Revisiting the Reliability of Language Models in Instruction-Following

Anonymous ACL submission

Abstract

Advanced LLMs have achieved near-ceiling instruction-following accuracy on benchmarks such as **IFEVAL**. However, these impressive scores do not necessarily translate to reliable services in real-world use, where users often vary their phrasing, contextual framing, and task formulations. In this paper, we study *nuance-oriented reliability*: whether models exhibit consistent competence across *cousin prompts* that convey analogous user intents but with subtle nuances. To quantify this, we introduce a new metric, `reliable@k`, and develop an automated pipeline that generates high-quality cousin prompts via data augmentation. Building upon this, we construct **IFEVAL++** for systematic evaluation. Across 20 proprietary and 26 open-source LLMs, we find that current models exhibit substantial insufficiency in nuance-oriented reliability—their performance can drop by up to 61.8% with nuanced prompt modifications. What's more, we characterize it and explore three potential improvement recipes. Our findings highlight nuance-oriented reliability as a crucial yet underexplored next step toward more dependable and trustworthy LLM behavior. Our code and benchmark are accessible: `https://anonymous.4open.science/r/IFEval-pp`.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable abilities to follow natural language instructions (Ouyang et al., 2022). The instruction-following ability is foundational for enabling faithful user interaction (Zhou et al., 2023; Lior et al., 2025; Zhang et al., 2024b), reliable agent behavior (Zhang et al., 2025a), and reduced harmful outcomes (Ruan et al., 2024). Evaluating this ability is essential for building reliable, trustworthy AI systems. Consequently, numerous benchmarks have been proposed to measure instruction-following performance across diverse task types and constraint categories (see Appendix A for a survey of 36 benchmarks).

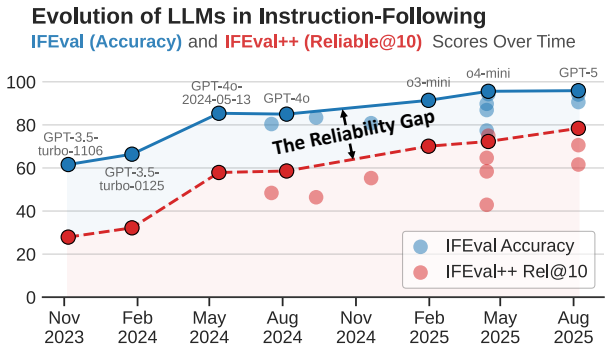

Figure 1: **Evolution of OpenAI's Models in Instruction-Following Abilities Across Generations**.

As illustrated in Figure 1, the instruction-following ability of LLMs has improved rapidly across generations. Recent LLMs achieve near-ceiling performance on widely adopted benchmarks like **IFEVAL** (Zhou et al., 2023), with `GPT-5` reaching an accuracy as high as 95.9%. However, growing evidence reveals that model performance can be highly sensitive to prompt wording (Sclar et al., 2024; Mizrahi et al., 2024; Cao et al., 2024), and that overfitting to benchmark test cases is possible (Zhang et al., 2024a). Thus, questions remain about how well benchmark accuracy translates to real-world reliability.

To illustrate this concern, consider two LLMs that both achieve perfect benchmark accuracy. When faced with user prompts expressing similar intent but differing subtly in phrasing, contextual framing, or task instantiation, one model may generalize appropriately while the other fails. Current benchmarks, which primarily emphasize task and constraint diversity, cannot capture this crucial **nuance-oriented dimension of reliability**. This motivates our central question: *Are current LLMs reliable when handling "cousin prompts" that convey similar user intents but vary in nuanced ways?*

To operationalize this, we propose a new metric,

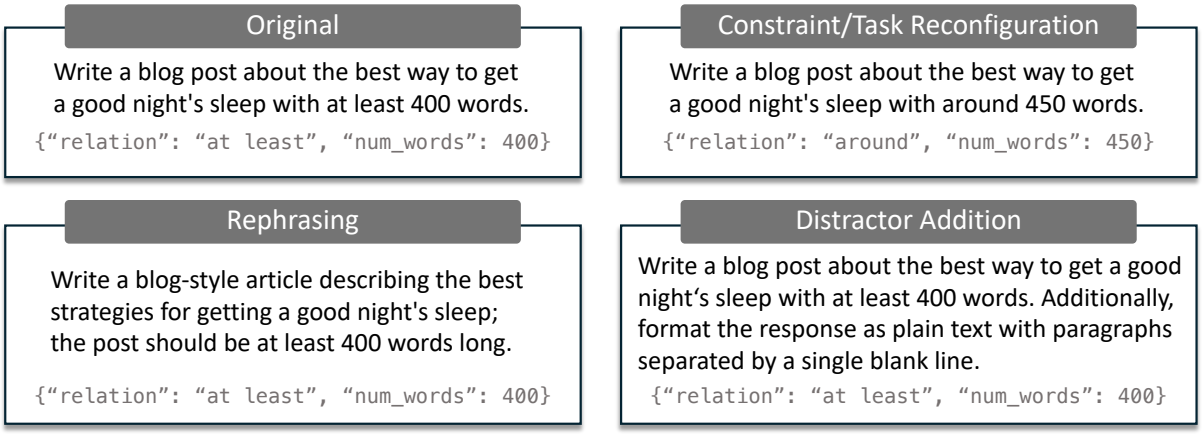

Figure 2: **Examples of Cousin Prompts**. The corresponding evaluation configurations are provided.

`reliable@k`, which quantifies whether an LLM can simultaneously solve a set of *cousin prompts*. We further design an automated pipeline for generating high-quality cousin prompts using three data augmentation strategies: rephrasing, distractor addition, and task/constraint reconfiguration. This is coupled with a code-assisted validity checker. Building on **IFEVAL** (Zhou et al., 2023), we leverage the pipeline to construct an extended benchmark, **IFEVAL++**[1], which consists of 541 test cases, each comprising 10 cousin prompts. This provides a testbed for assessing nuance-oriented reliability of LLMs in a systematic way.

Using **IFEVAL++**, we comprehensively evaluate 20 proprietary and 26 open-source LLMs. Our analysis reveals that current LLMs often exhibit considerable inconsistent competence across cousin prompts. The relative drop from accuracy on **IFEVAL** to `reliable@10` on **IFEVAL++** can be as large as 61.8% for `Qwen3-0.6B` and 54.7% for `GPT-3.5-turbo-1106`, while even the most reliable model, `GPT-5`, experiences a decrease of 18.3%. Notably, nuance-oriented reliability emerges as **a second-order property**: for instance, while `Gemma-3-IT-27B` ranks 17th in accuracy on **IFEVAL**, it rises to 7th when considering `reliable@10` on **IFEVAL++**. Furthermore, we empirically characterize how models' nuance-oriented reliability evolves with chronological development, model scale, reasoning capability, and augmentation type.

Observing the insufficiency of current LLMs in nuance-oriented reliability, we investigate three potential pathways for improvement: prediction-based methods, training-based methods, and test-time scaling. Among these, parallel test-time scaling through rejection sampling proves most effective, enabling a relatively weak model such as `Qwen3-4B` to surpass even the strongest open-source model, `LLaMA-3.3-70B-Instruct`.

We contend that improving this *nuance-oriented reliability* is a crucial next step toward dependable, trustworthy LLM behaviors. In summary, our contributions primarily lie in:

- We investigate the nuance-oriented reliability of LLMs in instruction-following and propose a new metric, `reliable@k`, to quantify it.
- We develop an automated pipeline for generating cousin prompts, and build **IFEVAL++**, a benchmark that enables actionable evaluation of nuance-oriented reliability.
- Using **IFEVAL++**, we conduct comprehensive experiments across a wide range of models and explore three distinct pathways to enhance reliability.

2 Preliminaries

2.1 Problem Statement

Let F be an auto-regressive language model that takes a prompt x and produces a response $y = F(x)$. Following prior work (Zhou et al., 2023), instruction-following ability is typically evaluated by checking whether y satisfies a set of constraints $C = \{c_1, c_2, \ldots, c_m\}$ specified in the prompt as natural language instructions. Please refer to Appendix A for related works.

A Pilot Experiment. To ground our discussion, we begin with a pilot experiment focusing on test cases involving word-length constraints on responses. From **IFEVAL**, we extract 48 suitable prompts characterized by a relation (*at least*, *at most*, or *around*) and a numerical value *num_word*, as exemplified in Figure 2. We then prompt `GPT-5-mini` to

[1]We name it to show respect for the renowned **IFEVAL** benchmark, as we position **IFEVAL++** as a successor of **IFEVAL**, similar to how C++ relates to C.

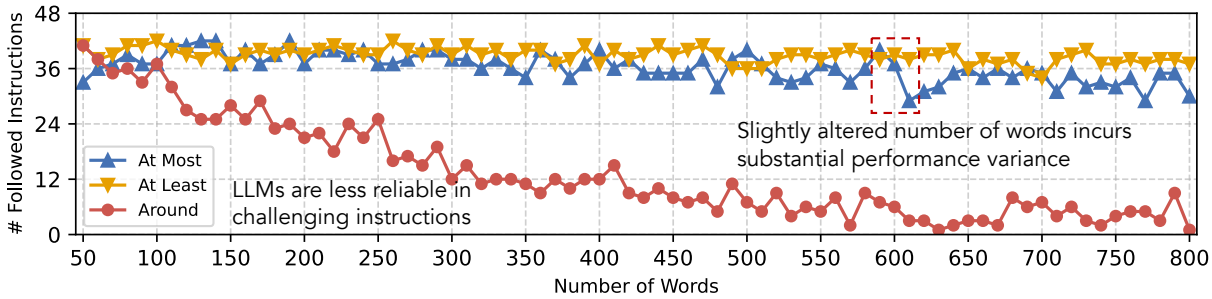

Figure 3: **Results of Requesting Varying Number of Response Words**. Experiments with `Qwen3-8B`.

convert the prompt in each test case into a template, leaving the relation and value configurable. Using a length interval of 10, ranging from 50 to 800, we program to instantiate the templates, yielding $3 \times 76 \times 48 = 10,944$ prompts. The evaluation configurations are adjusted accordingly. As shown in Figure 3, LLM performance is highly sensitive to the requested word count. A minor modification, *e.g.*, from "at most 600" to "at most 610", can incur many test cases to fail. This sensitivity is especially pronounced under the more challenging *around* constraint. This confirms that LLMs may not be reliable across prompts with subtle nuances. We ask: Shouldn't a reliable LLM tackle them all?

Two Reliability Dimensions in Instruction-Following. To situate our research, we review 36 benchmarks designed to evaluate instruction-following performance (see Appendix A.1). Most of these works focus on *benchmark comprehensiveness*—seeking evaluations that span diverse task types, domains, and constraint categories. This focus aligns with the view that a reliable assistant should handle diverse user demands across scenarios. However, our pilot study suggests that comprehensiveness alone is insufficient to capture the full notion of reliability. In real-world use, users may phrase identical requirements in multiple ways or express similar requirements with nuanced differences (see Figure 2 for examples). A truly reliable LLM should consistently produce correct outputs across these prompts. We therefore propose a complementary perspective: **nuance-oriented reliability**, which measures the model's stability when responding to "cousin prompts" that convey analogous user intents and differ only in subtle linguistic or semantic nuances.

2.2 Scope of This Work

New Metric: `reliable@k`. To capture it, we introduce a more challenging metric, `reliable@k`, designed to provide an actionable evaluation of LLMs' nuance-oriented reliability. Specifically, `reliable@k` quantifies how consistently an LLM can handle a set of closely related cousin prompts simultaneously. Formally, for k cousin prompts with model outputs $\{y_1, y_2, \ldots, y_k\}$, we define

$$\texttt{reliable@k} = \mathbb{I}\left(\sum_{j=1}^{k} \text{is_passed}(y_j) = k\right),$$

where $\text{is_passed}(y_j)$ indicates whether the model's response to the j-th cousin prompt satisfies the evaluation criterion. Cousin prompts can be obtained by augmenting the original prompts in existing benchmarks, which we detail in Section 3. The parameter k and the choice of cousin prompts together control the rigor of the nuance-oriented reliability evaluation, rendering `reliable@k` scalable (see Appendix D.2 for experimental evidence). Notably, when $k = 1$, `reliable@1` reduces to the standard accuracy metric, since only the original prompts are included.

Data Source. Specifically, we choose **IFEVAL** (Zhou et al., 2023) as our main testbed, since state-of-the-art LLMs have already achieved strong performance on it (*e.g.*, OpenAI's `GPT-5` reaches an accuracy of 95.9% as measured in Table 1), making it a meaningful setting for discussing reliability and the challenging `reliable@k` metric. This benchmark focuses on single-turn tasks with code-verifiable format constraints, such as "*Answer using at most {num_words} words*." Reliably tackling such ***simple yet foundational*** instructions is a necessary condition for addressing more complex, multi-turn, and multi-modal instruction-following settings. See Appendix B.1 for the 25 constraints.

3 Curating Cousin Prompts At Scale

We introduce three data augmentation methods for producing cousin prompts (Section 3.1), our checker for ensuring augmented test case validity (Section 3.2), and the operational procedures for finalizing the **IFEVAL++** benchmark (Section 3.3).

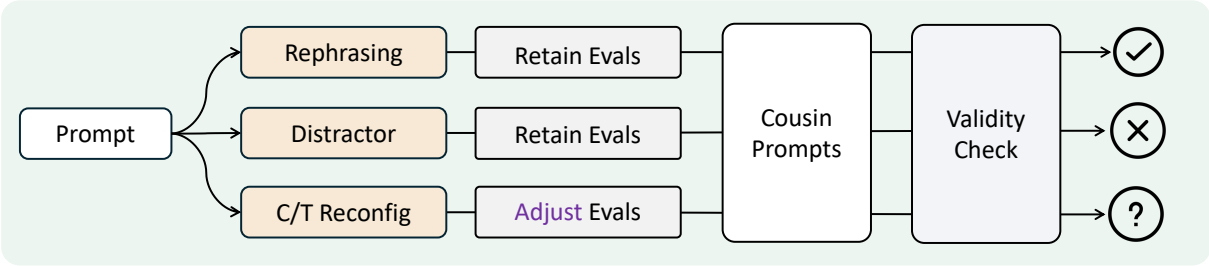

Figure 4: **Conceptual Illustration of the Augmentation-Filtering Data Curation Pipeline**.

3.1 Data Augmentation Methods

Data augmentation for cousin prompts primarily involves revisions on two components: rewriting the prompts and adjusting the evaluation configurations, if necessary. In this work, we mainly account for three types of augmentation methods and employ LLM-based augmentation. See examples of the augmented cousin prompts in Figure 2.

Type 1. Rephrasing. This reflects the scenario where different users may articulate the same task in varying ways. The augmenter rephrases each of the 541 prompts from **IFEVAL** into alternative wordings while preserving the core task and constraint semantics (see Prompt 2). For rephrasing, the corresponding evaluation configurations remain unchanged, since the rephrased prompts still request responses that satisfy the same requirements as the original prompts.

Type 2. Distractor Addition. The augmenter appends each prompt with a distractor constraint that does not interfere with the satisfaction of the original constraints (see Prompt 3). The augmenter is instructed to ensure that each distractor constraint remains compatible with the original prompt. In this setting, responses continue to be evaluated against the original constraints, with the added constraint functioning purely as a distractor, in contrast to prior work on multi-constraint composition (Jiang et al., 2024; Wen et al., 2024). This setup examines whether LLMs can integrate additional constraints in a plug-and-play manner.

Type 3. Constraint/Task Reconfiguration. In this setting, we aim to examine whether LLMs can reliably follow instructions when prompts are presented in varied scenarios. To this end, we craft cousin prompts that share the same requirement types but differ in subtle ways. For test cases with at least one configurable constraint (*e.g.*, *num_word*), we instruct the augmenter to modify the constraint and update the evaluation configurations accordingly (see Prompt 4). This mirrors the pilot experiment introduced in Section 2.1. For cases with only non-configurable constraints, such as requiring responses in *JSON* format, we instead configure the task to indirectly impose the same requirement in a different context (Prompt 1). In the case of task reconfiguration, the evaluation configurations remain unchanged.

3.2 Code-Assisted Validity Check

LLM-based augmentation enables us to rapidly acquire cousin prompts, but risks introducing flawed test cases. To mitigate this issue, we develop an automated validity checker.

The checker is designed with an emphasis on high recall: its primary objective is to flag any potentially erroneous cases, even at the cost of over-detection. This strategy allows us to curate a high-quality evaluation set by filtering candidates against the checker. Inspired by Chen et al. (2023), our central design is to embed the implementations of evaluation functions directly into the checker's prompt, clearly specifying how the evaluation operates (see Prompt 5). Empirically, this approach enables more deterministic and robust verification. The checker is implemented in a unified way to monitor all three types of augmented test cases.

Validating the Checker. Following principles in Appendix B.2, we iteratively refined the checker prompt on $3 \times 300 = 900$ test cases, where the augmenter was deliberately instructed to inject errors. This is an aggressive modeling of flawed prompts that the checker may encounter. Using `GPT-5-mini` (medium reasoning effort), the checker achieved a recall of 99.7% (898/900).

To further assess robustness, we evaluated it on $3 \times 1,000 = 3,000$ additional flawed cases, maintaining a recall of 99.9% (2,997/3,000); the few misses were due to unavoidable hallucinations (Sun et al., 2025). The checker also achieved 90.0% precision on 541 unrevised prompts[2]. Overall, the

[2]Many flagged cases were indeed flawed or ambiguous. Necessary corrections are detailed in Appendix B.3, and the revised dataset is used for subsequent experiments.

Table 1: **Nuance-Oriented Reliability of Representative LLMs**. We use `rel@k` for `reliable@k` and "C/T" for constraints or tasks. We report `reliable@k` across **IFEVAL++** subsets by augmentation method. Models are ranked by `reliable@10` on **IFEVAL++** (with the rankings of **IFEVAL** accuracy for reference). Reasoning models are in *blue*, non-reasoning in *beige*. Best and second-best scores are in **bold** and underline.

Model	IFEVAL		+Rephrasing		+Distractor		+C/T Reconfig		IFEVAL++	
	Acc	Rank	rel@2	rel@4	rel@2	rel@4	rel@2	rel@4	rel@10	Rel Δ
Open-Source LLMs										
LLaMA-3.3-70B-Instruct	**92.1**	1	**89.6**	**88.9**	**88.0**	**80.0**	**89.6**	**80.6**	**71.0**	**-22.9%**
DeepSeek-V3.1	90.6	5	86.0	81.3	85.6	77.3	86.5	76.0	63.4	-30.0%
Qwen3-Next-80B-A3B-TK	91.3	2	86.1	80.8	84.8	76.5	86.3	76.9	63.2	-30.8%
Kimi-K2-0905	91.3	2	86.9	81.5	85.8	78.0	85.2	76.0	63.0	-31.0%
Qwen3-Next-80B-A3B-IT	90.9	4	84.6	80.4	86.5	77.3	86.5	75.8	62.3	-31.5%
GLM-4.5	86.9	10	82.6	78.2	81.1	72.5	83.2	74.9	61.7	-28.9%
Gemma-3-IT-27B	84.3	17	80.4	77.8	79.1	73.4	80.4	70.2	61.6	-27.0%
Qwen3-14B	88.4	6	85.4	81.5	83.2	73.4	83.2	72.5	61.2	-30.8%
Gemma-3-IT-12B	84.3	17	81.1	78.2	79.7	73.0	79.3	69.5	59.5	-29.4%
Qwen3-8B	87.6	7	83.4	79.5	81.7	72.8	81.3	69.9	58.8	-32.9%
LLaMA-3.1-70B-Instruct	85.6	15	82.3	79.1	79.5	71.0	81.0	69.1	57.1	-33.3%
Qwen3-30B-A3B	87.6	7	80.6	75.6	81.5	70.6	82.6	71.2	57.1	-34.8%
GLM-4.5-Air	87.4	9	82.6	76.3	81.1	70.8	81.5	70.8	56.7	-35.1%
Qwen3-235B-A22B	86.3	12	81.0	73.8	78.6	69.7	81.3	70.1	56.6	-34.5%
GPT-oss-120b	86.0	14	81.9	75.4	80.0	72.6	77.3	67.1	56.4	-34.4%
Qwen2.5-72B-Instruct	86.7	11	82.6	76.3	77.6	68.2	80.0	69.1	55.6	-35.9%
Qwen3-32B	86.3	12	81.5	75.8	81.9	72.3	81.7	69.7	55.3	-36.0%
DeepSeek-R1	83.4	20	77.1	70.6	77.8	69.1	75.8	65.4	53.0	-36.4%
Qwen3-4B	85.2	16	79.9	74.5	78.7	67.8	79.1	65.4	52.1	-38.8%
Qwen2.5-32B-Instruct	82.4	21	77.6	71.2	72.5	62.8	75.4	64.1	50.3	-39.0%
GPT-oss-20b	83.9	19	76.5	68.2	76.9	66.2	73.2	61.9	46.8	-44.3%
Qwen2.5-14B-Instruct	78.2	22	72.6	67.1	69.5	59.0	69.7	57.3	44.7	-42.8%
LLaMA-3.1-8B-Instruct	75.8	23	70.8	64.7	65.8	57.1	67.1	53.8	41.4	-45.4%
Qwen2.5-7B-Instruct	73.0	24	66.2	60.1	62.5	50.5	63.8	50.6	34.8	-52.3%
Qwen3-1.7B	71.5	25	63.4	55.1	62.3	49.4	61.6	46.2	34.0	-52.5%
Qwen3-0.6B	58.0	26	47.5	41.4	46.0	33.3	46.6	34.2	22.2	-61.8%
Proprietary LLMs										
GPT-5	**95.9**	1	**93.3**	**90.4**	**93.0**	**86.5**	**93.7**	**88.7**	**78.4**	**-18.3%**
o3	94.3	3	93.5	89.1	92.1	84.5	91.7	85.0	75.0	-21.3%
o4-mini	95.6	2	91.9	87.1	90.4	81.7	91.9	84.8	72.3	-24.4%
Gemini-2.5-pro	93.3	5	89.6	87.4	88.4	82.6	91.3	83.0	71.9	-23.0%
GPT-5-mini	94.1	4	90.6	85.2	89.3	83.0	89.1	82.8	70.6	-25.0%
o3-mini	91.4	6	89.5	86.0	88.7	81.5	88.5	80.6	70.1	-25.2%
Gemini-2.5-flash	90.4	8	87.6	84.3	85.6	78.4	86.7	78.2	66.4	-26.5%
Gemini-2.0-flash	90.6	7	87.8	83.7	84.4	78.0	85.6	77.3	65.8	-27.3%
GPT-4.1	89.8	9	86.3	83.0	83.9	76.3	84.3	76.9	64.7	-28.0%
Gemini-2.5-flash-lite	88.0	10	83.9	81.5	83.2	75.8	83.0	73.2	62.7	-28.8%
GPT-5-nano	90.6	7	85.0	81.3	85.8	77.3	83.9	71.5	61.6	-32.0%
GPT-4o	85.0	13	80.2	76.3	79.7	71.3	79.3	69.1	58.6	-31.1%
GPT-4.1-mini	86.9	11	83.2	78.9	79.9	71.7	80.6	70.2	58.4	-32.8%
GPT-4o-2024-05-13	85.4	12	81.3	78.0	79.1	71.7	79.1	66.7	57.9	-32.3%
GPT-4o-2024-11-20	80.8	15	77.6	73.9	75.4	68.8	75.2	64.1	55.3	-31.6%
GPT-4o-mini	80.4	16	75.8	71.7	72.1	64.1	72.8	60.3	48.4	-39.8%
o1-mini	83.4	14	76.2	66.9	74.7	65.2	73.8	59.3	46.4	-44.3%
GPT-4.1-nano	77.4	17	72.5	67.8	70.1	60.6	68.8	55.6	42.9	-44.6%
GPT-3.5-turbo-0125	66.4	18	57.9	50.8	55.5	44.7	55.3	42.1	32.2	-51.5%
GPT-3.5-turbo-1106	61.6	19	53.0	48.4	50.5	41.0	51.8	37.8	27.9	-54.7%

checker rigorously detects invalid prompts with minimal valid loss. Moreover, since the augmenter is also explicitly guided to avoid introducing flaws, the two-layer mechanism minimizes the risk of flaws arising from the augmentation process.

3.3 Synthesizing the IFEVAL++ Benchmark

In this section, we synthesize **IFEVAL++**: To balance cost and coverage, each original test case is expanded into nine replicates, with three derived

from each augmentation method. Following the augmentation methodology introduced earlier, we automatically produce cousin prompts and verify them using the code-assisted validity checker to ensure test case integrity. This process is iterated until a sufficient number of valid cousin prompts are collected.

The **IFEVAL++** benchmark comprises 541 test cases, each containing one original prompt and nine cousin prompts. With **IFEVAL++**, we can characterize the nuance-oriented dimension of LLMs' reliability, which cannot be revealed by evaluating only standalone prompts. Accordingly, we report `reliable@1` (*i.e.*, accuracy on **IFEVAL**), `reliable@10` on **IFEVAL++**, and `reliable@4` and `reliable@2` on the subsets of **IFEVAL++** corresponding to each augmentation type.

4 Experiments

4.1 Experimental Setup

Models. In this work, we include a diverse set of 46 models to facilitate comparative analyses across four key dimensions: (a) differences in model scale, (b) chronologically different generations of models, (c) contrasts between open-source and proprietary models, and (d) distinctions between reasoning and non-reasoning models. Additional model details are provided in Appendix C.

Generation Strategies. We use official default system messages when provided in chat template (*e.g.*, for `Qwen-2.5` models); otherwise, we leave the system message empty to elicit the model's default behavior, as suggested by models like `DeepSeek-R1`[3]. We further validate this choice in Appendix D.1. We employ greedy decoding by default, which is suitable for reliability evaluation and provides a reproducibility guarantee. Additionally, we enable reasoning mode by default if supported.

Evaluation Methods. Following the practices of `DeepSeek-V3` (Liu et al., 2024), `Qwen3` (Yang et al., 2025a), and `Kimi K2` (Team et al., 2025), we use the *prompt strict* evaluation mode of **IFEVAL**. This requires LLMs to satisfy all requirements specified in the prompts. Based on this success signal, we compute the `reliable@k` metric.

4.2 Main Results with IFEVAL++

We answer the initial question: **Current LLMs do not guarantee nuance-oriented reliability.**

As shown in Table 1, the relative drop from accuracy on **IFEVAL** to `reliable@10` on **IFEVAL++** is substantial, reaching up to 61.8% for `Qwen3-0.6B` and 54.7% for `GPT-3.5-turbo-1106`. Even the most reliable model, `GPT-5`, suffers an 18.3% decline when confronted with prompts containing subtle variations. Importantly, we observe that higher accuracy on **IFEVAL** does not necessarily translate into higher `reliable@10` on **IFEVAL++**. For example, while `Gemma-3-IT-27B` ranks only 17th on **IFEVAL**, it rises to 7th place on **IFEVAL++**. This highlights that **nuance-oriented reliability is a second-order property beyond accuracy**, one that comprehensiveness-oriented instruction-following benchmarks fail to capture. Besides, we arrive at several empirical findings:

- **Chronological Development**: As expected, more recent LLMs tend to perform better, especially when developed by the same vendor. This underscores the importance of the underlying training methodology, as highlighted by the stark difference in reliability between `LLaMA-3.3-70B-Instruct` (`reliable@10`: 71.0) and its predecessor, `LLaMA-3.1-70B-Instruct` (`reliable@10`: 57.1).
- **Model Scale**: In general, larger-scale models tend to outperform smaller ones. An exception is observed within the `Qwen3` family, where `Qwen3-14B` demonstrates lower accuracy on **IFEVAL** but achieves a higher `reliable@10` score compared to the larger `Qwen3-32B`. This indicates that surpassing a certain parameter threshold is sufficient to attain strong reliability. Models below this threshold, such as `Qwen3-0.6B`, are unable to handle instructions effectively. At the same time, model performance is not determined by scale alone: Training data quality and methodology also play crucial roles. Notably, models with comparable parameter counts but developed by different vendors can show substantial differences in reliability, as exemplified by `Gemma-3-IT-27B` and `Qwen2.5-32B-Instruct`.
- **Reasoning**: As shown, recent reasoning models demonstrate increasingly stronger nuance-oriented reliability. However, `LLaMA-3.3-70B-Instruct`, though not a reasoning model, still achieves the highest ranking among open-source models. This suggests reasoning itself may not be a prerequisite for achieving high reliability. This motivates us to examine the specific contribution of reasoning to reliability, which we investigate in Section 5.3.
- **Augmentation Type**. Models exhibit varying robustness to the three types of cousin prompts.

[3] https://huggingface.co/deepseek-ai/DeepSeek-R1#usage-recommendations

Table 2: **AUROC Scores of Prediction Methods**.

Method	Qwen3-8B	Qwen2.5-7B
Verb. Conf.	0.549	0.518
Perplexity	0.497	0.529
Probing	0.757	0.759

Most models can effectively handle rephrased cousin prompts, which share the greatest similarity with the original ones. This result is expected, as rephrased prompts differ only in wording (understanding the instructions) while requesting the same underlying response (executing the instructions). In contrast, cousin prompts that include additional distractor constraints or C/T reconfiguration introduce greater complexity. Their corresponding `reliable@4` scores are typically lower than those of rephrased prompts. We hypothesize that, although distractor constraints are designed to remain compatible with the original instructions, and C/T reconfiguration introduces only slight variations in the underlying requirements, both inevitably increase the difficulty of response planning and execution (Dong et al., 2025b).

5 Advancing Nuance-Oriented Reliability

While LLMs are not yet reliable, we explore three possible directions for improving them.

5.1 Predicting the Selective Following

If we can predict how the LLM reacts to different prompts, we can proactively approach reliable LLM services by prompt selection. This case study goes in a similar vein to Stolfo et al. (2025); Cao et al. (2024); Heo et al. (2025a,b), but our focus is to predict the instruction-following prior to generation and across various cousin prompts. We experiment with `Qwen3-8B` and `Qwen2.5-7B-Instruct`, respectively. For each model, we randomly sample 1,200 of them as the validation set, with 600 followed and 600 unfollowed. We evaluate three potential prediction methods, with their performance presented in Table 2.

• **Verbalized Confidence**. We prompt the LLM to articulate its confidence in successfully following one given test case, on a scale from 0 to 9. However, the self-reported confidence yields performance close to random guessing. This outcome can be attributed to the LLMs' general tendency toward overconfidence, aligning with the observations of Xiong et al. (2024).

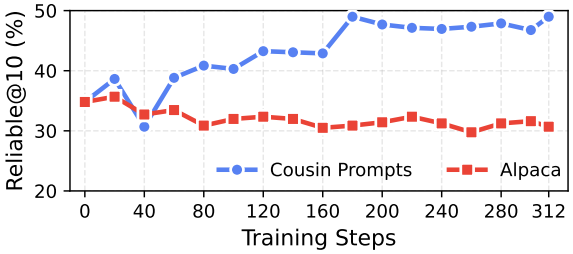

Figure 5: **Training on Datasets Affects Reliability Performance on IFEVAL++ in Varying Fashion**.

• **Prompt Perplexity**. We compute the perplexity of LLMs on each prompt and use it for predicting instruction-following. Yet, the close-to-chance AUROC reveals that low-perplexity prompts that are most familiar to the model do not necessarily correspond to successful instruction-following.

• **Probing Hidden States**. Causal LMs may plan their responses before generation, and the relevant signals can be probed from their hidden states (Dong et al., 2025b). Specifically, we train a linear logistic regression probe, which takes the hidden states of the last input token as input. To train the probe, we additionally collect a non-overlapping set of 200 followed and 200 unfollowed prompts. Compared with the previous two methods, probing provides a more effective prediction. The best validation performance is achieved when probing the 17th and the 27th layer on two models, respectively. Although probing shows potential in this direction, the probes are far from being a reliable predictor.

5.2 Training-Based Method

We adopt supervised fine-tuning (SFT) as our training method and evaluate two dataset configurations, which represent the two extremes of prompt relatedness. (1) We use the general-domain instruction-following dataset, **ALPACA**(Taori et al., 2023), whose prompts are largely unrelated to **IFEVAL++**. (2) We curate an additional set of prompts, also augmented from **IFEVAL** but decontaminated from **IFEVAL++** prompts via exact matching. Then, we collect responses via rejection sampling against `LLaMA-3.3-70B-Instruct`. We fine-tune `Qwen-2.5-7B-Instruct` for 312 steps on each dataset. Further details on data curation and training are provided in Appendix E.1.

Figure 5 reports the reliability performance on **IFEVAL++** across training steps. We observe that the two datasets lead to markedly different outcomes. Models fine-tuned on **ALPACA** exhibit a slightly declining performance. In contrast,

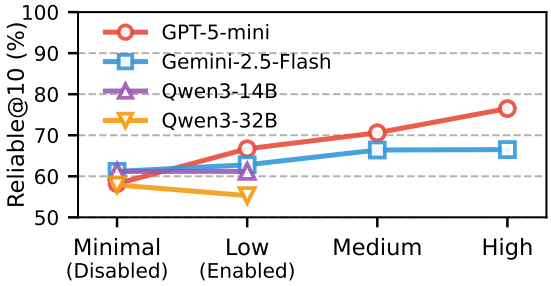

Figure 6: **Impact of Reasoning Effort on Nuance-Oriented Reliability of LLMs**.

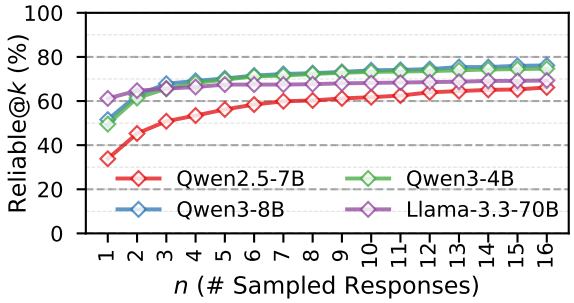

Figure 7: **Scaling Landscape of Rejection Sampling**.

training on the curated *cousin prompts* steadily improves reliability, with performance climbing above 45% after 200 steps. This reaffirms reliability as a second-order property from the training perspective. It benefits more from targeted fine-tuning on semantically adjacent samples rather than from the sheer scale of training data. This observation resonates with recent explorations on rewriting training data (Fujii et al., 2025; Team et al., 2025). In Appendix E.2, we additionally explore how different training techniques (SFT vs. DPO (Rafailov et al., 2024)) affect the nuance-oriented reliability.

5.3 Test-Time Scaling

We explore two forms of scaling up test-time compute (Snell et al., 2024): sequential and parallel.

• **Sequential (Reasoning Effort)**. Reasoning models inherently capture thinking patterns such as self-correction (Guo et al., 2025). Scaling up reasoning effort can be viewed as a sequential analogue to scaling up test-time compute. We include four models that support adaptive reasoning budget control: `GPT-5-mini`, `Gemini-2.5-Flash`, `Qwen3-14B`, and `Qwen3-32B`. As shown in Figure 6, increasing reasoning effort usually improves reliability, particularly for `GPT-5-mini`. Manual inspection of their intrinsic CoT reveals that, through reasoning, models can infer user intent and adapt responses accordingly. This explicit reasoning process enhances robustness against subtle prompt variations. However, inappropriate reasoning patterns may reduce reliability (Chen et al., 2024a; Dong et al., 2025a), as evidenced by the negative effects observed in `Qwen3-32B`.

• **Parallel (Rejection Sampling)**. To scale test-time compute in parallel, we adopt sampling rather than greedy decoding, generating n samples with a temperature of 1. We assume the availability of a response selector. For example, in prompts related to **IFEVAL++**, format requirements can be extracted with the assistance of an LLM, while Python code can be used for rapid response evaluation. With the response selector, the `reliable@10` metric is scored positively if the 10 cousin prompts within one group all have at least one response that passes among the n samples. The results are shown in Figure 7. Rejection sampling with a suitable response selector effectively enhances the reliability of LLMs in instruction-following. The `reliable@10` scores rise significantly as more samples are allowed, plateauing around $n = 12$. We further observe that reasoning models, such as `Qwen3-8B`, benefit more from additional compute via parallel scaling, given their more exploratory nature (Chen et al., 2025). For instance, `Qwen3-4B` and `Qwen3-8B`, with only $n = 3$ samples, already surpass `LLaMA-3.3-70B` (the strongest open-source model in Table 1). However, we note that extending this parallel scaling strategy to other domains may be challenging without suitable response selectors.

6 Conclusion

In this work, we present a systematic study on the nuance-oriented reliability of LLMs in instruction-following. To characterize this property, we introduce the new metric `reliable@k`, which can be instantiated with cousin prompts. We design and implement a fully automated pipeline to generate such cousin prompts through data augmentation methods, combined with a code-assisted validity checker. Leveraging this pipeline, we construct the **IFEVAL++** benchmark, which enables systematic evaluation. Extensive experiments across 46 models using **IFEVAL++** uncover a critical vulnerability: current LLMs frequently fail to ensure reliability across cousin prompts with nuanced differences. We further explore three directions for improving this reliability: prediction-based, training-based, and test-time scaling. Overall, our findings highlight the importance of moving toward nuance-oriented reliability as a key dimension of dependable, trustworthy LLM behaviors.

7 Limitations

We identify several points that are not fully solved in this work and are worth future investigation.

Measurement Efficiency. Our practice of evaluating nuance-oriented reliability is straightforward, where we introduce cousin prompts to avoid the risk of hacking certain test cases. This enables us to compute the `reliable@k` metric to monitor the reliability dimension. However, this also introduces a higher evaluation cost. For example, a full evaluation of **IFEVAL++** requires 10× as many responses to be generated as **IFEVAL**. Future work will focus on selecting or crafting those most differentiating test cases and cousin prompts, which enables a more cost-efficient evaluation.

Missing Content-Level Evaluation. Following common practice in prior studies, we primarily evaluate whether LLMs adhere to constraint instructions. Moreover, prompts often include one or multiple tasks, which are not explicitly assessed by either **IFEVAL** or **IFEVAL++**. This design choice is acceptable, as satisfying constraints is a necessary prerequisite for correctly following the entire prompt. Future work could extend this by jointly measuring content-level response quality along with constraint adherence.

Focus on IFEVAL. In this work, we primarily adopt **IFEVAL** as our testbed. It is widely recognized in the community as a standard benchmark for instruction-following evaluation and has approached performance saturation with the latest models. Nevertheless, our reliability evaluation methodology is not limited to this single benchmark. Future research may extend our approach to other existing benchmarks, or even beyond the instruction-following domain. We envision that evaluating nuance-oriented reliability will become a default practice, complementing traditional capacity-oriented evaluation.

Potential Bias in Automated Validity Check. Although the code-assisted validity checker demonstrates high recall and precision, its automation depends on LLM-based judgment, which may introduce subtle biases or misclassifications. We carefully and systematically validate the checker's performance in Section 3.2 and additionally dedicate substantial, though undocumented, manual effort to continuously monitor and ensure the quality of the checking process. Moreover, the strong performance of advanced models—such as `GPT-5`, which achieves a `reliable@10` of 78.4—suggests that most test cases are indeed valid and feasible; otherwise, invalid cases would cause even the strongest LLMs to fail consistently.

Limited Coverage of Improvement Recipes. While numerous efforts have been proposed to enhance instruction-following capabilities, reproducing all such methods to assess their combined impact on nuance-oriented reliability lies beyond the scope of this work. We focus on several representative improvement strategies, including prediction-based, training-based, and test-time scaling approaches. Future research could undertake a more comprehensive investigation to identify the most effective techniques for improving reliability across diverse enhancement paradigms.

8 Ethical Considerations

Our research adheres to the ethical guidelines established by the Association for Computational Linguistics (ACL)[4]. We have made every effort to conduct this work with the utmost respect for ethical principles and research integrity.

This research aims to advance the understanding of the reliability of large language models in instruction-following. All experiments were performed on publicly available models and datasets, ensuring compliance with relevant terms of service and data usage policies, *e.g.*, **IFEVAL** (Zhou et al., 2023). No proprietary or confidential information was accessed or reverse-engineered during this study. Our analyses do not involve human subjects, sensitive personal data, or the generation of harmful content. Our use of AI assistants is limited to writing polishing and grammar checking. We release the complete codebase and datasets (*e.g.*, **IFEVAL++**) to ensure full reproducibility of our results.

Finally, we remind the readers that any techniques introduced in this paper should be applied ethically and within appropriate research contexts.

References

Bowen Cao, Deng Cai, Zhisong Zhang, Yuexian Zou, and Wai Lam. 2024. On the worst prompt performance of large language models. *NeurIPS*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *TMLR*.

[4] https://aclrollingreview.org/responsibleNLPresearch/

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024a. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Xinyi Chen, Baohao Liao, Jirui Qi, Panagiotis Eustratiadis, Christof Monz, Arianna Bisazza, and Maarten Rijke. 2024b. The sifo benchmark: Investigating the sequential instruction following ability of large language models. In *EMNLP (Findings)*.

Yihan Chen, Benfeng Xu, Quan Wang, Yi Liu, and Zhendong Mao. 2024c. Benchmarking large language models on controllable generation under diversified instructions. In *AAAI*.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. 2025. Pass@ k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E Gonzalez, and 1 others. 2024. Chatbot arena: An open platform for evaluating llms by human preference. In *ICML*.

Jianshuo Dong, Yujia Fu, Chuanrui Hu, Chao Zhang, and Han Qiu. 2025a. Towards understanding the cognitive habits of large reasoning models. *arXiv preprint arXiv:2506.21571*.

Zhichen Dong, Zhanhui Zhou, Zhixuan Liu, Chao Yang, and Chaochao Lu. 2025b. Emergent response planning in LLMs. In *ICML*.

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

Elliot L Epstein, Kaisheng Yao, Jing Li, Xinyi Bai, and Hamid Palangi. 2024. Mmmt-if: A challenging multimodal multi-turn instruction following benchmark. *arXiv preprint arXiv:2409.18216*.

Thomas Palmeira Ferraz, Kartik Mehta, Yu-Hsiang Lin, Haw-Shiuan Chang, Shereen Oraby, Sijia Liu, Vivek Subramanian, Tagyoung Chung, Mohit Bansal, and Nanyun Peng. 2024. Llm self-correction with decrim: Decompose, critique, and refine for enhanced following of instructions with multiple constraints. In *EMNLP (Findings)*.

Tingchen Fu, Jiawei Gu, Yafu Li, Xiaoye Qu, and Yu Cheng. 2025. Scaling reasoning, losing control: Evaluating instruction following in large reasoning models. *arXiv preprint arXiv:2506.01776*.

Kazuki Fujii, Yukito Tajima, Sakae Mizuki, Hinari Shimada, Taihei Shiotani, Koshiro Saito, Masanari Ohi, Masaki Kawamura, Taishi Nakamura, Takumi Okamoto, and 1 others. 2025. Rewriting pre-training data boosts llm performance in math and code. *arXiv preprint arXiv:2505.02881*.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. 2024. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Qianyu He, Jie Zeng, Wenhao Huang, Lina Chen, Jin Xiao, Qianxi He, Xunzhe Zhou, Jiaqing Liang, and Yanghua Xiao. 2024a. Can large language models understand real-world complex instructions? In *AAAI*.

Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, and 1 others. 2024b. Multi-if: Benchmarking llms on multi-turn and multilingual instructions following. *arXiv preprint arXiv:2410.15553*.

Juyeon Heo, Christina Heinze-Deml, Oussama Elachqar, Kwan Ho Ryan Chan, Shirley You Ren, Andrew Miller, Udhyakumar Nallasamy, and Jaya Narain. 2025a. Do LLMs "know" internally when they follow instructions? In *ICLR*.

Juyeon Heo, Miao Xiong, Christina Heinze-Deml, and Jaya Narain. 2025b. Do LLMs estimate uncertainty well in instruction-following? In *ICLR*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *ICLR*.

Daniel Jaroslawicz, Brendan Whiting, Parth Shah, and Karime Maamari. 2025. How many instructions can llms follow at once? *NeurIPS Workshop*.

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 *ACL*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *SOSP*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Jinnan Li, Jinzhe Li, Yue Wang, Yi Chang, and Yuan Wu. 2025a. Structflowbench: A structured flow benchmark for multi-turn instruction following. In *ACL (Findings)*.

Zhenyu Li, Kehai Chen, Yunfei Long, Xuefeng Bai, Yaoyin Zhang, Xuchen Wei, Juntao Li, and Min Zhang. 2025b. Xifbench: Evaluating large language models on multilingual instruction following. *arXiv preprint arXiv:2503.07539*.

Gili Lior, Asaf Yehudai, Ariel Gera, and Liat Ein-Dor. 2025. Wildifeval: Instruction following in the wild. *arXiv preprint arXiv:2503.06573*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Yile Liu, Ziwei Ma, Xiu Jiang, Jinglu Hu, Jing Chang, and Liang Li. 2025. Maxife: Multilingual and cross-lingual instruction following evaluation. *arXiv preprint arXiv:2506.01776*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. State of what art? a call for multi-prompt llm evaluation. *Transactions of the Association for Computational Linguistics*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *NeurIPS*.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. 2025. Generalizing verifiable instruction following. *arXiv preprint arXiv:2507.02833*.

Yunjia Qi, Hao Peng, Xiaozhi Wang, Amy Xin, Youfeng Liu, Bin Xu, Lei Hou, and Juanzi Li. 2025. Agentif: Benchmarking instruction following of large language models in agentic scenarios. *arXiv preprint arXiv:2505.16944*.

Yanzhao Qin, Tao Zhang, Yanjun Shen, Wenjing Luo, Haoze Sun, Yan Zhang, Yujing Qiao, Weipeng Chen, Zenan Zhou, Wentao Zhang, and 1 others. 2025. Sysbench: Can large language models follow system messages? In *ICLR*.

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 *ACL (Findings)*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *NeurIPS*.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. 2024. Identifying the risks of LM agents with an LM-emulated sandbox. In *ICLR*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *ICLR*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Tingyu Song, Guo Gan, Mingsheng Shang, and Yilun Zhao. 2025. IFIR: A comprehensive benchmark for evaluating instruction-following in expert-domain information retrieval. In *NAACL*.

Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. 2025. Improving instruction-following in language models through activation steering. In *ICLR*.

Jiao Sun, Yufei Tian, Wangchunshu Zhou, Nan Xu, Qian Hu, Rahul Gupta, John Wieting, Nanyun Peng, and Xuezhe Ma. 2023. Evaluating large language models on controlled generation tasks. In *EMNLP*.

Wangtao Sun, Chenxiang Zhang, XueYou Zhang, Xuanqing Yu, Ziyang Huang, Pei Chen, Haotian Xu, Shizhu He, Jun Zhao, and Kang Liu. 2024. Beyond instruction following: Evaluating inferential rule following of large language models. *arXiv preprint arXiv:2407.08440*.

Zhongxiang Sun, Qipeng Wang, Haoyu Wang, Xiao Zhang, and Jun Xu. 2025. Detection and mitigation of hallucination in large reasoning models: A mechanistic perspective. *arXiv preprint arXiv:2505.12886*.

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

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

Peiding Wang, Li Zhang, Fang Liu, Lin Shi, Minxiao Li, Bo Shen, and An Fu. 2025a. Codeif-bench: Evaluating instruction-following capabilities of large language models in interactive code generation. *arXiv preprint arXiv:2503.22688*.

Zhaoyang Wang, Jinqi Jiang, Huichi Zhou, Wenhao Zheng, Xuchao Zhang, Chetan Bansal, and Huaxiu Yao. 2025b. Verifiable format control for large language model generations. In *NAACL*.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxing Xu, and 1 others. 2024. Benchmarking complex instruction-following with multiple constraints composition. *NeurIPS*.

Xiaodong Wu, Minhao Wang, Yichen Liu, Xiaoming Shi, He Yan, Lu Xiangju, Junmin Zhu, and Wei Zhang. 2025. LIFBench: Evaluating the instruction following performance and stability of large language models in long-context scenarios. In *ACL*.

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 *ACL*.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, YIFEI LI, Jie Fu, Junxian He, and Bryan Hooi. 2024. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *ICLR*.

Kaiwen Yan, Hongcheng Guo, Xuanqing Shi, Jingyi Xu, Yaonan Gu, and Zhoujun Li. 2025. Codeif: Benchmarking the instruction-following capabilities of large language models for code generation. *arXiv preprint arXiv:2502.19166*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Jian Yang, Wei Zhang, Shukai Liu, Linzheng Chai, Yingshui Tan, Jiaheng Liu, Ge Zhang, Wangchunshu Zhou, Guanglin Niu, Zhoujun Li, Binyuan Hui, and Junyang Lin. 2025b. Ifevalcode: Controlled code generation. *arXiv preprint arXiv:2507.22462*.

Shunyu Yao, Howard Chen, Austin W. Hanjie, Runzhe Yang, and Karthik R Narasimhan. 2024. COLLIE: Systematic construction of constrained text generation tasks. In *ICLR*.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. 2025. {τ}-bench: A benchmark for \underline{T}ool-\underline{A}gent-\underline{U}ser interaction in real-world domains. In *ICLR*.

Junjie Ye, Caishuang Huang, Zhuohan Chen, Wenjie Fu, Chenyuan Yang, Leyi Yang, Yilong Wu, Peng Wang, Meng Zhou, Xiaolong Yang, and 1 others. 2025. A multi-dimensional constraint framework for evaluating and improving instruction following in large language models. *arXiv preprint arXiv:2505.07591*.

Weizhe Yuan, Ilia Kulikov, Ping Yu, Kyunghyun Cho, Sainbayar Sukhbaatar, Jason Weston, and Jing Xu. 2024. Following length constraints in instructions. *arXiv preprint arXiv:2406.17744*.

Jie Zeng, Qianyu He, Qingyu Ren, Jiaqing Liang, Yanghua Xiao, Weikang Zhou, Zeye Sun, and Fei Yu. 2025. Order matters: Investigate the position bias in multi-constraint instruction following. In *ACL (Findings)*.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, William Song, Tiffany Zhao, Pranav Vishnu Raja, Charlotte Zhuang, Dylan Z Slack, Qin Lyu, Sean M. Hendryx, Russell Kaplan, Michele Lunati, and Summer Yue. 2024a. A careful examination of large language model performance on grade school arithmetic. In *NeurIPS Datasets and Benchmarks Track*.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. 2025a. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *ICML*.

Tao Zhang, Chenglin Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, and 1 others. 2024b. Cfbench: A comprehensive constraints-following benchmark for llms. *arXiv preprint arXiv:2408.01122*.

Wei Zhang, Zhenhong Zhou, Kun Wang, Junfeng Fang, Yuanhe Zhang, Rui Wang, Ge Zhang, Xavier Li, Li Sun, Lingjuan Lyu, Yang Liu, and Sen Su. 2025b. Lifebench: Evaluating length instruction following in large language models. *arXiv preprint arXiv:2505.16234*.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, and 1 others. 2025. Swift: a scalable lightweight infrastructure for fine-tuning. In *AAAI*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *NeurIPS*.

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. *arXiv preprint arXiv:2311.07911*.

Tao Zou, Xinghua Zhang, Haiyang Yu, Minzheng Wang, Fei Huang, and Yongbin Li. 2025. Eifbench: Extremely complex instruction following benchmark for large language models. *arXiv preprint arXiv:2506.08375*.

Table 3: **Key Designs of the Existing Instruction-Following Benchmarks**.

Benchmark	# Cases	Comprehensiveness-Oriented			Nuance-Oriented
		New Tasks	New Constraints	Complicated Scen.	
COLLIE-V1 (Yao et al., 2024)	2,080	◐	●	○	○
CELLO (He et al., 2024a)	523	○	◐	●	○
NPB (Sun et al., 2023)	–	●	●	○	○
FOLLOWBENCH (Jiang et al., 2024)	820	◐	●	○	○
IFEVAL (Zhou et al., 2023)	541	○	●	○	○
CODI-EVAL (Chen et al., 2024c)	9,060	○	●	○	◐
INFOBENCH (Qin et al., 2024)	500	○	●	◐	○
FOFO (Xia et al., 2024)	494	●	●	○	○
ALPACAEVAL-LI (Yuan et al., 2024)	802	○	●	◐	○
MT-BENCH-LI (Yuan et al., 2024)	240	○	●	◐	○
SIFO (Chen et al., 2024b)	800	◐	○	○	●
RULEBENCH (Sun et al., 2024)	–	●	○	●	○
COMPLEXBENCH (Wen et al., 2024)	1,150	○	◐	●	○
CFBENCH (Zhang et al., 2024b)	1,000	◐	●	●	○
SYSBENCH (Qin et al., 2025)	500	○	○	●	○
MMMT-IF (Epstein et al., 2024)	71	●	◐	●	○
REALINSTRUCT (Ferraz et al., 2024)	302	○	●	○	○
MULTI-IF (He et al., 2024b)	4,501	○	○	●	○
LIFBENCH (Wu et al., 2025)	2,766	○	◐	●	○
VFF (Wang et al., 2025b)	21k	○	●	○	○
STRUCTFLOWBENCH (Li et al., 2025a)	155	○	○	●	○
PBIF (Zeng et al., 2025)	24k	○	○	○	●
CODEIF (Yan et al., 2025)	1,200	●	●	◐	○
WILDIFEVAL (Lior et al., 2025)	11,813	○	●	◐	○
XIFBENCH (Li et al., 2025b)	465	○	◐	●	○
CODEIF-BENCH (Wang et al., 2025a)	–	○	●	●	○
MULDIMIF Ye et al. (2025)	1,200	○	●	◐	○
MATHIF (Fu et al., 2025)	420	●	◐	●	○
AGENTIF (Qi et al., 2025)	707	●	●	●	○
LIFEBENCH (Zhang et al., 2025b)	10,800	◐	●	○	○
MAXIFE (Liu et al., 2025)	18,285	○	○	●	○
EIFBENCH (Zou et al., 2025)	1,000	◐	◐	●	○
ORDERED COMMONGEN (Song et al., 2025)	27,648	●	●	◐	○
IFBENCH (Pyatkin et al., 2025)	300	○	●	○	○
IFSCALE (Jaroslawicz et al., 2025)	–	○	●	◐	◐
IFEVALCODE (Yang et al., 2025b)	1,600	●	◐	◐	○

A Related Works

A.1 Benchmarking Instruction-Following

We provide a chronological survey in Table 3.

COLLIE-V1 (Yao et al., 2024). This work proposes a grammar-based framework for constructing controlled text generation, allowing automatic conversion from formal constraint configurations to natural language instructions. The final compiled benchmark includes 2,080 instances comprising 13 constraint structures.

CELLO (He et al., 2024a). Accounting for the complexity of in-the-wild user instructions, this work focuses on long and complex instructions. The proposed **CELLO** benchmark contains 523 instructions that cover four types of response constraints: count limit, answer format, input dependency, and phrase callback.

NPB (Sun et al., 2023). This work evaluates the controllability of LLMs during generation tasks. It includes numerical planning constraints, requesting responses with varying lengths.

FOLLOWBENCH (Jiang et al., 2024). This work assesses the constraint-following abilities of LLMs, considering five types of constraints: content, situation, style, format, and example. It involves incrementally adding constraints when curating instructions, yielding a total of 820 instructions with different difficulty levels.

IFEVAL (Zhou et al., 2023). This work pioneers in proposing a broad taxonomy of code-verifiable constraints. The resulting benchmark consists of 541 instructions, covering 25 types of constraints.

CODI-EVAL (Chen et al., 2024c). This work introduces expansion and rewriting steps to enhance instruction diversity. The resulting **CODI-EVAL** benchmark consists of 9,060 instructions and covers a range of constraints, including sentiment, topic, keyword, length, and toxicity avoidance.

INFOBENCH (Qin et al., 2024). This work proposes a novel scoring method: preparing a check-

list for each instruction and employing LLM-as-a-Judge to decide the acceptance. Meanwhile, it introduces the **INFOBENCH** benchmark, which comprises 500 instructions and involves constraints like content constraint, linguistic constraint, style rules, format specifications, and number limits.

FOFO (Xia et al., 2024). This work specifically studies the format-following abilities of LLMs. It proposes the **FOFO** benchmark, which contains 494 instructions. Each test case requires the responses to adhere to domain-specific formats, *e.g.*, medical reports and Latex. LLM-as-a-Judge is employed for automated adherence evaluation.

ALPACAEVAL-LI & MT-BENCH-LI (Yuan et al., 2024). This work investigates LLMs' ability to follow length constraints in instructions. To facilitate this evaluation, the authors extend **ALPACAEVAL** (Dubois et al., 2024) and **MT-BENCH** (Zheng et al., 2023) by constructing the corresponding length-instructed variants

SIFO (Chen et al., 2024b). This work focuses on a specific aspect of instruction-following evaluation—ordering of multiple instructions (*i.e.*, constraints in this work). The **SIFO** benchmark comprises 800 test cases, with LLMs explicitly instructed to address multiple constraints in a prescribed sequential order. This is implemented as a single-turn interaction.

RULEBENCH (Sun et al., 2024). This work studies inferential rule following, where additional *if–then* rules are incorporated into the instructions. This setting differs from the affirmatively requested constraints in other works.

COMPLEXBENCH (Wen et al., 2024). This work studies LLMs' abilities to follow complex instructions involving multiple constraint compositions. The **COMPLEXBENCH** consists of 1,150 instructions, each associated with a constraint composite. The composition types considered include *And*, *Chain*, *Selection*, and *Nested*.

CFBENCH (Zhang et al., 2024b). This work emphasizes the comprehensiveness of constraint types for evaluating instruction following. It proposes a constraint system with 25 subcategories and meticulously curates 1K Chinese instructions.

SYSBENCH (Qin et al., 2025). This work echoes the real-world user practices involving system prompts. They evaluate the instruction-following ability of LLMs, involving 500 system prompts requesting six types of constraints.

MMMT-IF (Epstein et al., 2024). This work extends the instruction-following evaluation to multimodal and multi-turn settings. The **MMMT-IF** benchmark contains 71 test cases, with image inputs and multiple constraints scattered across chat.

REALINSTRUCT (Ferraz et al., 2024). This paper introduces real-world user queries as a source for instruction-following test cases. The authors meticulously curate instruction candidates from ShareGPT and then decompose them into tasks, contexts, and constraints. The final test set includes 302 instances, each of which typically contains multiple constraints.

MULTI-IF (He et al., 2024b). This work extends instruction-following evaluation to multi-turn and multilingual settings. The **MULTI-IF** benchmark is composed of 4,501 multilingual conversations, constructed by augmenting **IFEVAL** with multi-turn interactions and translations.

LIFBENCH (Wu et al., 2025). This work studies instruction-following abilities of LLMs in long-context tasks, *e.g.*, processing multiple documents. The benchmark consists of 2,766 instructions, involving long-context tasks and 6 types of instruction-following capabilities.

VFF (Wang et al., 2025b). This work exclusively explores the format-constraint following abilities of LLMs. It constructs 21K test cases with varying levels of difficulty by combining one to three verifiable constraints per instruction. The constraint types include limited word count, limited content, limited punctuation, limited structure, limited grammar, and specific number formatting.

STRUCTFLOWBENCH (Li et al., 2025a). This work explores the instruction-following performance of LLMs in multi-turn scenarios, emphasizing the structural dependencies that arise across turns. The benchmark incorporates eight types of constraints and six categories of structural flows: *follow-up*, *refinement*, *recall*, *expansion*, *summary*, and *unrelatedness*. In total, it includes 155 dialogues, evaluated via LLM-as-a-Judge.

PBIF Zeng et al. (2025). This work investigates how the ordering of multiple constraints influences the instruction-following performance of LLMs. It identifies an intriguing phenomenon: LLMs often perform better when the constraints are presented in a *hard-to-easy* order.

CODEIF (Yan et al., 2025). This work introduces a benchmark for evaluating LLMs' instruction-following in code generation, covering tasks like synthesis, debugging, refactoring, and explanation. It comprises 1,200 test cases across 8 categories and 50 sub-instructions, spanning multi-

ple programming languages.

WILDIFEVAL (Lior et al., 2025). This work curates a set of 12K complex instructions collected from real-world user queries on Chatbot Arena (Chiang et al., 2024). Each instruction is then decomposed into 8 categories of constraints, and LLM-as-a-Judge is employed for automated evaluation of instruction-following.

XIFBENCH (Li et al., 2025b). This work primarily explores the instruction-following abilities of LLMs across different language settings, from high-resource (English and Chinese) to low-resource (Hindi). It starts with five categories of constraints: content, style, situation, format, and numerical. By composing between one and five constraints into each seed instruction, the authors construct a total of 465 hard instructions. These are further extended into multiple languages through translation, resulting in a multilingual benchmark.

CODEIF-BENCH (Wang et al., 2025a). This work focuses on multi-turn interactive code generation. It covers 9 types of verifiable instructions aligned with real-world software development requirements for test-case validation, and supports both Static and Dynamic Conversation settings.

MULDIMIF Ye et al. (2025). This work proposes a multi-dimensional constraint framework, involving three types of constraint formatting patterns (*Example*, *Listing*, and *Incorporation*), four categories of constraints, and four difficulty levels. In total, this benchmark consists of 1,200 code-verifiable instructions.

MATHIF (Fu et al., 2025) is a dedicated benchmark for evaluating instruction-following in mathematical reasoning tasks. It contains 420 evaluation samples derived from 15 Python-verifiable constraints across 4 categories, applied to math problems of varying difficulty.

AGENTIF (Qi et al., 2025) evaluates LLMs' instruction-following abilities in agentic scenarios with long, complex, and constraint-heavy instructions. It comprises 707 human-annotated instructions drawn from 50 real-world agentic tasks.

LIFEBENCH (Zhang et al., 2025b). This work is conducted with a focus on length constraints, ranging from 16 to 8192 words. It contains 10,800 instructions, in English or Chinese, requesting responses with length satisfying three types of relations (*At Most*, *At Least*, and *Equal To*).

MAXIFE (Liu et al., 2025). This work focuses on multilingual and cross-lingual scenarios. It consists of parallel data in 23 languages, with each record combining a Basic Question and 1–3 Instructions, totaling 18,285 entries.

EIFBENCH (Zou et al., 2025) introduces a new evaluation paradigm—multi-instruction, multi-constraint—to assess the instruction-following capabilities of LLMs, reflecting the multifaceted requirements often encountered in real-world scenarios. The benchmark contains 1,000 instances, each featuring a complex scenario description accompanied by a set of N tasks, each governed by M constraints. For each instance, the model is required to respond to all $N \times M$ instructions.

ORDERED COMMONGEN (Song et al., 2025) introduces a benchmark that evaluates LLMs' ability to generate concepts in a specified order. It contains 27,648 instances from 192 concept sets and 6 instruction templates, using ordered coverage to measure adherence to the specified sequence.

IFBENCH (Pyatkin et al., 2025). This work highlights the limited generalization of LLMs' instruction-following abilities across constraint types, thus falling into comprehensiveness-oriented reliability. They curate a new set of 300 instructions, involving 58 new constraint types beyond the 25 types in **IFEVAL**.

IFSCALE (Jaroslawicz et al., 2025). This work introduces a benchmark for evaluating instruction-following performance as instruction density increases in professional business report generation. Each task contains 10–500 keyword-inclusion instructions drawn from a curated 500-term business vocabulary, with density increasing in steps of 10.

IFEVALCODE (Yang et al., 2025b) introduces a multilingual benchmark for controlled code generation, evaluating LLMs' ability to follow instructions beyond correctness in coding scenarios. It includes 1,600 samples in 8 languages with paired Chinese–English queries, assessing both correctness and instruction adherence.

Among these, several works are relevant to the techniques and the research problem studied in this work. We distinguish ours from several highly relevant ones:

- **CODI-EVAL** (Chen et al., 2024c): This benchmark mainly focuses on the controllability of LLMs in generation tasks. This work advocates an instruction diversification process to synthesize diverse forms of constraint expression (*i.e.*, the rewriting step). They aim to achieve finer-grained coverage of instructions rather than to study the reliability issue. What's more, they fail

to account for inter-prompt relationships.

- **SIFO** (Chen et al., 2024b) & **PBIF** (Zeng et al., 2025): These two works investigate the position bias problem of LLMs in the context of instruction-following. They examine how LLMs' compliance changes when the order of constraints within prompts is altered. This phenomenon can be viewed as a special case of our rephrasing-based augmentation approach.
- **LIFEBENCH** (Zhang et al., 2025b): This work specifically studies length constraints. Their study is analogous to our pilot study in Section 2.1. However, they focus on how long-context LLMs follow length constraints rather than on the general reliability.
- **IFSCALE** (Jaroslawicz et al., 2025): This work studies how many instructions LLMs can simultaneously follow. For the research question, the study also relates to reliability, but it stress-tests LLMs along the complexity-oriented dimension.

A.2 Evaluating LLMs' Reliability

There exist works revealing LLMs' sensitivity to prompt wordings (Sclar et al., 2024). Cao et al. (2024) specifically study the worst performance across semantically equivalent case-level queries. Mizrahi et al. (2024) advocate for multi-prompt LLM evaluation, which shares the same methodology as our curation of cousin prompts. Based on these works, we move beyond studying simply prompt sensitivity issues (the rephrasing augmentation may reflect) and towards more general nuance-oriented reliability.

Yao et al. (2025) introduce a new metric, `pass^k`, for evaluating agentic scenarios. This metric measures reliability across k independent trials. Although it appears similar, `pass^k` is orthogonal to the `reliable@k` metric, which instead evaluates consistency across distinct cousin prompts rather than independent runs. Notably, `pass^k` can be used in conjunction with `reliable@k` to enable a more rigorous assessment of reliability.

B Data Adaption

B.1 Constraint Types

We adopt the 541 test cases from **IFEVAL**. All of them are composed of tasks with several format constraints. The format constraints are categorized into 25 types, which we detail in Table 4. These **simple** constraints lay the foundation for LLMs' solving various more complex requirements. As most LLMs, as reported by the community and verified by our experiments (the *ACC* column in Table 1), have boosted performance in the original **IFEVAL**, it is a necessary next step to understand how solid and reliable their excellent instruction-following abilities are. This motivates the nuance-oriented reliability examination.

B.2 Principles for High-Quality Data

Recall that each test case mainly relates to three key components: the prompt which instructs LLMs, the evaluation function for checking the instruction-following behaviors (specifically, Python code and response formats in the context of **IFEVAL** and **IFEVAL++**), and the evaluation configurations which specify how the evaluation functions should examine the instruction-following behaviors.

We document four principles for high-quality test cases, which are also enforced in our data augmentation in Section 3.1 and implemented in our automated validity checker (see Prompt 5).

- **Clear Instruction Intents**: Each test case must contain an unambiguous and self-contained prompt that clearly specifies the task to be performed and the constraints to be satisfied. The prompt should minimize interpretation uncertainty and avoid overlapping objectives (*e.g.*, mixing summarization with classification). This ensures that the model's output can be attributed to understanding and following the instruction rather than guessing the task's intent.
- **Verifiability through Evaluation Implementations**: A high-quality test case must be verifiable by automated evaluation functions. The expected outputs or behavioral criteria should be formulated so that they can be programmatically validated without requiring human judgment. Since certain evaluation function implementations in **IFEVAL** possess inherent assumptions and inductive biases, the specified constraint requirements must remain within the boundaries of these evaluation functions. Meanwhile, the prompts must articulate the constraint requirements in a way that the downstream evaluation functions can verify them.
- **Alignment with Evaluation Configurations**: The evaluation configuration (*e.g.*, the *relation* and *num_words* for the length constraints) must be compatible with both the prompt and the evaluation function. Misalignments between prompt requirements and evaluation logic can produce

false positives or negatives, undermining the reliability of the benchmark. Thus, each test case should be explicitly validated against the intended constraint requirements (*i.e.*, the evaluation configurations) to ensure consistency.

- **Attainability**: A test case should be solvable by a capable model based solely on the given prompt. Consider a scenario where two length constraints coexist within a single test case: one requiring more than ten sentences and another requiring fewer than eight words. Such constraints are inherently incompatible, rendering the test case infeasible and therefore subject to removal. We observe that **IFEVAL** includes several test cases exhibiting these attainability issues. Moreover, since our data augmentation strategies—namely, adding distractors and altering constraints—may occasionally combine multiple constraints, we design and implement our validity checker to carefully filter out any combinations that result in unattainable or contradictory requirements.

B.3 Revising Infeasible Test Cases

Manual review and community feedback reveal that a small number of test cases in **IFEVAL** are flawed. We make necessary yet minimal revisions to ensure test case integrity. The overall information and examples are listed in Table 5. We identify that common issues in **IFEVAL** can be categorized into the following four classes.

Flawed Implementation of Evaluation Functions. A representative issue arises from mismatches between prompts and the underlying automatic checkers. For example, one test case requires the "#" character to appear at least four times. However, the original evaluation function only supports letters a-z, making it impossible to evaluate such symbols. As a result, this prompt is invalid for the evaluation function, despite being well-formed. We updated the evaluation function to accept any character—including symbols and digits—so that constraints targeting non-alphabetic characters can be correctly evaluated, ensuring the evaluation function aligns with the intended instructions.

Flawed Documentation of Evaluation Configurations. In some instances, the natural language prompt directly contradicts the evaluation arguments. For example, one prompt requires "the letter o to appear at least 6 times", but the evaluation argument specifies "less than 6 times. Such inconsistencies lead to an unavoidable conflict between what the prompt demanded and how correctness is assessed. We corrected these cases by rewriting the prompt to match the evaluation arguments.

Typos in Natural Language Prompts. Some test cases contain minor typographical errors that make them misleading. For instance, one prompt instructs to separate two tables using "exactly 6 asterisk symbols: *******", where the example actually has 7 asterisks, contradicting the requirement. This was revised to six asterisks "******" to match the stated constraint, thereby eliminating ambiguity introduced by typographical mistakes.

Unachievable Multi-Constraint Test Cases. Some prompts impose multiple constraints that are logically incompatible. A notable example requires the model first to repeat a sentence verbatim (including commas) and then simultaneously adhere to the instruction "avoid using commas". Since the two conditions could not be satisfied simultaneously, such prompts were fundamentally unachievable. We resolved this by removing the contradictory requirement, keeping the prompt feasible.

C Model Details

In this work, we include a total of 36 large language models to establish a representative benchmark. Our selection aims to comprehensively capture the development trends of LLMs in terms of reliability. To this end, we include models that are widely recognized for their performance or influence within the research and developer communities, covering a diverse range of providers, architectures, and scales. For those proprietary models, we access their LLM services via API, specifically the OpenAI-compatible interface. For the open-source ones, we locally deploy them using vLLM (Kwon et al., 2023) for inference. We use the instruction-tuned or chat version and adopt official chat templates. See Table 6 for the detailed information.

D More Experimental Results

D.1 Impact of the System Prompt

Certain LLMs are published with default system prompts. For instance, models in the `Qwen2.5` series include default prompts in their chat templates. In contrast, more recent reasoning models, such as `DeepSeek-R1` and `Qwen3`, do not specify system prompts; their providers even recommend using an empty prompt. To elicit the default (and arguably optimal) behaviors while avoiding evaluation bias,

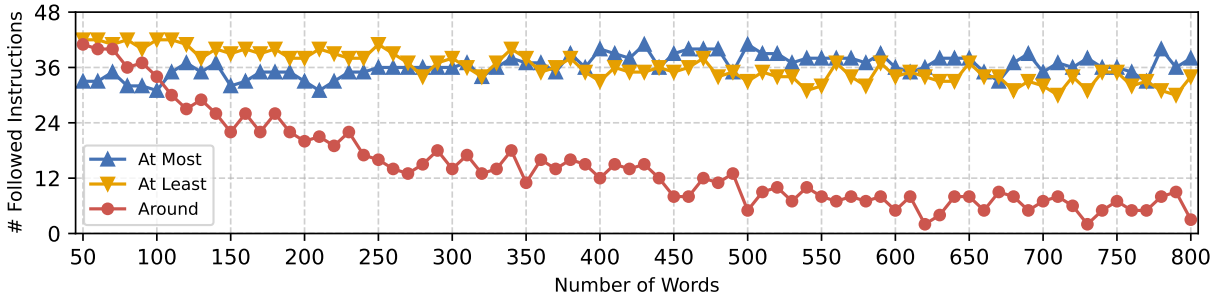

Figure 8: **Results of Requesting Varying Number of Response Words**. Experiments with `Qwen3-32B`.

our main experiments adopt the system prompt settings recommended by model providers.

Meanwhile, we notice that it is a common practice for service providers or end users to employ system prompts tailored for personalized LLM services (Ge et al., 2024; Qin et al., 2025). We are also curious about the impact of system prompt setting on the models' reliability in instruction-following. We include six settings as listed in Table 7: the default, three persons sampled from Ge et al. (2024), python coder, and encouraging better instruction-following. Except for the system prompt setting, other decoding strategies remain consistent.

The results, illustrated in Figure 10, show that system prompts have an implicit but notable impact on performance in **IFEVAL++**. Empirically, the default system prompt tends to yield the best instruction-following performance, confirming the representativeness of our choice in the main experiments. In contrast, personalization may harm reliability: certain persona prompts reduce `reliable@10` by up to 8.2%. Moreover, explicitly encouraging better instruction-following (the *follow-instructions* prompt) does not improve performance. Importantly, the same prompt may affect models differently—the *persona3* prompt significantly degrades the reliability of `Qwen3-8B`, but has little or no impact on `LLaMA-3.3-70B`.

This initial empirical measurement substantiates the validity of our choice of using the default system prompt. However, the mechanisms by which the setting of system prompts implicitly influence downstream tasks such as **IFEVAL++** remain poorly understood, highlighting an important direction for future research.

D.2 Scalability of the `reliable@k` Metric

The `reliable@k` metric is inherently configurable, depending on the choice of k and the underlying cousin prompts. By increasing k, we can construct more challenging test cases, as the LLM must handle a larger set of cousin prompts simultaneously

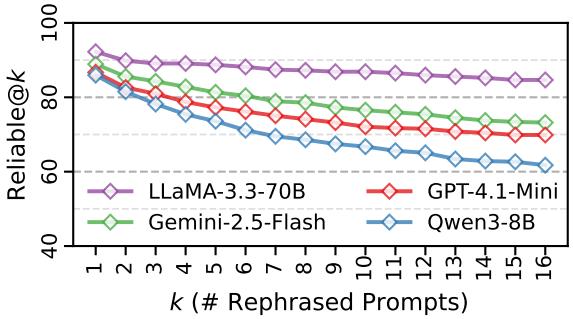

Figure 9: **The `reliable@k` Metric is Highly Scalable**.

to achieve a score.

As illustrated in Figure 9, simple augmentation via rephrasing (strictly following the automated validity checker described in Section 3.2) can steadily raise the difficulty up to approximately $k = 16$. Two key observations emerge: 1) Increasing k can better differentiate the reliability of different LLMs. For instance, `Qwen3-8B` and `GPT-4.1-mini` exhibit similar reliability when k is small, but as k grows, `GPT-4.1-mini` demonstrates higher reliability than `Qwen3-8B`. 2) This highlights a promising pathway for scalable oversight. Employing more sophisticated augmentation methods, such as introducing distractors, can further enhance the scalability of the evaluation.

An additional implicit advantage of this approach, particularly for nuance-oriented evaluation, is data efficiency: test cases can be augmented directly within existing benchmarks, following the recipe established in this work, without requiring large-scale new data collection.

D.3 Pilot Experiments on `Qwen3-32B`

We extend our pilot experiments to `Qwen3-32B`. As shown in Figure 8, the instability persists in the larger-scale LLM. Overall, for the more challenging *Around* relation, the LLM is far less reliable, which is aligned with the findings of Zhang et al. (2025b). The instability reveals the vulnerability of LLMs in nuance-oriented reliability, motivating us to study this critical yet underexplored dimension.

Table 4: **The List of 25 Code-Verifiable Constraints.**

Instruction Group	Instruction	Placeholders	Example
Keywords	Include Keywords	keywords	Write a blog post about how to train a dog that is geared towards kids. Include the keywords "{finale}" and "{less´ in the post.
Keywords	Keyword Frequency	relation, keyword, frequency	Write a fairy tale about a princess and a dragon, making sure the word "{replied}" appears {at least} {twice}.
Keywords	Forbidden Words	forbidden_words	Can you write a rap that doesn't include the keywords "{Yo}", "{check}", and "{peace}"?
Keywords	Letter Frequency	let_relation, letter, let_frequency	Make a tweet for playboy's twitter account without using capital letters. Include {at least} {4} hashtags, starting with "{#}"
Language	Response Language	language	what is the difference between a levee and an embankment? Please respond to me only in {Korean}.
Length Constraints	Number Paragraphs	num_paragraphs	Write a very angry letter to someone who's been trying to convince you that 1+1=3. There should be exactly {4} paragraphs. Separate the paragraphs with .
Length Constraints	Number Words	relation, num_words	Write a blog post about the best way to get a good night's sleep with {at least} {400} words.
Length Constraints	Number Sentences	relation, num_sentences	Write a template with {less than} {7} sentences for how to calculate the offset of an element in an array.
Length Constraints	Number Paragraphs + First Word in i-th Paragraph	first_word, num_paragraphs, nth_paragraph	How are you doing today? Could you write me exactly 4 paragraphs each separated by two new lines? Please start the first paragraph with the word "firm".
Detectable Content	Postscript	postscript_marker	Write me a template for a product description in the form of a poem and end it with a postscript starting with {P.P.S}
Detectable Content	Number Placeholder	num_placeholders	Draft a blog post about Ripley's Aquarium. Make sure your blog post contains at least {7} placeholders represented by square brackets, such as [location].
Detectable Format	Number Bullets	num_bullets	Make a rubric for a home theater installation targeting moms. Your answer must contain exactly {4} bullet points. Use markdown bullet points such as: * This is point 1
Detectable Format	Title	–	Please write a riddle about the inverse function with a title wrapped in double angular brackets, i.e. «title».
Detectable Format	Multiple Sections	section_spliter, num_sections	Write a {4} section resume for professional clown Phil Larkin. Each section should be explicitly noted as {Section} X.
Detectable Format	Choose From	–	Would you consider yourself to be smart? Choose from: My answer is yes. My answer is no. My answer is maybe. Just choose one phrase from above as your answer.
Detectable Format	JSON Format	–	What are the advantages and disadvantages of having supernatural powers? Make it short. Wrap the entire output in JSON format. You can use markdown ticks such as ` ` ` .
Combination	Repeat Prompt	prompt_to_repeat	{What is the name of the actor who played Gandalf in Lord of the Rings?} First repeat the question above without change of words, then give your answer.
Combination	Two Responses	–	What is a name that people call God? Please give exactly two different responses. Separate the responses with 6 asterisk symbols: ******.
Change Cases	All Uppercase	–	Create a riddle about the name Sheldon using only 10 words. Make sure to only use capital letters in your entire response.
Change Cases	All Lowercase	–	What is another word for Engravings? Answer in lowercase letters only, throughout your entire answer.
Change Cases	Frequency of All capital Words	capital_relation, capi-tal_frequency	Write a serious riddle about trips and stitches in a poem style that includes at least 15 words in all capital letters.
Start with / End with	End Checker	end_phrase	Write a poem about two people who meet in a coffee shop and end your entire response with the exact phrase "{Is there anything else I can help with?}"
Start with / End with	Quotation	–	Write a review of IBM's 1956 chess program. Make sure your entire response is wrapped in double quotation marks.
Punctuation	No Commas	–	Write a limerick about writing a limerick. Don't use any commas in your entire reply.

Table 5: **Examples of Revised Infeasible Test Cases from the IFEVAL Benchmark**. We identify four major categories of issues: (1) flawed implementation of constraint checkers, (2) flawed implementation of constraint arguments, (3) typos in prompts, and (4) unachievable multi-constraint cases.

Flaw Type	Original Prompt	Revised Prompt	Reason for Revision	Involved Cases
Prompt–checker misalignment	Write a 2 paragraph critique of the following sentence in all capital letters, no lowercase letters allowed: "If the law is bad, you should not follow it". Label each paragraph with PARAGRAPH X.	Write a 2 paragraph critique in English of the following sentence in all capital letters, no lowercase letters allowed: "If the law is bad, you should not follow i". Label each paragraph with PARAGRAPH X.	For the **english_capital** and **english_lowercase** constraint, the original test cases did not explicitly specify the use of English. We revised the prompts to align with the checker's evaluation criteria.	1021, 1051, 1122, 1132, 1153, 1287, 1516, 1535, 1571, 1593, 1645, 1779, 1813, 1999, 202, 2100, 2391, 2531, 2563, 2943, 296, 3069, 3380, 3434, 3703, 3617, 3456, 3186, 3276
Prompt–checker misalignment	Write two jokes about rockets. Do not contain commas in your response. Separate the two jokes with 6 asterisk symbols: ******.	Write two different jokes about rockets. Do not contain commas in your response. Separate the two jokes with 6 asterisk symbols: ******.	For the **two_responses** constraint, prompts were revised to explicitly require distinct responses, aligning with the checker's evaluation.	1107, 1582, 1591, 3287
Prompt–checker misalignment	Explain the difference between a city and a village in a rap style to a kid. The words with all capital letters should appear at least 10 times. Put the response into at least 5 sections, separated using 3 asterisks ***.	Explain the difference between a city and a village in a rap style to a kid. The words with all capital letters should appear at least 10 times. Put the response into exactly 5 sections, separated using 3 asterisks ***.	For the **number_paragraphs** constraint, the evaluation requires exactly n paragraphs, whereas the original natural language prompt only specifies "at least n," creating a mismatch.	2180
Prompt–checker misalignment	Tell a joke that has the words thursday and amalgamation in it, but use Swahili language only, no other language is allowed.	Tell a joke that has the words thursday and amalgamation in it, the response should be primarily in Swahili, though you may include a few words from other languages if needed.	Revised to relax the **language** requirement: the response should be primarily in the specific language, allowing a few words from other languages if necessary (e.g., English keywords). This ensures the prompt is feasible while preserving the intended linguistic constraint.	2309, 3063, 3311
Prompt–arguments contradiction	... Use the letter o as a keyword in the syntax of the template. The letter o should appear at least 6 times....	...Use the letter o as a keyword in the syntax of the template. The letter o should appear less than 6 times. ... (Evaluation augments: {"let_relation": "less than", "letter": "o", "let_frequency": 6})	Revised to resolve **contradictions** between the natural language prompt and evaluation augments.	1174, 1217, 1964, 3369, 337
Prompt typos	Create a table with a 7 day trip itinerary for India, and a 7 day trip itinerary for China. Separate them with exactly 6 asterisks symbols: *******	Create a table with a 7 day trip itinerary for India, and a 7 day trip itinerary for China. Separate them with exactly 6 asterisks symbols: ******	Revised to correct **typos** in the natural language prompt.	1332
Incompatible constraints	**Write a plot for a story about two people who swap fingerprints. Include a title wrapped in double angular brackets, i.e. «title». In your response please avoid using commas.** First, repeat the request above word for word without change. Do not say any words or characters before repeating the request above. After you repeated the request, you can give your response next.	**Write a plot for a story about two people who swap fingerprints. Include a title wrapped in double angular brackets, i.e. «title».** First, repeat the request above word for word without change. Do not say any words or characters before repeating the request above. After you repeated the request, you can give your response next.	The original prompt imposes **multiple constraints that are mutually incompatible**, making it unachievable. For instance, it requires repeating a specific text fragment containing commas while simultaneously prohibiting the use of commas in the response.	1627, 2216, 3305, 3371, 3718, 374

Table 6: **Model Details**.

Model	Vendor	Release Date	# Params	Context Size	Knowledge Cutoff
Proprietary LLMs (API-Based)					
GPT-5	OpenAI	2025-08	?	400k	2024-09
GPT-5-mini	OpenAI	2025-08	?	400k	2024-05
GPT-5-nano	OpenAI	2025-08	?	400k	2024-09
GPT-5-chat-latest	OpenAI	2025-08	?	125K	2024-09
Gemini-2.5-Pro	Google	2025-06	?	1M	2025-01
Gemini-2.5-Flash	Google	2025-06	?	1M	2025-01
Gemini-2.5-Flash-Lite	Google	2025-06	?	1M	2025-01
o4-mini	OpenAI	2025-04	?	?	2024-06
GPT-4.1	OpenAI	2025-04	?	1M	2024-06
GPT-4.1-mini	OpenAI	2025-04	?	1M	2024-06
GPT-4.1-nano	OpenAI	2025-04	?	1M	2024-06
o3	OpenAI	2025-04	?	195K	2024-06
o3-mini	OpenAI	2025-01	?	195K	2023-10
GPT-4o-2024-11-20	OpenAI	2024-11	?	125K	2023-10
GPT-4o	OpenAI	2024-11	?	125K	2023-10
GPT-40-2024-05-13	OpenAI	2024-11	?	125K	2023-10
o1-mini	OpenAI	2024-09	?	125K	2023-10
GPT-4o-mini	OpenAI	2024-07	?	125K	2023-10
GPT-3.5-turbo-0125	OpenAI	2024-01	?	16K	2021-09
GPT-3.5-turbo-1106	OpenAI	2023-11	?	16K	2021-09
Open-Source LLMs					
Qwen3-Next-80B-A3B-TK	Alibaba	2025-09	80B (A3B)	262K	?
Kimi-K2-0905	Moonshot	2025-09	1T(A32B)	256K	2025-3
Qwen3-Next-80B-A3B-Instruct	Alibaba	2025-09	80B (A3B)	256K	?
GPT-oss-120b	OpenAI	2025-08	120B (A5.1B)	128K	2024-06
GPT-oss-20b	OpenAI	2025-08	20B (A3.6B)	128K	2024-06
GLM-4.5	Z.AI	2025-07	355B (A32B)	128K	?
Qwen3-30B-A3B	Alibaba	2025-07	30B (A3B)	256K	?
GLM-4.5-Air	Z.AI	2025-07	106B	128K	?
Qwen3-235B-A22B	Alibaba	2025-07	235B (A22B)	256K	?
Qwen3-14B	Alibaba	2025-05	14.8B	32K	?
Qwen3-8B	Alibaba	2025-05	8B	128K	?
Qwen3-32B	Alibaba	2025-05	32B	128K	?
Qwen3-4B	Alibaba	2025-04	4B	32K	?
Qwen3-1.7B	Alibaba	2025-04	2B	32k	?
Qwen3-0.6B	Alibaba	2025-04	0.8B	32k	?
Gemma-3-IT-27B	Google	2025-03	27B	128K	2024-08
Gemma-3-IT-12B	Google	2025-03	12B	128K	2024-08
DeepSeek-V3.1	DeepSeek	2025-01	671B (A37B)	128K	2025-07
DeepSeek-R1	DeepSeek	2025-01	671B (A37B)	128K	2024-12
LLaMA-3.3-70B-IT	Meta	2024-12	70B	128K	2023-12
Qwen2.5-72B-Instruct	Alibaba	2024-09	72B	131k	?
Qwen2.5-32B-Instruct	Alibaba	2024-09	32B	128K	?
Qwen2.5-14B-Instruct	Alibaba	2024-09	14.7B	128K	?
Qwen2.5-7B-Instruct	Alibaba	2024-09	7B	32K	?
LLaMA-3.1-70B-Instruct	Meta	2024-07	70B	128K	2023-12
LLaMA-3.1-8B-Instruct	Meta	2024-07	8B	128K	2023-12

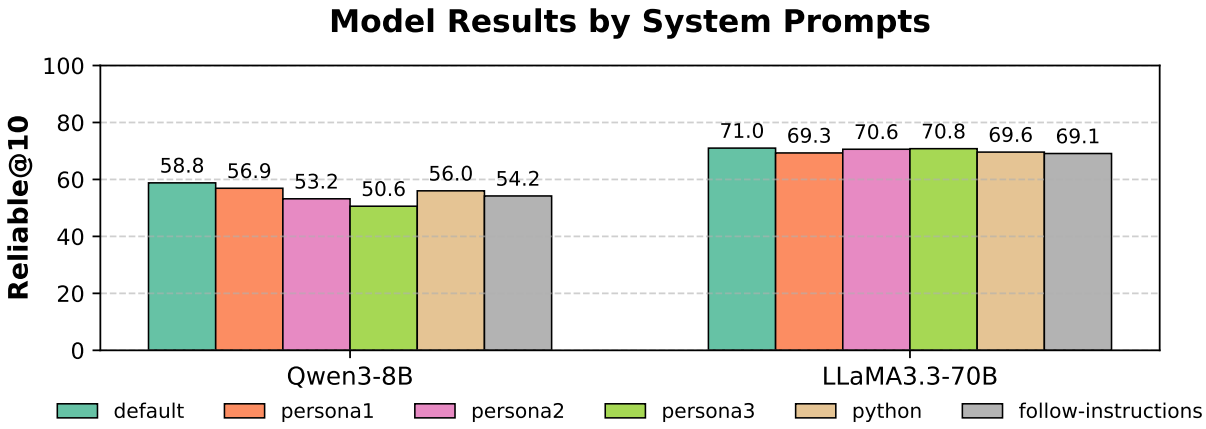

Figure 10: **Impact of System Prompts on Nuance-Oriented Reliability**.

E Training-Based Methods

E.1 Training Details

In this section, we detail the training recipe for the experiments in Section 5.2. Recall that we resort to supervised fine-tuning as the training method, which teaches the LLMs to mirror the expected responses. This is supervised by a cross-entropy loss on the response tokens. Formally, let x denote the input prompt (after applying the chat template) and $y = (y_1, \ldots, y_T)$ the corresponding target response consisting of T tokens. The model parameterized by θ defines a conditional distribution $p_\theta(y_t \mid x, y_{<t})$ over the next token given the input and previously generated tokens. The supervised fine-tuning objective minimizes the negative log-likelihood of the ground-truth responses, *i.e.*,

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}(x, y) \sim \mathcal{D} \sum_{t=1}^{T} \log p_\theta(y_t \mid x, y_{<t}), \tag{1}$$

where $\mathcal{D}$ denotes the training dataset. This objective is equivalent to minimizing the token-level cross-entropy loss between the predicted distribution and the true response sequence. During optimization, we employ teacher forcing, feeding the gold tokens $y_{<t}$ at each step, to stabilize training and accelerate convergence.

We train the `Qwen2.5-7B-Instruct` using the Swift framework (Zhao et al., 2025). We use LoRA (Hu et al., 2022) for parameter-efficient fine-tuning. We introduce adapters, with rank $r = 16$ and coefficient $\alpha = 32$, to all linear modules. Model parameters are updated using the AdamW optimizer (Loshchilov and Hutter, 2019). The peak learning rate is 0.00005, and the global batch size is 64. To ensure training stability, we set a warmup ratio of 0.05, cosine learning rate decay, and gradient clipping of 1.0. We save checkpoints every 20 steps throughout the training process.

We include two dataset settings in the SFT experiments: (1) general-domain instruction-following, for which we use the representative alpaca (Taori et al., 2023) as the data source; (2) self-curated datasets tailored for improving reliability, for which we employ another pool of cousin prompts for the **IFEVAL**. Across the two settings, the system prompt is consistently set to the default one for both training and evaluation.

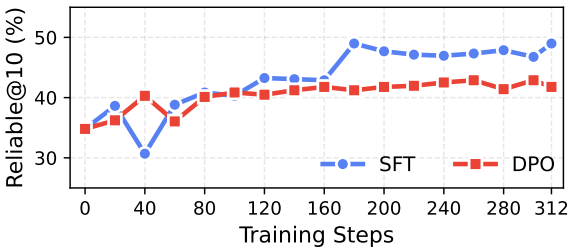

Figure 11: **Training Methods Affects Reliability Performance on IFEVAL++ in Varying Fashion**.

We further elaborate on the curation process for the *Cousin Prompts* training set, following the same augmentation-then-filtering paradigm introduced in Section 3. For the rephrasing augmentation, we reuse the 16 replicates of cousin prompts described in Appendix D.2. For the other two augmentation methods, each original prompt is expanded into five additional replicates. After augmentation, we first remove any prompts that exactly match those in **IFEVAL++**, and then apply the automated validity check. To maintain balance across the three augmentation methods, we randomly down-sample the cousin prompts belonging to each augmentation method to 2,400 examples.

We then employ the `LLaMA-3.3-70B-Instruct`

Table 7: **System Prompts Used in Appendix D.1**.

Persona 1	**Persona 3**	**Persona 2**
You are a helpful assistant serving the following persona, and you should adjust your response to the persona. A person who is interested in European Union law, particularly the concept of EU citizenship and its rights. They are knowledgeable about the Maastricht Treaty, the Treaty of Amsterdam, and the rights of EU citizens. They are also interested in the concept of 'union citizenship' and how it complements national citizenship. They may have a background in law or politics and are interested in understanding how EU laws are implemented and enforced.	You are a helpful assistant serving the following persona, and you should adjust your response to the persona. A finance manager who is interested in understanding the different inventory accounting methods used in business, particularly the advantages and disadvantages of each method. They are looking for a detailed explanation of the cost of goods sold and how it affects inventory physical flow and value. They are also interested in learning about the specific examples given in the text and how they can apply these concepts in their own business.	You are a helpful assistant serving the following persona, and you should adjust your response to the persona. A conservation biologist who specializes in the study of aquatic ecosystems, specifically focusing on the restoration of migratory fish habitats and the monitoring of aquatic species. They have extensive experience working with the Virginia Department of Game and Inland Fisheries (DGIF) and have a deep understanding of the complexities of dam removal and its impact on aquatic habitats. They have a passion for conservation and have dedicated their career to protecting and restoring natural ecosystems. They have a keen eye for detail and are skilled at analyzing data to identify patterns and trends in the ecosystem. They have a strong work ethic and are committed to making a positive impact on the environment.
Python	**Following-Instructions**	
You are a highly skilled and experienced software engineer specializing in Python programming.	You are a highly skilled and experienced assistant in following the user's instructions.	

Table 8: **Composition of the Dataset Used in Section 5.2 for the *Cousin Prompts* Case**.

Augmentation	Count
Rephrasing	2,303
Distractor Addition	2,274
C/T Reconfig	2,046

as the teacher model, and run sampling 16 times with a temperature of 1.0. Then, we account for only those prompts with at least one response that can pass the evaluation. The resulting dataset composition is presented in Table 8. We run SFT on this dataset for three epochs, totaling 312 training steps. For the *Alpaca* dataset, we apply the same configuration, training on a randomly sampled subset of the complete dataset. This setup enables us to isolate the effect of the training data itself, independent of other confounding factors, on reliability.

E.2 Training Methods: SFT vs. DPO

Additionally, we advance an exploratory attempt around distinct training methods. We include the direct preference optimization (DPO) algorithm (Rafailov et al., 2024), an alignment method that directly optimizes a model's parameters toward preferred responses without requiring explicit reward modeling. Unlike reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022), which relies on a separately trained reward model and policy optimization, DPO leverages pairwise preference data. This encourages the model to assign higher relative likelihood to preferred responses compared to rejected ones, enabling alignment without explicit reward modeling. The experiments are conducted on the same *Cousin Prompts* dataset. For DPO, the rejected samples are collected from `LLaMA-3.3-70B-Instruct`. When multiple rejected responses are available, we randomly select from those that fail to meet the evaluation criteria. If none explicitly fail, we instead choose the response with the largest character count, as it intuitively represents the least efficient way of satisfying the requirements. All other training configurations are shared across the two training methods, including the learning rate.

As illustrated in Figure 11, DPO in this setting does not further boost the reliability and even underperforms compared to vanilla SFT. We hypothesize that this phenomenon arises from the construction of the rejected responses. Specifically, the current setup may fail to capture the nuanced discrepancy that the policy model should learn to minimize between preferred and rejected outputs. In the context of instruction-following, a rejected response may still reflect partial progress toward fulfilling the given constraints. Penalizing such outputs too strongly could inadvertently steer the model away from learning the correct alignment direction.

We leave this non-trivial exploration of training methods in the Appendix, as we cannot draw conclusions about best practices. Future works may extend this investigation to other training paradigms, *e.g.*, reinforcement learning from verifiable rewards (Lambert et al., 2024; Guo et al., 2025; Zou et al., 2025).

Prompt 1: Task Reconfiguration

You are a helpful assistant that revise the prompt to test an LLMs' generalization ability.

The prompt typically contain one task and a few constraints. The only part that should be revised is the task description.

You are free to revise the task description but keep the new task compatible with the original constraints. Typically, the revised task should be more challenging than the original task.

You should make the minimal changes to the constraints unless absolutely necessary, such that the new prompt is still able to request the same set of constraints.

Both the responses to the original prompt and the new prompt will be evaluated by the following function:
{evaluation_function_implementation}

And the necessary arguments for the evaluation function (i.e., the requirements to be satisfied by the response) are:
{evaluation_function_arguments}

The original prompt is:
{prompt}

You should directly output the new prompt, without any other text.

Prompt 2: Rephrasing the Prompts

You are a helpful assistant whose task is to rephrase a given prompt into another prompt.

You will be provided with:
1. A complete natural language prompt consisting of a task and one or more constraints.
2. A set of evaluation functions that will be used to evaluate responses to the prompt.
3. The arguments for the evaluation functions.

Your task
Rewrite the given prompt while following these rules:
- Treat the input as a single, complete prompt. Keep all constraints intact, but rephrase them naturally.
- Do not alter numbers, thresholds, links, counts, or formatting requirements.
- Do not add new constraints or information.
- Preserve explicit structural rules (e.g., "repeat exactly", "use this format", "separate items with ******").
- Always output the rewritten prompt in English, even if the original is in another language.
- If a constraint specifies a casing requirement (e.g., ALL CAPS, all lowercase), rephrase it using normal sentence casing.
- For constraints in the **repeat_prompt** category, do not modify the sentence to be repeated (the one provided in the arguments).

—

Input
Original prompt:
{prompt}

Evaluation functions:
{evaluation_function_implementation}

Arguments:
{evaluation_function_arguments}

—

Output
Directly produce the rewritten prompt, without any additional text.

Prompt 3: Add a Distractor Constraint

You are a helpful assistant tasked with adding one additional distractor constraint to the original prompt.

Instructions:
- Preserve all contents of the original prompt exactly as they are.
- At the end of the new prompt, append one extra constraint sentence.
- This extra constraint must introduce an additional **format requirement** for the response.
- The new constraint must not interfere with or alter the original constraints.

Original Constraints:
{constraints}

Evaluation Functions:
{evaluation_functions}

Original Prompt:
{prompt}

Prompt 4: Constraint Reconfiguration

You are a helpful assistant that revise the prompt to test an LLMs' generalization ability.

The prompt typically contain one task and a few constraints. You are allowed to revise the constraints and correspondingly adjust the evaluation arguments. If necessary, you can also revise the task description.

If multiple constraints exist, you should revise at least one of them.

Typically, the revised constraints should be **more challenging** than the original constraints, but not beyond the **allowable range** of the evaluation functions.

Keep using the original constraint types of the original prompt and the original evaluation functions. Only change the fill-in values (arguments) of the constraints.

To help you better understand the constraints, here are the evaluation functions that will be used to evaluate the responses to the original prompt and the new prompt:
{evaluation_function_implementations}

The original prompt is:
{prompt}

The original instruction id list is:
{instruction_id_list}

The evaluation arguments for the original prompt are:
{evaluation_function_arguments}

You should output a list of JSON objects with the following fields:
- "random_seed": The random seed you choose to generate the new prompt.
- "reasoning": The reasoning process for generating the new prompt and the new kwargs.
- "revised_prompt": The new prompt.
- "revised_instruction_id_list": The new instruction id list. If you have added or removed some constraints, you should also update the instruction id list. The length of the list should be the same as that of the revised kwargs.
- "revised_kwargs": The new kwargs in string format, which should be valid JSON. The format should be the same as the original kwargs as follows:
{evaluation_function_arguments_format}

The total number of the revised prompts should be {num_prompts}. The revised prompts should be **distinct** in constraints, **reasonable**, and increasingly difficult.

Keep in mind that do not use constraints beyond the provided evaluation functions.

Prompt 5: Checking the Integrity of Test Cases

You are tasked with verifying whether a given test case for evaluating LLMs' instruction-following behavior is valid.

Each test case consists of:
- **Prompt**: The instruction provided to the LLM.
- **Evaluation Arguments**: The specific parameters required by the evaluation function.
- **Evaluation Function**: The function that checks the LLM's response against the evaluation arguments and returns a boolean judgment.

Your task is to determine if the test case is valid by checking the following conditions:

A test case is valid only if it satisfies all three conditions below:

1. **Argument Reflection**
Every evaluation argument must be explicitly required or constrained in the Prompt.
If the Prompt does not impose a constraint corresponding to a given evaluation argument, the test case is invalid.
You should understand how the evaluation function works and the constraints (evaluation arguments) to be evaluated.

2. **Format Compatibility**
The Prompt's format and placeholder conventions must allow a response that can pass the evaluation function.
- For arguments of type 'keywords', 'keyword', 'forbidden_words', or 'prompt_to_repeat', the Prompt must clearly require or forbid these elements in such a way that the evaluation function can check them.
- Be careful with the punctuation marks.

3. **Logical Consistency**
Logically, the context in the Prompt (e.g., the tasks or other constraints that are not evaluated) must not contradict the constraints to be evaluated.
All constraints are independent of each other and should be met at the same time. Do not interpret one constraint conditioned on another.
For example, requiring JSON-formatted output but additionally asking for a plain-text line is invalid.

Notes:
- Your judgement should be harsh and rigorous. You should not be lenient or allow for any exceptions. Default to "is_valid": false.
- A stricter or looser Prompt constraint is **not** equivalent to an Evaluation Argument.
- The test case is valid only if **all three conditions** above are satisfied.

Output:
Return your judgment in the following JSON format:

```json
{{
    "reasoning": "Your reasoning process",
    "is_valid": true/false
}}
```

—

Prompt:
{prompt}

Evaluation Arguments:
{evaluation_args}

Evaluation Function:
{evaluation_function}