

Beyond Single Commands: Evaluating LLMs on Multiple Instruction Sequences

Stanford CS224N Custom Project

Sagnik Bhattacharya
sagnikb@stanford.edu

Vaastav Arora
arora74@stanford.edu

Prateek Varshney
vprateek@stanford.edu

Abstract

Rapid advancements in large language models (LLMs) have significantly improved their capabilities in NLP tasks. However, these models often struggle with practical real-world scenarios that require following complex, multi-instruction sequences in a single pass. To quantify this capability and address a critical gap in current LLM evaluation methods, we propose a novel benchmark to evaluate LLMs' performance on their ability to adhere to multiple instructions within a single inference call. We present a synthetically generated dataset covering two multi-instruction formats (multi-part and multi-step), three task domains (coding, open ended text generation, and causal inference), and two prompt subcategories (with and without context). Our evaluation metrics focus on the count of instruction adherence and the first occurrence of non-compliance. Our contributions are two-fold: (1) We demonstrate a scaling law of multi-instruction adherence for LLMs of varying sizes and training regimes. (2) We empirically verify our hypothesis that mainstream LLMs have not been trained on datasets derived from multi-instruction text samples.

1 Key Information

- TA mentor: Tony Lee
- External collaborators: No
- External mentor: Harshit Joshi
- Sharing project: No
- Github Code: Shared with mentor (<https://github.com/sagnik119/HybridInstructionLLM>)

2 Introduction

In recent years, the rapid advancements in large language models (LLMs) have significantly enhanced their capabilities across a wide array of natural language processing (NLP) tasks. These models, including GPT-4 (Brown et al., 2020), PaLM (Chowdhery et al., 2022), and Claude (Enis and Hopkins, 2024), have demonstrated impressive performance in generating coherent and contextually relevant text, and even engaging in complex dialogues. However, a critical gap remains: these models often struggle with practical real-world scenarios that require following complex, multi-instruction sequences within a single inference call. This limitation is particularly problematic given the increasing demand for LLMs to perform tasks that involve multiple sub-tasks or criteria simultaneously, such as generating a comprehensive report from a set of detailed instructions or executing a series of related commands provided in a zero-shot manner to the model.

The primary challenge lies in the inherent training objectives of LLMs. These models have been primarily trained on tasks involving in-context next word prediction based on large corpora (Ouyang et al., 2022), which often misaligns with the objective for the models to follow complex and massive "human" instructions. The emerging field of instruction tuning (IT) addresses this by training LLMs on specific (Instruction, Output) pairs, aiming to bridge the gap between traditional language

modeling and practical instruction-based applications. However, since such IT approaches are directed at fine-tuning LLMs to follow a single instruction per inference call, their ability to follow complex instructions that necessitate handling multiple sub-tasks remains dubious. Existing methods, including multi-hop reasoning and multi-turn conversation (Ho et al., 2020; Duan et al., 2023) evaluation suites, primarily focus on sequential or iterative task execution but do not adequately address the complexity of simultaneous multi-instruction tasks. Moreover, evaluating the performance of LLMs in such contexts often involves expensive and time-consuming processes, such as human annotations and fact-checking, which are not scalable. For instance, using professional fact-checkers to manually inspect the outputs of LLMs can be both time-consuming and expensive (Zhang and Gao, 2023).

To address this gap, we propose a novel benchmark designed to evaluate the multi-instruction adherence capabilities of LLMs. Our approach involves the creation of a synthetically generated dataset that encompasses two multi-instruction formats (multi-part and multi-step), three task domains (coding, open-ended text generation, and causal inference), and two prompt subcategories (with and without context). This benchmark is intended to rigorously assess how well LLMs can handle complex instruction sequences in a single inference call, providing a comprehensive evaluation that goes beyond traditional single-instruction tests. To minimize the human-in-the-loop, we use another LLM (which we call the evaluator) of sufficient quality to evaluate the output of the LLM under consideration using single-instruction inference calls.

Our contributions are two-fold. First, we demonstrate a scaling law of multi-instruction adherence for LLMs of varying sizes and training regimes, highlighting the impact of model scale and training methods on their ability to follow complex instructions. This suggests that multi-instruction following ability might be an emergent property. Second, we empirically verify that mainstream LLMs have not been adequately trained on datasets derived from multi-instruction text samples, underscoring a significant area for improvement in current training paradigms. In this project, we introduce our **Multi-Instruction** benchmark, which sets itself apart from existing evaluation suites by providing a variety of multi-instruction prompt templates.

Summary of our Contributions:

- We propose a novel evaluation benchmark and metric that analyzes the single prompt multi-instruction adherence capabilities of Large Language Models. We have fully open-sourced our dataset, the data generation pipeline, and the evaluation pipeline.
- To the best of our knowledge, we propose the first fully-automated evaluation pipeline with minimal human-in-the-loop (which is restricted to creating seed examples for the dataset generation).
- Our evaluation and proposed metrics demonstrate a scaling law: multi-instruction adherence capability scales with model size. Further, closed-sourced models such as Gemini Pro fare better than open-sourced models like Llama 2.

3 Related Work

The evaluation of large language models (LLMs) has traditionally been domain-specific, focusing on particular tasks or datasets. For instance, the works by (Hendrycks et al., 2021) exemplify task-specific evaluations that assess LLM performance within narrowly defined contexts. However, the evolving landscape of natural language processing (NLP) has seen a shift towards a more holistic evaluation of LLMs, assessing high-level capabilities and properties across multiple facets (Liang et al., 2023; He et al., 2024a).

Existing studies such as HotpotQA (Yang et al., 2018) and StrategyQA (Geva et al., 2021) have introduced datasets that require multi-hop reasoning. These datasets are instrumental in training and testing the internal reasoning processes of LLMs. Nonetheless, they do not comprehensively evaluate the accuracy of intermediate steps or the efficacy of concurrent versus sequential processing of tasks. This limitation is significant because real-world applications often involve executing multiple instructions either sequentially or concurrently.

Several researchers have explored the instruction-following capabilities of LLMs across different dimensions. Works by Son et al. (2024), Zhou et al. (2023a), and He et al. (2024b) analyze how LLMs handle various instruction settings, such as multi-step or multi-part instructions, and different types of tasks, including those with language-level constraints and complex real-world scenarios.

The evaluation metrics used in these studies, such as token-level accuracy, perplexity scores, and COPY-BLEU, provide a range of qualitative and quantitative assessments. However, these metrics often rely on human intervention, making them time-consuming and not easily scalable.

The comprehensive survey by Zhang et al. (2024) reviews the methodologies, datasets, and applications of instruction tuning (IT), presenting a critical analysis of current approaches. This survey highlights the efficiency and limitations of existing methods, which primarily focus on single-instruction objectives. Consequently, these methods do not fully address the challenges of handling multiple instructions within a single inference call, an area increasingly relevant in practical applications.

Our work seeks to bridge this gap by proposing a novel benchmark specifically designed to evaluate the multi-instruction adherence capabilities of LLMs. Unlike traditional benchmarks, our benchmark encompasses a broader range of scenarios with two distinct multi-instruction formats (multi-part and multi-step), spanning three task domains (coding, open-ended text generation, and causal inference), and including both context-dependent and context-independent prompts. This comprehensive approach allows for a more rigorous assessment of how well LLMs can handle complex instruction sequences in a single inference call. To minimize the human-in-the-loop aspect, we employ another high-quality LLM as an evaluator to assess the outputs of the LLM under consideration.

In summary, our work builds on and extends existing research by providing a novel benchmark that evaluates LLMs’ ability to follow multiple instructions in a single inference call. This benchmark addresses the limitations of current single-instruction and multi-hop reasoning datasets and contributes to the ongoing efforts to develop more advanced and versatile NLP systems.

4 Approach

Benchmark Features Analogous to the feature design framework established by CELLO Son et al. (2024), we developed our own set of features for constructing our benchmark. We categorize multi-instructions test samples into two task types—multi-step (sequential execution) and multi-part (non-sequential execution)—across three task domains: coding, open-ended text generation, and causal inference. Additionally, each multi-instruction may require context, which can include a guiding text description or a master task or both. We designed these features and criteria to emulate real work scenarios, ensuring that the models are tested in environments that closely mimic practical applications.

$$\text{sample} \in \{\text{multi-part, multi-step}\} \times \{\text{coding, causal inference, text generation}\} \\ \times \{\text{requires context, does not require context}\},$$

Tasks in the Multi-Step subset demand a sequential approach, with the accuracy of each step being vital for the following ones. This subset assesses LLMs’ proficiency in managing interdependent tasks. Conversely, the Multi-Part subset consists of contextually related but independent sub-tasks, evaluating LLMs’ capacity to process multiple, disparate tasks simultaneously.

The task domains within these subsets include coding, open-ended text generation, and causal inference. Coding tasks test the LLMs’ ability to understand and generate code, requiring a strong grasp of syntax and logical structuring. Open-ended text generation assesses the models’ creativity

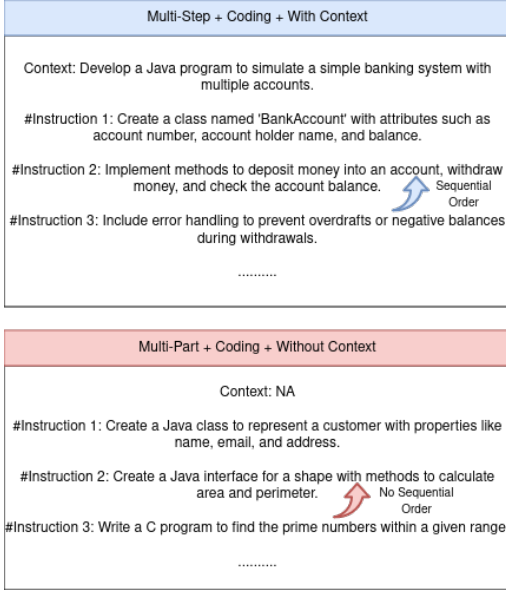


Figure 1: An example that compares Multi-Part and Multi-Step examples in our dataset. Note how Multi-Step requires the Invoker model to solve step-by-step due to a sequential order among the sub-tasks, and how Multi-Part is devoid of any such ordering.

and coherence in producing narrative or descriptive text based on provided prompts. Causal inference challenges the models to deduce relationships and causations from given data, a crucial skill for scientific and analytical applications.

Each multi-instruction task may incorporate context, such as a guiding text description or a master task. Context provides necessary background information, situating the instructions within a coherent narrative or logical framework. This setup assesses the LLMs’ ability to interpret context, which is crucial for understanding nuances, maintaining relevance, and ensuring that tasks are executed accurately and appropriately within a given domain.

Large-scale Synthetic Multi-instruction Generated Dataset Each multi-instruction sample comprises at least 15 instructions (See Appendix A.3 for examples). We first initialize the dataset with 24 seed examples evenly distributed across all categories and then expand the dataset Rajpurkar et al. (2016); Zhou et al. (2023b). The synthetic multi-instruction sample generation pipeline consists of:

1. **Few-Shot Prompting:** A LLM is prompted using examples sampled from the current iteration of the dataset (i.e. seed examples + synthetic examples from previous iterations). Prompts are tailored to specific task types and categories to generate new multi-instructions. (See Appendix A.2 for examples of few shot generation prompt, and Appendix A.3 for the corresponding generated outputs.)
2. **Filtering:** We then eliminate low-quality generated instructions based on inadmissible tasks (e.g., non-language tasks like “draw a diagram”) or high similarity (cosine distance of instruction embedding) to existing instructions. We use voyage-larg2-instruct (Muennighoff et al., 2022) as the embedding model to compute instruction embeddings.
3. **Instruction Augmentation:** (If needed) We generate additional instructions for samples whose instruction counts falls below a threshold, for our experiments, 15, after filtering. This requires prompt tuning for specific category and the number of instructions required as of the current iteration. It takes roughly 2-3 iterations of augmentation to generate complete multi-instructions containing more than 15 instructions each. Examples of prompts provided to LM for augmentation, and the generated instructions, are given in Appendix A.4 and Appendix A.5 respectively.

Test Case Generation The synthetic multi-instructions generated above are used to create "families" of tests. For the MULTI-PART case, we generate a family of test cases from a single multi-instruction by selecting any subset of instructions, where order of instructions is not considered. In a case with N instructions, this would lead us to 2^N test cases. However, for MULTI-STEP case, the order of instructions needs to be preserved. Consequently, for a multi-step multi-instruction with N instructions, we can generate N test cases, where the i -th test case comprises instructions 1 to i .

Model Invocation We designate the LLM responsible for generating responses to our multi-instruction test cases as the "Invoker." To rigorously explore the scaling laws of instruction adherence, we incorporate LLMs with diverse parameter sizes, training methodologies, and instruction-tuning capabilities in our list of Invokers. This approach enables us to systematically investigate how variations in model complexity and training paradigms impact the performance and fidelity of the LLMs in adhering to intricate, multi-faceted instructions.

Model Evaluation We define two evaluation metrics:

1. **Instruction Adherence Proportion (IAP)** This represents the ratio of the number of instructions that were adhered to out of the total number of instructions present in the multi-instruction prompt. It is a ratio measurement and doesn’t emphasize sequential position or nature of the instruction, but provides a quantitative measure of how many instructions the model was able to handle correctly.
2. **First Instruction Deviance (FID)** This represents the position of the first instruction within the list of instructions that was not adhered to by the model. This facilitates our analysis of multi-step instruction prompts by pinpointing the specific step in the sequence where the model did not comply with instruction following.

Given that we want to minimize the human in the loop aspect due to its inability to scale, we use a state-of-the-art LLM, specifically gemini-1.0-pro-002, as the evaluator with the expectation that its assessment will be closest to that of a human evaluator.

5 Experiments

5.1 Data

For our experiment, we manually generated 24 seed examples of multi-instructions, evenly distributed among the 12 categories. Each seed example contains at least 15 instructions. To remove instructions similar to existing ones, we used embeddings generated by the voyage-large-2-instruct model and calculated cosine similarities. We set a similarity threshold of 0.9, beyond which we removed the instruction. This threshold was derived experimentally to achieve an optimal balance between semantic meaning and token-based similarities.

We conducted 14 sequential iterations for generations, all running simultaneously for 12 multi-instruction categories. The model used for generation was GPT-3.5, with a temperature set to 0.7, empirically shown to provide the best balance between diversity and rule adherence for the generated multi-instructions. The final generated and filtered dataset comprises 1008 multi-instructions, distributed equally among the 12 categories. Each generated synthetic multi-instruction comprises 15-50 instructions. However, for our test cases, we use this synthetic dataset to generate variable length multi-instructions by selecting any subset of instructions, as explained in 4. Thus we have multi-instructions with number of instructions varying between 1 to 50 for our experimental purposes.

5.2 Experimental details

We benchmark a diverse set of LLMs covering the LLaMA 3 family chat models (Llama-3-8b-chat-hf, Llama-3-70b-chat-hf), Mistral family models (Mixtral-7B-Instruct-v0.3, Mixtral-8x7B-Instruct-v0.1, Mixtral-8x22B-Instruct-v0.1), Microsoft Phi 2 (2.7B), and Qwen 1.5 chat (0.5B). Additionally, we incorporated GPT-3.5-turbo-0125 and Gemini-1.0-pro-002, although their specific sizes and technical specifications were not available for inclusion in our detailed analysis.

All models, except for GPT-3.5-turbo-0125 and Gemini-1.0-pro-002, were executed using Together AI's inference API¹. For GPT-3.5-turbo-0125, we leveraged the Chat Completion API provided by OpenAI, while for Gemini-1.0-pro-002, we utilized the Vertex AI inference API through Google Cloud Platform. In each instance, we adhered to the default configuration parameters (e.g., temperature) established by the respective providers. No additional constraints regarding time or token length were imposed.

To facilitate the Invoker's task, we design a minimalistic prompt that includes essential details describing the multi-instruction format (MULTI-PART and MULTI-STEP), task domain (coding, open-ended text generation, and causal inference), and context when present. This minimalistic prompt is engineered to provide the bare minimum necessary for the LLM to initiate its response generation. Our objective is to evaluate the performance of the LLM when it is given only the most essential information, thereby assessing its capability to handle complex instructions without extensive guidance (See Appendix A.6 for examples).

5.3 Evaluation method

Adhering to the paradigm of using one LLM to evaluate the responses of another in a scalable manner, the evaluator (gemini-1.0-pro-002) is provided with the complete response generated by the original ("Invoker") LLM, along with each individual instruction from the multi-instruction prompt. The evaluator determines whether each instruction has been adhered to, and outputs a true or false verdict for each instruction. Consequently, we obtain a list of true or false values for each instruction in every test case. These values are then synthesized to produce the IAP and FID values.

Given that our multi-instruction tests can belong to any of the 12 categories, we develop a custom prompt for each category combination. These prompts explicitly describe the instruction format, domain, and context to the evaluator LLM, ensuring it is fully informed for its evaluation task.

¹www.together.ai

Examples of these custom prompts can be found in Appendix A.7. The evaluator LLM is instructed to output a 'T' or 'F' as its final character to indicate a true or false assessment. However, due to occasional inconsistencies in its output, we employ a series of string-matching algorithms to accurately extract the evaluator's answers, informed by common formatting irregularities observed during our experiments. Please refer to Appendix A.8 for examples of the evaluator's output.

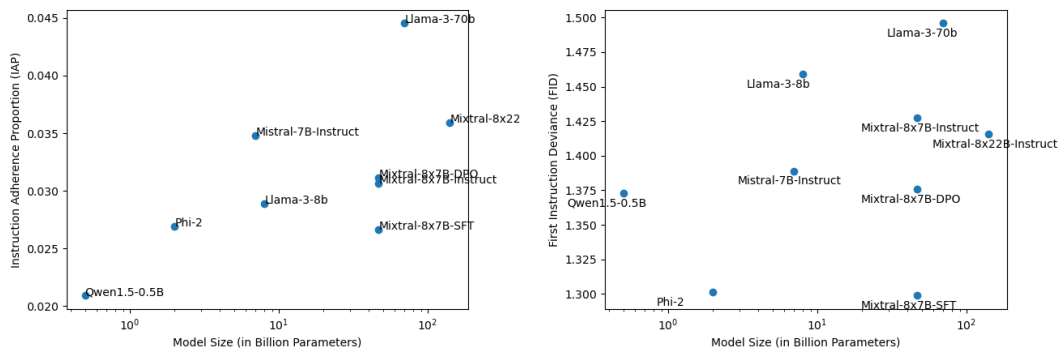
5.4 Results

Figure 2 demonstrates the 0-shot multi-instruction following capabilities of different LLMs. While there is an increase in task-adherence capability of LLMs as we increase the model size, it manifests more visibly in IAP than the FID value. We can see a strong correlation between model size and IAP and no significant correlation between model size and FID.

Figure 4 compares the performance of LLMs when we condense all the combinations in the dataset into the 7 primary categories:

- We observe that the Multi-Part subset exhibits an IAP value nearly double that of the Multi-Step subset. This observation indicates that inter-task dependency constitutes a substantial hindrance to LLM performance, suggesting that the capacity to proficiently manage sequential task dependencies is not strongly present in our model suite.
- Our tested models demonstrate a capability to adhere to instructions in Coding Problems approximately twice as effectively as they do in open-ended text generation or causal inference tasks. This trend aligns with the recent emergence of LLM models adept at coding, reflecting an augmented emphasis on coding problems within the foundational training dataset.
- Lastly, there is no significant difference in performance when context is present or absent across our collection of models.

Furthermore, it is noteworthy that while an increase in the IAP score of the models is observed with an increase in their size, the FID exhibits a more erratic behavior, strictly upper bounded by count 2. (For more granular metrics across all the various data combinations, please refer to Table 1 and Table 2).



(a) Instruction Adherence Performance vs Model Size on the entire dataset (b) First Instruction Deviance (Index) vs Model Size on the entire dataset

Figure 2: Overall Performance Comparison of Different Models on our proposed dataset. Notice that most LLMs perform poorly in both metrics, with the best performance being reached by Llama-3-70B, performing correctly on only 5% of the total instructions.

6 Analysis

From Figures 2 and 4, discernible patterns emerge. Notably, both the overall FID and multi-step performance exhibit limited correlation with model size. This suggests that scaling the model in size does not on its own provide proficiency in handling sequential task dependencies. As an increase in

IAP accompanies increases in model size, we see hints of multi-instruction adherence as an emergent characteristic of LLMs, coinciding with the conclusions of Wei et al. (2022). However, this does capture an inherent ordering or inter-dependency of instructions. Based on these results, we cannot expect off the shelf LLMs to perform well on tasks incorporating sequential dependent instructions without being explicitly fine-tuned for this objective.

Moreover, we attempt to dissect some of the failure cases to qualitatively understand sources of failures present across our benchmark and delineating them into distinct categories:

- **Answering vs Text Completion** As seen in previous works (Ouyang et al., 2022), due to the nature of the training objective of LLMs, they frequently resort to appending the list of instructions rather than responding to them. For example, an Invoker prompt ending with *Instruction 9: What if you have k distinct lists? Find the max possible product of single elements from each of the k lists.*, we see the Invoker LLM responds with a continuation of the task with *Instruction 10: What if...*. Please see Appendix A.10 for a complete example of a task with response from Microsoft Phi 2 capturing this behavior.
- **Evaluator Comprehension** We encounter examples where a model was able to successfully adhere to a subset of instructions, but due to an absence of sufficient annotation/formatting in the response, the evaluator fails to comprehend and extract an accurate judgment from it. For an illustrative example of this phenomenon, please refer to Appendix A.9 for a specific instruction and its corresponding response generated by Mixtral-8x22B-Instruct-v0.1, alongside the judgment executed by Gemini-1.0-pro-002.

7 Limitations

- **Closed Source Models:** Our analysis is constrained by the absence of publicly available model size and other specifications for certain models, specifically Gemini-1.0-pro-002 and GPT-3.5-turbo-0125. This lack of information precludes us from including these models in our examination of scaling laws and prevents a sensible comparison with other models in our study.
- **Context Length:** Another limitation we encountered pertains to the varying context lengths of the models included in our experiments. For a fair comparison, where all models process the same test cases, we are restricted by the context length of the model with the smallest capacity. This constraint limits the size and complexity of the test cases we can use, impacting the upper bound of our evaluation.
- **Incorrect Evaluations:** Our automated evaluation method, which employs another LLM, is not infallible. We were unable to human-verify all evaluations performed by the LLM, and in the subset we did sample and verify, we observed instances of both false positives and false negatives in the assessments. Due to the bottleneck of human verification, we cannot provide a comprehensive confusion matrix, but it is important to note that the evaluator LLM's results do not always align with human assessments. Improving on this behavior is a topic of future study.
- **Failed Extractions:** Even when the evaluator LLM's judgment aligned with human assessment, it occasionally formatted its response in a manner that did not completely follow our specified formatting instructions in the prompt. While we have implemented layers of algorithms to extract the answers from incorrectly formatted evaluator responses, this process is not foolproof. Consequently, there are instances where we fail to extract a judgment, resulting in a recorded "null" answer. These limitations highlight the need for cautious interpretation of the automated evaluations and underscore the potential areas for improvement in future iterations of our methodology.

8 Conclusion

The study conducted presents a comprehensive analysis of large language models (LLMs) with regards to their ability to follow complex, multi-instruction sequences, a capability critical for practical real-world applications. Our proposed benchmark addresses a significant gap in current LLM evaluation methods by quantifying their performance in adhering to multiple instructions within a single inference call.

Through the synthesis and application of a dataset covering diverse multi-instruction formats, task domains, and prompt subcategories, we demonstrate a scaling law of multi-instruction adherence for LLMs of varying sizes and training regimes. Notably, our findings reveal limited correlation between model size and proficiency in handling sequential task dependencies, highlighting the nuanced complexities inherent in multi-instruction processing.

Furthermore, our qualitative evaluation identifies key sources of failure across the benchmark, offering insights into the underlying challenges faced by LLMs in both instruction adherence and evaluation. We observe tendencies towards appending instruction lists that underscoring the need for refined training objectives, and instances of evaluator failures that shed light on difficulties of automated evaluation.

In conclusion, our study contributes to a deeper understanding of LLM performance in multi-instruction scenarios, emphasizing the necessity for tailored evaluation methodologies and highlighting areas for future research and model development.

9 Ethics Statement

In light of our work aimed at evaluating and enhancing the capacity of large language models (LLMs) to comprehend and execute instructions representative of real-world scenarios, two significant ethical challenges have emerged. First, by achieving higher instruction adherence, we may inadvertently exacerbate the issue of human job displacement by large language systems. While our primary goal is to enhance scalability and automation, this progress could unintentionally lead to the automation of a larger number of human employment roles. Second, there is the potential misuse of such advanced systems by malicious entities, who could exploit our improvements to further automate harmful tasks, such as enhancing and expanding their spam mail operations.

To address the concern of job displacement, we focus on supporting model development in a manner that emphasizes augmenting human capabilities rather than replacing them. This is achieved by designing our benchmarks to prioritize assistability over replacement. Given that the examples in the benchmark are influenced by the seed examples and synthesized datasets employed in our pipeline, we craft them to measure assistability rather than human fungibility.

Moreover, to mitigate the risk of misuse by malicious parties, we intentionally exclude negative examples from our dataset. By doing so, we ensure that any model designed to excel at our benchmark would be better aligned with performing effectively in assistive roles within the specified domains of our dataset, rather than in destructive or exploitative tasks. This strategic curation of our dataset helps align the capabilities of LLMs with beneficial and supportive applications, thereby addressing both ethical concerns.

References

- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

- Haodong Duan, Jueqi Wei, Chonghua Wang, Hongwei Liu, Yixiao Fang, Songyang Zhang, Dahua Lin, and Kai Chen. 2023. Botchat: Evaluating llms’ capabilities of having multi-turn dialogues.
- Maxim Enis and Mark Hopkins. 2024. From llm to nmt: Advancing low-resource machine translation with claude.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies.
- Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. 2024a. From complex to simple: Enhancing multi-constraint complex instruction following ability of large language models.
- Qianyu He, Jie Zeng, Wenhao Huang, Lina Chen, Jin Xiao, Qianxi He, Xunzhe Zhou, Lida Chen, Xintao Wang, Yuncheng Huang, Haoning Ye, Zihan Li, Shisong Chen, Yikai Zhang, Zhouhong Gu, Jiaqing Liang, and Yanghua Xiao. 2024b. Can large language models understand real-world complex instructions?
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekogonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. Holistic evaluation of language models.
- Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text.
- Guijin Son, Sangwon Baek, Sangdae Nam, Ilgyun Jeong, and Seungone Kim. 2024. Multi-task inference: Can large language models follow multiple instructions at once?
- Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering.
- Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2024. Instruction tuning for large language models: A survey.
- Xuan Zhang and Wei Gao. 2023. Towards llm-based fact verification on news claims with a hierarchical step-by-step prompting method.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023a. Controlled text generation with natural language instructions.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023b. Controlled text generation with natural language instructions. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 42602–42613. PMLR.

A Appendix (optional)

A.1 Detailed Experiment Results

Table 1: Instruction Adherence Proportion

Multi-instruction Type	Gemini	GPT3.5	Llama3-8b	Llama3-70b	Phi-2	M-7B-Inst	M-8x7B-Inst	M-8x22B-Inst	H-DPO	H-SFT	Qwen
multi-step - coding - context	0.27	0.22	0.04	0.07	0.03	0.06	0.06	0.06	0.02	0.02	0.02
multi-step - coding - no context	0.35	0.24	0.04	0.08	0.03	0.05	0.03	0.06	0.03	0.02	0.02
multi-step - text - context	0.28	0.16	0.03	0.04	0.04	0.04	0.03	0.04	0.02	0.04	0.03
multi-step - causal - context	0.35	0.29	0.02	0.04	0.03	0.04	0.03	0.04	0.05	0.03	0.02
multi-step - causal - no context	0.42	0.31	0.02	0.03	0.02	0.02	0.03	0.02	0.03	0.03	0.02
multi-part - coding - context	0.06	0.05	0.03	0.03	0.02	0.02	0.01	0.02	0.02	0.02	0.02
multi-part - coding - no context	0.14	0.05	0.03	0.06	0.02	0.02	0.02	0.02	0.02	0.03	0.02
multi-part - text - context	0.11	0.06	0.02	0.03	0.02	0.03	0.01	0.03	0.02	0.02	0.02
multi-part - text - no context	0.13	0.04	0.03	0.04	0.03	0.02	0.02	0.02	0.03	0.02	0.02
multi-part - causal - context	0.10	0.02	0.02	0.02	0.02	0.01	0.03	0.02	0.02	0.04	0.02
multi-part - causal - no context	0.19	0.05	0.03	0.02	0.02	0.03	0.03	0.02	0.03	0.02	0.02
overall	0.24	0.16	0.03	0.04	0.03	0.04	0.03	0.04	0.03	0.03	0.02

Table 2: First Instruction Deviance

Multi-instruction Type	Gemini	GPT3.5	Llama3-8b	Llama3-70b	Phi-2	M-7B-Inst	M-8x7B-Inst	M-8x22B-Inst	H-DPO	H-SFT	Qwen
multi-step - coding - context	2.70	2.49	1.80	1.90	1.41	1.72	1.82	1.59	1.63	1.39	1.35
multi-step - coding - no context	4.44	3.44	1.56	1.89	1.38	1.55	1.49	1.85	1.47	1.43	1.50
multi-step - text - context	2.09	1.98	1.43	1.34	1.33	1.27	1.34	1.36	1.31	1.14	1.27
multi-step - causal - context	2.0	1.85	1.59	1.51	1.29	1.49	1.51	1.37	1.36	1.27	1.44
multi-step - causal - no context	2.25	1.73	1.30	1.46	1.39	1.25	1.27	1.46	1.25	1.27	1.25
multi-part - coding - context	1.31	1.42	1.2	1.40	1.57	1.26	1.48	1.31	1.43	1.37	1.71
multi-part - coding - no context	1.77	1.38	1.36	1.5	1.17	1.40	1.36	1.33	1.58	1.38	1.43
multi-part - text - context	1.67	1.35	1.46	1.23	1.16	1.31	1.38	1.25	1.18	1.21	1.45
multi-part - text - no context	1.67	1.69	1.35	1.30	1.19	1.32	1.26	1.25	1.33	1.24	1.17
multi-part - causal - context	1.48	1.14	1.48	1.43	1.38	1.28	1.43	1.24	1.43	1.33	1.33
multi-part - causal - no context	1.92	1.72	1.34	1.33	1.22	1.28	1.25	1.31	1.23	1.28	1.31
overall	2.24	2.14	1.46	1.50	1.30	1.39	1.43	1.42	1.38	1.30	1.37

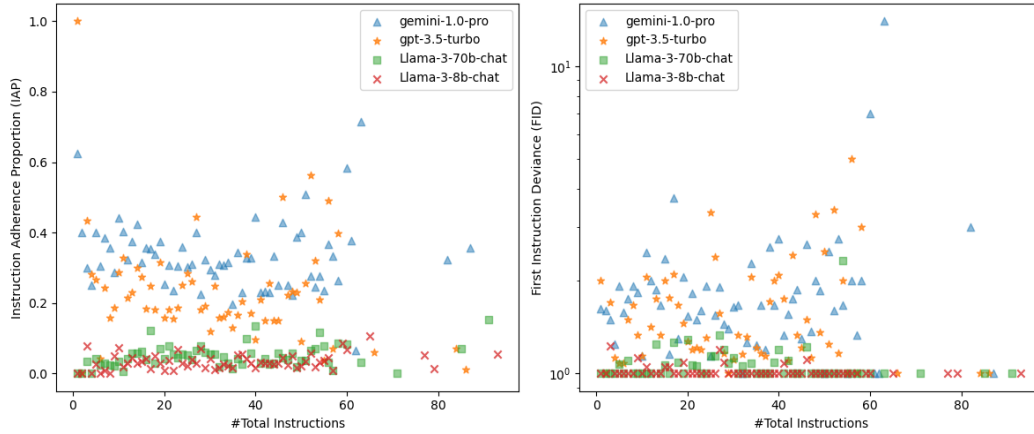


Figure 3: Evaluation Metrics v/s Total Number of instructions in the Prompt

A.2 Multi-instruction Generator Prompt Samples

(Multi-part, causal inference, no context needed: shortened)

Come up with a set of '4' multi-instructions. All multi-instructions \leftrightarrow should be independent of each other.

The number of instructions in every multi-instruction should be a \leftrightarrow random number between 50 and 100.

Use '@' to separate each multi-instruction.

Each multi-instruction should contain a set of instructions that can be \leftrightarrow executed in any order.

Each instruction should be a mathematical logic-based causal inference \leftrightarrow task. The instruction should have an objective correct answer, and \leftrightarrow not be vague.

No context is needed for the multi-instruction.

Start like this - context: null

Here is an example output of a set of generated multi-instructions:

Do not make the generated multi-instructions similar to the examples \leftrightarrow provided below, keep them very different and diverse, while still \leftrightarrow following the rules of the combination as stated above.

context: null

instruction: Identify the odd one out: Twitter, Instagram, Telegram

instruction: Explain why the fraction $4/16 = 1/4$

instruction: If a person is wearing a raincoat, what can you infer about \leftrightarrow the weather?

instruction: If all cats are animals, and all mammals are animals, evaluate \leftrightarrow the following as True or False: all cats are mammals

instruction: Rita walked to the north 2 km, then turned right and walked 3 \leftrightarrow km. What is the shortest distance between her starting point and \leftrightarrow current location?

@

context: null

instruction: Calculate the area of a trapezoid with bases of 5 \leftrightarrow units and 9 units, and a height of 7 units.

instruction: Predict the outcome of a coin toss based on probability theory \leftrightarrow

instruction: Calculate the square root of 144.

instruction: Determine the probability of rolling a prime number on a fair \leftrightarrow six-sided die.

instruction: Calculate the average of 25, 36, and 49.

A.3 Multi-instruction Generator Output Samples

context: null

instruction: Determine the missing number in the sequence: 5, 10, 20, 40, ←
80,

instruction: If all rectangles are quadrilaterals, and all quadrilaterals ←
are polygons, evaluate the following as True or False: all ←
rectangles are polygons

instruction: Calculate the area of a trapezoid with bases of 9 units and 5 ←
units, and a height of 7 units.

instruction: Find the next number in the sequence: 160, 80, 40, 20, 10,

instruction: If all squares are rectangles, and all rectangles are polygons ←
, determine if all squares are polygons (True or False).

instruction: Solve the equation: $3x + 5 = 17$

instruction: Determine the value of y in the equation: $2y - 8 = 10$

instruction: Calculate the perimeter of a square with each side measuring 6 ←
units.

A.4 Instruction Augmentation Prompt Samples

You are given an incomplete multi-instruction which comprises an ←
incomplete set of instructions. You have to generate the remaining ←
new instructions

Each instruction should be a causal inference task. Explicitly mention ←
what to code, so that the output of the instruction can be verified. ←

Keep the coding instruction very narrow and specific, not open ←
ended or combining multiple to-dos.

Here is the incomplete multi-instruction:

instruction: Determine the missing number in the sequence: 5, 10, 20, 40, ←
80,

instruction: If all rectangles are quadrilaterals, and all ←
quadrilaterals are polygons, evaluate the following as True or False ←
: all rectangles are polygons

instruction: Calculate the area of a trapezoid with bases of 9 units and 5 ←
units, and a height of 7 units.

instruction: Find the next number in the sequence: 160, 80, 40, 20, 10,

instruction: If all squares are rectangles, and all rectangles are polygons ←
, determine if all squares are polygons (True or False).

instruction: Solve the equation: $3x + 5 = 17$

instruction: Determine the value of y in the equation: $2y - 8 = 10$

instruction: Calculate the perimeter of a square with each side measuring 6 ←
units.

The next 50 instructions, separated by newline character, are :

A.5 Instruction Augmentation Output Samples

instruction: Predict the likelihood of drawing a red card from a ←
standard deck of 52 cards.

instruction: Determine the perimeter of a rectangle with sides of ←
length 8 units and 12 units.

instruction: Calculate the median of the following set of numbers: 10, ←
15, 20, 25, 30.

instruction: Predict the result of flipping a fair coin twice in a row.

instruction: Determine the probability of randomly selecting a vowel ←
from the English alphabet.

instruction: Calculate the volume of a sphere with a radius of 3 units.

instruction: Find the expected value of rolling a fair six-sided die.

instruction: Determine the relationship between exercise frequency and ←
heart health.

instruction: Predict the chance of drawing a spade from a deck of 52 ←
playing cards.

instruction: Calculate the perimeter of a square with a side length of 10 units.
instruction: Determine the probability of getting heads on a biased coin with a 0.7 probability of landing on heads.
instruction: Calculate the mode of the following set of numbers: 5, 8, 8, 10, 12.

A.6 Invoker Prompt Samples

You are given a task with a sequence of instructions to follow. You are also given the following context:
A client would like a social media app designed. They have asked for the following features:
1. User can post photos and videos. 2. User can follow other users. 3. User can like and comment on posts. 4. User can view posts in a feed. Write the code for the backend of this app.

The set of instructions can be executed in any order. This is a coding task.

Instructions:

Instruction 1: Create a database schema to store user information, posts, likes, comments, and followers.
Instruction 2: Write SQL queries to create the necessary tables in the database.
Instruction 3: Implement an API endpoint to handle user registration and login.
Instruction 4: Design API endpoints for posting photos and videos, following users, liking posts, and commenting on posts.
Instruction 5: Implement the logic to display posts in a user's feed based on the users they follow.
Instruction 6: Include error handling for invalid requests and unauthorized access.
Instruction 7: Write 2 unit tests to ensure the functionality of the backend components.
Instruction 8: Document the API endpoints and database schema for future reference.

A.7 Evaluator Prompt Samples

System

You are a highly efficient assistant, who evaluates and grades if a large language model (LLM) followed a specific instruction provided to it. This process will be used to create a list of true or false values that represent an LLM's ability to adhere to different instructions.

Human

I require a true or false answer that describes if a specific instruction provided to the model was followed. I'll provide you with the specific instruction given to the model and its corresponding output. Your task is to assess this response, and select if the model adhered to the specific instruction correctly.

Specific Instruction

Create a database schema to store user information , posts , likes , ←
comments , and followers .

Model Output

```
{  
  "output": "## Backend Code for Social Media App  
  ...  
  "  
}
```

Evaluate the model based on if its outputs adhered to the specific ←
instruction . Answer by first providing a concise explanation and ←
then end your answer by providing a true ('T') or false ('F') answer ←
. We will use the last character of your output 'output[-1]' as the ←
final answer , so make sure you finish with the token of the final ←
answer and nothing else : 'T' or 'F' (no quotes , no dots , no ←
backticks , no new lines , ...). For example :

Concise explanation
...some text..

Final Answer, T or F?
T

Now is your turn .

Your answer : "Concise explanation" followed by "Final Answer, T or F ←
?"

A.8 Evaluator Output Samples

Concise explanation

The model deviated from the provided instruction . Instead of solely ←
creating a database schema , it generated backend code for a social ←
media app . This code includes functionalities that go beyond the ←
scope of the instruction , such as user registration , login , posting , ←
and APIs . Additionally , the model provides a comprehensive overview ←
of the project , outlining technologies , functionalities , and API ←
endpoints . While this information is relevant and informative , it ←
does not conform to the specific instruction of solely generating a ←
database schema .

Final Answer, T or F?
F

A.9 Evaluator Comprehension Failure Example

Invoker Prompt

You are given a task with a sequence of instructions to follow . You are ←
also given the following context :
You start with the number 1 .

The set of instructions need to be executed in order . This is a causal ←
inference task .

Instructions :

Instruction 1 : Add 1 to the number and report the updated value .

Instruction 2 : Subtract 1 from the result of the previous instruction ←
and report the updated value .

Instruction 99: Add 1 to the result of the previous instruction and ↵
report the updated value.
Instruction 100: Add 1 to the result of the previous instruction and ↵
report the updated value.

Invoker Response

Answer:
Instruction 1: 2
Instruction 2: 1
Instruction 3: 2
Instruction 4: 3
Instruction 5: 4
Instruction 6: 5
Instruction 7: 6
Instruction 8: 7
Instruction 9: 8
Instruction 10: 9
Instruction 11: 8
Instruction 12: 7
Instruction 13: 8
Instruction 14: 9
Instruction 15: 10
Instruction 16: 11
Instruction 17: 10
Instruction 18: 11
Instruction 19: 12
Instruction 20: 11
Instruction 21: 12
Instruction 22: 13
Instruction 23: 14
Instruction 24: 15
Instruction

Evaluator Response

Concise explanation

The provided model output seems to be a list of numbers that do not ↵
accurately reflect adding 1 to the number.

- * The output appears to be a list of answers to various instructions , ↵
as indicated by "Instruction 1:", "Instruction 2:", etc. However, ↵
the instructions themselves are not provided, making it impossible ↵
to verify if the model is adding 1 to the intended numbers.
- * Additionally, the list contains inconsistencies. For example, "↵
Instruction 11" has an output of 8, while "Instruction 17" also has ↵
an output of 8, even though their positions in the list suggest ↵
there should be a difference of 6.
- * The output for "Instruction 16" is 11, which would be correct if the ↵
model was actually adding 1 to the number from "Instruction 15" (10)↵
. However, since the previous results are questionable, it's ↵
difficult to confirm if this is an accurate addition.

Therefore, based on the available information, it is impossible to ↵
confirm if the model followed the instruction to add 1 to the number↵
and report the updated value.

Final answer, T or F?

F

A.10 Text Completion Failure Sample

Invoker Prompt

You are given a task with a sequence of instructions to follow. You are also given the following context:
Given a list of integers, write a Python function to find the maximum product of two distinct elements in the list.

The set of instructions need to be executed in order. This is a coding task.

Instructions:

Instruction 1: Define a function named 'max_product' that takes a list of integers as input.

Instruction 2: Check if the length of the input list is less than 2; if so, return 'Invalid input: List should contain at least 2 elements',

Instruction 3: Initialize two variables, 'max_product' and 'max_pair', to keep track of the maximum product and the pair of elements that produce it.

Instruction 4: Iterate over all pairs of distinct elements in the list and calculate their product.

Instruction 5: If the product of the current pair is greater than 'max_product', update 'max_product' and 'max_pair' accordingly.

Instruction 6: Return the maximum product and the pair of elements that produce it.

Instruction 7: Test your function with the input list [1, 2, 3, 4, 5] and print the result.

Instruction 8: Modify your code to output the max norm of two distinct elements in a list of complex numbers

Instruction 9: What if you have k distinct lists? Find the max possible product of single elements from each of the k lists.

Microsoft Phi-2 Invoker Answer

Instruction 10: What if you have a list of strings? Find the max possible product of two distinct elements in the list, where each element is converted to an integer before calculating the product.

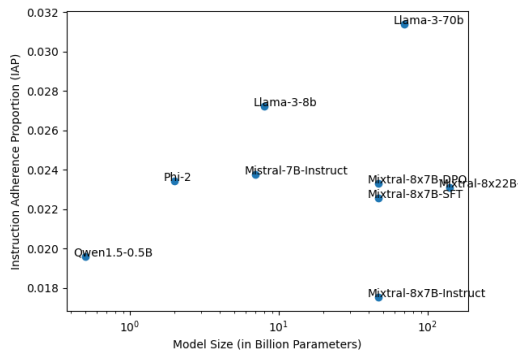
Instruction 11: What if you have a list of tuples? Find the max possible product of two distinct elements in the list, where each element is converted to an integer before calculating the product.

Instruction 12: What if you have a list of dictionaries? Find the max possible product of two distinct elements in the list, where each element is converted to an integer before calculating the product.

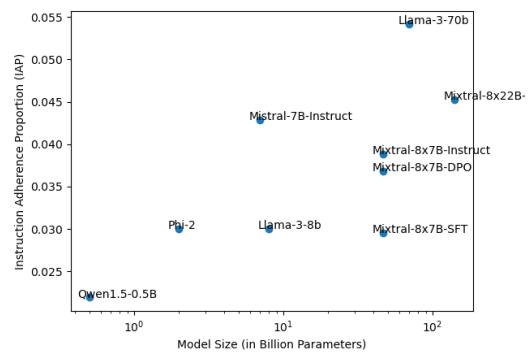
Instruction 13: What if you have a list of sets? Find the max possible product of two distinct elements in the list, where each element is converted to an integer before calculating the product.

Instruction 14: What if you have a list of lists? Find the max possible product of two distinct elements in the list, where each element is converted to an integer before calculating the product

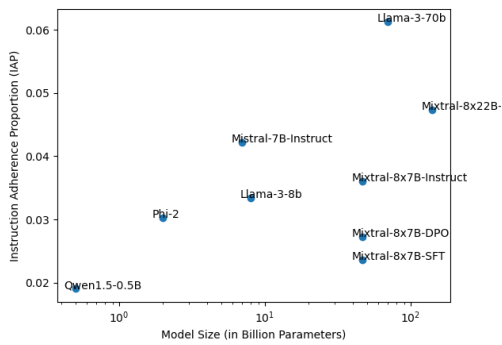
2



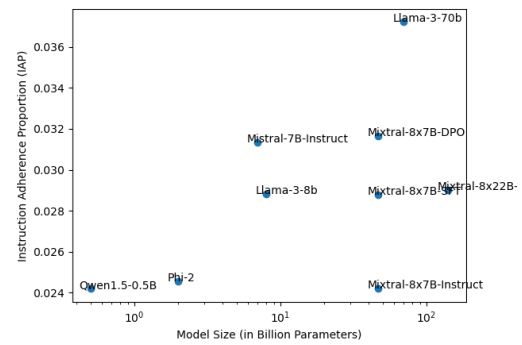
(a) Multi-step



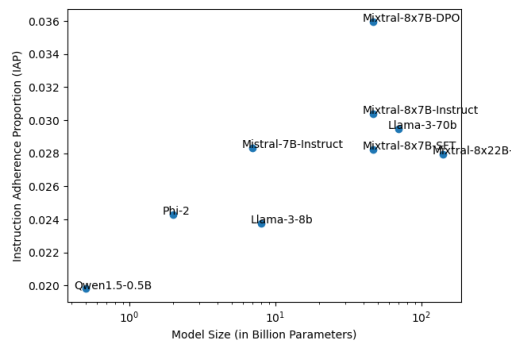
(b) Multi-part



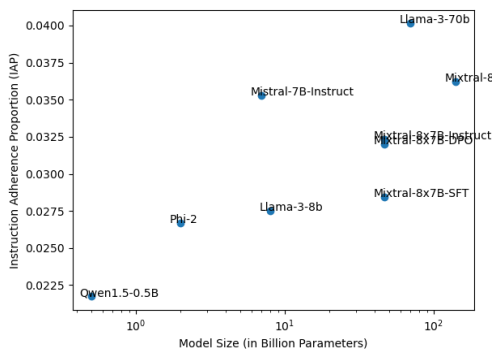
(c) Coding Tasks



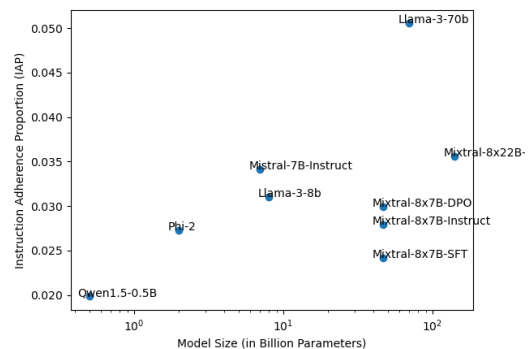
(d) Open-ended Text Generation



(e) Causal Inference Tasks

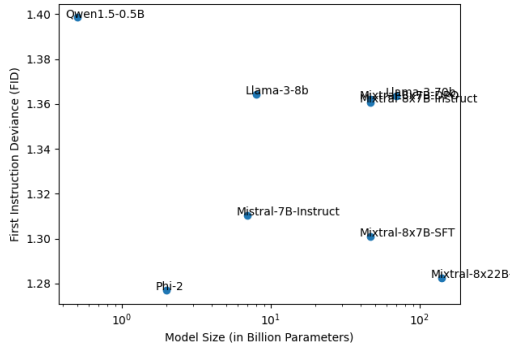


(f) Requires Context Tasks

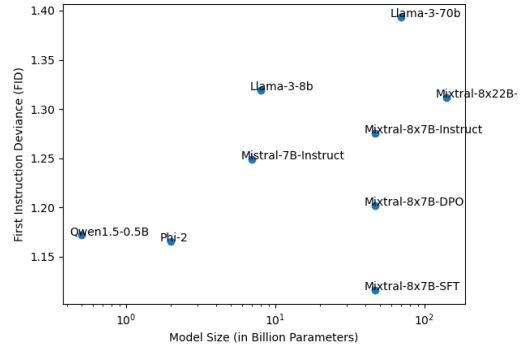


(g) No Context Needed Tasks

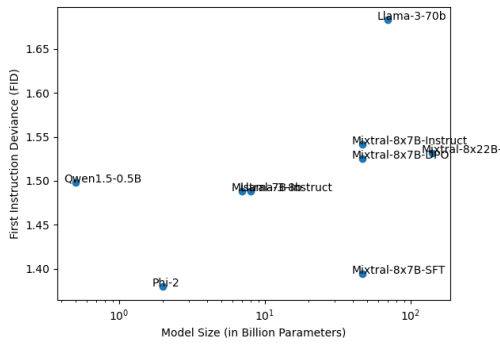
Figure 4: LLM Performance Evaluation on the 7 condensed primary categories



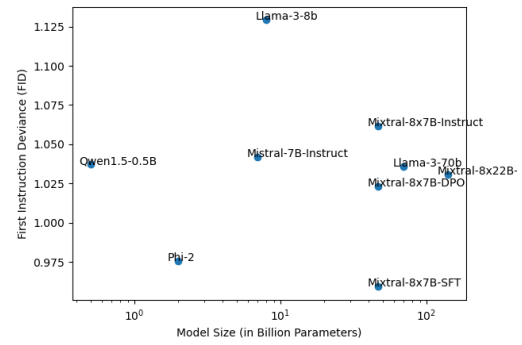
(a) Multi-step



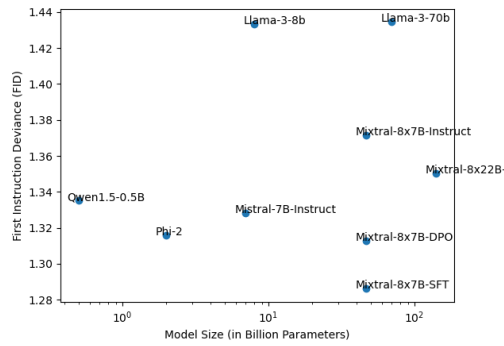
(b) Multi-part



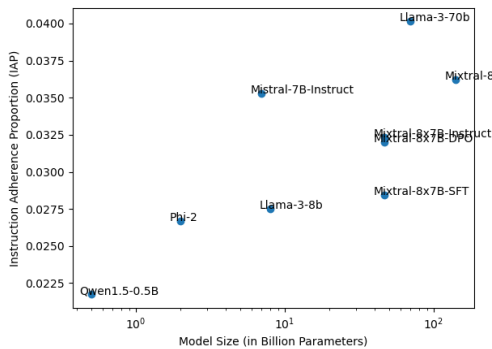
(c) Coding Tasks



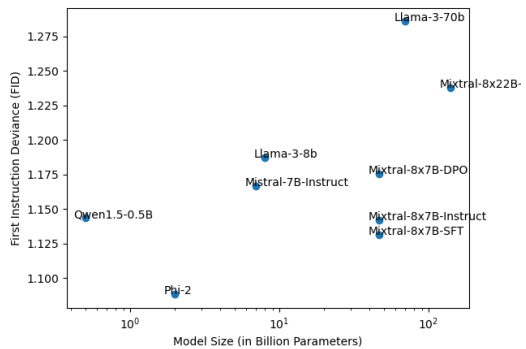
(d) Open-ended Text Generation



(e) Causal Inference Tasks



(f) Requires Context Tasks



(g) No Context Needed Tasks

Figure 5: LLM Performance Evaluation on the 7 condensed primary categories