of a general theme or topic, provide a more specific subset.
- Set a Higher Standard: Raise the bar for what's considered acceptable or successful.
- Limit Resources: Restrict the number or type of resources someone can use.
- Introduce Specific Criteria: Mandate particular components or features that must be included.
- Specifying Sequence: Dictate the order in which certain steps or actions should be taken.

### Prompt Template (Open-ended Question Answering in Style Constraints)

You are an Instruction Rewriting Expert. You need to rewrite #Given Instruction# based on #Rewriting Requirement#, in order to obtain a #Rewritten Instruction#. Basically, #Rewritten Instruction# should adhere to the following guidelines:

1. Your rewriting cannot omit the non-text parts such as the table and code in #Given Instruction#.
2. #Rewritten Instruction# must be reasonable and must be understood and responded by humans.
3. You should try your best not to make the #Rewritten Instruction# become verbose. #Rewritten Instruction# can only add 10 to 20 words into #Given Instruction#.

#Given Instruction#

(given_instruction)

#Rewritten Instruction#

Asking the AI to mimic a specific author's writing style.

- Tone and Emotion: Specify the desired emotional tone for the response.
- Writing Style: Ask the AI to mimic a specific author’s writing style.
- Contradiction: Ask the AI to provide a response that contradicts the previous statement or take a stance opposite to its prior response.
- Ambiguity: Teach the AI to create responses with intentional ambiguity or double meanings.
- Humor or Satire: Request that the response be humorous or satirical, requiring the AI to generate jokes or witty remarks.

### Prompt Template (Open-ended Question Answering in Format Constraints)

You are an Instruction Rewriting Expert. You need to rewrite #Given Instruction# based on #Rewriting Requirement#, in order to obtain a #Rewritten Instruction#. Basically, #Rewritten Instruction# should adhere to the following guidelines:

1. Your rewriting cannot omit the non-text parts such as the table and code in #Given Instruction#.
2. #Rewritten Instruction# must be reasonable and must be understood and responded by humans.
3. You should try your best not to make the #Rewritten Instruction# become verbose. #Rewritten Instruction# can only add 10 to 20 words into #Given Instruction#.

#Given Instruction#

(given_instruction)

#Rewritten Instruction#

Asking the AI to mimic a specific author's writing style.

- Length: Imposing constraints on the length of individual words, sentences, or paragraphs.
- Hierarchical Instructions: Providing instructions that have a hierarchical structure, where the AI needs to understand and follow a hierarchy of tasks to construct a response.
- Special Output Format: Asking the AI to respond by using data format like table, json, HTML, LaTeX, etc.
- Morphological Constraints: Asking the AI to avoid or use specific morphemes.
- Multi-lingual Constraints: Asking the AI to respond in multiple languages or switch between languages according to complex patterns.
- Incorporation of Specific Literary Devices: Requesting the inclusion of specific, and perhaps numerous, literary devices.
- Following a Specific Grammatical Structure: Requesting the AI to create responses that strictly follow a particular grammatical structure.
<table>
<thead>
<tr>
<th>Model</th>
<th>HSR (%)</th>
<th>SSR (%)</th>
<th>CSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-4-Preview-1106</td>
<td>84.0</td>
<td>76.0</td>
<td>3.5</td>
</tr>
<tr>
<td>GPT-3.5-Turbo-1106</td>
<td>72.0</td>
<td>68.0</td>
<td>2.7</td>
</tr>
<tr>
<td>Qwen-Chat-72B</td>
<td>84.0</td>
<td>72.0</td>
<td>3.2</td>
</tr>
<tr>
<td>LLama2-Chat-70B</td>
<td>48.0</td>
<td>44.0</td>
<td>3.2</td>
</tr>
<tr>
<td>Qwen-Chat-14B</td>
<td>56.0</td>
<td>64.0</td>
<td>3.2</td>
</tr>
<tr>
<td>WizardLM-13B-V1.2</td>
<td>68.0</td>
<td>56.0</td>
<td>2.1</td>
</tr>
<tr>
<td>LLama2-Chat-13B</td>
<td>48.0</td>
<td>44.0</td>
<td>2.4</td>
</tr>
<tr>
<td>Vicuna-13B-V1.5</td>
<td>60.0</td>
<td>52.0</td>
<td>1.9</td>
</tr>
<tr>
<td>Qwen-Chat-7B</td>
<td>56.0</td>
<td>50.0</td>
<td>1.7</td>
</tr>
<tr>
<td>LLama2-Chat-7B</td>
<td>44.0</td>
<td>40.0</td>
<td>1.6</td>
</tr>
<tr>
<td>Vicuna-7B-V1.5</td>
<td>60.0</td>
<td>48.0</td>
<td>1.3</td>
</tr>
<tr>
<td>Baichuan2-Chat-7B</td>
<td>60.0</td>
<td>48.0</td>
<td>1.2</td>
</tr>
<tr>
<td>ChatGLM3-6B</td>
<td>68.0</td>
<td>44.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 7: Results of Content Constraints across 5 difficulty levels. Proprietary LLMs, open-sourced LLMs (large), open-sourced LLMs (medium), and open-sourced LLMs (small) are distinguished by different colors.

<table>
<thead>
<tr>
<th>Model</th>
<th>HSR (%)</th>
<th>SSR (%)</th>
<th>CSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qwen-Chat-72B</td>
<td>84.0</td>
<td>76.0</td>
<td>3.5</td>
</tr>
<tr>
<td>LLama2-Chat-70B</td>
<td>48.0</td>
<td>44.0</td>
<td>2.7</td>
</tr>
<tr>
<td>Qwen-Chat-14B</td>
<td>56.0</td>
<td>64.0</td>
<td>2.7</td>
</tr>
<tr>
<td>WizardLM-13B-V1.2</td>
<td>68.0</td>
<td>56.0</td>
<td>2.1</td>
</tr>
<tr>
<td>LLama2-Chat-13B</td>
<td>48.0</td>
<td>44.0</td>
<td>2.4</td>
</tr>
<tr>
<td>Vicuna-13B-V1.5</td>
<td>60.0</td>
<td>52.0</td>
<td>1.9</td>
</tr>
<tr>
<td>Qwen-Chat-7B</td>
<td>56.0</td>
<td>50.0</td>
<td>1.7</td>
</tr>
<tr>
<td>LLama2-Chat-7B</td>
<td>44.0</td>
<td>40.0</td>
<td>1.6</td>
</tr>
<tr>
<td>Vicuna-7B-V1.5</td>
<td>60.0</td>
<td>48.0</td>
<td>1.3</td>
</tr>
<tr>
<td>Baichuan2-Chat-7B</td>
<td>60.0</td>
<td>48.0</td>
<td>1.2</td>
</tr>
<tr>
<td>ChatGLM3-6B</td>
<td>68.0</td>
<td>44.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 8: Results of Situation Constraints across 5 difficulty levels. Proprietary LLMs, open-sourced LLMs (large), open-sourced LLMs (medium), and open-sourced LLMs (small) are distinguished by different colors.

<table>
<thead>
<tr>
<th>Model</th>
<th>HSR (%)</th>
<th>SSR (%)</th>
<th>CSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qwen-Chat-72B</td>
<td>84.0</td>
<td>76.0</td>
<td>3.5</td>
</tr>
<tr>
<td>LLama2-Chat-70B</td>
<td>48.0</td>
<td>44.0</td>
<td>2.7</td>
</tr>
<tr>
<td>Qwen-Chat-14B</td>
<td>56.0</td>
<td>64.0</td>
<td>2.7</td>
</tr>
<tr>
<td>WizardLM-13B-V1.2</td>
<td>68.0</td>
<td>56.0</td>
<td>2.1</td>
</tr>
<tr>
<td>LLama2-Chat-13B</td>
<td>48.0</td>
<td>44.0</td>
<td>2.4</td>
</tr>
<tr>
<td>Vicuna-13B-V1.5</td>
<td>60.0</td>
<td>52.0</td>
<td>1.9</td>
</tr>
<tr>
<td>Qwen-Chat-7B</td>
<td>56.0</td>
<td>50.0</td>
<td>1.7</td>
</tr>
<tr>
<td>LLama2-Chat-7B</td>
<td>44.0</td>
<td>40.0</td>
<td>1.6</td>
</tr>
<tr>
<td>Vicuna-7B-V1.5</td>
<td>60.0</td>
<td>48.0</td>
<td>1.3</td>
</tr>
<tr>
<td>Baichuan2-Chat-7B</td>
<td>60.0</td>
<td>48.0</td>
<td>1.2</td>
</tr>
<tr>
<td>ChatGLM3-6B</td>
<td>68.0</td>
<td>44.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 9: Results of Style Constraints across 5 difficulty levels. Proprietary LLMs, open-sourced LLMs (large), open-sourced LLMs (medium), and open-sourced LLMs (small) are distinguished by different colors.
### Table 12: Results of Mixed Constraints across 5 difficulty levels.

<table>
<thead>
<tr>
<th>Model</th>
<th>HSR (%)</th>
<th>SSR (%)</th>
<th>CSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-4-Preview-1106</td>
<td>90.0</td>
<td>93.3</td>
<td></td>
</tr>
<tr>
<td>GPT-3.5-Turbo-1106</td>
<td>90.0</td>
<td>76.7</td>
<td>50.0</td>
</tr>
<tr>
<td>Qwen-Chat-7B</td>
<td>86.7</td>
<td>80.0</td>
<td>50.0</td>
</tr>
<tr>
<td>LLaMA2-Chat-70B</td>
<td>83.3</td>
<td>66.7</td>
<td>50.0</td>
</tr>
<tr>
<td>Qwen-Chat-14B</td>
<td>86.7</td>
<td>80.0</td>
<td>50.0</td>
</tr>
<tr>
<td>WizardLM-13B-V1.2</td>
<td>83.3</td>
<td>93.3</td>
<td>50.0</td>
</tr>
<tr>
<td>LLaMA2-Chat-13B</td>
<td>86.7</td>
<td>80.0</td>
<td>50.0</td>
</tr>
<tr>
<td>Vicuna-13B-V1.5</td>
<td>86.7</td>
<td>80.0</td>
<td>50.0</td>
</tr>
<tr>
<td>Qwen-Chat-7B</td>
<td>76.7</td>
<td>80.0</td>
<td>50.0</td>
</tr>
<tr>
<td>LLaMA2-Chat-7B</td>
<td>80.0</td>
<td>80.0</td>
<td>50.0</td>
</tr>
<tr>
<td>Vicuna-7B-V1.5</td>
<td>80.0</td>
<td>80.0</td>
<td>50.0</td>
</tr>
<tr>
<td>Baichuan2-Chat-7B</td>
<td>80.0</td>
<td>80.0</td>
<td>50.0</td>
</tr>
<tr>
<td>ChatGLM3-6B</td>
<td>80.0</td>
<td>80.0</td>
<td>50.0</td>
</tr>
<tr>
<td><strong>Avg.</strong></td>
<td>86.0</td>
<td>80.0</td>
<td>50.0</td>
</tr>
</tbody>
</table>

Proprietary LLMs, open-sourced LLMs (large), open-sourced LLMs (medium), and open-sourced LLMs (small) are distinguished by different colors.
The response and evaluation result of GPT-4-Preview-1106 for a level-5 instruction

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Response</th>
</tr>
</thead>
</table>
| Talk about domestic animal:  
- The norwegian elkhound is one of the northern spitz type breed of dog and is the national dog of Norway.  
- The Irish terrier (Irish: brocaire rua) is a dog breed from Ireland, one of many breeds of terrier.  
- A hearing dog is a type of assistance dog specifically selected and trained to assist people who are deaf or hard of hearing by alerting their handler to important sounds, such as doorbells, smoke alarms, ringing telephones, or alarm clocks.  
Could you list three more domestic animals that satisfy all of following conditions:  
1. Do not talk about dog.  
2. Their weights are less than 100kg.  
3. Can swim.  
4. Have 4 legs.  
5. Are hairy. | 1. The Maine Coon cat is a large domesticated cat breed that originates from the United States. Known for their size, they typically weigh less than 100kg. They have a thick, water-resistant fur coat, which allows them to swim if necessary.  
2. The American Mink is a semi-aquatic, fur-bearing mammal that is often domesticated for its fur. It has four legs, can swim, and weighs significantly less than 100kg.  
3. The Norwegian Forest Cat, also known as the "Wegie", is a breed of domestic cat originating in Northern Europe. This breed is adapted to a very cold climate, with a top coat of glossy, long, water-shedding hairs, and a woolly undercoat for insulation. Despite their large size, they weigh less than 100kg and are known to be good swimmers. |

| Evaluation | 1) The 5 added constraints are:  
- Do not talk about dog.  
- Their weights are less than 100kg.  
- Can swim.  
- Have 4 legs.  
- Are hairy.  
2) For the 5 added constraints, the #Answer of Initial Instruction + 5 constraints# satisfies each constraint as follows:  
- Do not talk about dog: Yes, the answer talks about Maine Coon cat, American Mink, and Norwegian Forest Cat, not dogs.  
- Their weights are less than 100kg: Yes, all the animals mentioned weigh less than 100kg.  
- Can swim: Yes, all the animals mentioned can swim.  
- Have 4 legs: Yes, all the animals mentioned have 4 legs.  
- Are hairy: Yes, all the animals mentioned are hairy.  
3) ['YES', 'YES', 'YES', 'YES', 'YES'] |

Table 13: The response and evaluation result of GPT-4-Preview-1106 for a level-5 instruction belonging to Content Constraints.
### Instruction

Talk about domestic animal:

- **the norwegian elkhound** is one of the northern spitz type breed of dog and is the national dog of norway.
- **the irish terrier** (irish: *brocaire rua*) is a dog breed from ireland, one of many breeds of terrier.
- **a hearing dog** is a type of assistance dog specifically selected and trained to assist people who are deaf or hard of hearing by alerting their handler to important sounds, such as doorbells, smoke alarms, ringing telephones, or alarm clocks.

Could you list three more domestic animals that satisfy ALL of following conditions:

1. Do not talk about dog.
2. Their weights are less than 100kg.
3. Can swim.
4. Have 4 legs.
5. Are hairy.

### Response

1. **Cat:** Cats are domestic animals that weigh less than 100kg, have four legs, and are hairy. They are also known for their swimming abilities, making them suitable for households with access to water.
2. **Rabbit:** Rabbits are small, furry animals that weigh less than 100kg and have four legs. They are also excellent swimmers, thanks to their powerful hind legs and water-resistant fur.
3. **Hamster:** Hamsters are small, furry rodents that weigh less than 100kg and have four legs. They are popular pets due to their gentle nature and low maintenance care.

### Evaluation

1) The 5 added constraints are:
- Do not talk about dog.
- Their weights are less than 100kg.
- Can swim.
- Have 4 legs.
- Are hairy.

2) For the 5 added constraints:
- Do not talk about dog: The answer does not mention any dogs. It talks about cats, rabbits, and hamsters. So, it satisfies this constraint.
- Their weights are less than 100kg: The animals mentioned in the answer (cats, rabbits, and hamsters) all weigh less than 100kg. So, it satisfies this constraint.
- Can swim: The answer mentions that cats and rabbits can swim. However, hamsters are not natural swimmers. So, it does not satisfy this constraint.
- Have 4 legs: All the animals mentioned in the answer (cats, rabbits, and hamsters) have four legs. So, it satisfies this constraint.
- Are hairy: All the animals mentioned in the answer (cats, rabbits, and hamsters) are described as being hairy or furry. So, it satisfies this constraint.

3) ['YES', 'YES', 'NO', 'YES', 'YES']

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Response</th>
<th>Evaluation</th>
</tr>
</thead>
</table>
| Talk about domestic animal: | 1. Cat: Cats are domestic animals that weigh less than 100kg, have four legs, and are hairy. They are also known for their swimming abilities, making them suitable for households with access to water.  
2. Rabbit: Rabbits are small, furry animals that weigh less than 100kg and have four legs. They are also excellent swimmers, thanks to their powerful hind legs and water-resistant fur.  
3. Hamster: Hamsters are small, furry rodents that weigh less than 100kg and have four legs. They are popular pets due to their gentle nature and low maintenance care. | 1) The 5 added constraints are:
- Do not talk about dog.
- Their weights are less than 100kg.
- Can swim.
- Have 4 legs.
- Are hairy.

2) For the 5 added constraints:
- Do not talk about dog: The answer does not mention any dogs. It talks about cats, rabbits, and hamsters. So, it satisfies this constraint.
- Their weights are less than 100kg: The animals mentioned in the answer (cats, rabbits, and hamsters) all weigh less than 100kg. So, it satisfies this constraint.
- Can swim: The answer mentions that cats and rabbits can swim. However, hamsters are not natural swimmers. So, it does not satisfy this constraint.
- Have 4 legs: All the animals mentioned in the answer (cats, rabbits, and hamsters) have four legs. So, it satisfies this constraint.
- Are hairy: All the animals mentioned in the answer (cats, rabbits, and hamsters) are described as being hairy or furry. So, it satisfies this constraint.

3) ['YES', 'YES', 'NO', 'YES', 'YES'] |

Table 14: The response and evaluation result of WizardLM-13B-V1.2 for a level-5 instruction belonging to Content Constraints.
Table 15: An overview of FollowBench-zh. “Avg Len” is the average character number of instructions. ◦ refers to rule-based evaluation, while ◻ refers to model-based evaluation.

Table 16: Results across five difficulty levels of FollowBench-zh. For each level, we compute the average score of all constraint categories. Proprietary LLMs, open-sourced LLMs (large), open-sourced LLMs (medium), and open-sourced LLMs (small) are distinguished by different colors.
给定一个初始指令，我们每次添加一个{constraint_type}约束，获得
了有{level_n}个附加约束的最终指令。

#初始指令#
{initial_instruction}

#初始指令+ ... {level_n}个约束的答案#
{answer_of_level_n_instruction}

系统
1) 请识别出添加的{level_n}个约束。
2) 对于添加的{level_n}个约束，请判断初始指令 + {level_n}个约束的
   答案是否满足了每个约束。
3) 在最后一行，只输出一个包含{level_n}个元素（"YES"或"NO"）的
   Python列表，表明答案是否满足添加的每一个约束。

Figure 10: Prompt template for model-based evaluation
of FollowBench-zh.

Figure 11: HSR (%) results in diverse constraint cate-
gories of FollowBench-zh. For each category, we
compute the average score of all difficulty levels.