Demystifying Instruction Mixing for Fine-tuning Large Language Models

Renxi Wang[1,2] **Haonan Li**[1,2] **Minghao Wu**[3] **Yuxia Wang**[1,2]
Xudong Han[1,2] **Chiyu Zhang**[4] **Timothy Baldwin**[1,2,5]
[1]Mohamed bin Zayed University of Artificial Intelligence [2]LibrAI
[3]Monash University [4] University of British Columbia [5] The University of Melbourne
{renxi.wang,haonan.li,yuxia.wang,xudong.han,timothy.baldwin}@mbzuai.ac.ae
minghao.wu@monash.edu chiyuzh@mail.ubc.ca

Abstract

Instruction tuning significantly enhances the performance of large language models (LLMs) across various tasks. However, the procedure to optimizing the mixing of instruction datasets for LLM fine-tuning is still poorly understood. This study categorizes instructions into three primary types: NLP downstream tasks, coding, and general chat. We explore the effects of instruction tuning on different combinations of datasets on LLM performance, and find that certain instruction types are more advantageous for specific applications but can negatively impact other areas. This work provides insights into instruction mixtures, laying the foundations for future research.[1]

1 Introduction

Instruction tuning has been shown to have surprising efficacy for aligning large language models (LLMs) with human instructions (Chung et al., 2022; Li et al., 2023; Wu et al., 2023; Xu et al., 2023; Touvron et al., 2023; Muennighoff et al., 2023a; Gunasekar et al., 2023). Recent studies highlight the diverse ways in which instructions can enhance the different capabilities of LLMs. For instance, using general-purpose, chat-like instructions can improve the performance of LLMs as chat assistants (Chiang et al., 2023; Ouyang et al., 2022; Taori et al., 2023; Ding et al., 2023), while training LLMs on instructions based off NLP tasks improves their performance on NLP benchmarks (Sanh et al., 2022; Chung et al., 2022; Muennighoff et al., 2023b), and incorporating coding instructions enhances LLM code generation (Fu and Khot, 2022; Gunasekar et al., 2023). However, a key unresolved issue is determining how to combine various instruction datasets to optimize overall LLM performance.

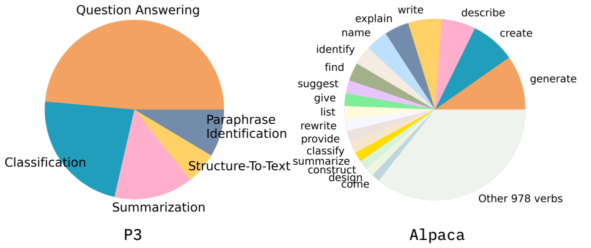


Figure 1: Instruction type distribution of P3 and Alpaca. For P3, the statistics come from the original dataset, while for Alpaca, we use a dependency parsing approach to extract the root verb of each instruction.

In this paper, we aim to better understand the impact of instruction mixing across three critical areas: NLP downstream tasks, coding, and chat. The core of our investigation revolves around understanding the influence of instruction dataset distributions on model performance in these different areas. We first select representative instruction datasets: P3 (Sanh et al., 2022) for NLP downstream tasks, CodeAlpaca (Chaudhary, 2023) for code generation, and Alpaca (Taori et al., 2023) for general-purpose instructions. As shown in Figure 1, P3 is focused primarily on five tasks (including QA and classification), whereas Alpaca contains a vast array of instructions. Using a dependency parser, we identify over 1K unique root verbs from Alpaca's instructions, with *generate*, *create*, and *describe* being the most frequent. CodeAlpaca, by contrast, is exclusively focused on coding tasks, and exhibits less variation compared to the others as exemplified in Table 3. We fine-tune models across all eight potential combinations of these instruction datasets, and carry out detailed evaluation of model performance in terms of NLP downstream tasks, coding proficiency, and chat capabilities.

Our main contribution in this work is to shed light on instruction mixing when fine-tuning LLMs through comprehensive experimentation. Our findings can be summarized as follows:

[1] Code and data are available at: https://github.com/Reason-Wang/InstructLLM.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop), pages 162–169
August 11-16, 2024 ©2024 Association for Computational Linguistics

- Specific instruction datasets enhance LLM performance in their respective task areas. However, combining all instruction types does not uniformly improve performance across all tasks.

- Instructions reformulated from NLP downstream tasks (such as P3) can negatively impact the model's conversational abilities. In contrast, instructions focused on coding not only improve coding proficiency but also enhance chat capabilities.

- Larger models, with their increased capacity, are able to make more effective use of a diverse range of instructions.

2 Related Work

Recent work has demonstrated that vanilla LLMs can follow general instructions if tuned with instructions and corresponding responses (Mishra et al., 2022; Sanh et al., 2022; Wang et al., 2022). For instance, Sanh et al. (2022) crafted an instructional dataset by reformulating supervised datasets with various prompts to create P3. However, despite their effectiveness in NLP tasks, these LLMs often diverge from human-like interactions in chatbot applications.

To facilitate general-purpose LLM fine-tuning, researchers has create general-purpose instructional data by human annotation (Conover et al., 2023) and automatic approaches (Wang et al., 2023b; Taori et al., 2023). Recent work has further expanded the dataset size (Wu et al., 2023), language coverage (Li et al., 2023), and task types (Chaudhary, 2023; Yue et al., 2023).

With increasing capabilities of LLMs and availability of instruction datasets, researchers have aimed to imbue a single model with diverse capabilities. Sengupta et al. (2023) attempted to blend different instruction datasets without considering the data volume and task types. Longpre et al. (2023) suggested that increasing the number of tasks and instruction diversity can enhance performance. In contrast, Anand et al. (2023) excluded P3 from their fine-tuning dataset, seemingly to enhance alignment. Nevertheless, none of these papers systematically studied the impact of the instruction mixture on the resulting LLM.

Concurrent to our work, Wang et al. (2023a) fine-tuned LLaMA models on 12 instruction-tuning datasets separately. By evaluating those model on 7 tasks, they found that different datasets can enhance model performance on individual tasks. They further identified the optimal dataset combination, and trained a single model to achieve the best overall performance. Novel to this work, we classify the instructions and model skills into three types, and conduct a deep analysis of the influence of data mixture on the models.

3 Experimental Setup

Datasets We select Alpaca (Taori et al., 2023) as the *general instruction dataset* to align models, in the form of 52K instruction–response pairs. We use P3 (Sanh et al., 2022) as our *NLP task instruction dataset*, which is reformatted for a wide range of NLP downstream tasks using diverse human-written templates. Since the number of samples in each task varies vastly, we randomly sample 1K instances from each subtask formatted with several corresponding prompts for diversity, resulting in 660K samples. For *coding data*, we choose CodeAlpaca (Chaudhary, 2023), which is an instruction dataset focusing on code generation. It contains 20K samples in different programming languages. To ensure a balanced comparison, we randomly sample a 20K subset from each dataset. Examples are provided in Table 3 in the Appendix.

Evaluation We divide the evaluation into three parts: NLP benchmark performance, code generation, and alignment evaluation (i.e., chat ability evaluation). For NLP benchmarks, we use ARC (Clark et al., 2018), Winogrande (Sakaguchi et al., 2021), PIQA (Bisk et al., 2020), MMLU (Hendrycks et al., 2020), RACE (Lai et al., 2017), and HellaSwag (Zellers et al., 2019). For coding, we use HumanEval (Chen et al., 2021), which tests the pass rate of the generate codes. For alignment evaluation, we use the FLASK (Ye et al., 2023) framework to score model alignment. We keep the eight most frequent alignment skills from the original evaluation set, resulting in 1,180 samples. Then we employ GPT-4 to assess model responses to each instruction sample based on human-written principles. See Appendix B for details of these skills.

Models We fine-tune LLaMA-2 7B and 13B (Touvron et al., 2023) models for two epochs in a generative way as in Radford et al. (2018), using a linear scheduler with a 3% warmup rate and a batch size of 64. The maximum learning rate is

Model	Data	ARC	Wino-	PIQA	MMLU	Race	Hella-	Average	HumanEval	
		(challenge)	grande				Swag		@1	@10
LLaMA-2-7B	None	43.1	69.5	78.0	40.8	39.2	57.2	54.6	13.7	21.3
	A	47.8	67.6	78.2	42.2	44.5	**61.1**	56.9	13.5	17.1
	C	46.1	69.5	78.5	41.0	41.1	61.0	56.2	16.2	24.4
	P	49.6	**71.4**	**79.0**	**46.0**	43.5	59.4	**58.2**	4.6	7.9
	AC	47.1	66.9	78.1	40.4	44.2	59.7	56.1	**17.5**	**25.0**
	AP	48.4	70.0	78.1	43.8	42.9	58.5	56.9	13.8	17.7
	CP	48.0	71.3	78.4	44.9	44.4	60.7	57.9	16.8	20.1
	ACP	**49.7**	68.0	77.9	43.5	**44.6**	58.7	57.1	16.0	23.8
LLaMA-2-13B	None	48.6	71.9	79.2	**52.1**	40.7	60.1	58.8	15.4	26.2
	A	54.1	71.2	80.0	47.9	**47.1**	**65.6**	61.0	15.1	20.7
	C	49.7	73.4	**80.8**	51.5	45.4	63.6	60.7	17.9	24.4
	P	54.3	74.2	80.0	50.3	45.6	62.5	61.1	0.3	1.8
	AC	51.6	68.8	80.6	48.7	44.4	63.0	59.5	17.1	27.4
	AP	54.8	71.7	80.3	51.2	45.2	62.7	61.0	8.3	14.6
	CP	**55.4**	**74.6**	80.5	51.4	45.6	63.9	**61.9**	18.2	25.0
	ACP	54.4	71.5	80.0	50.0	**47.1**	63.1	61.0	**20.2**	**32.9**

Table 1: Results on NLP and code generation benchmarks. All experiments are done in a zero-shot setting. The best result is in **bold**, and the second best result is underlined.

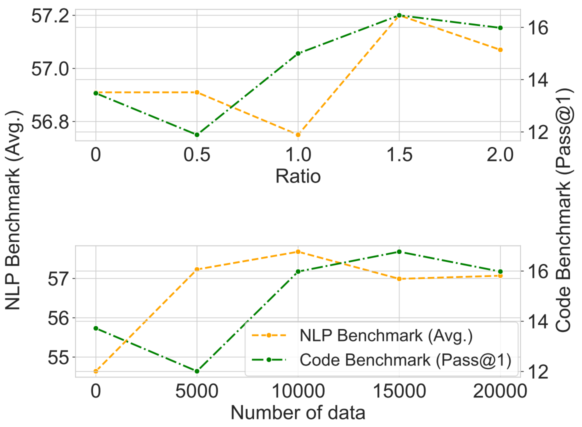


Figure 2: NLP benchmark scores (avg) and Code benchmark (HumanEval) scores for LLaMA-2-7B tuned with different mixing ratios and different numbers of instances. We keep the number of Alpaca instances constant at 20K and change the number of P3 and CodeAlpaca instances to get different ratios.

5×10^{-5}. The resources for training and evaluations are detailed in Appendix C.

4 Results

For the remainder of this paper, we denote the Alpaca, CodeAlpaca, and P3 datasets as A, C, P, respectively. For each model, we compare eight different data mixing strategies, denoted as None, A, C, P, AC, AP, CP, ACP, where *None* represents the vanilla model without fine-tuning, and each of the other settings represents the model fine-tuned with the corresponding dataset. For example, *AC* means the model is fine-tuned with both Alpaca and CodeAlpaca.

4.1 NLP Tasks and Code Benchmark Results

Table 1 shows the zero-shot results on the NLP and code generation benchmarks. Predictably *each specialized instruction dataset improves the performance on the benchmarks they are designed for*. In the no-mixture setting (comparing A, C, and P), models fine-tuned on P3 achieve the highest average score for NLP tasks, while models fine-tuned on CodeAlpaca excel in code generation benchmarks. Examining specific tasks reveals that *a model's performance on a specific task heavily relies on the similarity between the target task and the tasks it was fine-tuned on*. For instance, Alpaca fine-tuned models excel in Race and HellaSwag, which involve the story completion task, similar to the Alpaca instruction format. On the other hand, P3 fine-tuned models perform well on ARC and Winogrande, which involve multiple-choice QA and cloze tests, which are well represented in P3.

In the mixture setting, it's evident that *including specialized data consistently boosts model performance in corresponding benchmarks compared to models without such data.* For example, P, PA, PC, and PCA perform better than None, A, C, and CA on NLP downstream tasks. Focusing on the code benchmarks, *incorporating general instructions consistently improves coding performance*. For the 7B model, AC improves performance by $+1.28$ and $+0.61$ compared to C, while the improvements are -0.80 (outlier) and $+3.05$ for the 13B models. Another interesting finding is that the 13B models perform best with the ACP mixture, while the 7B models perform best with AC. This

Model	Data	Corr.	Fact.	Comm.	Compr.	Compl.	Insight.	Read.	Conc.	Avg.
LLaMA-2-7B	A	47.6	**55.4**	58.8	54.8	48.0	**50.4**	**88.0**	81.6	60.6
	C	48.8	52.0	58.4	52.0	40.2	46.2	83.8	78.4	57.4
	P	47.2	40.0	48.8	38.4	29.0	30.4	64.4	68.6	45.8
	AC	49.0	54.4	**59.6**	**56.4**	**48.2**	49.8	86.6	**85.6**	**61.2**
	AP	48.4	51.4	57.6	52.6	45.0	46.0	84.2	80.8	58.2
	CP	47.0	49.6	54.2	48.8	36.2	41.8	78.2	77.2	54.2
	ACP	**50.4**	53.0	59.0	53.8	47.2	46.8	85.0	81.8	59.6
LLaMA-2-13B	A	53.6	58.8	63.8	60.0	47.6	**55.2**	**89.2**	84.0	64.0
	C	**57.2**	58.8	61.0	57.8	43.8	52.4	85.6	82.2	62.4
	P	49.4	42.4	51.8	42.0	28.2	32.0	66.8	70.4	47.8
	AC	55.6	**61.0**	**66.6**	**61.2**	**51.4**	54.0	88.4	**86.6**	**65.6**
	AP	53.0	55.4	60.6	56.2	47.0	48.0	85.0	83.4	61.0
	CP	53.0	53.2	57.4	53.4	39.0	45.2	81.2	82.6	58.2
	ACP	51.6	55.6	61.8	57.0	47.0	48.6	87.0	83.0	61.4

Table 2: GPT-4 evaluation results on alignment skill assessment. We report eight dimensions: logical correctness, factuality, commonsense understanding, comprehension, completeness, insightfulness, readability, and conciseness, as well as average scores. Since the vanilla model cannot follow instructions, we exclude it from this table. The best result is in **bold**, and the second best result is underlined.

suggests that *larger models can better learn from varied instructions more effectively than smaller models*.

These findings highlight the importance of considering model size and target usage when designing the instruction mixture.

Mixing with Different Ratios While it is clear that mixing specialized instructions is vital for benchmark performance, how the mixing ratio correlates with the performance is also important. As Figure 2 shows, with the number of general instructions fixed to 20K, scores in both NLP task and code benchmarks first decrease and then increase as the ratio of specialized instructions increases. They both peak when the ratio is 1.5, and drop back slightly when the ratio is increased further to 2.0. We hypothesize that this is because the model overfits to the specialized instructions when there are too many such instructions.

Number of instances Figure 2 also shows the performance change with respect to the number of fine-tuning data instances. We mix each type of instruction with the same number. We find that the performance over both benchmarks plateaus when the number of instances is larger than 10K.

4.2 Alignment Skill Results

Table 2 shows the alignment skills results. We adopt the same setup as FLASK, using `GPT-4-0613` to access the alignment skills and scaling the scores to the range $[0, 100]$.

From Table 2 we make the following observations: (1) *All three types of instructions improve model alignment compared to the vanilla LLM.* Among these instructions, Alpaca stands out as the most effective. It contains general-purpose instructions and human-like responses, making it a better fit for aligning models with humans. (2) *While CodeAlpaca alone doesn't notably enhance alignment abilities, combining it with general instructions results in a substantial improvement of* $+0.6$ *(7B) and* $+1.6$ *(13B) points*; these improvements are mainly due to better compression, commonsense understanding, completeness, and conciseness. (3) *Mixing P3 data causes a drop of* -2.8 *(7B) and* -3.6 *(13B) in alignment skills,* suggesting that P3 has a negative impact on fine-tuning chatbot LLMs.

5 Conclusion

In this paper, we investigated different data mixing strategies in instruction fine-tuning. We measured models against diverse benchmarks and alignment skills. We find that general instructions provide better alignment as well as performance on NLP benchmarks, code instructions improve coding and alignment skills, while NLP task instructions hinder alignment skills when combined with other instruction types.

Limitations

Our work is subject to several limitations that should be addressed in future research. (1) We only use LLaMA-2 7B and 13B models in our experiments. Other models of varying sizes should be used to further verify our findings. We acknowledge that the model's behavior may vary with different sizes, and that usually, larger models have stronger capabilities, and hence may be able to handle more instructions without performance degradation. (2) In this paper, we limit our instruction dataset to 20K and mainly compare the 1:1 ratio of all instruction types. We leave the exploration of the impact of more instructions and mixing ratios to future work.

We acknowledge these limitations and propose that future work should focus on addressing them to help the community better understand the impact of instruction mixture on LLMs.

References

Yuvanesh Anand, Zach Nussbaum, Brandon Duderstadt, Benjamin Schmidt, and Andriy Mulyar. 2023. GPT4All: Training an assistant-style chatbot with large scale data distillation from GPT-3.5-Turbo. https://github.com/nomic-ai/gpt4all.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on Artificial Intelligence*, pages 7432–7439.

Sahil Chaudhary. 2023. Code Alpaca: An instruction-following LLaMA model for code generation. https://github.com/sahil280114/codealpaca.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try Arc, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free Dolly: Introducing the world's first truly open instruction-tuned LLM. https://github.com/databrickslabs/dolly.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations.

Hao Fu, Yao; Peng and Tushar Khot. 2022. How does GPT obtain its ability? tracing emergent abilities of language models to their sources. *Yao Fu's Notion*.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. Textbooks are all you need.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Haonan Li, Fajri Koto, Minghao Wu, Alham Fikri Aji, and Timothy Baldwin. 2023. Bactrian-X: Multilingual replicable instruction-following models with low-rank adaptation.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The Flan collection: Designing data and methods for effective instruction tuning.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. 2023a. Crosslingual generalization through multitask finetuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15991–16111, Toronto, Canada. Association for Computational Linguistics.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. 2023b. Crosslingual generalization through multitask finetuning.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. http://openai-assets.s3.amazonaws.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2022. Multitask prompted training enables zero-shot task generalization. In *ICLR 2022-Tenth International Conference on Learning Representations*.

Neha Sengupta, Sunil Kumar Sahu, Bokang Jia, Satheesh Katipomu, Haonan Li, Fajri Koto, William Marshall, Gurpreet Gosal, Cynthia Liu, Zhiming Chen, Osama Mohammed Afzal, Samta Kamboj, Onkar Pandit, Rahul Pal, Lalit Pradhan, Zain Muhammad Mujahid, Massa Baali, Xudong Han, Sondos Mahmoud Bsharat, Alham Fikri Aji, Zhiqiang Shen, Zhengzhong Liu, Natalia Vassilieva, Joel Hestness, Andy Hock, Andrew Feldman, Jonathan Lee, Andrew Jackson, Hector Xuguang Ren, Preslav Nakov, Timothy Baldwin, and Eric Xing. 2023. Jais and Jais-chat: Arabic-centric foundation and instruction-tuned open generative large language models.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. LLaMA 2: Open foundation and fine-tuned chat models.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023a. How far can camels go? exploring the state of instruction tuning on open resources.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Minghao Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Fikri Aji. 2023.

LaMini-LM: A diverse herd of distilled models from large-scale instructions.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. WizardLM: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. 2023. FLASK: Fine-grained language model evaluation based on alignment skill sets.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. MAmmoTH: Building math generalist models through hybrid instruction tuning.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

A Examples of Instruction Types

Table 3 list examples from Alpaca, CodeAlpaca, and P3 dataset.

B Alignment Skills Demonstration

The FLASK framework annotates each instruction with three skills that is needed to respond to the instruction. We select 8 most frequent skills and filter out instructions annotated with other skills, resulting 1,180 instructions in the evaluation set. The following are demonstrations of each alignment skill from the annotation prompt.

Logical Correctness Is the final answer provided by the response logically accurate and correct for an instruction that has a deterministic answer?

Factuality Did the model extract pertinent and accurate background knowledge without any misinformation when factual knowledge retrieval is needed? Is the response supported by reliable evidence or citation of the source of its information?

Commonse Understanding Is the model accurately interpreting world concepts for instructions that require a simulation of the expected result or necessitate commonsense or spatial reasoning?

Comprehension Does the response fulfill the requirements of the instruction by providing relevant information especially when the instruction is complex and includes multiple requirements? This includes responding in accordance with the explicit and implicit purpose of given instruction.

Completeness Does the response provide a sufficient explanation? Comprehensiveness and thoroughness of the response should be considered, which depends on the breadth of topics covered and the level of detail provided within each topic.

Insightfulness Is the response creative, original or novel, including new perspectives or interpretations of existing information?

Readability Is the response structured to promote readability and coherence? Does the response exhibit excellent organization?

Conciseness Is the response presented in a concise manner for the reader without any unnecessary information?

The prompt for alignment skill assessment are provided in Figure 3. For how a response corresponds to a specific level of an alignment skill and other details, please refer to their repository.[2]

C Resources

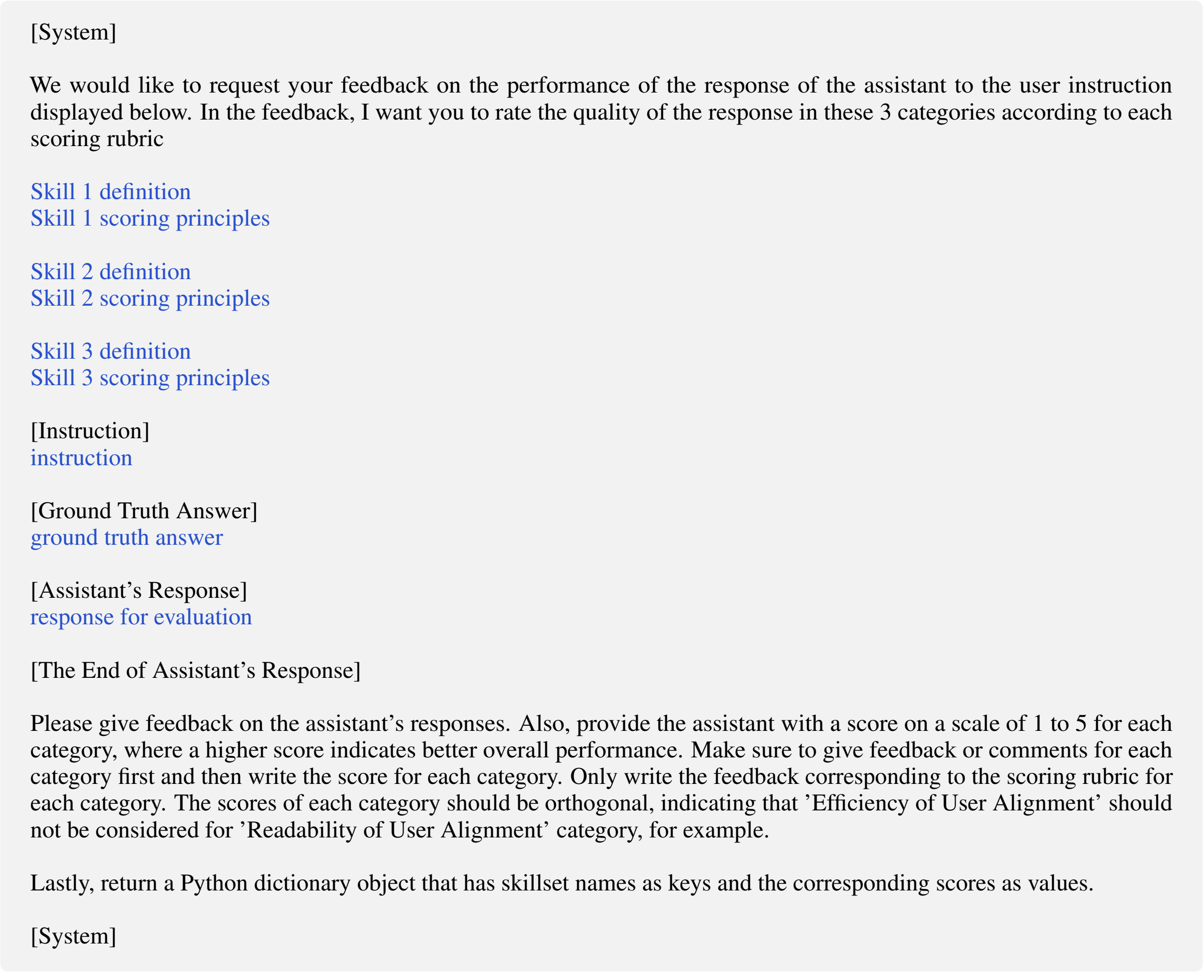
We use 4×A100 to train LLaMA-2-7B and 8×A100 to train LLaMA-2-13B. Each 20k data takes about 2 hours. For all experiments, training takes about 288 A100 GPU hours.

For evaluation, we use GPT-4, where each input has an average of 950 tokens and each output has an average of 293 tokens. All evaluations cost about $760.

[2] https://github.com/kaistAI/FLASK

Source	Example
Alpaca	**Instruction**: List 5 reasons why someone should learn to code. **Response**: 1. High demand for coding skills in the job market \n2. Increased problem-solving and analytical skills \n3. Ability to develop new products and technologiesn \n4. Potentially higher earning potential \n5. Opportunity to work remotely and/or freelance
CodeAlpaca	**Instruction**: Write a function to find the number of distinct states in a given matrix. Input: matrix = [[1, 0, 0],\n [1, 0, 1],\n [1, 1, 1]] **Response**: def find_num_distinct_states(matrix):\n states = set()\n for row in matrix:\n state = "".join([str(x) for x in row])\n states.add(state)\n return len(states)
P3	**Instruction**: Answer the following question.\nAnna Kournikova, Michelangelo, ILOVEYOU, Melissa, and Stuxnet are all examples of what? **Response**: Computer virus/worm

Table 3: Examples from Alpaca, CodeAlpaca, and P3.



[System]

We would like to request your feedback on the performance of the response of the assistant to the user instruction displayed below. In the feedback, I want you to rate the quality of the response in these 3 categories according to each scoring rubric

Skill 1 definition
Skill 1 scoring principles

Skill 2 definition
Skill 2 scoring principles

Skill 3 definition
Skill 3 scoring principles

[Instruction]
instruction

[Ground Truth Answer]
ground truth answer

[Assistant's Response]
response for evaluation

[The End of Assistant's Response]

Please give feedback on the assistant's responses. Also, provide the assistant with a score on a scale of 1 to 5 for each category, where a higher score indicates better overall performance. Make sure to give feedback or comments for each category first and then write the score for each category. Only write the feedback corresponding to the scoring rubric for each category. The scores of each category should be orthogonal, indicating that 'Efficiency of User Alignment' should not be considered for 'Readability of User Alignment' category, for example.

Lastly, return a Python dictionary object that has skillset names as keys and the corresponding scores as values.

[System]

Figure 3: Alignment skill assessment prompt (from FLASK (Ye et al., 2023)). The blue parts are filled by corresponding content.