Assessing Instruction Following Capabilities of Large Language Models

Piyushi Goyal and **Garima Singh** and **Baidyanath Kundu**
pgoyal@ethz.ch, garsingh@ethz.ch, bkundu@ethz.ch
D-INFK
ETH Zurich
Switzerland

Abstract

LLMs are increasingly used for problem-solving tasks such as coding with programming language constraints and text translations. These tasks rely heavily on a model's instruction-following capability, which is how well the model adheres to given instructions. Factors like question difficulty, the number of extra instructions, and how they are given to the LLM can affect this capability, so understanding these effects is important for creating effective prompts. In this work, we aim to assess the instruction-following capabilities of popular LLMs and identify effective prompt-engineering techniques to improve instruction-following accuracy. To this end, we systematically evaluate the instruction-following abilities of five popular LLMs using four instruction-annotated datasets along with two evaluation metrics. Overall, we find that the number of additional instructions is more detrimental to the instruction-following accuracy than the difficulty of the underlying prompt. Finally, we present analyses based on re-prompting and instruction reordering to test the effectiveness of prompt-engineering techniques, aiming to enhance accuracy solely through changes in user input.

1 Introduction

Large Language Models (LLMs) are becoming increasingly popular due to their exceptional problem-solving abilities. Nevertheless, it has been noted that when queries are complex or contain a large amount of irrelevant information, these models also have a tendency to provide inaccurate responses (Zhou et al., 2023). This can be especially detrimental if these LLMs are deployed in the real world as often the answers they provide (even if incorrect) come across as extremely confident and convincing (Kaddour et al., 2023). The first step in remedying these issues is to understand the inherent instruction-following abilities of these LLMs and identify scenarios in which they fail. This information can then be further analyzed to develop better fine-tuning methods or more effective prompt engineering techniques to unlock the full potential of these models.

In this project, we aim to perform a comprehensive study on the instruction-following capabilities of some popular LLMs over different datasets and evaluation metrics. Our goal is to investigate how well LLMs follow instructions and what factors influence their accuracy. We also investigate 3 different prompt-engineering techniques and comment on their effectiveness through experimental results. [1]

2 Related Work

A considerable amount of work has already been done for this problem statement. People have tried to evaluate the performance of LLMs when they are presented with easy questions in a twisted framing of the language. One such work is Qin et al. (2024), where the authors propose a novel evaluation metric that decomposes complex instructions into a series of output constraints. These constraints are then used to score LLM responses over a diverse dataset of instruction prompts spread over 72 domains. While their work presents good insights into the comprehension abilities of language models, the size of the dataset they evaluate on is rather limited and, hence, is not sufficient to draw general conclusions. Another work in this domain is by Zhou et al. (2023). Here, the authors identify a set of verifiable (quantitative) instructions that can be used as a part of the original prompt. The response to these questions can then be evaluated using a score that can be calculated for each response. The authors also suggest a further categorization of the accuracy of generated responses. This work presents a simple and concrete metric to evaluate

[1] All code can be found here on our github repo.

performance; however, the paper only presents a limited number of general instructions that may not always apply to check for more domain-specific correctness. Chia et al. (2023) present a more comprehensive evaluation suite in order to assess how well fine-tuned models do on a subset of instruction domains. While their analysis is more holistic compared to other works, they majorly use automated evaluation metrics, which may often not reflect human annotations (Zeng et al., 2023).

In contrast, there has also been some work to automatically develop more complex instructions from an initial stub prompt (Xu et al., 2023). These complex prompts can then be used to fine-tune LLMs for instruction-following. The authors of this work report impressive performance of the model fine-tuned using this approach. However, they only evaluate the resulting responses using an automated GPT-based evaluator and, hence, suffer from some of the issues outlined by Zeng et al. (2023). Zeng et al. (2023) build meta-evaluators in order to identify shortcomings in the response scoring functions. While human correctness annotations are typically considered the most relevant and accurate, they can be costly and time-consuming. Hence, it is important to identify automated evaluators that are not only efficient but also sound. In this work, we aim to provide a more comprehensive view of LLMs instruction following capabilities while keeping in mind the shortcomings of the evaluation metrics we use.

3 Experiments and Results

We divide our work into 5 experiments performed over 4 diverse datasets using 5 of the top-performing closed and open-source LLMs. In this section, we first establish the context of our experiments by discussing the LLMs and evaluation metrics used, then we give a brief overview of the datasets we use, and finally, we discuss each of our experiments/analyses in detail:

3.1 LLMs Analysed

We use 5 different LLMs for our experiments: 2 closed source, 2 open source, and 1 open weight. These were GPT4 (`gpt-4-turbo-2024-04-09`) (OpenAI et al., 2024), GPT-4o (`gpt-4o-2024-05-13`) (OpenAI, 2024), Llama3-8B-Instruct (AI@Meta, 2024), Mistral-7B-Instruct-v0.2 (Jiang et al., 2023) and Gemma-7B (Gemma Team et al., 2024) respectively. For the GPTs, we use the OpenAI API, and for the open-source/weights models (hereby referred to as open models), we use the Ollama API (Ollama, 2024).

3.2 Evaluation Metrics

We mainly use 2 different evaluation metrics - IFEval (Zhou et al., 2023), and InfoBench (Qin et al., 2024). The following subsections briefly discuss each metric and how it works. Further, Fig. 1 gives an overview of each of the metrics discussed.

IFEval. This metric relies on atomic and verifiable instructions that can be checked purely algorithmically. Each instruction is mapped to a 'checker' class that implements a function to check whether the given instruction is being followed or not. Since this checking is algorithmic, it only supports a limited set of instructions.

InfoBench. Unlike IFEval, InfoBench uses an LLM (called a Judge LLM) to assess whether the given response follows all the given instructions using a set of 'decomposed' questions. These decomposed questions are just a way to ask the Judge LLM whether an instruction is being followed or not. For each instruction in the given prompt, the Judge LLM gives its judgment through a 'YES' or 'NO' response. If the Judge LLM does not respond in the aforementioned way, it is re-fed the prompt to try again. Because InfoBench uses a Judge LLM that typically has more context on the prompt and response, it can handle much more nuanced instructions than IFEval. In our experiments, we use Llama 3 as a judge for all LLMs except itself (here, we use mistral instead) unless stated otherwise.

As noted in Fig. 1, both evaluation metrics return a list of 'following' judgments for each instruction in the prompt. Hence, there are two ways to look at the accuracy of an LLM being evaluated. One way is to only count the prompts that follow all the instructions (we also call this the ***full* accuracy**). Another way is to only count the number of instructions being followed (we call this the ***partial* accuracy**). In all our experiments, we report the *full* accuracy unless stated otherwise.

3.3 Datasets

In total, we use 4 datasets from different domains. A brief overview of them is as follows:

MMLU by Hendrycks et al. (2021). Here, we only take the math subset of MMLU. This dataset primarily consists of multiple choice questions

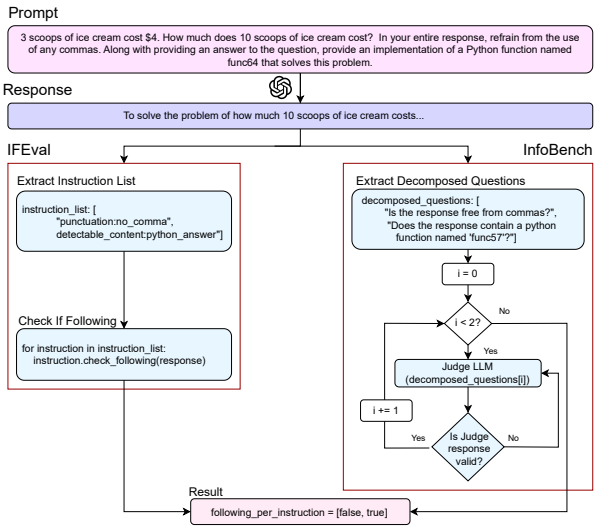

Figure 1: **Overview of IFEval and InfoBench**. This figure illustrates how the different evaluation metrics handle an example set of prompt and response.

from mathematics across various levels of difficulty - Elementary (I), High School math and statistics (II), Abstract Algebra (III), and College (IV). For our experiments, we sample 100 questions from different levels to use in our analysis. We only select questions that are valid standalone (i.e., they do not require specification of answer choices). For one of the mentioned analyses (complexity analysis), we further sample $\approx$ 50 prompts per category/difficulty level.

IFEval dataset by Zhou et al. (2023). This dataset comes from the set of prompts used to test the IFEval framework. This dataset contains general and conversational questions, such as asking the LLM to write a poem, plan an itinerary, etc. To lower computational costs, we sampled 100 prompts from the original dataset for our use. Here, each prompt comes with a preset list of instructions.

InfoBench dataset by Qin et al. (2024). This is fairly similar to the last dataset, wherein it mainly consists of general conversational questions. Here too, we sample 100 prompts for use in our experiments. Each prompt comes with a preset list of instruction/decomposed questions.

Mathwell by Christ et al. (2024). Lastly, this dataset consists mainly of elementary-level math word problems where the solvability factor is annotated for each question. We derive this dataset from Mathwell's testing samples, and out of these samples, we pick 100 prompts at random to use in our experiments.

For the math datasets described above, we create a set of annotated datasets where each prompt is randomly annotated with a number of instructions (between 1 and 5) from the IFEval list of verifiable instructions. Moreover, we complement the existing instructions with a set of math-specific verifiable instructions summarised in table 1. These instruction-annotated prompts are then processed to create another dataset compatible with InfoBench, where each occurrence of instruction is appended with a question to check its following. These questions are used by the InfoBench evaluation script to check if the instructions are being followed. We discuss the annotations in detail in the following section.

3.4 Experiments

In order to assess the instruction-following capabilities of LLMs, we conduct 2 different analyses. First, we use IFEval and InfoBench to evaluate the instruction annotated prompts across the different datasets to get an initial look at each LLM's accuracy. Second, in order to further understand where LLMs fail, we try to look at the effects of prompt complexity vs. instruction complexity on accuracy.

We further perform 3 different experiments to assess different prompt-engineering techniques in order to improve the instruction-following accuracy of LLMs. First, we analyze whether correcting and reprompting (rather than re-feeding the same prompt) leads to better following accuracy. Then, we assess whether reordering the instructions helps with accuracy. Lastly, we gauge whether prompting the LLM step-by-step leads to better instruction following. In the following sections, we discuss each of the aforementioned analyses in more detail.

3.4.1 IFEval and InfoBench Evaluations

In this experiment, we aim to assess two qualities relating to the instruction following capabilities of LLMs. First, how do different LLMs perform across domains, and second, how well are the evaluation metrics able to asses this performance. To do so, we setup 2 different benchmarks as described below:

InfoBench to IFEval. InfoBench is mainly aimed at assessing the qualitative properties of a response. In turn, most instructions within the InfoBench datasets are fairly qualitative and are not directly convertible to IFEval-style instructions. For example, to convert to IFEval, we only have a limited set of qualitative instructions we can retain.

Instruction	Description	Parameters
Answer Steps	Instructs the LLM to respond in a step-by-step manner following a specific formatting	Minimum and maximum number of steps
Answer Highlight	Instruct the LLM to (only) highlight the final answer in bold using markdown	-
Answer Equation	Instructs the LLM to represent the given question as a Latex style mathematical formula formatted in a specific way.	-
Answer Round	Instructs the LLM to either round or truncate the answer to a specific number of decimal places. The prompt needs to result in a numeric, decimal answer.	Formatting type (round or truncate), number of decimal places to format
Python Function	Instructs the LLM to generate a Python function with a specific name that solves the given problem.	Name of the python function

Table 1: **Verifiable instructions for math datasets.** Here, the instruction refers to the name of the instruction, the description gives a brief overview of what the instruction is meant for, and parameters define the aspects of the instruction that are dynamic, i.e., different across multiple instantiations of the same instruction.

One example of such an instruction can be "list down 6 things..." where we can use the 'list' quality to generate a verifiable instruction that checks whether the answer is formatted as a list or not. Another verifiable instruction here could be that there must be exactly 6 or ≥ 6 bullets/list items in the response. To make up for the instructions that cannot be converted, we additionally supplement each prompt with 1 or 2 IFEval-compatible instructions.

IFEval to InfoBench. The next step is to understand how and if the atomic, verifiable instruction from IFeval can be ported to be compatible with the InfoBench evaluation metric. At the surface level, most instructions from IFEval can directly be converted to InfoBench through a mapping between the instruction type and the decomposed question. For example, the 'Answer Highlight' instruction from Tab. 1 can be mapped to the following decomposed question – "Is the final answer (and the final answer only) in bold?". Another example can be found in Fig.1, where we show the same prompt and instruction set for both IFEval and InfoBench. Further, we can also add more qualitative instructions. For example, if a prompt says, 'Write a funny poem.', we could generate a decomposed question to determine whether the generated poem is actually funny or not (which isn't possible in IFEval).

Here, we used ChatGPT to interconvert the pre-provided prompts of the IFEval dataset to decomposed questions compatible with InfoBench (and vice-versa). Moreover, to level the comparison, we generated a minimum of one qualitative instruction per prompt in the 'IFEval to InfoBench' dataset (along with the existing verifiable instructions). Tab. 2 gives an overview of results from the aforementioned datasets across the various models.

We observe that the GPT4 family greatly outperforms the open counterparts. Within the open models, Llama 3 performs the best, followed by Gemma and Mistral. It is important to note here that since all the datasets (or instances of datasets) contain slightly different instructions, they are not comparable, i.e., we cannot compare the evaluation metrics here directly. However, we do this in the following sections by fixing the instructions between the datasets to be the same.

InfoBench vs. IFEval. As mentioned in the previous section, the next step is to understand how both different metrics compare to each other. To do this, we use our annotated math datasets and evaluate them using both IFEval and InfoBench. The instructions per prompt for both evaluation metrics are the same (i.e., no qualitative instructions) to make sure we make a fair comparison.

Why InfoBench on verifiable instructions? While we do realize that InfoBench was primarily

created to evaluate qualitative instructions, it can be very useful in examining quantitative instructions as well. For instance, algorithmic metrics such as IFEval are very generic and, hence, limited in their ability to check if an instruction is truly being followed or not. For example, the 'Answer Highlight' instruction from Tab. 1 directs the LLM to generate a response in which the answer to the question is in bold; however, we can only check for the existence of the bold formatting and not the content within that. For example, the LLM might embolden a phrase that is not the answer, and IFEval would still consider that instruction to be followed within the given scheme. A judge LLM, on the other hand, might be able to identify this issue and mark that instruction as not being followed. Moreover, IFEval does not exhaustively capture all possible verifiable instructions; hence, an AI-based system here would also benefit scalability.

Tab. 3 gives an overview of the results. Here, we can see that IFEval is more optimistic in its assessment of instruction-following than InfoBench. While it may seem like IFEval is more 'lenient', this is not actually the case. IFEval leads to higher accuracy because it is closer to the true measure of instruction-following when compared to InfoBench. We do not see this trend in the previous sections because – a) the previous datasets contain fewer instructions per prompt on average and b) the InfoBench versions contain more qualitative questions than quantitative ones.

InfoBench also suffers from several issues when it comes to quantitative/verifiable instructions. One major issue is that the Judge LLM is unable to follow the decomposed questions correctly. For instance, for the decomposed question 'Does the response contain 1 paragraph separated by the markdown divider: ***?' and a response like: '<response text> -new line character- <response text>', the Judge LLM classifies this instruction as not followed only because technically there are multiple paragraphs (because of the newline character) in the response (even though the decomposed question specifies that paragraphs are separated by a special symbol). There are other such instances of the LLM not being able to fully follow the decomposed questions (which we omit here in the interest of saving space) and this plays a big part in bringing down the accuracy reported by InfoBench.

Another aspect where InfoBench-style metrics falter is that the accuracies or judgments they produce on the instruction-following of a particular LLM are only as good as the underlying Judge LLM. For example, for a decomposed question, 'Is the response free from any commas?', even if the response is truly free of commas, the Judge LLM (here Llama 3) hallucinates and reports that there are commas throughout the response and hence, classifies this instruction as not being followed. Moroever, this is repeated several times in the evaluation (in other instructions as well), affecting the final accuracy.

To conclude this section, InfoBench suffers from several issues, which leads it to report lower instruction-following accuracy across several models. While InfoBench can be a great metric to evaluate the qualitative or intangible aspects of instructions, it is not (yet) suitable to replace or integrate with algorithmic metrics such as IFEval. Therefore, for the following experiments, we mainly use IFEval as the evaluation metric (unless stated otherwise).

3.4.2 Prompt vs. Instruction Difficulty

Now that we have an overview of how various LLMs perform across different datasets, the next natural step is to ask what makes these LLMs fail. While this is a difficult question to answer, and there can be various (valid) reasons/explanations, in this experiment, we try to examine 2 aspects that make a prompt and instruction set 'difficult' for an LLM to follow.

The first aspect we assess is the inherent difficulty of the prompt. For example, a prompt dealing with mathematical concepts from the elementary level is going to be easier to process than a prompt dealing with mathematical concepts at the university level. Following this example, we use a similar logic in determining the difficulty of prompts. We use our MMLU sub-dataset and determine the difficulty of each prompt by the category it belongs to (elementary math, college math, etc.). The second aspect we assess is the difficulty or complexity of the instructions. Since this is fairly hard to quantify, we use the number of instructions per prompt as the difficulty/complexity of instruction. Here, more instructions to one prompt implies higher instruction complexity.

Tab. 4 summarises the results of this analysis across 4 models. We can clearly see that instruction complexity has a higher effect on the accuracy of instruction following than the difficulty of the

Models	IFEval	InfoB to IFEval	IFEval to InfoB	InfoB
GPT4	67	64	61	74
GPT4o	66	51	63	80
Llama 3	52	57	56	69
Mistral	53	43	54	46
Gemma	63	45	54	52

Table 2: **Accuracy of different LLMs over InfoBench and IFEval datasets.** Here, the first 2 columns are evaluated using IFEval and the other 2 using InfoBench. We see that GPT4o performs the best overall, closely followed by GPT4.

Models	Mathwell		MMLU	
	IFEval	InfoB	IFEval	InfoB
GPT4	65	24	47	22
GPT4o	56	28	56	20
Llama 3	42	19	31	14
Mistral	24	22	12	13
Gemma	25	20	21	13

Table 3: **Accuracy reported by the different evaluation metrics over the math datasets.** Here, we see that IFEval is more optimistic compared to InfoBench, leading to a higher (and more precise) accuracy rate.

underlying prompt, even for less complex instruction. Even in between levels, accuracy drops faster as the instruction complexity increases as opposed to question difficulty. As for specific LLMs, the GPT family performs the best by far, followed by Llama 3, then Gemma and Mistral.

Seeing how increasing instruction complexity (number of instructions here) leads to a quicker decline in performance, a natural next step is to understand how to prevent or even reduce this decline. In the following sections, we perform some experiments to test out 3 different prompt engineering techniques in order to improve the instruction following capabilities of LLMs.

3.4.3 Guided vs. Blind Reprompting

One idea to improve the instruction following rate is to determine the best way to get the LLM 'back on track', i.e., prompt the LLM to fix its response to follow certain instructions. While there can be several ways to do this, we compare 2 – Guided and Blind reprompting. Guided reprompting involves prompting the LLM to fix the specific instructions it did not previously follow. Blind reprompting refers to re-feeding the same initial prompt back to the LLM multiple times, allowing it to recognize the issues itself. Fig. 2 illustrates this concept with an example. We reprompt a maximum of 5 times (beyond which instruction following is marked fail) or till the LLM's response follows all instructions.

Tab. 5 describes the results of this analysis for 2 of the most popular open-source LLMs. Here, we see that guided reprompting has marginal or no benefits over blind reprompting. While we are not fully sure why, we believe the additional context from the guiding could increase the instruction complexity and thereby reduce the performance benefits. Compared to no reprompting (or only

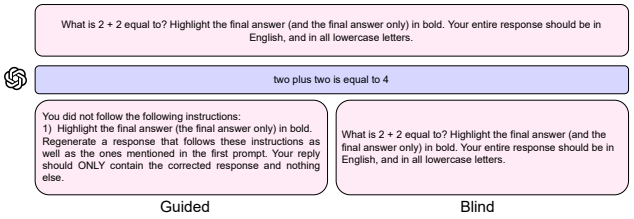

Figure 2: **Blind vs. guided reprompting.** This figure illustrates the difference in how a given LLM is reprompted within the two reprompting schemes we discuss here.

regenerating from scratch), both methods perform significantly better for Llama 3. This implies that the added context from past responses (and the reprompts) somehow helps the LLM to follow more instructions. For Gemma, the benefits of reprompting are limited.

3.4.4 Instruction Reordering

Another idea to improve instruction following is understanding whether certain relative orderings of the instructions affect accuracy. For this experiment, we try to assess the accuracy of instruction following over all the possible combinations of the added instruction.

To limit the complexity, we generate another annotated dataset using Mathwell such that each prompt has exactly 3 instructions. We then generate 6 different instances of the same annotated dataset where the instructions for each prompt are reordered (i.e., each instance contains one combination of instructions per prompt). We then generate the responses over all 6 files and evaluate them. After this, we calculate the maximum difference in instructions followed per prompt across all 6 instances of our dataset. For example, for 3 instructions per prompt, we have 4 possible max differences in instruction following – 0 (no differ-

Models	Prompt Difficulty	Instruction Difficulty					
		1	2	3	4	5	6
GPT4o	I	90 ±1	73 ±5	54 ±5	43 ±3	31 ±3	18 ±4
	II	86 ±2	62 ±5	50 ±5	38 ±3	25 ±3	17 ±4
	III	87 ±3	62 ±4	45 ±4	30 ±1	17 ±3	13 ±2
	IV	75 ±3	57 ±3	32 ±1	17 ±3	11 ±2	8 ±1
Llama 3	I	79 ±2	53 ±3	44 ±4	24 ±4	17 ±6	10 ±2
	II	83 ±4	53 ±4	37 ±5	21 ±4	11 ±2	7 ±4
	III	73 ±4	50 ±4	33 ±6	15 ±2	11 ±3	6 ±1
	IV	72 ±1	41 ±4	19 ±4	6 ±2	4 ±2	2 ±2
Mistral	I	75 ±6	36 ±5	24 ±2	14 ±1	9 ±2	2 ±2
	II	71 ±5	39 ±4	21 ±3	10 ±4	3 ±2	2 ±3
	III	65 ±2	37 ±3	18 ±5	6 ±2	5 ±2	2 ±2
	IV	60 ±2	26 ±5	5 ±2	3 ±1	2 ±2	0 ±1
Gemma	I	79 ±2	37 ±3	33 ±4	20 ±1	11 ±3	6 ±2
	II	68 ±1	42 ±3	27 ±3	15 ±3	6 ±3	3 ±2
	III	63 ±3	42 ±2	19 ±3	5 ±1	5 ±2	3 ±2
	IV	58 ±4	26 ±2	8 ±1	0 ±0	1 ±1	0 ±0

Table 4: **Evaluation of effects of instruction vs. prompt difficulty on *full* accuracy for several LLMs using IFEval metric.** In this table, we report the mean ± std. deviation over 5 runs. Each difficulty level contains 46 data instances (we subsample here as not all prompts are compatible with more than 2/3 instructions). We can clearly see that increasing instruction complexity has a much larger effect on 'following' accuracy as compared to increasing the inherent difficulty of the question. We also perform this analysis using the InfoBench evaluation metric but due to it's inefficiency discussed in earlier sections, we do not present it here.

LLMs	Reprompting		
	None	Blind	Guided
Llama 3	40 ± 4	67 ± 2	72 ± 2
Gemma	23 ± 3	26 ± 2	35 ± 3

Table 5: **Full accuracy for various reprompting techniques.** We use the annotated Mathwell dataset and report the full accuracy for the various reprompting techniques. Here, we report the mean ± std. deviation over 5 runs.

ence in following), 1 (at most one instruction is followed/not followed), 2 (at most 2 instructions followed/not followed) and 3 (all instructions followed/not followed). We then count the number of prompts per difference 'class' (averaged over 3 runs) and report these numbers in Fig. 3.

Due to high uncertainty, we are unable to come to a conclusion on whether instruction reordering actually helps. It seems that reordering has close to no effect on the *partial* accuracy. It is also possible that 3 instructions per prompt are simply too low to see the actual benefits of reordering. Due to time and computing constraints, we leave these investigations for future work.

3.4.5 Step-by-step Prompting

Lastly, we explore how differences in feeding instructions to the LLM can affect accuracy. In this experiment, we specifically consider two broad ways of providing instructions to LLMs – all together or step-by-step. In the former, we simply append all instructions to the base prompt (as seen in earlier examples) and then feed this to the LLM. In the latter, we split the instructions over different 'steps' (each step constitutes one instruction) and then feed these and the base prompt to the LLM one by one. We further expand this step-by-step technique by introducing an extra statement: 'Along with this instruction, follow all previous instructions as well' to be appended to every instruction prompt. We call this specific prompting method 'step-by-step (aided)'.

We evaluate the *full* accuracy of these techniques on the 3 open LLMs and present these results in Tab. 6. We observe that step-by-step prompting does not massively increase the *full* accuracy for Llama 3. For models like Gemma and Mistral, step-by-step prompting actually (marginally) decreases the accuracy. Our belief is that the drop occurs because step-by-step prompting requires a larger

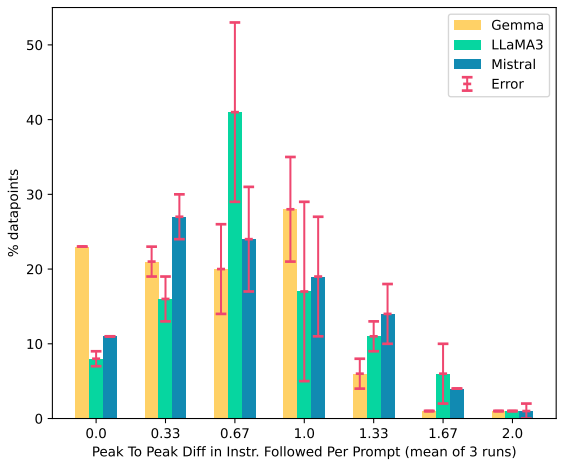

Figure 3: **Effect of reordering on instruction following.** For each prompt, we take the best and the worst performing order and get the difference in their performance (in terms of the number of instructions followed). This is then averaged over three runs and plotted above. The error bars represent the points with std. dev. more than the difference in bin centers.

context window.

Model	All at once	Step-by-step	Step-by-step (aided)
Llama 3	42 $\pm$3	48 $\pm$3	50 $\pm$1
Mistral	29 $\pm$1	15 $\pm$2	18 $\pm$1
Gemma	25 $\pm$0	12 $\pm$0	11 $\pm$2

Table 6: **Comparison of step-by-step prompting vs. all at once.** Here, we report the mean $\pm$ std. deviation over 3 runs. We observe little to no difference between the accuracy of the different prompting methods.

4 Limitations

As with every study, our work also has a couple of limitations listed as follows:

- **Data and Runs:** Due to compute and time constraints, we only present our analyses on smaller datasets and only report averages across 5 runs in most of the places. Since LLMs are highly variable and non-deterministic, we might have to scale our analyses even more to get more substantial results.
- **Analyses Parameters:** Due to time constraints, we were only able to try a few parameters for the analyses we performed. For example, in the reordering analysis, we only evaluated prompts with 3 instructions each. The results of the analysis may be different if evaluated on other instruction complexities/number of instructions.

5 Conclusion

In this work, we present a comprehensive look into the instruction following capabilities of popular LLMs across various domains. We also present a set of math-compatible verifiable instructions and use them to comment on the evaluation metrics used. We try to show how certain evaluation metrics that use LLMs as a Judge, can be unnecessarily harsh and inaccurate compared to programmatic methods. Further, to understand where LLMs fail in following instructions, we compare the effects of instruction complexity vs. inherent question complexity on the instruction-following accuracy of various LLMs. Here, we show that following accuracy is more heavily affected by the instruction complexity, even for small increases. Additionally, we also explore several prompt-engineering techniques in order to improve the instruction-following accuracy of LLMs. We cover 3 different techniques and provide insights through experiments on if and how well they work.

In conclusion, we present a detailed evaluation of the instruction following capabilities of modern LLMs and also assess the effectiveness of different prompt engineering techniques.

6 Future Work

Although in this project we only provide a small set of proof of concepts, we believe our work can be greatly extended through follow-up/future work. One such work could be scaling up the presented analyses here to more massive datasets; this would provide a more solid look into the instruction following capabilities of LLMs. Moreover, other prompt engineering techniques can be explored and possibly reasoned about. This would not only help the community better understand how LLMs process instructions but also help in maximizing the LLM's full potential. Another interesting aspect of research could be understanding the best ways to rework instructions in order to maximize following.

References

AI@Meta. 2024. Llama 3 model card.

Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. 2023. Instructeval: Towards holistic evaluation of instruction-tuned large language models.

Bryan R Christ, Jonathan Kropko, and Thomas Hartvigsen. 2024. Mathwell: Generating educational math word problems at scale.

Thomas Mesnard Gemma Team, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, and et al. 2024. Gemma.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*.

Ollama. 2024. Ollama.

OpenAI. 2024. Hello gpt-4o.

OpenAI et al. 2024. Gpt-4 technical report.

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.

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

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2023. Evaluating large language models at evaluating instruction following.

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.