

When Instructions Multiply: Measuring and Estimating LLM Capabilities of Multiple Instructions Following

Keno Harada¹, Yudai Yamazaki^{2*}, Masachika Taniguchi^{3*}, Edison Marrese-Taylor¹, Takeshi Kojima¹, Yusuke Iwasawa¹, Yutaka Matsuo¹

¹The University of Tokyo, ²Kyoto University, ³University of the Ryukyus

{keno.harada, emarrese, t.kojima, iwasawa, matsuo}@weblab.t.u-tokyo.ac.jp

yamazaki.yudai.82m@st.kyoto-u.ac.jp, k248443@eve.u-ryukyu.ac.jp

Abstract

As large language models (LLMs) are increasingly applied to real-world scenarios, it becomes crucial to understand their ability to follow multiple instructions simultaneously. To systematically evaluate these capabilities, we introduce two specialized benchmarks for fundamental domains where multiple instructions following is important: Many Instruction-Following Eval (ManyIFEval) for text generation with up to ten instructions, and Style-aware Mostly Basic Programming Problems (StyleMBPP) for code generation with up to six instructions. Our experiments with the created benchmarks across ten LLMs reveal that performance consistently degrades as the number of instructions increases. Furthermore, given the fact that evaluating all the possible combinations of multiple instructions is computationally impractical in actual use cases, we developed three types of regression models that can estimate performance on both unseen instruction combinations and different numbers of instructions which are not used during training. We demonstrate that a logistic regression model using instruction count as an explanatory variable can predict performance of following multiple instructions with approximately 10% error, even for unseen instruction combinations. We show that relatively modest sample sizes (500 for ManyIFEval and 300 for StyleMBPP) are sufficient for performance estimation, enabling efficient evaluation of LLMs under various instruction combinations.¹

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across various tasks (Brown et al., 2020; OpenAI, 2024a; Google, 2023; Llama Team, 2024; Gemma Team, 2024;

* Equal contribution

¹ManyIFEval/StyleMBPP data and evaluation codes are available at <https://github.com/kenoharada/Multiple-Instructions-Following>

Anthropic, 2024; Gemini Team, 2024). However, their practical utility depends on their ability to simultaneously follow multiple instructions. Simple task descriptions such as “Write a piece of code” or “Summarize this text” typically produce outputs with limited practical value. Adding specific instructions transforms these basic prompts into practically useful tools - for instance, code generation with style guide compliance enables effective team collaboration, while text summarization with formatting requirements (e.g., bullet points, length limits) ensures human-readable outputs. These kinds of instructions are ubiquitous in real-world applications, and models must be able to follow multiple instructions to be useful in practice.

While some existing benchmarks have made valuable contributions to evaluate multiple-instructions-following capabilities for real-world applications, there are several limitations towards accurate and fair evaluation. For instance, IFEval (Zhou et al., 2023) and ComplexBench (Wen et al., 2024) assign a different task description across different number of instructions, which prevents from isolating the effect of instruction counts as a controlled variable. FollowBench (Jiang et al., 2024) utilized LLM-as-a-judge for measuring multiple-instructions-following performance, which prevents from reliable and stable verification compared to rule-based and programmable one. Without controlled experimental design, it becomes difficult to systematically and precisely analyze how the number of instructions impacts model performance.

Our work focuses on measuring the simultaneous multiple-instructions-following performance, which we consider crucial in real-world applications. Specifically, we introduce two specialized benchmarks: ManyIFEval for text generation (up to 10 instructions) extended from IFEval (Zhou et al., 2023) and StyleMBPP for code generation (up to 6 instructions) extended from MBPP (Austin et al.,

2021). Our benchmarks maintain consistent task descriptions while varying the number of instructions, allowing us to isolate and analyze the effect of instruction counts on model performance (Figure 1 and Table 1). Our benchmarks also can objectively measure multiple-instructions-following ability of LLMs with a programmatically verifiable approach (Table 1). In addition, our benchmarks have balanced and sufficient sample sizes for each instruction count (Figure 2).

Based on the created benchmarks, we conducted a broad range of experiments to evaluate the multiple-instructions-following performance with a wide variety of LLMs. Specifically, our experiments used ten state-of-the-art LLMs including API based models (GPT-4o, Claude3.5 Sonnet, Gemini 1.5 Pro, o3-mini) and open models (Llama3.1-8B, Gemma2-9B, Gemma2-2B, Qwen-2.5 72B, DeepSeek-V3, DeepSeek-R1), revealing that performance consistently and drastically degrades as the number of instructions increases.

Furthermore, given the fact that evaluating all the possible combinations of multiple instructions is computationally impractical in actual use cases, we developed regression models that can estimate performance on both unseen instruction combinations and different numbers of instructions which are not used during training. Specifically, we explore three modeling approaches: naive estimators, beta-binomial, and logistic regression. Through this estimation, we demonstrate that a logistic regression model using instruction count as an explanatory variable can predict performance with approximately 10% error, even for unseen instruction combinations. The approach generalizes effectively to unseen numbers of instructions, achieving a mean absolute error of 0.03 ± 0.04 when predicting performance on 10 instructions using training data from up to 9 instructions. Our experiments reveal that relatively modest sample sizes are sufficient for accurate performance estimation - 500 samples for ManyIFEval and 300 for StyleMBPP. This suggests that our benchmarks can provide reliable performance estimates for LLMs under various instruction combinations, enabling efficient evaluation and guiding future model development.

2 Related Work

2.1 Instruction-Following Benchmarks

Instruction-following capability is a crucial aspect of large language models (LLMs) as it influences

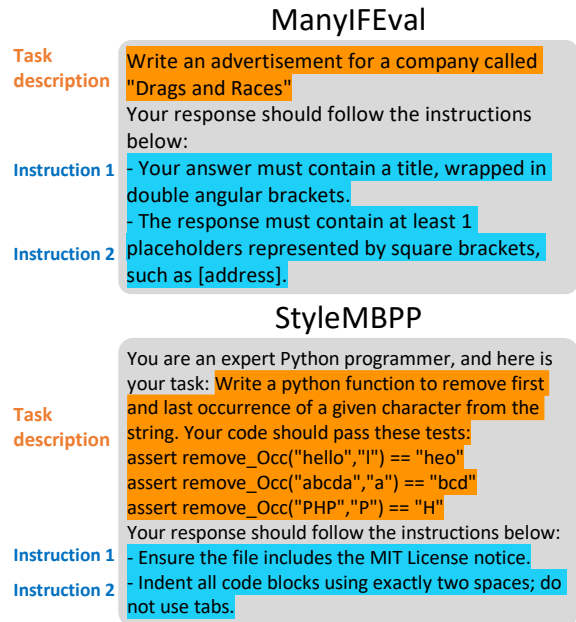


Figure 1: Examples from our proposed ManyIFEval (top) and StyleMBPP (bottom) benchmarks. Task description defines the primary objective (e.g., text generation, code creation), while instructions specify additional constraints (e.g., formatting rules, style guidelines) that must be simultaneously followed. Our controlled experimental design, where the core task description is kept consistent while varying the number of instructions, allows us to systematically investigate the impact of instruction counts on LLM performance.

how well they can meet human expectations. Many early benchmarks focused on evaluating responses to different types of instructions, using relatively simple prompts (Wang et al., 2022; Li et al., 2023; Zheng et al., 2023).

As LLMs are increasingly expected to handle more complex real-world tasks, studies have attempted to evaluate models’ instruction-following capabilities from various aspects (Xu et al., 2024; Hayati et al., 2024; He et al., 2024a; Li et al., 2024; Wen et al., 2024; Jiang et al., 2024; Zhang et al., 2024; Qin et al., 2025; He et al., 2024b; Yan et al., 2025). ComplexBench (Wen et al., 2024) categorizes instructions into four types based on their characteristics: “And” (requiring multiple conditions to be satisfied simultaneously), “Chain” (following steps in a specific order), “Selection” (choosing actions based on conditions), and “Nested” (combining And, Chain, and Selection in a hierarchical structure). The “And” type closely aligns with our focus on satisfying multiple instructions simultaneously. Additionally, there are existing benchmarks that assess LLMs’ multiple-instructions-following

Benchmark	#Samples	#Instructions per sample (max)	Same task description with various # of instructions	Rule-based verifier for all instructions	Rule-based verifier for task description
FollowBench (Mixed)	85	5	✓ Yes	✗ No	✗ No
IFEval	541	3	✗ No	✓ Yes	✗ No
CELLO	523	6	✗ No	✓ Yes	✗ No
CFBench	1000	14	✗ No	✗ No	✗ No
ComplexBench	1150	14	✗ No	✗ No	✗ No
InfoBench	500	15	✗ No	✗ No	✗ No
CodeIF	1200	21	✗ No	✗ No	✗ No
ManyIFEval (Ours)	2160	10	✓ Yes	✓ Yes	✗ No
StyleMBPP (Ours)	3000	6	✓ Yes	✓ Yes	✓ Yes

Table 1: Benchmarks for evaluating multiple-instructions-following ability of LLMs. Our benchmarks (ManyIFEval and StyleMBPP) can conduct fair evaluation with the same task description across different number of instructions. In addition, ours can objectively measure the ability with rule-based, programmatically verifiable approach.

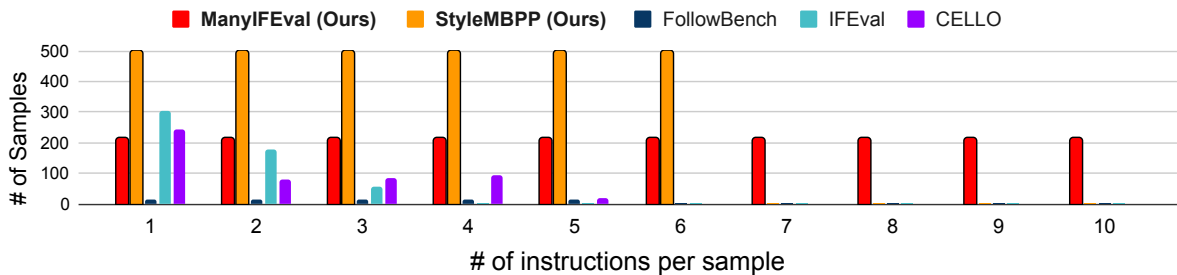


Figure 2: Comparison of sample size distribution per instruction count between our proposed ManyIFEval and StyleMBPP benchmarks and existing benchmarks (FollowBench, IFEval, and CELLO). Our benchmarks ensure balanced and sufficient sample sizes for each instruction count, enabling a robust analysis of the performance trends as the number of instructions increases.

ability, as described in Table 1 and Appendix A. Multi-turn benchmarks such as Multi-IF (He et al., 2024b) and SysBench (Qin et al., 2025) evaluate whether all instructions are followed throughout the entire conversation. In contrast, our benchmark focuses on a more fundamental single-turn problem setting, which is the basis for the multi-turn instruction following ability.

While existing benchmarks use different terminology to refer to the additional requirements that models must satisfy, with some using "constraint" as in ComplexBench (Wen et al., 2024) and FollowBench (Jiang et al., 2024), and others using "instruction" as in IFEval (Zhou et al., 2023) and Multi-IF (He et al., 2024b), we adopt the term "instruction" following IFEval’s convention throughout this paper.

2.2 Instruction-Following Performance with Multiple Instructions

While results of previous benchmarks have suggested that following multiple instructions becomes more challenging as their number increases, existing evaluations face limitations in isolating and measuring this effect. For instance, Com-

plexBench (Wen et al., 2024) shows lower performance on Chain, Selection, and Nested instruction types which contain more instructions, but this could be attributed to various factors beyond instruction count, such as task complexity. FollowBench’s Mixed dataset (Jiang et al., 2024), while conceptually similar to our focus, has limited samples (17 per instruction count) and relies on model-based evaluation, leading to inconsistent results where higher instruction levels sometimes outperform lower ones. IFEval (Zhou et al., 2023) provides objective programmatic verification but varies both instruction count and task descriptions across samples, making it difficult to attribute performance changes specifically to increased instruction count. In summary, existing benchmarks have the following three limitations: (1) unbalanced or insufficient sample sizes across instruction counts, (2) reliance on LLM-based evaluation methods that lack objectivity and reliability (Zhou et al., 2023; Zeng et al., 2024; Zheng et al., 2023; Wang et al., 2024), and (3) inconsistency in task descriptions across varying instruction counts.

Our benchmarks are specifically designed to address these limitations by systematically varying

the number of instructions added to each task description, ensuring balanced sample sizes across different instruction counts. In addition, by employing programmatic rule-based verification for each added instruction, we enable objective and reliable assessment of the relationship between instruction count and instruction-following performance. See [Table 1](#) and [Figure 2](#) for details.

2.3 Performance Estimation

As parameters of language models grow, the computational cost of evaluation on benchmarks increases. While some studies attempt to reduce evaluation costs through strategic sampling ([Perlitz et al., 2024](#); [Vivek et al., 2024](#); [Maia Polo et al., 2024](#)) or combining existing benchmarks ([Ni et al., 2024](#)), we focus on modeling and estimating how instruction-following performance degrades as the instruction count increases, using our benchmarks.

3 Benchmark Creation

To systematically investigate multiple instructions-following capabilities of LLMs, we introduce two benchmarks: ManyIFEval for general writing task with up to ten instructions and StyleMBPP for code generations with up to six instructions. Both benchmarks are constructed with programmatically verifiable instructions to ensure objective and reliable evaluation. Our benchmarks are specifically created using sets of instructions that are non-conflicting and simultaneously satisfiable. The primary goal is to investigate how performance degrades as the number of instructions increases, keeping the instructions themselves compatible. We ensured this compatibility through careful curation. ManyIFEval extends IFEval ([Zhou et al., 2023](#)), while StyleMBPP builds upon MBPP ([Austin et al., 2021](#)). ManyIFEval and StyleMBPP are under the CC BY 4.0 International license same as original MBPP and IFEval.

3.1 ManyIFEval

ManyIFEval (Many Instruction-Following Eval) is designed to evaluate text generation tasks under multiple instructions, and is an extension of IFEval ([Zhou et al., 2023](#)), which comprises task descriptions such as *"Write a blog post about a trip to Japan."* and a set of objectively verifiable instructions shown in [Figure 1](#) (top).

We started with the 25 instructions from the original IFEval dataset. We removed instructions that

# Instructions	Rule-based (Ground Truth)	LLM-as-a-Judge (GPT-4o)
5	0.574	0.815
10	0.213	0.657

Table 2: Comparison of Prompt-level Accuracy on ManyIFEval using rule-based verification vs. LLM-as-a-Judge (GPT-4o zero-shot). LLM-as-a-Judge tends to inflate accuracy scores and makes it difficult to accurately measure performance degradation as instruction count increases.

were inherently difficult to combine or prone to conflict (e.g., specifying the output language, repeating the prompt, generating two distinct responses) or had very low individual success rates (e.g., specifying first word in i -th paragraph). This resulted in a set of 15 instructions. We mapped out potential conflicts between these 15 (e.g., character count limits vs. paragraph count requirements; uppercase vs. lowercase). When generating samples for our benchmark, we algorithmically ensured that only mutually compatible instructions were selected and combined for any given prompt.

Finally, we selected 216 task descriptions from IFEval. For each task description, we generated samples with varying numbers of non-conflicting and simultaneously satisfiable instructions, ranging from 1 to 10. The instructions in ManyIFEval fall into six categories: (1) keyword requirement for including/excluding specific words, (2) length instructions for controlling text length, (3) formatting requirements like bullet points and placeholders, (4) letter case specifications for uppercase/lowercase usage, (5) text beginning/ending requirements like quotation marks, and (6) punctuation rules for allowing/prohibiting specific marks. [Table 5](#) in [Appendix E](#) shows the success rates of individual instructions.

To demonstrate that rule-based verification is more reliable than LLM-as-a-Judge, we compared both evaluation methods on ManyIFEval. As shown in [Table 2](#), LLM-as-a-Judge tends to inflate accuracy scores, particularly as instruction count increases. This inflation makes it difficult to accurately measure the true degradation in multiple-instructions-following capabilities, reinforcing the importance of objective, programmatic verification for reliable benchmark evaluation.

3.2 StyleMBPP

StyleMBPP (Style-aware Mostly Basic Programming Problems) is designed for evaluating code generation tasks under multiple style-related instructions, as shown in Figure 1 (bottom). This benchmark extends MBPP (Austin et al., 2021), a code generation benchmark focused on basic Python programming problems, such as "Write a Python function to sort the given array by using merge sort.". While MBPP assesses code correctness based on test cases, StyleMBPP adds instructions to evaluate models' adherence to coding style guidelines while maintaining functional correctness. We selected common Python coding style guidelines, primarily focusing on those verifiable using Pylint (Pylint contributors, 2003). We chose non-conflicting instructions and filtered out any with extremely low individual success rates. Finally, we add a new instruction to this original set, where we require the model to incorporate a "MIT License notice" to the generated code.

For StyleMBPP, we augmented the 500 test problems from MBPP with style instructions. For each MBPP problem, we created samples with instruction counts ranging from 1 to 6, resulting in 3,000 samples. The style instructions require specific code formatting aspects: (1) Generated code files must include the MIT License notice. (2) Code must follow strict indentation rules, using spaces instead of tabs. (3) Functions must include docstrings. (4) Specific conditional comparison operators may be required or prohibited. (5) Lines must not exceed a maximum character length. (6) Variable names must adhere to length restrictions. Table 6 in Appendix E shows the success rate of following individual instructions.

4 Evaluation

4.1 Setup

To evaluate the multiple-instructions-following capabilities of LLMs, we conducted experiments using both closed and open models. For closed models (accessed via APIs), we evaluated GPT-4o (gpt-4o-2024-05-13) (OpenAI, 2024b), Claude 3.5 Sonnet (claude-3-5-sonnet-20240620) (Anthropic, 2024), Gemini 1.5 Pro (gemini-1.5-pro-002) (Gemini Team, 2024) and o3-mini (OpenAI, 2025). For open models, we assessed Gemma 2 (gemma-2-9b-it, gemma-2-2b-it) (Gemma Team, 2024), Llama 3.1 (Meta-Llama-3.1-8B-Instruct) (Llama Team, 2024), Qwen2.5-

72B (Qwen2.5-72B-Instruct) (Qwen Team, 2025), DeepSeek-V3 (DeepSeek-V3-0324) (DeepSeek-AI, 2025b) and DeepSeek-R1 (DeepSeek-AI, 2025a). All models were evaluated using zero-shot prompting presenting the task description along with varying numbers of instructions. See Appendix C for the detailed evaluation setup. We report results of Qwen2.5-72B, DeepSeek-V3, DeepSeek-R1 and o3-mini in Appendix F.

4.2 Evaluation Metrics

To evaluate instruction-following performance, we adopted metrics based on IFEval (Zhou et al., 2023) and FollowBench (Jiang et al., 2024).

Prompt-level Accuracy Prompt-level Accuracy is the success rate of following all given instructions simultaneously for a particular prompt (Equation 1). This assesses the model's capability to handle multiple instructions at once. This is Hard Satisfaction Rate (HSR) in FollowBench. Prompt-level Accuracy is defined as follows:

$$\text{Prompt-level Accuracy (n)} = \frac{1}{m} \sum_{i=1}^m \prod_{j=1}^n s_i^j, \quad (1)$$

where m represents the number of prompts and n represents the number of instructions per prompt, s_i^j represents a binary metric if the target instruction $j \in n$ in a task $i \in m$ is successfully followed. Therefore, $\prod_{j=1}^n s_i^j = 1$ if all instructions for prompt i are satisfied, and 0 otherwise. In other words, this metric computes whether all the instructions in a prompt have been followed.

Instruction-level Accuracy (Inst-level accuracy)

Instruction-level Accuracy is the success rate of following individual instructions in its response (Equation 2). This metric assesses the model's ability to adhere to each instruction separately. This is Soft Satisfaction Rate (SSR) in FollowBench. This accuracy is defined as follows:

$$\text{Inst-level Accuracy (n)} = \frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n s_i^j, \quad (2)$$

where $s_i^j = 1$ if the j -th instruction of the i -th task is satisfied, and $s_i^j = 0$ otherwise.

4.3 Result

ManyIFEval Figure 3 shows the Instruction-level Accuracy (left panel) and Prompt-level Accuracy (right panel) across different instruction counts for ManyIFEval. Results consistently demonstrate

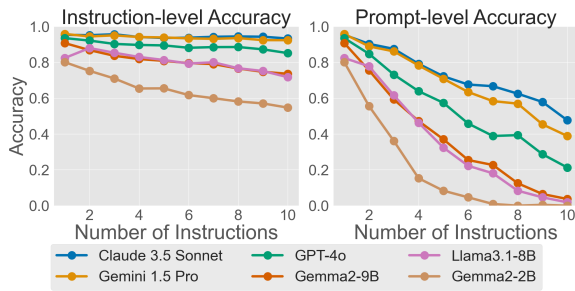


Figure 3: Evaluation results on ManyIFEval. The left panel shows Instruction-level Accuracy (the average success rate of following individual instructions), and the right panel shows Prompt-level Accuracy (the success rate of satisfying all instructions in a prompt simultaneously). Prompt-level Accuracy consistently shows a degrading trend as the number of instructions increases, while Instruction-level Accuracy remains relatively

stable. Performance degradation across all evaluated models as the number of instructions increases, for both metrics. While models exhibit relatively high accuracy in following individual instructions in isolation, their ability to satisfy all instructions simultaneously diminishes significantly as the instruction count rises. Notably, the relative performance ranking among models remains consistent between Instruction-level and Prompt-level accuracy metrics - models that perform better at following individual instructions also demonstrate superior performance when required to follow multiple instructions simultaneously. This indicates that the presence of multiple instructions introduces complexity for LLMs, even when the individual instructions are intrinsically simple. Figure 14 in Appendix E shows detailed results of following individual instructions. Through comparison with experimental results from IFEval and FollowBench, we demonstrated that our benchmark provides more stable performance measurement across a higher number of instructions (see Figure 10 in Appendix B).

StyleMBPP Secondly, we evaluated performance on StyleMBPP, which requires models to follow multiple style guide instructions while generating Python code to pass test cases. Result in Figure 4 shows similar degradation patterns as in ManyIFEval. While models exhibit relatively high accuracy in following individual instructions in isolation, their performance of following all instructions degrade as the instruction count rises.

Compared to GPT-4o, both Gemini 1.5 Pro and Claude 3.5 show much lower Prompt-level Accuracy when given six instructions simultaneously.

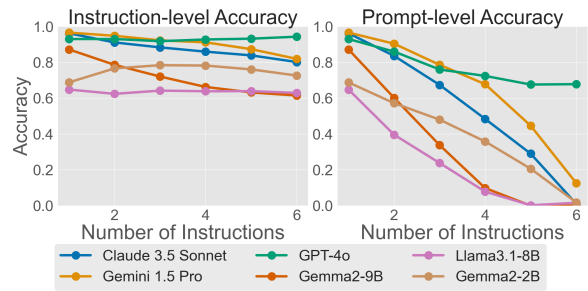


Figure 4: Evaluation results on StyleMBPP. Similar to ManyIFEval, the Prompt-level Accuracy shows consistent degradation trend as the number of instructions increases.

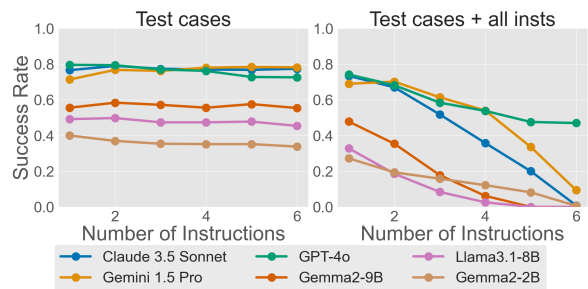


Figure 5: Evaluation results of code generation success rate on StyleMBPP. Left panel: success rate of passing all test cases for the task description, which remains relatively stable even as more instructions are added. Right panel: success rate of passing all test cases and satisfying all instructions simultaneously ("Test cases + all insts"), which shows a significant performance drop as instruction count increases.

This performance gap is attributed to certain instructions becoming significantly more challenging for these models as the instruction count increases. Specifically, while both models can follow the "Characters per line" instruction with high success rates (99% for Gemini 1.5 Pro and 97% for Claude 3.5) when it is presented in isolation, their performance drops dramatically to 20% and 2% respectively when this instruction is combined with five other instructions. For open models like Gemma 2 and Llama 3.1, we observe that their success rate for the "Indentation" instruction is extremely low, making it difficult for these models to follow all instructions successfully when this instruction is included in the prompt. Figure 15 in Appendix E shows detailed results of following individual instructions.

As shown in Figure 5, the functional correctness of the generated code measured by passing all test cases remains relatively stable even as more style instructions are added (left panel). However,

for more practical cases where a base task description must be executed while following multiple instructions simultaneously, we observe performance degradation as the number of instructions increases (right panel). This indicates that while models can maintain their core programming capabilities, their ability to simultaneously satisfy multiple style guide instructions becomes increasingly challenging with more instructions.

Improvements by Reasoning As shown in Tables 8, 9, 11, 12 of Appendix F, we observed that reasoning models outperform non-reasoning models (DeepSeek-V3 vs DeepSeek-R1, GPT-4o vs o3-mini). We observed that setting "reasoning effort" in o3-mini to "high" yielded the best results. In reasoning traces, DeepSeek-R1 explicitly checks each given instruction one by one to formulate a plan of approach as shown in Table 14. This suggests that reasoning helps models better understand and adhere to multiple instructions.

Comparison with General Benchmarks While models show comparable performance on standard benchmarks such as MMLU (Hendrycks et al., 2021), our benchmarks reveal substantial performance gaps on the instruction-following capabilities. For instance, DeepSeek-V3, GPT-4o, and Qwen2.5-72B achieve similar scores on standard benchmarks but there are performance gaps measured by Prompt-level Accuracy on ManyIFEval (n=10) and on StyleMBPP (n=6). See Appendix H for detailed comparison.

5 Performance Prediction

As parameters of models grow, their evaluation becomes increasingly computationally expensive. In multiple-instruction settings, evaluating all of them would require substantial computational resources because the number of possible instruction combinations are massive. To address this challenge, we develop models capable of estimating performance on both unseen instruction combinations and different numbers of instructions. We explore three modeling approaches: **Naive Estimators**, **Beta-Binomial**, and **Logistic Regression**, each offering different perspectives on modeling multiple-instructions-following behavior.

5.1 Models

Naive Estimators We model the success or failure of following a single instruction as a Bernoulli

trial with probability $p \in [0, 1]$. For multiple instructions presented together, the simplest approach assumes independent Bernoulli trials. The probability of successfully following all n instructions is the product of their individual success probabilities, $\prod_{i=1}^n p_i$, where p_i represents instruction i 's success probability. We explore two variants of this estimator. In the first variant, which we refer to as **Product(Each, n=1)**, we use the empirical success rates of following individual instructions when each instruction is presented in isolation. In the second variant, referred to as **Product(Each, n=n)**, we use the empirical success rates of following individual instructions when the instruction is presented together with $n - 1$ other instructions.

Beta-Binomial Similar to Naive Estimators, the Beta-Binomial approach models each instruction's success as a Bernoulli trial with probability p , but treats p itself as a random variable drawn from a Beta distribution $p \sim \text{Beta}(\alpha, \beta)$, and estimates α , β via maximum likelihood from the training data.

Logistic Regression We train logistic regression models that predict the probability of following all instructions successfully, based on features such as the number of instructions and instruction identifiers. We investigate several configurations of the model. One configuration, denoted as **Logistic (w/ n)**, includes only the instruction count as a feature. Another, referred to as **Logistic (w/ n) (trained n≤k)**, is trained on data containing at most k instructions, also using instruction count as a feature. A third configuration, **Logistic (w/ n, IDs)**, incorporates both the instruction count and instruction identifiers, unique numerical labels assigned to each distinct type of instruction, as features.

Beta-Binomial and Logistic (w/ n) models do not account for each instruction's difficulty, while Product and Logistic (w/ n, IDs) incorporate this information through instruction identifiers.

5.2 Training and Evaluation Setup

For all methods, we evaluate performance using the predicted probability of successfully following all instructions, compared to the actual success rate measured by Prompt-level Accuracy on the test set. This allows us to assess how well each approach generalizes to both unseen instruction combinations and different number of instructions. For both ManyIFEval and StyleMBPP benchmarks, we split the datasets into train and test sets to fit the models on the train split and evaluate their performance on

the test split. When splitting the data, we focused on task descriptions and instruction combinations. Specifically, for ManyIFEval, we created sets of instruction combinations with 5 instructions, while for StyleMBPP, we created sets with 3 instructions. We then divided these combination sets between train and test splits ensuring no overlap in combinations. The splits were created such that there were no shared task descriptions between train and test sets, and no shared instruction combinations for the 5-instruction cases in ManyIFEval and 3-instruction cases in StyleMBPP. This resulted in 1,070 training samples and 1,090 test samples for ManyIFEval, and 1,458 training samples and 1,542 test samples for StyleMBPP.

We estimate parameters for the Beta-Binomial model using NumPy and SciPy (Harris et al., 2020; Virtanen et al., 2020), while the logistic regression model is trained with statsmodels (Seabold and Perktold, 2010). We used GPT-4o, Claude 3.5 Sonnet, Gemini 1.5 Pro, Gemma 2 (2B, 9B) and Llama 3.1 for experiments.

5.3 Results

Figure 6 illustrates the empirically-observed Prompt-level accuracy and the values predicted by models on ManyIFEval and StyleMBPP. As shown in Table 3, despite its simplicity, the Logistic (w/ n) model achieves mean absolute errors consistently within 0.1 across both ManyIFEval and StyleMBPP benchmarks, demonstrating effective generalization to unseen instruction combinations.

To assess the relationship between sample size and estimation error, we conducted an analysis varying the size of the training set, as shown in Figure 7. For each training sample size, we run five iterations changing random seeds. Our results demonstrate that relatively modest sample sizes are sufficient for reliable performance estimation. Specifically, ManyIFEval requires approximately 500 samples (50 task descriptions for each number of instructions) to achieve stable estimates, while StyleMBPP needs around 300 samples (50 task descriptions for each number of instructions). Beyond these thresholds, increasing the sample size yields diminishing returns in terms of estimation accuracy. This finding has practical implications for future benchmark development and evaluation, suggesting that comprehensive multiple-instructions-following assessment can be achieved with manageable dataset size that saves computational resources.

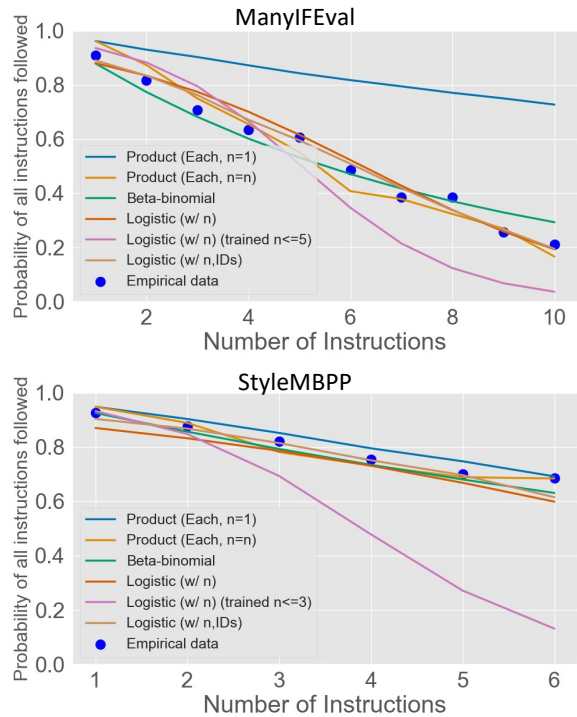


Figure 6: Comparison of empirical Prompt-level Accuracy data (points) and predictions from various estimation models (lines) on ManyIFEval (top) and StyleMBPP (bottom) for GPT-4o. Simple models such as Beta-binomial, Logistic regression model using instruction count as a feature capture the performance degradation trend as the number of instructions increases.

For unseen number of instructions, we evaluated the Logistic (w/ n) model on ManyIFEval. Table 4 shows that the model exhibits a consistent decrease in estimation error as the number of instructions used for training increases. Same trend is observed for StyleMBPP (See Table 13 in Appendix F).

5.4 Discussion

Although our study has systematically analyzed multiple-instructions-following ability of LLMs, several important questions remain for future work. First, whether similar relationships between instruction count and performance hold for more complex instruction types not covered in our benchmarks, such as semantic instructions, conditional logic, or multi-step procedures. Second, further investigation is needed into the mechanisms behind the performance degradation observed with increasing instruction count. Based on previous works (Venkateswaran and Contractor, 2025; Heo et al., 2025; Stolfo et al., 2025) which suggested a relationship between instruction following and the activation values of specific neurons or attention scores, understanding these failure modes through

Method	ManyIFEval			StyleMBPP		
	n=5	n=10	Corr (r)	n=3	n=6	Corr (r)
Product (Each, n=1)	0.21 ± 0.07	0.34 ± 0.13	0.904	0.15 ± 0.09	0.31 ± 0.32	0.728
Product (Each, n=n)	0.04 ± 0.03	0.02 ± 0.02	0.994	0.05 ± 0.01	0.01 ± 0.01	0.996
Beta-binomial	0.05 ± 0.05	0.06 ± 0.03	0.974	0.05 ± 0.02	0.19 ± 0.13	0.922
Logistic (w/ n)	0.04 ± 0.03	0.02 ± 0.03	0.993	0.06 ± 0.05	0.05 ± 0.03	0.980
Logistic (w/ n, IDs)	0.03 ± 0.04	0.02 ± 0.03	0.994	0.05 ± 0.05	0.03 ± 0.02	0.988

Table 3: Mean absolute error ± standard deviation and Pearson correlation (r) of Prompt-level Accuracy predictions by various performance estimation models.

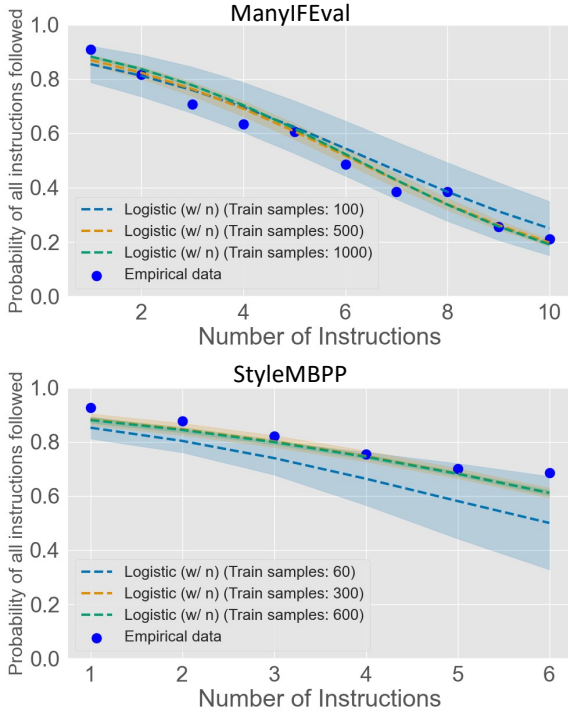


Figure 7: Train sample size variation and estimation error of ManyIFEval (top) and StyleMBPP (bottom) tested on GPT-4o. ManyIFEval requires 500 samples (50 task descriptions for each # of instructions) and StyleMBPP requires 300 samples (50 task descriptions for each # of instructions) to achieve stable estimates.

analysis of attention patterns and internal model representations could inform the development of models more robust to multiple simultaneous instructions.

6 Conclusion

We introduced two benchmarks, ManyIFEval and StyleMBPP, to evaluate models’ ability to follow multiple instructions simultaneously. Our experiments have demonstrated that performance degrades as the number of instructions increases, with the degradation pattern varying across models and instruction types. We also showed that this degra-

Logistic (w/ n)	Abs Err (n=10)	Corr (r)
trained n≤5	0.15 ± 0.16	0.944
trained n≤7	0.09 ± 0.09	0.989
trained n≤9	0.03 ± 0.04	0.988

Table 4: Result of estimation for unseen number of instructions. Absolute error between predictions and empirical result of successfully following all instructions on unseen number of instructions aggregated over six LLMs on ManyIFEval.

dation can be modeled using simple approaches like logistic regression, enabling accurate prediction of model performance on unseen instruction combinations with relatively small sample sizes. These findings provide insights for understanding and improving models’ ability to handle multiple instructions simultaneously, an essential capability for real-world applications.

Limitations

While our benchmarks cover a range of instruction types, they are still limited in scope. We focused on relatively simple instructions that can be objectively evaluated, such as keyword inclusion, character counts, and formatting rules. More complex instruction types involving semantic understanding, conditional logic (e.g., if/then rules), or multi-step procedures (e.g., first summarize then analyze) were not included. Second, our analysis of performance degradation mechanisms is primarily empirical; we observe the patterns but cannot fully explain the underlying causes. A deeper investigation of model internals, such as analyzing attention scores between instruction tokens and output tokens, studying activation patterns in different model layers, and examining how token representations evolve during the generation process, would be valuable for understanding why certain models struggle more with multiple instructions.

Acknowledgments

We used ChatGPT for translation and paraphrasing of our original text, and GitHub Copilot to help draft plotting scripts for our experimental visualizations

References

- Anthropic. 2024. [Claude 3.5 sonnet](#).
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. [Program synthesis with large language models](#).
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.
- DeepSeek-AI. 2025a. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#).
- DeepSeek-AI. 2025b. [Deepseek-v3 technical report](#).
- Fireworks AI. 2023. [Fireworks AI](#).
- Gemini Team. 2024. [Gemini 1.5: Unlocking multi-modal understanding across millions of tokens of context](#). *Preprint*, arXiv:2403.05530.
- Gemma Team. 2024. [Gemma 2: Improving open language models at a practical size](#). *Preprint*, arXiv:2408.00118.
- Google. 2023. [Palm 2 technical report](#). *Preprint*, arXiv:2305.10403.
- Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'io, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. [Array programming with NumPy](#). *Nature*, 585(7825):357–362.
- Shirley Anugrah Hayati, Taehee Jung, Tristan Bodding-Long, Sudipta Kar, Abhinav Sethy, Joo-Kyung Kim, and Dongyeop Kang. 2024. [Chain-of-instructions: Compositional instruction tuning on large language models](#). *Preprint*, arXiv:2402.11532.
- 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18188–18196.
- Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, et al. 2024b. [Multi-if: Benchmarking llms on multi-turn and multilingual instructions following](#). *arXiv preprint arXiv:2410.15553*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#). In *International Conference on Learning Representations*.
- Juyeon Heo, Christina Heinze-Deml, Oussama Elachqar, Kwan Ho Ryan Chan, Shirley You Ren, Andrew Miller, Udhyakumar Nallasamy, and Jaya Narain. 2025. [Do LLMs "know" internally when they follow instructions?](#) In *The Thirteenth International Conference on Learning Representations*.
- Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. [Follow-Bench: A multi-level fine-grained constraints following benchmark for large language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688, Bangkok, Thailand. Association for Computational Linguistics.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. [Efficient memory management for large language model serving with pagedattention](#). In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Kenneth Li, Tianle Liu, Naomi Bashkansky, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. [Measuring and controlling instruction \(in\)stability in language model dialogs](#). In *First Conference on Language Modeling*.
- Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. [AlpacaEval: An automatic evaluator of instruction-following models](#). https://github.com/tatsu-lab/alpaca_eval.
- Llama Team. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.

- Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. 2024. [tinyBenchmarks: evaluating LLMs with fewer examples](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 34303–34326. PMLR.
- Jinjie Ni, Fuzhao Xue, Xiang Yue, Yuntian Deng, Mahir Shah, Kabir Jain, Graham Neubig, and Yang You. 2024. [Mixeval: Deriving wisdom of the crowd from llm benchmark mixtures](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 98180–98212. Curran Associates, Inc.
- OpenAI. 2024a. [Gpt-4 technical report](#). *Preprint*, arXiv:2303.08774.
- OpenAI. 2024b. [Gpt-4o system card](#).
- OpenAI. 2025. [Openai o3-mini system card](#).
- Yotam Perlit, Elron Bandel, Ariel Gera, Ofir Arviv, Liat Ein-Dor, Eyal Shnarch, Noam Slonim, Michal Shmueli-Scheuer, and Leshem Choshen. 2024. [Efficient benchmarking \(of language models\)](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2519–2536, Mexico City, Mexico. Association for Computational Linguistics.
- Pylint contributors. 2003. [Pylint](#).
- Yanzhao Qin, Tao Zhang, Tao Zhang, Yanjun Shen, Wenjing Luo, sunhaoze, Yan Zhang, Yujing Qiao, weipeng chen, Zenan Zhou, Wentao Zhang, and Bin CUI. 2025. [Sysbench: Can LLMs follow system message?](#) In *The Thirteenth International Conference on Learning Representations*.
- Qwen Team. 2025. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Skipper Seabold and Josef Perktold. 2010. [statsmodels: Econometric and statistical modeling with python](#). In *9th Python in Science Conference*.
- Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. 2025. [Improving instruction-following in language models through activation steering](#). In *The Thirteenth International Conference on Learning Representations*.
- Praveen Venkateswaran and Danish Contractor. 2025. [Spotlight your instructions: Instruction-following with dynamic attention steering](#).
- Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. [SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python](#). *Nature Methods*, 17:261–272.
- Rajan Vivek, Kawin Ethayarajh, Diyi Yang, and Douwe Kiela. 2024. [Anchor points: Benchmarking models with much fewer examples](#). In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1576–1601, St. Julian’s, Malta. Association for Computational Linguistics.
- Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. [Large language models are not fair evaluators](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, Bangkok, Thailand. Association for Computational Linguistics.
- Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022. [Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiabin Xu, et al. 2024. [Benchmarking complex instruction-following with multiple constraints composition](#). *arXiv preprint arXiv:2407.03978*.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei

Lin, and Daxin Jiang. 2024. [WizardLM: Empowering large pre-trained language models to follow complex instructions](#). In *The Twelfth International Conference on Learning Representations*.

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](#).

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2024. [Evaluating large language models at evaluating instruction following](#). In *The Twelfth International Conference on Learning Representations*.

Tao Zhang, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, Bin Cui, et al. 2024. [Cfbench: A comprehensive constraints-following benchmark for llms](#). *arXiv preprint arXiv:2408.01122*.

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

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Sidhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. [Instruction-following evaluation for large language models](#). *arXiv preprint arXiv:2311.07911*.

A Relationship between Multi-turn Benchmarks and Our Benchmarks

Multiple instructions following is a practically important capability, and in recent years, benchmarks have been proposed in multi-turn settings to evaluate whether all instructions are followed throughout the entire conversation such as Multi-IF (He et al., 2024b) and SysBench (Qin et al., 2025). In contrast, our benchmark focuses on a more fundamental single-turn problem setting, which is the basis for the multi-turn instruction following ability.

Our benchmarks can be easily extended to the problem setting where instructions are added turn by turn, as seen in Multi-IF and SysBench. The results are presented in Figure 8 and Figure 9. Similar to the single-turn setting, we observed that performance in following instructions declines as the number of instructions increases.

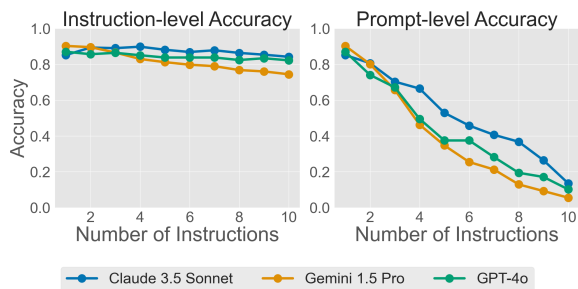


Figure 8: Evaluation results on ManyIFEval in multi-turn settings. Similar to the single-turn setting, we observed that performance in following instructions declines as the number of instructions increases.

B Reliable Performance Measurement with Increasing Instruction Counts using ManyIFEval

Evaluating the capability of Large Language Models (LLMs) to follow an increasing number of simultaneous instructions requires benchmarks that ensure both reliable verification and sufficient data. Previous benchmarks, such as IFEval and FollowBench, while valuable, have sometimes faced limitations due to smaller sample sizes for certain instruction configurations or reliance on evaluation methods that can introduce variability.

Our benchmarks are specifically designed to address these challenges by combining (1) programmatic, rule-based verification for objective assessment of instruction following, and (2) a substantial and balanced number of samples for each count of instructions. This design ensures that performance changes can be confidently attributed to the model’s ability to handle an increasing instruction load, rather than artifacts of small sample sizes or subjective evaluation.

To illustrate the reliability of ManyIFEval in measuring performance degradation, we evaluated the GPT-4o-mini model. Figure 10 presents the average Prompt-level Accuracy and standard deviation across three different random seeds on IFEval, FollowBench and ManyIFEval. The consistent trend and minimal standard deviation observed underscore the benchmark’s capacity to provide a stable and dependable measure of how an LLM’s performance on a given task description is impacted by the addition of multiple, concurrent instructions.

C Details of Evaluation Setup

We used transformers (Wolf et al., 2020) and vLLM (Kwon et al., 2023) libraries for Gemma

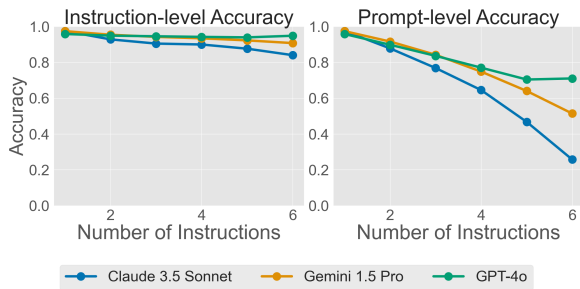


Figure 9: Evaluation results on StyleMBPP in multi-turn settings. Similar to the single-turn setting, we observed that performance in following instructions declines as the number of instructions increases.

2, Llama 3.1 models’ inference. For inference of Qwen2.5-72B, DeepSeek-V3 and DeepSeek-R1, we used endpoints cloud hosted by Fireworks AI (Fireworks AI, 2023). All models were evaluated using zero-shot prompting presenting the task description along with varying numbers of instructions to each model. During decoding, we employed greedy decoding (top-k=1) to ensure deterministic outputs across models, except for GPT-4o, o3-mini due to API limitations. For open models we tested locally, we used one H100 GPU(80 GB) and inference for each model takes within 1 hour.

D Task Description

We manually selected 216 task descriptions from the original IFEval (Zhou et al., 2023) benchmark for ManyIFEval. Original IFEval’s task descriptions are collected by few-shot prompting and manual curation. For StyleMBPP, we used all 500 task descriptions from MBPP benchmark (Austin et al., 2021). MBPP’s task descriptions are collected from crowdworkers who have basic knowledge of Python. Details of how the task descriptions are collected can be found in Section 2.1 of MBPP paper (Austin et al., 2021).

We show some example task descriptions in Figure 11 and Figure 12. All data are available at <https://github.com/kenoharada/Multiple-Instructions-Following>.

E Success Rate of Each Instruction Following

We present the success rates of following each instruction in isolation (when it is the only instruction given alongside the task description) using the GPT-4o model.

Table 5 lists the success rates for each type of

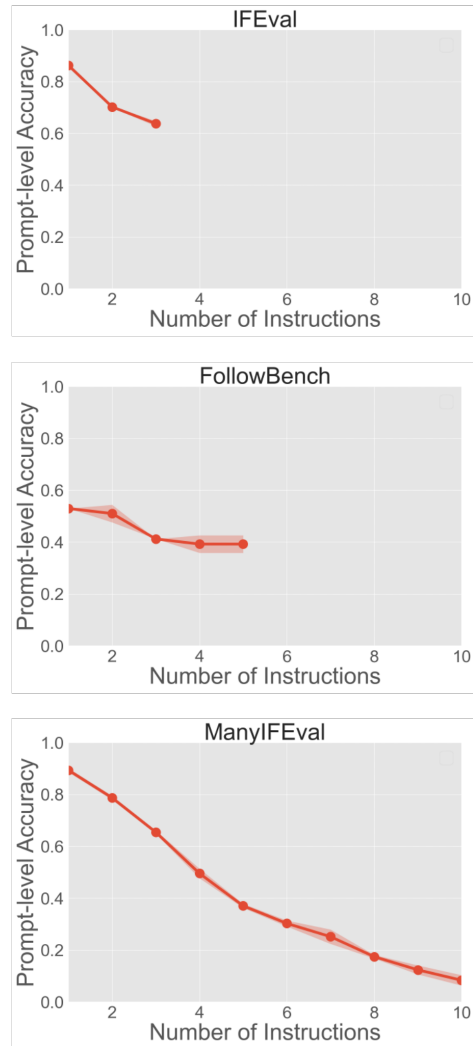


Figure 10: Average Prompt-level accuracy and standard deviation for GPT-4o-mini on IFEval, FollowBench and ManyIFEval across 3 random seeds. The shaded area represents the standard deviation. The results demonstrate ManyIFEval’s ability to reliably measure performance trends as instruction counts increase, facilitated by rule-based verification and ample sample sizes.

instruction within ManyIFEval benchmark when presented individually to GPT-4o. Please refer to Table 1 in IFEval paper (Zhou et al., 2023) for each instruction’s description. We used the same rule-based verifier as Zhou et al. (2023), which are written in Python. Every instruction has its program to judge the instruction is successfully followed or not. We show example rule-based verification code in Figure 13. Table 6 lists the success rates for each type of instruction within StyleMBPP benchmark when presented individually to GPT-4o. Table 7 shows each instruction’s description in StyleMBPP. Similar to ManyIFEval, we used rule-based verifiers written in Python to evaluate whether each

ManyIFEval Task Descriptions

Example 1: Write a blog post about the best way to get a good night’s sleep.

Example 2: Are the weather conditions in the Arctic very cold most of the year?

Figure 11: Example task descriptions from ManyIFEval benchmark.

StyleMBPP Task Descriptions

Example 1: Write a python function to find the maximum sum of elements of list in a list of lists. Your code should pass these tests: ... (omitted for brevity) ... `assert maximum_Sum([[0,1,3],[1,2,1],[9,8,2],[0,1,0],[6,4,8]]) == 19`

Example 2: Write a python function to check whether the first and last characters of a given string are equal or not. Your code should pass these tests: ... (omitted for brevity) ... `assert check_Equality("mad") == "Not Equal"`

Figure 12: Example task descriptions from StyleMBPP benchmark.

instruction is successfully followed or not using Pylint (Pylint contributors, 2003).

Instructions that require inserting specific text (Keywords, Title, MIT License, Quotation, Function docstring) or maintaining a simple pattern (UpperCase, Lowercase, Indentation) are easier for the model to follow. In contrast, instructions related to length or count (e.g., character count, number of sentences, variable name length) tend to have lower success rates.

F Detailed Experimental and Estimation Results

In this section, we present detailed aggregate experimental results for Prompt-level Accuracy and Instruction-level Accuracy across all evaluated LLMs on both ManyIFEval and StyleMBPP benchmarks. Additionally, we show detailed results for our performance estimation models when predicting performance on an unseen number of instructions for StyleMBPP.

Instruction	Success Rate
Keywords to include	0.98
Keyword occurrence count	0.96
Forbidden keywords	0.95
Specific character count	0.62
Character count	0.81
Sentence count	0.73
Paragraph count	0.87
Placeholder	0.95
Bullet points	0.88
Title	1.0
Uppercase	0.97
Lowercase	0.97
Uppercase word count	0.87
Quotation marks	1.0
Comma prohibition	1.0

Table 5: Single instruction following success rates for ManyIFEval instructions using GPT-4o.

Instruction	Success Rate
MIT License notice	1.000
Indentation	1.000
Function docstring	1.000
Conditional comparison	0.992
Characters per line	0.868
Variable name length	0.794

Table 6: Single instruction following success rates for StyleMBPP instructions using GPT-4o.

F.1 ManyIFEval Evaluation Results

Table 8 and Table 9 provide Prompt-level Accuracy and Instruction-level Accuracy, respectively, for all evaluated LLMs on ManyIFEval benchmark across varying numbers of instructions. Figure 14 shows the success rate of following each specific instruction type in ManyIFEval changes as the total number of simultaneously presented instructions increases. We show an example of GPT-4o’s response in Table 10.

F.2 StyleMBPP Evaluation Results

Table 11 and Table 12 provide Prompt-level Accuracy and Instruction-level Accuracy, respectively, for all evaluated LLMs on StyleMBPP benchmark across varying numbers of instructions. Figure 15 shows the success rate of following each specific instruction type in StyleMBPP changes as the total number of simultaneously presented instructions increases.

```

class BulletListChecker(Instruction):
    """Checks the bullet list in the prompt."""
    ... (omitted for brevity) ...

    def check_following(self, value):
        """Check if the number of bullet lists meets the requirement.

        Args:
            value: A string representing the response. The response is expected to
                contain some bullet lists that start with `*`.

        Returns:
            True if the actual number of bullet lists in the response meets the
            requirement.
        """
        bullet_lists = re.findall(r"^\s*\[^\]*.*$", value, flags=re.MULTILINE)
        bullet_lists_2 = re.findall(r"^\s*-.*$", value, flags=re.MULTILINE)
        num_bullet_lists = len(bullet_lists) + len(bullet_lists_2)
        return num_bullet_lists == self._num_bullets

```

Figure 13: Python code for evaluating ‘Bullet points’ instruction in ManyIFEval. We used the same code as IFEval (Zhou et al., 2023).

F.3 Estimation for Unseen Number of Instructions (StyleMBPP)

Table 13 shows the performance of logistic regression model in estimating the Prompt-level Accuracy for an unseen number of instructions ($n=6$) on the StyleMBPP benchmark, based on training with fewer instructions.

G Reasoning Trace Example

Table 14 demonstrates how reasoning models like DeepSeek-R1 approach multiple-instructions-following tasks. The reasoning trace reveals that the model explicitly identifies and processes each instruction separately, developing a plan before generating the response.

H Performance Gap Between Standard and Multiple-Instructions-Following Benchmarks

To better understand the capability differences between models, we compared their performance on standard benchmarks versus our multiple-instructions-following benchmarks. Table 15 presents results for three representative models across different evaluation tasks. On standard benchmarks (results from DeepSeek-AI (2025b)) such as MMLU (general knowledge), HumanEval-Mul (coding), and IFEval, the three models demonstrate comparable performance. However, our evaluation on benchmarks reveals substantial performance gap.

Instruction Name	Instruction
MIT License notice	Ensure the file includes the MIT License notice.
Indentation	Indent all code blocks using exactly two spaces; do not use tabs.
Function docstring	Ensure each function has a docstring describing its purpose.
Conditional comparison	Avoid comparing against True, False, or None using == or !=. Instead, use the variable’s truthiness (e.g., ‘if variable:’) or ‘is None’/‘is not None’ for None checks.
Characters per line	Limit all lines to a maximum of 79 characters.
Variable name length	All variable names should be at least three characters long.

Table 7: The list of 6 instructions in StyleMBPP.

Model	n = 1	n = 2	n = 3	n = 4	n = 5	n = 6	n = 7	n = 8	n = 9	n = 10
Claude 3.5 Sonnet	0.95	0.90	0.88	0.79	0.72	0.68	0.67	0.62	0.58	0.48
Gemini 1.5 Pro	0.96	0.89	0.86	0.78	0.71	0.63	0.58	0.57	0.45	0.39
GPT-4o	0.94	0.85	0.73	0.64	0.57	0.46	0.39	0.39	0.29	0.21
Gemma2-9B	0.91	0.75	0.59	0.47	0.37	0.25	0.23	0.12	0.06	0.04
Llama3.1-8B	0.82	0.78	0.62	0.46	0.32	0.22	0.18	0.08	0.05	0.02
Gemma2-2B	0.80	0.56	0.36	0.15	0.08	0.05	0.01	0.00	0.00	0.00
Qwen2.5-72B	0.95	0.83	0.71	0.52	0.40	0.28	0.19	0.12	0.04	0.02
DeepSeek-V3	0.96	0.90	0.85	0.70	0.63	0.48	0.43	0.33	0.29	0.19
DeepSeek-R1	0.95	0.89	0.81	0.74	0.71	0.64	0.59	0.49	0.44	0.38
o3-mini (low)	0.97	0.93	0.92	0.84	0.81	0.75	0.69	0.70	0.64	0.53
o3-mini (medium)	0.98	0.95	0.94	0.89	0.87	0.83	0.78	0.73	0.73	0.64
o3-mini (high)	1.00	0.96	0.95	0.93	0.90	0.88	0.83	0.82	0.79	0.78

Table 8: Prompt-level Accuracy for ManyIFEval

Model	n = 1	n = 2	n = 3	n = 4	n = 5	n = 6	n = 7	n = 8	n = 9	n = 10
Claude 3.5 Sonnet	0.95	0.95	0.96	0.94	0.94	0.94	0.94	0.94	0.94	0.93
Gemini 1.5 Pro	0.96	0.94	0.95	0.94	0.94	0.93	0.93	0.94	0.92	0.92
GPT-4o	0.94	0.92	0.90	0.90	0.89	0.88	0.88	0.89	0.87	0.85
Gemma2-9B	0.91	0.87	0.84	0.82	0.81	0.80	0.79	0.77	0.75	0.74
Llama3.1-8B	0.82	0.88	0.85	0.83	0.81	0.79	0.80	0.77	0.75	0.72
Gemma2-2B	0.80	0.75	0.71	0.65	0.65	0.62	0.60	0.58	0.57	0.55
Qwen2.5-72B	0.95	0.91	0.90	0.86	0.85	0.82	0.80	0.78	0.77	0.75
DeepSeek-V3	0.96	0.95	0.95	0.92	0.91	0.89	0.89	0.89	0.89	0.87
DeepSeek-R1	0.95	0.94	0.94	0.92	0.93	0.92	0.93	0.92	0.92	0.92
o3-mini (low)	0.97	0.96	0.97	0.96	0.96	0.95	0.95	0.96	0.95	0.94
o3-mini (medium)	0.98	0.97	0.98	0.97	0.97	0.97	0.97	0.96	0.96	0.96
o3-mini (high)	1.00	0.98	0.98	0.98	0.98	0.98	0.97	0.98	0.98	0.98

Table 9: Instruction-level Accuracy for ManyIFEval

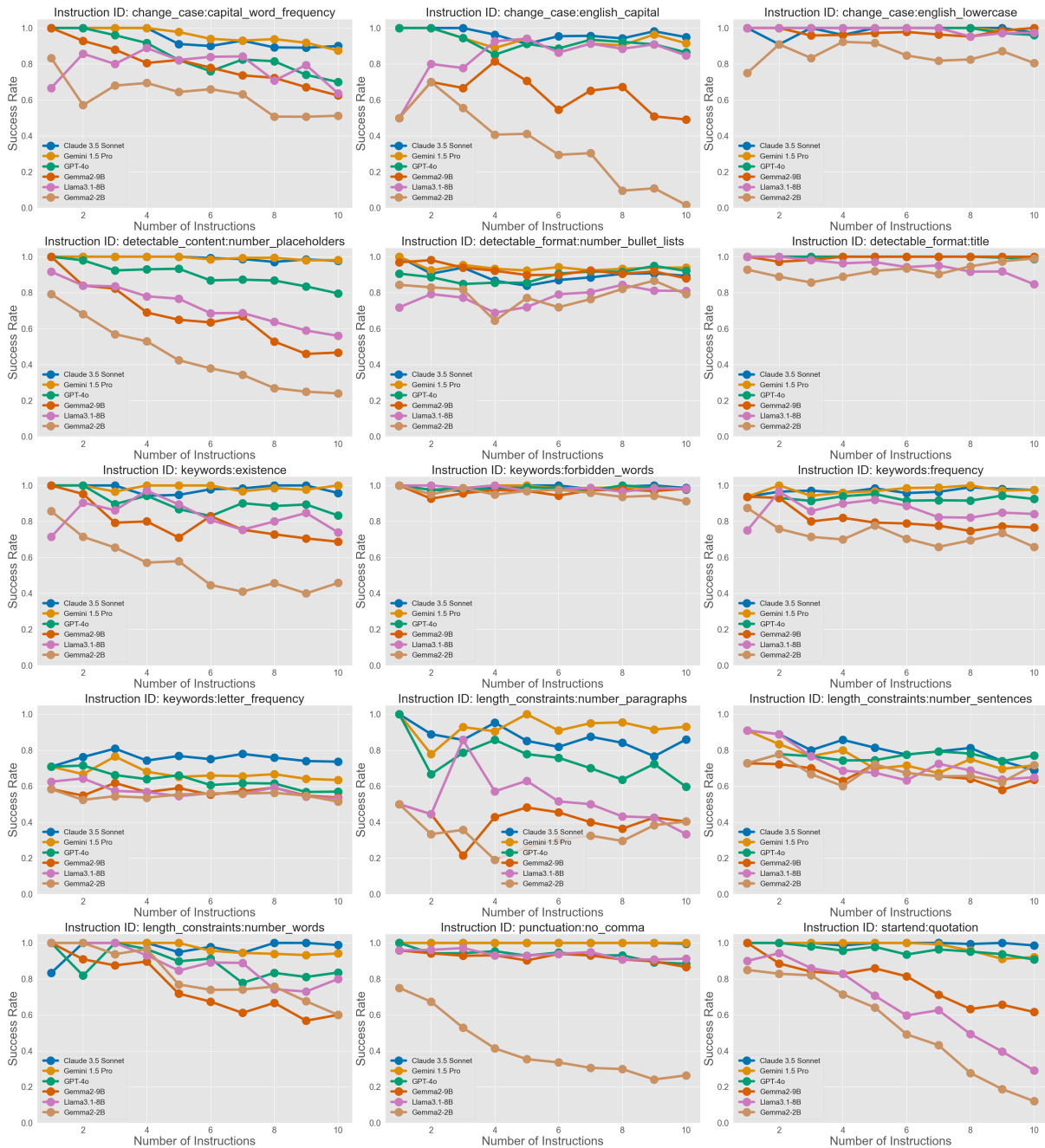


Figure 14: ManyIFEval instruction-following success rate of each instruction.

Example response of GPT-4o zero-shot for 5 instructions	
Prompt	<p>Write a blog post about the best way to get a good night’s sleep.</p> <p>Your response should follow the instructions below:</p> <ul style="list-style-type: none"> - Your answer must contain exactly 2 bullet points. Use the markdown bullet points such as: * This is point 1. * This is point 2 - In your response, the letter m should appear at least 6 times. - The response must contain at least 2 placeholders represented by square brackets, such as [address]. - Answer with at least 400 words. - Do not include keywords [’reason’, ’review’] in the response.
Response	<p># The Best Way to Get a Good Night’s Sleep</p> <p>Getting a good night’s sleep is essential for maintaining overall health and well-being.</p> <p>... (omitted for brevity) ...</p> <p>* **Create a Relaxing Bedtime Routine:** Developing a calming pre-sleep routine can signal to your body that it’s time to wind down. ... (omitted for brevity) ...</p> <p>* **Limit Caffeine and Alcohol Intake:** What you consume during the day can significantly impact your sleep quality. ... (omitted for brevity) ...</p> <p>... (omitted for brevity) ...</p> <p>* **Invest in a Comfortable Mattress and Pillows:** The quality of your mattress and pillows can make a significant difference in your sleep quality. ... (omitted for brevity) ...</p> <p>* **Control the Room Temperature:** The temperature of your bedroom can also affect your sleep. ... (omitted for brevity) ...</p> <p>... (omitted for brevity) ...</p> <p>For more personalized advice, feel free to reach out to [healthcare professional] or visit [website] for additional resources.</p>

Table 10: Example response of GPT-4o zero-shot for 5 instructions. Red text indicates instructions that are not being followed. Tested on ManyIFEval.

Model	n = 1	n = 2	n = 3	n = 4	n = 5	n = 6
Claude 3.5 Sonnet	0.96	0.84	0.67	0.48	0.29	0.01
Gemini 1.5 Pro	0.97	0.90	0.79	0.68	0.45	0.13
GPT-4o	0.93	0.86	0.76	0.72	0.68	0.68
Gemma2-9B	0.87	0.60	0.34	0.10	0.00	0.00
Llama3.1-8B	0.65	0.40	0.24	0.08	0.00	0.02
Gemma2-2B	0.69	0.57	0.48	0.36	0.21	0.02
Qwen2.5-72B	0.86	0.64	0.39	0.19	0.13	0.11
DeepSeek-V3	0.92	0.77	0.56	0.32	0.18	0.01
DeepSeek-R1	0.90	0.82	0.66	0.48	0.30	0.12
o3-mini (low)	0.96	0.86	0.68	0.56	0.34	0.18
o3-mini (medium)	0.95	0.84	0.69	0.57	0.38	0.15
o3-mini (high)	0.95	0.89	0.79	0.68	0.58	0.37

Table 11: Prompt-level Accuracy for StyleMBPP

Model	n = 1	n = 2	n = 3	n = 4	n = 5	n = 6
Claude 3.5 Sonnet	0.96	0.91	0.88	0.86	0.84	0.80
Gemini 1.5 Pro	0.97	0.95	0.92	0.91	0.87	0.82
GPT-4o	0.93	0.93	0.92	0.93	0.93	0.94
Gemma2-9B	0.87	0.79	0.72	0.66	0.63	0.61
Llama3.1-8B	0.65	0.62	0.64	0.64	0.64	0.63
Gemma2-2B	0.69	0.77	0.78	0.78	0.76	0.73
Qwen2.5-72B	0.86	0.81	0.76	0.72	0.72	0.69
DeepSeek-V3	0.92	0.89	0.84	0.80	0.77	0.74
DeepSeek-R1	0.90	0.89	0.86	0.83	0.79	0.74
o3-mini (low)	0.96	0.93	0.89	0.88	0.85	0.82
o3-mini (medium)	0.95	0.92	0.89	0.89	0.85	0.82
o3-mini (high)	0.95	0.95	0.93	0.91	0.90	0.87

Table 12: Instruction-level Accuracy for StyleMBPP

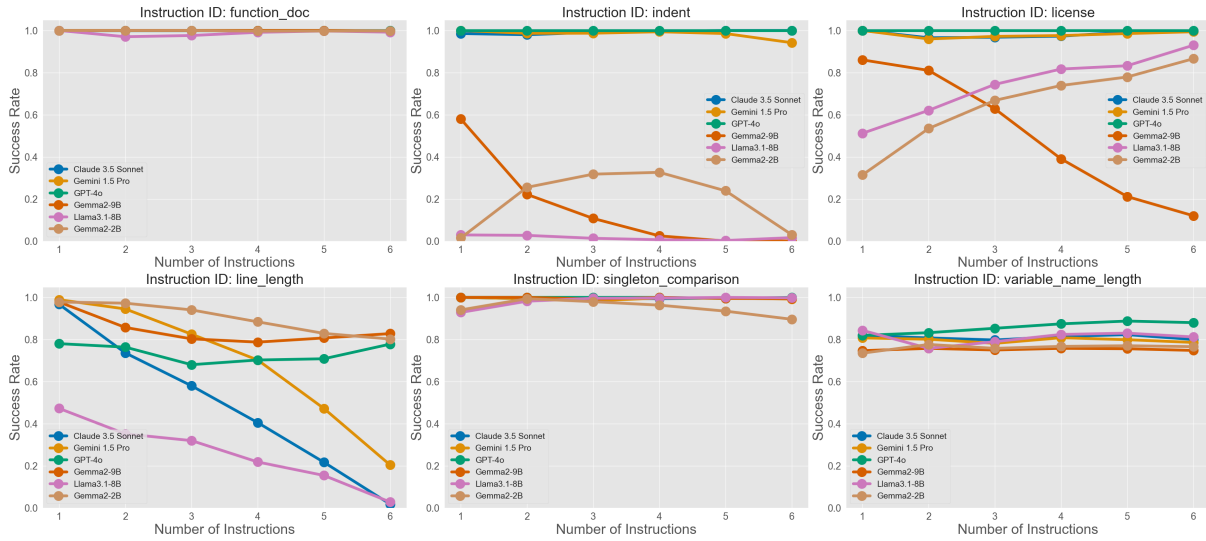


Figure 15: StyleMBPP instruction-following success rate of each instruction.

Logistic (w/ n)	Abs Err (n=6)	Corr (r)
trained $n \leq 3$	0.14 ± 0.22	0.282
trained $n \leq 5$	0.09 ± 0.07	0.924

Table 13: Result of estimation for unseen number of instructions. Absolute error between predictions and empirical result of successfully following all instructions on unseen number of instructions aggregated over six LLMs on StyleMBPP.

Example reasoning trace of DeepSeek-R1 zero-shot for 5 instructions	
Prompt	<p>Write a blog post about the best way to get a good night’s sleep.</p> <p>Your response should follow the instructions below:</p> <ul style="list-style-type: none"> - Your answer must contain exactly 2 bullet points. Use the markdown bullet points such as: * This is point 1. * This is point 2 - In your response, the letter m should appear at least 6 times. - The response must contain at least 2 placeholders represented by square brackets, such as [address]. - Answer with at least 400 words. - Do not include keywords [’reason’, ’review’] in the response.
Reasoning trace	<p><think></p> <p>Okay, the user wants me to write a blog post about the best way to get a good night’s sleep. Let me start by understanding the requirements. They need exactly two bullet points using markdown, the letter ’m’ has to appear at least six times, include two placeholders in square brackets, and the total should be over 400 words. Also, I must avoid the words ’reason’ and ’review’.</p> <p>... (omitted for brevity) ...</p> <p>Let me think of words with ’m’—maybe ’mindfulness’, ’environment’, ’calming’, ’moment’, ’maximum’, ’minimize’.</p> <p>... (omitted for brevity) ...</p> <p>Placeholders: For the first bullet, maybe [specific time] and in the second, [ideal temperature].</p> <p>... (omitted for brevity) ...</p> <p>First bullet: Consistent schedule.</p> <p>Second bullet: Environment. Mattress, lighting, temperature, mindfulness.</p> <p>... (omitted for brevity) ...</p> <p>Okay, that should meet all requirements. Now write it out, keeping paragraphs concise but detailed enough for the word count.</p> <p></think></p>

Table 14: Example reasoning trace of DeepSeek-R1 zero-shot for 5 instructions. Tested on ManyIFEval.

Model	Standard Benchmarks			Multiple Instructions	
	MMLU	HumanEval-Mul	IFEval	ManyIFEval (n=10)	StyleMBPP (n=6)
DeepSeek-V3	88.5	82.6	86.1	0.19	0.01
GPT-4o	87.2	80.5	84.3	0.21	0.68
Qwen2.5-72B	85.3	77.3	84.1	0.02	0.11

Table 15: Performance comparison across standard benchmarks and multiple-instructions-following benchmarks. While models show comparable performance on standard benchmarks (MMLU for general knowledge, HumanEval-Mul for coding, IFEval for single instruction following, results from [DeepSeek-AI \(2025b\)](#)). However, our evaluation on benchmarks reveals substantial performance gap.