Understanding the Instruction Mixture for Large Language Model Fine-tuning

Anonymous ACL submission

Abstract

While instructions fine-tuning of large language models (LLMs) has been proven to enhance performance across various applications, the influence of the instruction dataset mixture on LLMs has not been thoroughly explored. In this study, we classify instructions into three main types: NLP downstream tasks, coding, and general chatting, and investigate their impact on LLMs. Our findings reveal that specific types of instructions are more beneficial for particular uses, while it may cause harms to other aspects, emphasizing the importance of meticulously designing the instruction mixture to maximize model performance. This study sheds light on the instruction mixture and paves the way for future research.

1 Introduction

Supervised fine-tuning (SFT) has been proven to be an effective approach to align large language models (LLMs) with human instructions, enhancing downstream task performance, and facilitating code generation (Touvron et al., 2023; Muennighoff et al., 2023; Gunasekar et al., 2023). Previous research has demonstrated that directly transforming NLP downstream tasks (e.g., coreference resolution) into instruction-response pairs, followed by fine-tuning models on such datasets, leads to improvements in performance across various benchmarks (Sanh et al., 2022). Recent studies show that general-purpose instructions can enhance both the performance of LLMs on downstream tasks and their alignment with human intents (Taori et al., 2023; Touvron et al., 2023; Zeng et al., 2023). Additionally, incorporating code datasets has been shown to enhance a model's logical reasoning abilities (Fu and Khot, 2022; Gunasekar et al., 2023).

As LLMs continue to advance, researchers are keen to endow a single model with diverse abilities. One straightforward approach is to combine multiple specialized instruction datasets. For instance, to enhance a model's code generation ability, we might incorporate CodeAlpaca into the SFT data (Chaudhary, 2023). However, there is no standard way of selecting instruction datasets explicitly. The process of mixing different datasets and understanding how various instruction types interact with each other remains underexplored.

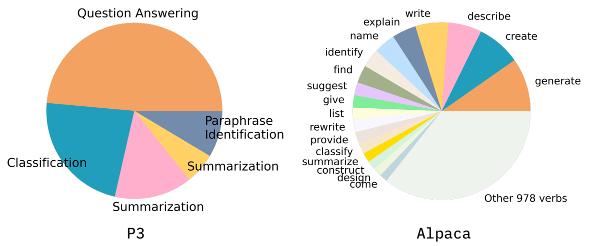

Figure 1: Instruction type distribution of P3 and Alpaca. For P3, the statistics come from the original dataset. For Alpaca, we utilize a dependency parsing approach to extract the root verb of each instruction.

In this paper, our focus is on evaluating the model's performance in three key areas: NLP downstream task performance, coding ability, and chat capabilities. We aim to investigate how different distributions of instruction datasets can impact model performance across these diverse aspects. Our selected representative SFT datasets are P3 (Sanh et al., 2022) for NLP downstream tasks, CodeAlpaca (Chaudhary, 2023) for code generation, and Alpaca for general-purpose instructions.

According to Figure 1, P3 mainly contains five types of tasks (including QA, classification, summarization, etc.), while Alpaca can be classified into about 1K types based on the root verb of each instruction, where the top 3 root verbs are *generate, create* and *describe*. Additionally, codeAlpaca is exclusively focused on code-related tasks. Notably, when compared to the general instructions in Alpaca, instruction-response pairs in both P3 and CodeAlpaca exhibit narrower variations, whether in format or content.

We conduct experiments with eight different mixture settings involving these instruction datasets, assessing model performance in NLP downstream tasks, coding proficiency, and alignment skills (i.e., chat abilities). Our extensive experiments yield the following insights:: (1) Using a single type of STF data consistently improves the performance of the model on the corresponding task, while all three types of instructions can be used to improve NLP downstream task performance. (2) Incorporating instructions that are simply reformatted from NLP downstream tasks (e.g., P3) downgrades the model's alignment skills, resulting in a worse chat experience. (3) Exploiting code instructions can improve the model's coding ability and boost the alignment skills.

Based on our findings, we suggest that researchers should carefully design the instruction mixture to maximize the model's performance on the target usage while taking model size into consideration. We also appeal to the community to evaluate LLMs not only based on the benchmark performance but also on the alignment skills.

2 Related Work

Recent work has demonstrated that vanilla LLMs can be effective at following general language instructions if tuned with instructions alongside their corresponding outcomes (Mishra et al., 2022; Sanh et al., 2022; Wang et al., 2022). To construct such instructional datasets, researchers have used a variety of approaches. For example, Sanh et al. (2022) reformat a large set of supervised datasets using multiple prompt templates to create P3, a dataset of instruction-response pairs covering a wide range of NLP downstream tasks. Even though such LLMs perform well on NLP downstream tasks, they do not align well with human behavior as chatbots.

To facilitate the general-purpose LLMs finetuning, Conover et al. (2023) introduced a high-quality human-annotated instruction dataset tailored for LLMs. However, this method is resource-intensive and lacks scalability. To address such issue, Wang et al. (2023) and Taori et al. (2023) firstly use an automatic data collection approach to collect a large-scale general instruction dataset. Based on the proposed datasets or approaches, later work expands the dataset size (Wu et al., 2023), language coverage (Li et al., 2023), and task types (Chaudhary, 2023; Yue et al., 2023).

With the increasing capability of LLMs and availability of instruction datasets, researchers aim to imbue a single model with diverse capabilities. Sengupta et al. (2023) have attempted to blend different instruction datasets without considering the data volume and task types. Longpre et al. (2023) propose 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 skills.

Nevertheless, none of these works systematically investigate the impact of instruction mixture on LLMs. Our work aims to find the impact of mixing together different instructions to align models.

3 Experimental Setup

SFT Datasets We select Alpaca (Taori et al., 2023) as the *general instruction dataset* for aligning models, which contains 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 data 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 with different programming languages. To ensure a balanced comparison, we utilize a 20K subset from each type of dataset, randomly sampled. Unless explicitly mentioned, the experiments and discussions will be based on these subsets for the rest of this paper.

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) datasets. For coding, we use HumanEval (Chen et al., 2021), which tests the pass rate of generated codes. For alignment evaluation, we utilize FLASK (Ye et al., 2023) framework to score models' alignment skills. 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 models' responses to each instruction sample based

Model	Data	ARC (challenge)	Wino-grande	PIQA	MMLU	Race	Hella-Swag	Average	HumanEval @1	HumanEval @10
LLaMA-2-7B	None	43.09	69.53	77.97	40.81	39.23	57.20	54.64	13.72	21.34
	A	47.78	67.64	78.24	42.19	44.50	**61.09**	56.91	13.48	17.07
	C	46.08	69.46	78.50	40.99	41.05	60.96	56.17	16.22	24.39
	P	49.57	**71.43**	**79.00**	**45.98**	43.45	59.44	**58.15**	4.63	7.93
	AC	47.10	66.93	78.13	40.42	44.21	59.70	56.08	**17.50**	**25.00**
	AP	48.38	70.01	78.07	43.84	42.87	58.46	56.94	13.84	17.68
	CP	47.95	71.27	78.40	44.91	44.40	60.69	57.94	16.77	20.12
	ACP	**49.66**	68.03	77.86	43.52	**44.59**	58.73	57.07	15.98	23.78
LLaMA-2-13B	None	48.55	71.90	79.16	**52.12**	40.67	60.12	58.75	15.43	26.22
	A	54.10	71.19	80.03	47.86	**47.08**	**65.58**	60.97	15.06	20.73
	C	49.66	73.40	**80.79**	51.50	45.36	63.63	60.72	17.87	24.39
	P	54.27	74.19	80.03	50.30	45.55	62.46	61.13	0.30	1.83
	AC	51.62	68.75	80.58	48.68	44.40	62.97	59.50	17.07	27.44
	AP	54.79	71.74	80.30	51.15	45.17	62.72	60.98	8.29	14.63
	CP	**55.38**	**74.59**	80.52	51.42	45.55	63.85	**61.89**	18.23	25.00
	ACP	54.44	71.51	80.03	49.98	**47.08**	63.14	61.03	**20.24**	**32.93**

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.

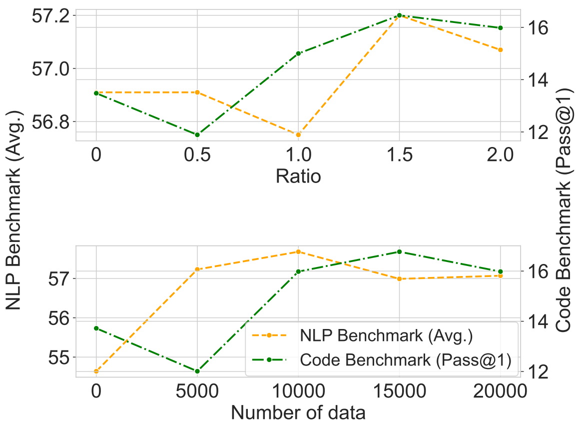

Figure 2: NLP benchmark scores (avg) and Code benchmark (HumanEval) scores for LLaMA-2-7B tuned with different mixing ratios and different number of data. We keep the number of Alpaca to 20K and change the number of P3 and CodeAlpaca to get different ratios.

on human-written principles. See appendix A for demonstrations of these skills.

SFT Setup We employ LLaMA-2 (7B, and 13B) models (Touvron et al., 2023). We fine-tune the models for two epochs in a generative way as in Radford et al. (2018). We use a linear scheduler with a 3% warmup rate and a batch size of 64. The maximum learning rate is 5×10^{-5}.

4 Results

For short, we denote 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. We use the same naming convention for the rest of the paper.

4.1 NLP Tasks and Code Benchmark Results

Table 1 shows the zero-shot results on NLP and code generation benchmarks. Notably *each type of specialized instructions improve 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 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 story completion task similar to the Alpaca instruction format. On the other hand, P3 fine-tuned models outperform in ARC and Winogrande, which involve multiple-choice QA and cloze tests, aligning with P3's data.

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 code benchmarks, *incorporating general instructions*

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 accessment. We report eight dimensions, i.e., logical correctness, factuality, commonsense understanding, comprehension, completeness, insightfulness, readability, and conciseness, as well as average scores. Since vanilla model cannot follow instructions, we exclude its result here. The best result is in **bold**, and the second best result is underlined.

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 intereting finding is that the 13B models achieve their best results with the ACP mixture, while the 7B models perform best with AC. This suggests that *larger models have greater capacities and can better leverage various instructions*.

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

Mixing with Different Ratios Despite knowing mixing specialized instructions are vital for better benchmark performance, how the mixing ratio correlates with the performance is also important for the best training strategy. As Figure 2 shows, given the number of general instructions fixed to 20K, scores of both NLP task benchmarks and code benchmarks first decrease and then increase as the ratio of specialized instructions increases. They both reach the maximum when the ratio is 1.5, while slightly decrease when the ratio continues to increase to 2.0. We think this is because the model is overfitted to the specialized instructions when there are too many such instructions.

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

4.2 Alignment Skills Results

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

From Table 2 we have the following findings: (1) *All three types of instructions improves 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 significantly 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 attributed to better compression, commonsense understanding, completeness, and conciseness. (2) *Mixing P3 data causes a drop of -2.8 (7B) and -3.6 (13B) in average alignment skills.* This indicates that P3 tends to have 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 with diverse benchmarks and alignment skills. We find that general instructions provide better alignment skills as well as performance on NLP benchmarks, code instructions improve coding and alignment skills, while NLP task reformated instructions hinder alignment skills when combined with other instruction types.

Limitation

Our work is subject to several limitations that should be addressed in future research:

- We only use LLaMA-2 7B and 13B models in our experiments. Other models in various sizes can be used to verify our findings. We acknowledge that the model's behavior may vary with different sizes, usually, larger models have better capabilities, and hence may be able to handle more instructions without performance drop in any evaluation setting.

- 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*, volume 34, 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*.

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.

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. 2023. 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.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

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, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. 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.

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.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023. GLM-130b: An open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations (ICLR)*.

A 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?

For how a response corresponds to a specific level of an alignment skill and other details, please refer to their repository [1].

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