

# Alpaca – An Inexpensive Instruction-Tuned Model to Compete with ChatGPT

Tim Lindenau

June 20, 2023

## Abstract

This paper introduces Alpaca, a novel instruction-tuned language model that aims to challenge established models like ChatGPT. Unlike existing models, Alpaca is openly available, compact in size, and can be trained at a significantly lower cost of just 600\$. Despite its cost-effectiveness, Alpaca strives to deliver comparable performance to ChatGPT. The paper delves into the internal workings of Alpaca and conducts a comprehensive evaluation to determine whether it can match the claims of performance comparable to ChatGPT.

## 1 Introduction

When Sam Altman visited TUM in Mai 2023 and was asked about the success of ChatGPT<sup>1</sup> compared to the previous GPT-3 model [3], he attributed it to the introduction of *a new interface* to interact with the chatbot. This *new interface* refers to the model’s ability to function as a chatbot, understanding user intentions from simple input, and generating high-quality output. This interface was unlocked through instruction tuning: a fine-tuning process where a pre-trained model is fine-tuned with natural language instructions similar to those that might be provided by users. Since the release of ChatGPT a few months ago, some comparable instruction-tuned models have been released, such as Bard by Google. Although these models show impressive performance, they are all closed-source and owned by large corporations. The model presented in this work – Alpaca – has been one of the first approaches for an open alternative. Based on the smallest of the LLaMA models [19] (7b parameters) and using an innovative way to create its fine-tuning dataset called *self-instruct* [20], Alpaca achieves performance “qualitatively similar to ChatGPT” [18] for just 600\$ total training cost. Alpaca is important as it is a first step to breaking the monopoly of big tech corporations and because it brings back the capability of doing research with instruction-tuned models to independent institutions in academia.

This work begins with an introduction to instruction tuning and the LLaMA models. It then presents an overview of the Alpaca architecture. A significant portion of the paper focuses on evaluating Alpaca’s performance, which was insufficiently addressed in the original publication. Finally, the paper discusses one risk associated with Alpaca.

## 2 Related Work

### 2.1 Instruction Tuning

Large language models (LLMs) such as GPT-3 demonstrate impressive few-shot capabilities across various tasks without any fine-tuning. What they lack is strong zero-shot performance when it comes to responding to unseen tasks [3]. This limitation becomes especially apparent in the chatbot setting where a model must answer to prompts given by a user in natural language and must provide convincing answers just from the prompt without any more in-context examples. Instruction tuning is a fine-tuning step applied to vanilla LLMs that significantly improves zero-shot performance and allows instruction-tuned models to perform well as chatbots without the necessity for tricks like prompting.

To motivate the idea behind instruction tuning the concept of measuring AI-Progress along two axes [17] can be helpful. The first axis is *capabilities*, meaning what can a model do, i.e., how good is its underlying understanding of language (grammar, common sense reasoning but also knowledge). The second axis is *alignment*, meaning how well is the model aligned with the user and its intentions, i.e., is the model behaving as a user expects it to be. Most LLMs perform well on the first axis due to their large model capacity and intensive pre-training. It is, however, the second axis where they struggle because their training objective of predicting the next token is not aligned with the intended use case of responding to human prompts in a chatbot setting [17]. During instruction-tuning alignment is improved by fine-tuning the model with examples of tasks a chatbot has to answer in the real world. These training tasks are provided in natural language, e.g., *explain the moon landing to a 6-year-old* and are referred to as user instructions [12, 22, 21, 13].

The process of instruction tuning can be performed in multiple ways. In the traditional supervised manner, instruction tuning is performed using a dataset of training samples, so-called instruction instances. Each instance comprises a) instruction tasks, b) input, and c) desired output. For example, one valid instance is: a) *Translate to German:* → b) *I love learning about language models* → c) *Ich liebe es über Sprachmodelle zu lernen*. Additional examples are available in Appendix A. During training, instruction tasks and inputs are fed to the model and the model is optimized by comparing its response to

---

<sup>1</sup>In this work ChatGPT always refers to the ChatGPT model based on GPT-3.5, not the model based on GPT-4

the reference answer and minimizing a loss. Besides being used for Alpaca, this supervised approach is widely employed by models such as FLAN [22]. Another mostly unsupervised approach for instruction tuning known as reinforcement learning from human feedback also exists and was, for example, used in ChatGPT. For details refer to Ouyang et al. [13].

## 2.2 Self-Instruct

The effectiveness of instruction tuning in enhancing user alignment, thereby improving task comprehension and achieving better zero-shot performance, has been demonstrated in various studies [17, 22, 21]. Moreover, research has consistently shown that instruction tuning outperforms traditional fine-tuning approaches when using the same dataset but not in an instruction style [22]. Additionally, instruction tuning offers advantages for addressing issues, such as toxicity and hallucinations, and has shown success in mitigating these challenges [17]. However, despite its capabilities, there was one major obstacle so far that prevented academia and smaller enterprises to create their own instruction-tuned models. This obstacle was coming up with a good dataset for fine-tuning.

That data is an important factor for fine-tuning should not surprise. However, in instruction tuning the issue is worse than in many other disciplines because coming up with a good dataset is hard. So did Wang et al. [21] show that while as expected, the performance of an instruction-tuned model is dependent on the size of the fine-tuning dataset (i.e., the number of instances in the dataset), it is even more dependent on the diversity of the instruction tasks itself. What this means is that it does not help to have multiple instances of the same task, for example, multiple examples of the task "Translate this text to German:". Instead, effective instruction tuning requires a large dataset of diverse instruction tasks with only a few instances of each one. Achieving this diversity provides a major challenge for the construction of a dataset. It cannot be easily done by web scraping but also using crowd-workers is difficult because they are expensive and likely to generate instruction tasks very similar to one another.

Wang et al.'s work on *self-instruct* [20] proposes one way to overcome the data bottleneck using a new, almost human-free, data generation pipeline. Their idea is inspired by the observation that even non-instruction-tuned LLMs can come up with new instruction tasks and corresponding examples of input and output when provided with reference examples in their context (few-shot) [26, 24]. As shown in Figure 1, the corresponding data-generation pipeline starts with a seed dataset of a few hand-written instruction instances and a LLM (e.g., GPT-3). Some examples of the seed dataset are sampled to be used as in-context examples before the LLM is then tasked to come up with a new instruction instance (new instruction task, input, and desired output) taking the context as reference. The newly generated instance is stored in the output but is also fed back to extend the seed dataset. The same process is repeated many times until the desired output size is reached.

Wang et al. [20] have shown that their process is effective. The generated dataset is diverse showing only a small overlap with the seed set. Consequently, fine-tuning on it achieves a 33% performance increase on the *Super Natural Instructions* benchmark [21] when first using the non-instruction-tuned GPT-3 to create a dataset of over 50.000 instructions and then fine-tuning the same GPT-3 model on this data; hence the name *self-instruct*.

## 2.3 LLaMA Models

The LLaMA [19] models are a collection of strong-performing, open-source, foundational models trained by Meta ranging in size from a relatively small 7 billion parameters to about 65 billion parameters. All models are based on an autoregressive transformer decoder architecture and were trained exclusively on publicly available datasets. In comparison to other models, LLaMA models were trained on significantly more training data, which allows them to perform on par with much larger models. For example, the LLaMA-13b model outperforms the 175 billion parameters GPT-3 model in many benchmarks and the

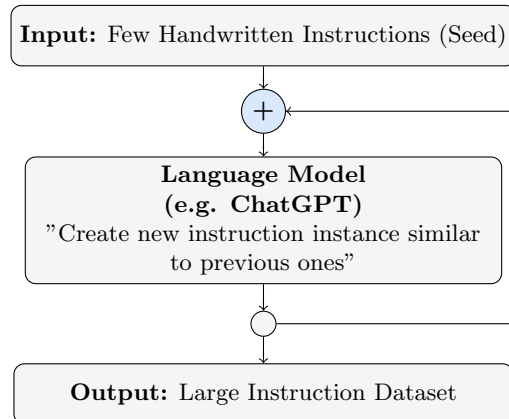


Figure 1: Self-Instruct: LM generates instruction dataset by recursively feeding back its outputs to inspire the generation of new instructions.

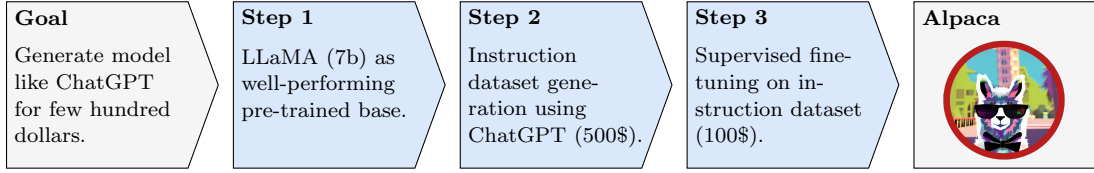


Figure 2: Alpaca is created in three steps for an overall cost of just 600\$. Alpaca image from [18].

largest LLaMA model with 65 billion parameters is competitive with Chinchilla [10] (70 billion parameters) and Palm [19] (540 billion parameters).

### 3 Alpaca Model in Detail

The Alpaca model [18], is a small and easy-to-train instruction-tuned model that challenges models like ChatGPT, by achieving qualitatively similar performance on a significantly smaller size and for a fraction of the training cost. As shown in Figure 2 three steps are necessary for the construction of Alpaca. The first step is to already start with a well-performing pre-trained model as the base. In the case of Alpaca, the smallest 7 billion parameters LLaMA model was chosen. This model was selected because its small size makes it easy to fine-tune, requiring less compute and less training data, while still delivering good performance. The second step involves generating a high-quality instruction dataset. This was achieved through a process similar to self-instruct but with a slightly different objective. Instead of aiming to self-improve the LLaMA model, Alpaca aims to distill knowledge [9] from a more powerful model by using the highest quality dataset possible. Therefore, the instruction-tuned text-davinci-003 [13] model was chosen to create the dataset, rather than GPT-3. Finally, the last step is supervised instruction tuning using the previously generated dataset.

With this setup, the training of Alpaca is very cost-effective and can be done for just 600\$, where 500\$ are required for the creation of the instruction dataset and another 100\$ for the costs of fine-tuning.

### 4 Performance Evaluation of the Alpaca Model

Since its release, the Alpaca model has generated significant excitement with its promise of ChatGPT-like performance for just 600\$. However, Taori et al. [18] did not provide any quantifiable evaluations of the model’s performance. Instead, all performance claims relied only on qualitative anecdotal evidence from the authors. This section aims to offer a more comprehensive evaluation of the model’s true capabilities by collecting, comparing, and interpreting performance data across three benchmarks.

**Benchmark 1 – Evaluation of language understanding:** Using the *Language Model Evaluation Harness* [6] performance is reported in four traditional benchmarks measuring qualities in general language understanding. Language understanding here means capabilities such as common-sense reasoning, pronoun resolution, but also the internal knowledge of the model and how well it can apply this knowledge. The benchmarks are 1) HellaSwag [25], 2) WinoGrande [16], 3) BoolQ [5], and 4) PIQA [2]. The results for LLaMA, Alpaca, and text-davinci-003 (model strongly related to ChatGPT that serves as a proxy due to the unavailability of data) are reported in Table 1a.

**Benchmark 2 – Evaluation as a chatbot with GPT-4 as the judge:** The primary quality expected from a chatbot is not language understanding but its ability to effectively respond to novel user prompts, comprehend their intentions, and provide high-quality responses. Few established benchmarks for this goal exist and the existing ones are too simple, making it hard to distinguish performance amongst SOTA models [4]. Recently Chiang et al. [4] proposed a new benchmark idea that leverages the observation that “GPT-4 can produce highly consistent ranks and detailed assessment when comparing chatbots’ answers” [4]. In detail, in this benchmark always two models are presented with a challenging prompt from different categories a chatbot might have to answer in the real world. Both responses are collected and GPT-4 is tasked to evaluate them side-by-side on an integer scale from 1 to 10. The results for LLaMA, Alpaca, and Bard (serving as a proxy for ChatGPT due to the unavailability of data) are presented in Table 1b. As an important note, this benchmark uses a newer and larger Alpaca version based on the 13 billion parameters LLaMA model. Although this model is expected to perform better than the 7 billion parameters version, it is a good proxy of its performance because the fine-tuning is the same, and compared to models like ChatGPT this model remains small and cheap to train.

Model	HellaSwag	WinoGrande	BoolQ	PIQA	Model	Score
LLaMA (7b)	73	66.9	73.1	77.4	LLaMA (13b)	68
Alpaca (7b)	73.9	66.1	73.1	77.4	Alpaca (13b)	76
text-davinci-003	83.3	75.5	88.1	83.19	Bard	93

(a) Benchmark 1

(b) Benchmark 2

Table 1: Performance comparison LLaMA, Alpaca, text-davinci-003, and Bard (the last two models can be seen as comparable to ChatGPT). Benchmark 1: Comparison in language understanding using four traditional LM benchmarks; data from [1]. Benchmark 2: Performance evaluation as a chatbot with GPT-4 as the judge; data from [4].

**Benchmark 3 – Evaluation by manual inspection:** To get a better intuition and to observe how differences in the first two benchmarks translate to differences in real-world usage, some examples of interactions with the models are provided. These examples aim to highlight general observations the author made while experimenting with the models. All examples can be found in Appendix B.

## 4.1 Results

The results are divided into two sections. The first section compares Alpaca to its base LLaMA model to determine if instruction tuning provides a measurable advantage. The second section then compares Alpaca with ChatGPT to analyze how similar their performance is.

### 4.1.1 Comparison of Alpaca and LLaMA

Instruction tuning does not help in building a better language understanding with both models performing similarly well in the first benchmark. However, better performance in the first benchmark was also not the goal of instruction tuning. Instead, instruction tuning should improve performance in the chatbot setting where mere language understanding might not be that important but rather user alignment. Results from the second benchmark show that instruction tuning does improve alignment leading to an 8 percentage point better performance. Finally, the practical examples in benchmark three validate the expectations obtained from the first two benchmarks. Firstly, it is evident that instruction tuning enhances alignment, as Alpaca demonstrates a better understanding of user intentions in cases where LLaMA encounters difficulties (see examples B.1.2, B.1.3). However, it is also notable that once the intention is understood, both models show similar response quality (see example B.1.1). It is likely, that similar language understanding is one factor contributing to the similar response quality.

### 4.1.2 Comparison of Alpaca and ChatGPT

Again, starting with the first benchmark there is a clear difference between Alpaca and text-davinci-003 indicating a significantly better language understanding of the latter. This result is not surprising given that text-davinci-003 also performs much better than LLaMA in these tests. Still, what this difference suggests is that there could be tasks that text-davinci-003 and therefore ChatGPT can solve, that Alpaca just cannot due to its limited understanding of language. The more surprising result is observed in the second benchmark. Since this benchmark measures capabilities as a chatbot, it was expected that performance would be similar. This is not the case with Alpaca performing a large 17 percentage points worse than Bard. To put this into context, this difference is more than twice as large as the difference between Alpaca and LLaMA of 8 percentage points, which as shown before, already had significant implications for understanding user intent.

The important question now is whether these differences are also visible in real-world utilization. Here, benchmark three depicts a more optimistic scenario showing that both Alpaca and ChatGPT are good at understanding user intent with ChatGPT getting the intent right always and Alpaca mostly. However, ChatGPT is significantly better when it comes to the quality of the responses. Better language understanding as shown in the first benchmark is one factor contributing to this difference. This can, for example, be seen in translation tasks where ChatGPT generally performs better due to a better language understanding stemming from its larger model size (see example B.2.1). In the future, this factor could be an issue since language understanding might be a barrier unable to be overcome by smaller models and could therefore manifest an advantage of big corporations.

At the same time, however, it is also likely that some differences in the response quality stem from ChatGPT using better instruction tuning. For example, one can observe that the responses of Alpaca are oftentimes short (see example B.2.2), which can be a disadvantage since longer responses are

generally desirable because they leave more space for details. This tendency to short responses is a consequence of the instruction tuning dataset used in Alpaca since in this dataset all responses are limited to just 100 words [18]. Additionally, being generated by another LM, the fine-tuning dataset of Alpaca occasionally suffers from issues that can be disadvantageous. For example, there are some instances where responses do not match the instructions [15].

It is important to note that these second types of disadvantages should not demotivate but rather allow for optimism about the future of models like Alpaca since those are areas where further work is possible. For example, there are already new models similar to Alpaca but with improvements in instruction tuning that manage to further close the gap to ChatGPT [4, 7, 14]. An interesting question for future work could therefore be to evaluate how much each factor (language understanding, instruction tuning) contributes to the model performance and if there is a limiting factor just because of scale.

## 4.2 Discussion of Approach to Benchmarking

The approach used to benchmark is just one of many possible methods, and it is therefore important to critically reflect on its effectiveness. In particular, it should be noted that the second benchmark is not yet rigorous and has known flaws. For example, it has been observed that GPT-4 strongly prefers longer answers [23], an observation that helps to understand why Alpaca performed so badly in the second benchmark, despite showing good performance in the third benchmark.

Despite this limitation, overall the chosen approach to benchmarking is supported since it aligns with the findings made in other recent studies. For example, *The False Promise of Imitating Proprietary LLMs* [8], also investigated the performance of models like Alpaca and validates the findings that Alpaca performs well to many questions, especially simple ones. However, they have also shown that the small model size can hold back performance due to weaker language understanding. This limitation is particularly notable for difficult tasks that require capabilities like complex common-sense reasoning.

## 5 Inheritance of Negative Traits via Self-Instruct

It is well-known that all current language models suffer from issues like bias, toxicity, and hallucinations. Even though research is ongoing, the assumption is that these issues are (at least partially) the results of flawed datasets [3]. Naturally, datasets created using LMs (i.e., self-instruct) are likely to contain instances that suffer from these very issues, which poses a potential risk of amplification of negative traits when such datasets are used for fine-tuning. A first example demonstrating this amplification can be found when looking at Alpaca and its tendency to hallucinate.

While originally publicly available through a web interface the Alpaca model had to be taken down quickly due to a high tendency to hallucinate [18]. For instance, using the *TruthfulQA* [11] benchmark, Alpaca achieves a poor result of only 26.7% compared to models like text-davinci-003 with a score of 40.0%. There are at least two possible explanations for this tendency to hallucinations. Firstly, the small model size may play a role as smaller versions of the same model tend to hallucinate more [19]. However, this is unlikely to be the primary factor here since the same LLaMA model performs significantly more truthful than Alpaca, scoring 41% in TruthfulQA. A second hypothesis is that as previously mentioned flawed fine-tuning data may amplify hallucinations. The *Cleaned Alpaca Dataset* initiative [15] provides strong evidence for this second hypothesis by having shown that the Alpaca dataset suffers from many instances of hallucinations. They then filtered out all instances of hallucinations and trained a cleaned version of Alpaca on this new dataset. Again, tested for truthfulness, this cleaned version performed significantly better, improving performance from the previously achieved 26.7% to now 37%.

## 6 Conclusion

In conclusion, Alpaca is an innovative idea that, although not completely on the same level as ChatGPT, represents an important initial step towards well-performing, open-source instruction-tuned models. As such Alpaca already had a big influence and inspired the creation of a new class of language models all based on the same idea of combining small foundational models and dataset generation using LLMs. Examples of these models are Koala [7] & Vicuna [4]. However, there remain open questions. Firstly, there is evidence for an upper limit to the performance of models like Alpaca just because of their small size and consequently worse language understanding. Additionally, the use of other language models for dataset creation carries risks, such as increased hallucinations, which do not yet find enough attention in discourse and require more future work to be done.

## References

- [1] Y. Anand, Z. Nussbaum, B. Duderstadt, B. Schmidt, and A. Mulyar. GPT4All: Training an Assistant-style Chatbot with Large Scale Data Distillation from GPT-3.5-Turbo, 2023. URL <https://github.com/nomic-ai/gpt4all>. Publication Title: GitHub repository.
- [2] Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi. PIQA: Reasoning about Physical Commonsense in Natural Language, Nov. 2019. URL <http://arxiv.org/abs/1911.11641>. arXiv:1911.11641 [cs].
- [3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language Models are Few-Shot Learners, July 2020. URL <http://arxiv.org/abs/2005.14165>. arXiv:2005.14165 [cs].
- [4] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%\* ChatGPT Quality, Mar. 2023. URL <https://lmsys.org/blog/2023-03-30-vicuna/>.
- [5] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *NAACL*, 2019.
- [6] L. Gao, J. Tow, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, K. McDonell, N. Muennighoff, J. Phang, L. Reynolds, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou. A framework for few-shot language model evaluation, Sept. 2021. URL <https://doi.org/10.5281/zenodo.5371628>.
- [7] X. Geng, A. Gudibande, H. Liu, E. Wallace, P. Abbeel, S. Levine, and D. Song. Koala: A Dialogue Model for Academic Research, Apr. 2023. URL <https://bair.berkeley.edu/blog/2023/04/03/koala/>. Published: Blog post.
- [8] A. Gudibande, E. Wallace, C. Snell, X. Geng, H. Liu, P. Abbeel, S. Levine, and D. Song. The False Promise of Imitating Proprietary LLMs, May 2023. URL <http://arxiv.org/abs/2305.15717>. arXiv:2305.15717 [cs].
- [9] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network, Mar. 2015. URL <http://arxiv.org/abs/1503.02531>. arXiv:1503.02531 [cs, stat].
- [10] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. v. d. Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre. Training Compute-Optimal Large Language Models, Mar. 2022. URL <http://arxiv.org/abs/2203.15556>. arXiv:2203.15556 [cs].
- [11] S. Lin, J. Hilton, and O. Evans. TruthfulQA: Measuring How Models Mimic Human Falsehoods, May 2022. URL <http://arxiv.org/abs/2109.07958>. arXiv:2109.07958 [cs].
- [12] S. Mishra, D. Khashabi, C. Baral, and H. Hajishirzi. Cross-Task Generalization via Natural Language Crowdsourcing Instructions, Mar. 2022. URL <http://arxiv.org/abs/2104.08773>. arXiv:2104.08773 [cs].
- [13] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, Mar. 2022. URL <http://arxiv.org/abs/2203.02155>. arXiv:2203.02155 [cs].
- [14] B. Peng, C. Li, P. He, M. Galley, and J. Gao. Instruction Tuning with GPT-4, Apr. 2023. URL <http://arxiv.org/abs/2304.03277>. arXiv:2304.03277 [cs].
- [15] G. Ruebsamen. Cleaned Alpaca Dataset, June 2023. URL <https://github.com/gururise/AlpacaDataCleaned>. original-date: 2023-03-21T16:30:07Z.
- [16] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale, Nov. 2019. URL <http://arxiv.org/abs/1907.10641>. arXiv:1907.10641 [cs].
- [17] Scale AI. OpenAI’s InstructGPT: Aligning Language Models with Human Intent, Aug. 2022. URL <https://www.youtube.com/watch?v=QGpaBW0aHQI>.

- [18] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford Alpaca: An Instruction-following LLaMA model, 2023. URL [https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca). Publication Title: GitHub repository.
- [19] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. LLaMA: Open and Efficient Foundation Language Models, Feb. 2023. URL <http://arxiv.org/abs/2302.13971>. arXiv:2302.13971 [cs].
- [20] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. Self-Instruct: Aligning Language Model with Self Generated Instructions, Dec. 2022. URL <http://arxiv.org/abs/2212.10560>. arXiv:2212.10560 [cs].
- [21] Y. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Arunkumar, A. Ashok, A. S. Dhanasekaran, A. Naik, D. Stap, E. Pathak, G. Karamanolakis, H. G. Lai, I. Purohit, I. Mondal, J. Anderson, K. Kuznia, K. Doshi, M. Patel, K. K. Pal, M. Moradshahi, M. Parmar, M. Purohit, N. Varshney, P. R. Kaza, P. Verma, R. S. Puri, R. Karia, S. K. Sampat, S. Doshi, S. Mishra, S. Reddy, S. Patro, T. Dixit, X. Shen, C. Baral, Y. Choi, N. A. Smith, H. Hajishirzi, and D. Khashabi. Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks, Oct. 2022. URL <http://arxiv.org/abs/2204.07705>. arXiv:2204.07705 [cs].
- [22] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned Language Models Are Zero-Shot Learners, Feb. 2022. URL <http://arxiv.org/abs/2109.01652>. arXiv:2109.01652 [cs].
- [23] C. Xu, D. Guo, N. Duan, and J. McAuley. Baize: An Open-Source Chat Model with Parameter-Efficient Tuning on Self-Chat Data, May 2023. URL <http://arxiv.org/abs/2304.01196>. arXiv:2304.01196 [cs].
- [24] S. Ye, D. Kim, J. Jang, J. Shin, and M. Seo. Guess the Instruction! Flipped Learning Makes Language Models Stronger Zero-Shot Learners, Dec. 2022. URL <http://arxiv.org/abs/2210.02969>. arXiv:2210.02969 [cs].
- [25] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. HellaSwag: Can a Machine Really Finish Your Sentence?, May 2019. URL <http://arxiv.org/abs/1905.07830>. arXiv:1905.07830 [cs].
- [26] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba. Large Language Models Are Human-Level Prompt Engineers, Mar. 2023. URL <http://arxiv.org/abs/2211.01910>. arXiv:2211.01910 [cs].



## A Composition of Instruction Tuning Dataset

The following shows multiple examples of how instances of an instruction tuning dataset can look like. It shows that different kinds of instruction styles are possible. For example, text-generation and text-classification tasks, but also tasks with and without input.

Example	Comment
<b>Instruction:</b> Find out if the given text is in favor of or against abortions. <b>Input:</b> I believe that women should have the right to choose whether or not they want to have an abortion. <b>Output:</b> Class Label: Pro-abortion	The first kind of instructions are classification tasks. These instructions generally consist of the instruction, the input to classify, and a desired class label.
<b>Instruction:</b> Give me a quote from a famous person on this topic. <b>Input:</b> Topic: The importance of being honest. <b>Output:</b> Honesty is the first chapter in the book of wisdom – Thomas Jefferson.	The second type of instructions are text-generation instructions.
<b>Instruction:</b> Explain the moon landing to a 6-year-old in a few sentences. <b>Input:</b> <code>&lt;empty&gt;</code> <b>Output:</b> People went to the moon and they took pictures of what they saw and sent them back to the earth so we could all see them.	Some instances omit the input if it is already contained in the instruction.

## B Additional Material for Benchmark 3: Examples of Real-World Interactions with Different Language Models

The following provides an overview of actual interactions with different language models (LLaMA, Alpaca, and ChatGPT). All examples were collected to best possible demonstrate the strengths and weaknesses of each model. Section B.1) compares the results of LLaMA and Alpaca to the same prompt, while section B.2 compares ChatGPT and Alpaca. Each example is accompanied by a brief description explaining the observations.

### B.1 Comparison of Responses between LLaMA and Alpaca

#### B.1.1 Example 1: Alpaca and LLaMA Show Similar Response Quality

<b>Instruction Task:</b> How does backpropagation work?	
<b>Response LLaMA</b>	<b>Response Alpaca</b>
Backpropagation is an iterative process that works by repeatedly calculating the error between the network’s output and the desired output, then adjusting the weights of the artificial neurons.	Backpropagation is a form of neural network training where the error of the model is propagated back through the network, adjusting the weights of each layer accordingly.

Both models understand the prompt correctly and generate a well-written response. There is no clear difference in the quality of the responses. Similar response quality could be the effect of similar language understanding performance.

#### B.1.2 Example 2: LLaMA Has Problems in Understanding User Intentions

<b>Instruction Task:</b> I want to bake a lemon cake!	
<b>Response LLaMA</b>	<b>Response Alpaca</b>
I’m sorry I did not understand you correctly. Can you repeat the request?	Here is a recipe for a lemon cake: Ingredients: - 2 cups all-purpose flour - 1/2 teaspoon baking powder [...]

This prompt is more involved than the first one and the intention is not as clear. The non-instruction-tuned LLaMA model struggles and does not understand the user intent. Alpaca understands the prompt well and delivers a well-written response. This example shows well the positive effect-of instruction tuning on alignment.

#### B.1.3 Example 3: LLaMA Has Problems in Understanding User Intentions. Alpaca Has Problems in Response Quality

<b>Instruction Task:</b> Translate to German: I love learning about language models.	
<b>Response LLaMA</b>	<b>Response Alpaca</b>
I didn’t quite catch that. Please try again	Ich liebe lernen über Sprachmodelle

Again a simple prompt with a clear user intention. Still, LLaMA does not understand the intention. Instruction tuning is again effective for alignment and Alpaca understands the intention well. However, the response of Alpaca is with some grammatical mistakes which could be an indication of insufficient language understanding.

## B.2 Comparison of Responses between ChatGPT and Alpaca

### B.2.1 Example 1: ChatGPT Shows Superior Performance in Translation Task

<b>Instruction Task:</b> Translate to German: I love learning about language models.	
<b>Response ChatGPT</b>	<b>Response Alpaca</b>
Ich liebe es, über Sprachmodelle zu lernen.	Ich liebe lernen über Sprachmodelle.

This is the same prompt as before where Alpaca gets the intention right but struggles with the response. ChatGPT also gets the intention right and surpasses Alpaca in response quality, delivering a response without any grammatical mistakes. This difference is likely a consequence of the better language-understanding capabilities of ChatGPT.

### B.2.2 Example 2: ChatGPT Shows Superior Response Quality

<b>Instruction Task:</b> Draft an apology email to a customer who experienced a delay in their order and provide reassurance that the issue has been resolved.	
<b>Response ChatGPT</b>	<b>Response Alpaca</b>
<p>Dear [Customer's Name],</p> <p>I hope this email finds you well. I am writing to sincerely apologize for the delay you experienced with your recent order. [...]</p> <p>We have identified the root cause of the delay and have implemented measures to prevent similar occurrences in the future.</p> <p>To make it up to you for the inconvenience caused, we have expedited the processing of your order, and it will be dispatched today. You should receive a separate email with the tracking details shortly. Additionally, we have applied a discount of 15% to your next purchase as a gesture of goodwill.</p> <p>We appreciate your understanding and patience throughout this process. If you have any further questions or concerns, please don't hesitate to reach out to our customer support team at [customer support contact details]. We are always here to assist you.</p> <p>Thank you for your continued trust in our products and services. We are committed to providing you with an exceptional customer experience going forward.</p> <p>Warm regards, [Your Name] [Company Name]</p>	<p>We apologize for the delay you experienced with your order and thank you for your patience. We have identified the cause of the issue and have taken steps to ensure it does not happen again. We value your business and hope to be able to serve you better in the future.</p>

Both models understand the user intention correctly and deliver a correct response. However, the response quality of ChatGPT is significantly better. While Alpaca omits important details like the customer name and the name of the service worker, ChatGPT includes all these details. Also, the response of ChatGPT is more creative coming up with the idea of a discount on the next purchase. It is likely that the difference in response quality is partially due to different language-understanding performances but also partially due to differences in instruction tuning. The response of Alpaca is relatively short which is a common observation with this model. Longer responses are, however, often preferable since they leave more space for details. This disadvantage of Alpaca preferring short responses is a direct consequence of the instruction tuning dataset since all responses in this dataset are limited to a length of just 100 words.

### B.2.3 Example 3: Only ChatGPT Can Infer True User Intention from Instruction Task

<b>Instruction Task:</b> Explain why this joke is funny: I'm a big fan of whiteboards. I find them quite remarkable.	
<b>Response ChatGPT</b>	<b>Response Alpaca</b>
This joke is a play on words, specifically a pun. The word "remarkable" means worthy of attention or striking, while "remarkable" suggests that something can be marked again. The humor comes from the double meaning of "remarkable." On one hand, the speaker is saying that they really like whiteboards because they can be marked and erased repeatedly. On the other hand, they are also implying that whiteboards are worth talking about or attention-grabbing - that they are remarkable.	I'm glad you appreciate the joke. Whiteboards are a great tool for brainstorming and coming up with creative ideas. They are also versatile and can be used for a variety of tasks. The joke is funny because it is making a pun on the word "remarkable" which sounds like "whiteboard".

Only ChatGPT understands the user intention correctly. Alpaca has issues understanding the intent and first gives unnecessary information. At the end, it tries to answer the question but also the answer is wrong and does not explain the joke. ChatGPT understands the intention better due to better instruction tuning. That Alpaca then fails to explain the joke is most likely an issue of language understanding, i.e., benchmark 1.