Prompt Weight Experiments for LLM Instruction Fine-Tuning

Mathew Huerta-Enochian
EQ4ALL
11 Nonhyeon-ro 76-gil, Gangnam-gu, Seoul
`mathew@eq4all.co.kr`

Abstract

We present a small study analyzing how weighting prompt token classification loss affects the performance of 7B-size LLaMA models fine-tuned on instruction tasks. We recreated Stanford's Alpaca experiment with both LLaMA 1 and LLaMA 2 using multiple instruction datasets. We found that models fine-tuned on our short-completion dataset benefited most from a small prompt loss weight of ~ 0.1 during training while models fine-tuned on long-completion datasets was unaffected by prompt loss weight.

1 Introduction

Due to the recent successes in language modeling, especially those of large language models (LLMs) and instruction-following training, the amount of research and number of applications devoted towards language modeling has skyrocketed. In 2023, the number of LLM-related articles submitted to arXiv.org climbed from less than 1% at the end of 2022 up to 5% and 6% at the end of 2023 (for reference see figure 1).[1] However, while machine learning solutions advance by leaps and bounds, our understanding of them and standards of practice are slow to catch up.

To train LLMs for instruction tasks, a prompt and target completion are concatenated together and the model learns its parameters by optimizing for next-token maximal likelihood classification. HuggingFace's Transformers library (Wolf et al., 2020), the de facto library for training LLMs, allows users to mask select tokens when calculating prediction loss. Token optimization in Transformers is therefore binary—either a token's classification loss is used for optimization or it is not. As such, researchers using Transformers generally use only the completion token classification loss (by masking the loss from the prompt tokens) or use loss from all tokens.

[1]"LLM-related articles" were identified by matching any of the following search terms in any search field on arXiv (search is not case dependant): LLM, "Large Language Model", GPT, OpenAI, "Open AI".

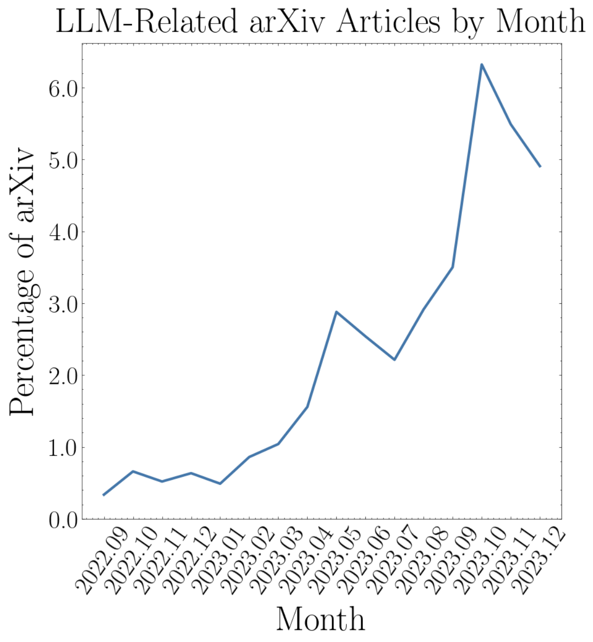

Figure 1: Percentage of total arXiv articles related to LLMs. The percentage increases from 2022 through 2023 as interest and research in LLMs grew.

On the other hand, some libraries implement explicit loss weighting for prompt tokens. BloomAI's (Blo) API supports a `prompt_loss_weight` parameter used to re-weight the prompt token losses, using a default value of 0.01 if not specified. OpenAI (Ope) also supported a similar `prompt_loss_weight` parameter in their API, but it was officially removed along with the v1 fine_tune API in early January, 2024. OpenAI's `prompt_loss_weight` also used a default value of 0.01 with the following parameter explanation: *"The weight to use for loss on the prompt tokens. This controls how much the model tries to learn to generate the prompt (as compared to the completion which always has a weight of 1.0), and can add* *a stabilizing effect to training when completions are short. If prompts are extremely long (relative to completions), it may make sense to reduce this weight so as to avoid over-prioritizing learning the prompt."*

We could not find a study validating OpenAI's claim nor could we find existing literature evaluating prompt loss weighting (PLW) for fine-tuning LLMs. How does this parameter affect training? How important is it for model performance on instruction tasks?

This research is a targeted evaluation of how utilizing PLW for LLM instruction fine-tuning affects downstream LLM performance. We base our experiments on the Alpaca (Taori et al., 2023) instruction fine-tuning experiment, and results should be interpreted against this context. We make the following contributions:

- We show that PLW has a statistically significant relationship with model performance on downstream tasks when training on short-completion instruction data. This relationship is negatively quadratic with performance increasing for small PLW to a maximum at some critical value and decreasing for PLW beyond the critical value.

- We show that PLW has no significant effect on downstream performance for models trained on the original Alpaca dataset and the cleaned Alpaca dataset. We conclude that PLW as a parameter (including prompt loss masking) can be safely ignored when training on long-completion data.

2 Motivation and Hypotheses

We define *instruction data* as one or many instances of structured text data with each instance containing an instruction text, an optional context or input text, and a target completion text. Generally, models trained for instruction completion are given the instruction and input text and trained to predict the completion. For the rest of the paper, we use the term *prompt* to refer to the concatenation of instruction and input text.

We define two types of general datasets. *Short-completion* data refers to data where the average prompt length is longer than the average completion length. On the other hand, *long-completion* data refers to data where the average prompt length is shorter than the average completion length.

For fine-tuning on short-completion data, we postulate that using a non-zero PLW would help the model learn better sequence generation, potentially generalizing to longer completions. This would suggest a positive relationship between pLW and generation performance.

However, unless the model is expected to generate prompt-like data in downstream tasks, the training and downstream use cases will not match. I.e., training the model to generate prompt tokens would be most useful for generating prompt tokens (such as for generating instruction data). Due to this train-test mismatch, we postulate that PLW also has a negative influence on generation performance.

Limiting PLW to the range of [0, 1], we postulate that there is a critical value λ for PLW with $0 <= \lambda <= 1$. For PLW less than λ, the positive effect dominates the negative effect and for values greater than λ, the negative effect dominates the positive effect. If $\beta = 0$, then PLW's contribution to model performance is strictly negative. Conversely, if $\lambda = 1$, then PLW contributes strictly positively. Thus, we expect that this relationship could be modeled by either a negative quadratic, a positive linear, or negative linear model, depending on the critical point λ.

We further postulate that λ is lower for long-completion fine-tuning data and higher for short-completion fine-tuning data. Note that in practice, λ would be dependent on a number of factors including the training data, downstream tasks, evaluation metrics, pre-trained model, etc., and would not be a fixed intrinsic vaue.

We will test the relationship between PLW and model instruction-following performance using three datasets that we characterize by their completion-prompt length ration, where the dataset with the highest completion-prompt length ratio is over 7 and the dataset with the lowest is less than 1. We expect the threshold λ to be near 1 for long-completion data and > 0 for the short-completion data. See section 3 for more details about the data.

To test this, we present the following hypotheses to be tested three times, once per a dataset.

Null Hypothesis ($\mathbf{H_0}$)
Prompt loss weight has no relationship with model performance on our target tasks.

Alternative 1 ($\mathbf{H_1}$: Linear Relationship)
Prompt loss weight has a ***linear*** *relationship with model performance on our target tasks.*

Alternative 2 ($\mathbf{H_2}$: Quadratic Relationship)

*Prompt loss weight has a **quadratic** relationship with model performance on our target tasks.*

3 Methodology

To evaluate the effect of instruction fine-tuning with PLW on generation performance, we recreated the Alpaca experiment using multiple parameter settings and evaluated the fine-tuned models on several benchmarks. We use the original Alpaca code and Transformers library, modified to allow setting PLW during training. We then experiment with training on six PLWs, two different pre-trained language models (PTLMs), and three different datasets for a total of thirty six experimental training runs. Training was performed exactly as per the original Alpaca experiment. We used the hyperparameters suggested by the authors,[2] modifying only the three experimental parameters (PLW, PTLM, dataset).

3.1 Parameter: Prompt Loss Weight

We performed a set of experimental training runs for each the following PLWs:
{0.0 , 0.01 , 0.05 , 0.1 , 0.5 , 1.0}
Note that 0.0 is identical to the masking used in the original Alpaca project and 1.0 is equivalent to the unmasked training (i.e., the default behavior of the Transformers library when token masks are not specified).

Though existing literature is sparse, given the little that is available and the default weight values from BLoomAI and OpenAI APIs, we chose to focus on the PLWs listed above. Though they do not provide their reasoning, Kozachek (2023) reported results fine-tuning GPT-3 with a `prompt_loss_weight` of 0.1. Dodgson et al. (2023) reported using the default value of 0.01 when fine-tuning GPT models. Wang et al. (2023) reported that a PLW of 0 performed best for them when working on the Self-Instruct framework. Interestingly, Wutschitz et al. (2023) reported hyperparameter search results for next-sentence-prediction on Elsevier data using PLWs of 0.1 and 0.5 and found 0.5 to give the best results.

3.2 Parameter: Pre-Trained Language Model

We use both LLaMA V1 (Touvron et al., 2023a) to recreate the original Alpaca experiment and LLaMA V2 (Touvron et al., 2023b) to provide more relevant results with contemporary research.

The LLaMA models are used for two reasons. The first is for continuity with the Alpaca experiment. Using the same model line puts results into a context with which most researcher are familiar and improves comparability between results. The second is that 7-billion parameter models perform at or near SOTA on many tasks while being more accessible than larger models.

3.3 Parameter: Fine-Tuning Dataset

In addition to running all experiments with the original Alpaca prompt data, we also complete training runs using two other datasets: the cleaned prompt dataset from AlpacaDataCleaned[3], and a custom version of the cleaned prompt dataset that has been modified to have short completion sequences.

We refer to these datasets as AlpacaData, AlpacaDataCleaned, and AlpacaDataShort, respectively.

AlpacaData was generated using the Self-Instruct approach to demonstrate creating an instruction-following language model on an academic budget.

AlpacaDataCleaned is being created by manually cleaning AlpacaData and has recently adopted data from the GPT4 LLM dataset (Peng et al., 2023). AlpacaDataCleaned includes includes fixes for the following issues in AlpacaData: hallucinations, merged instructions, empty outputs, empty code examples, instructions to generate images, N/A outputs, inconsistent input fields, wrong answers, nonsensical instructions, and extraneous control characters. According to the curator, cleaning is ongoing, so please note the hash reference in footnote 3.

We generated AlpacaDataShort from AlpacaDataCleaned by swapping instructions and completions for all instances where completion length was greater than prompt length. To swap text fields, we used one of two strategies that both rephrase instruction-input-completion instances as instruction-prediction tasks. If the instance has no input field, we used "`Predict the prompt that generated the following AI output.`" as the new instruction field. If the instance has an input field, we used "`Predict the prompt that generated the below AI output given the`

[2] See the "Fine-tuning" subsection of the Stanford Alpaca's readme: `github.com/tatsu-lab/stanford_alpaca/tree/761dc5bfbdeeffa89b8bff5d038781a4055f796a`

[3] We use the file named "alpaca_data_cleaned.json" from this repository and commit `github.com/gururise/AlpacaDataCleaned/tree/791174f63efa9aa54ca6c38639dc6e5bb5657d80`

Data	Instruction	Input	Completion	Total Tokens	Completion/Prompt
AlpacaData	13.40 (4.97)	6.25 (14.38)	64.51 (64.85)	4,376,482	3.27
AlpacaDataCleaned	14.21 (9.95)	6.42 (17.65)	162.74 (150.89)	9,490,837	7.83
AlpacaDataShort	16.93 (13.10)	162.34 (151.69)	14.62 (10.99)	10,035,667	0.081

Table 1: Mean instruction, input, and completion tokenized sequence lengths across each dataset. Standard deviations for relevant fields are included in parentheses.

Task	Version	N Shots
ARC Challenge*	0	25
PIQA	0	0
TruthfulQA (MC1)*	1	0
WinoGrande*	0	5

Table 2: Benchmark tasks used for evaluation.
*Task also used in HuggingFace's Open LLM Leaderboard.

`following context. Context: <input>`" with `<input>` replaced with the original input field, as the new instruction field. In all cases we then used the original completion as the new input field and the original instruction as the new completion field.

Descriptive statistics for tokenized instruction, input, and completion sequences are presented in table 1. Note that AlpacaDataCleaned is completion-dominated at a ratio of 7.83 while AlpacaDataShort is prompt-dominated at a ratio of 0.081. It is also worth noting that both AlpacaDataCleaned and AlpacaDataShort are over twice the size of AlpacaData.

3.4 Evaluation

We evaluated the thirty six models on four instruction-based benchmarks: ARC Challenge (Clark et al., 2018), PIQA (Bisk et al., 2020), TruthfulQA (MC1) (Lin et al., 2022), and WinoGrande (Sakaguchi et al., 2019). All evaluations are performed with EleutherAI's Language Model Evaluation Harness. [4] For comparison, we used the same checkpoint that HuggingFace uses to run their Open LLM Leaderboard [5] and use the same number of shots for matching benchmarks. See table 2 for details on benchmark tasks.

By selecting multiple performance indicators, we can get a broader analysis of generation performance, but it increases experimental complexity. To keep our regression model simple, we will test up to three predictors: PLW, PLW^2, and the benchmark task as a categorical variable. With three predictor variables (including a categorical with four categories) and three different fine-tuning dataset scenarios, we will actually be testing fifteen different relationships. In our opinion, thirty six samples is on the low, but acceptable range. To reduce the chance of making Type I errors, we will use the Holm-Bonferroni method for p-value correction. Since we are using such a small sample size relative to the number of variables, it is crucial that statistical significance is reported correctly.

4 Results and Discussion

We trained thirty six models and evaluated each on the four benchmarks tasks.

A visualization of performance against `prompt_loss_weight` for benchmark, dataset, and PTLM is presented in figure 2. Visualization of a simple aggregate score via the unweighted mean of the four performance indicators is also presented in figure 3. We show the aggregate score as it gives a nice qualitative summary of the different models and data scenarios, but this score was not properly validated and is only used for visualization.

Furthermore, based on visual inspection of both the composite scores and each individual benchmark score, we observe that performance varies with PLW the most closer to zero than in the rest of the [0, 1] interval. Therefore, for visualization and regression, we transform the selected weights to be uniform on the interval [0, 1].

We next performed regression analysis for each of the three groups. To effectively capture potential quadratic and linear relationships between PLW and model performance (H_2 and H_1, respectively), we shifted and scaled the PLW values to cover [-1, 1].

Regression was then performed using the equation
"$\texttt{score} \sim \texttt{weight}^2 + \texttt{weight} + \texttt{C(benchmark)}$",

[4]See the LM Evaluation Harness here: github.com/EleutherAI/lm-evaluation-harness/tree/b281b0921b636bc36ad05c0b0b0763bd6dd43463

[5]At the time of this research, Huggingface's Open LLM Leaderboard found at huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard has instructions for reproducing their results by using EleutherAI's Language Model Evaluation Harness.

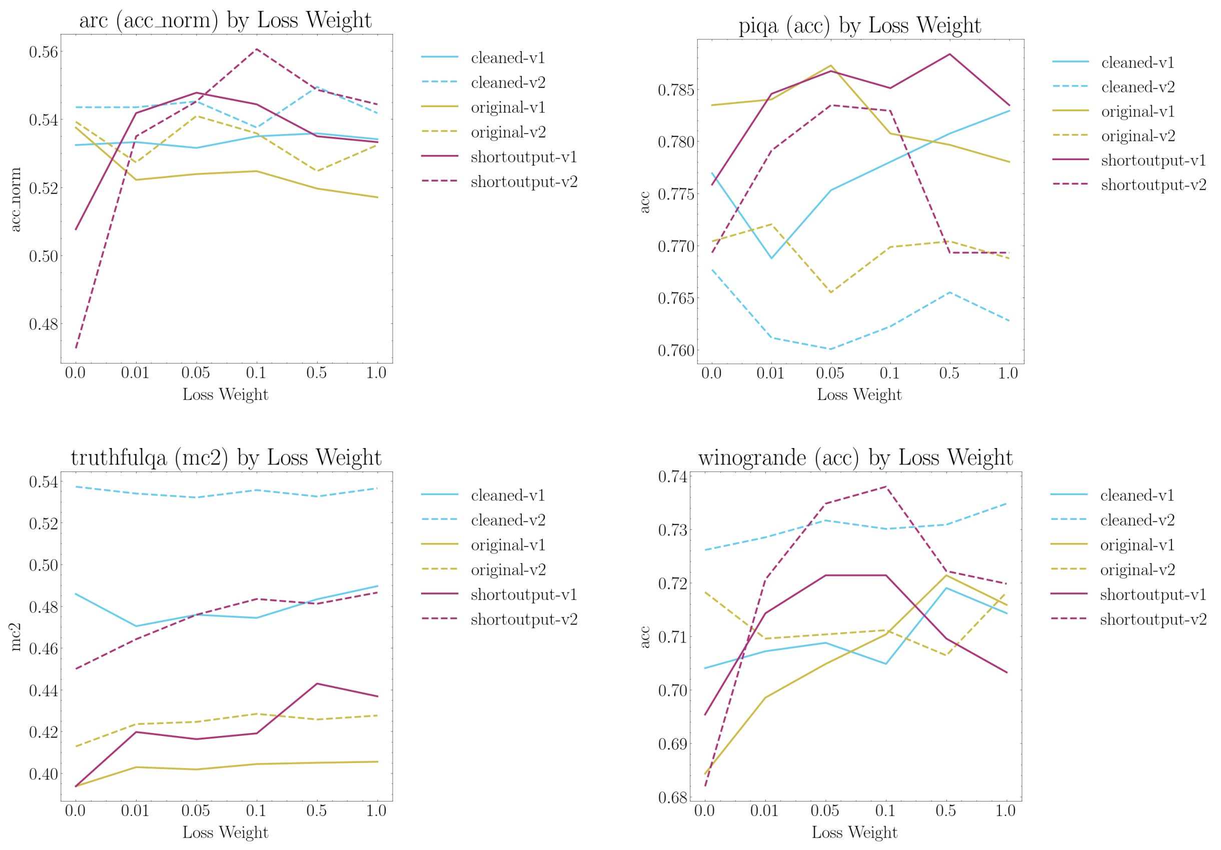

Figure 2: Performance by prompt loss weight for each benchmark, dataset, and PTLM. Best viewed in color.

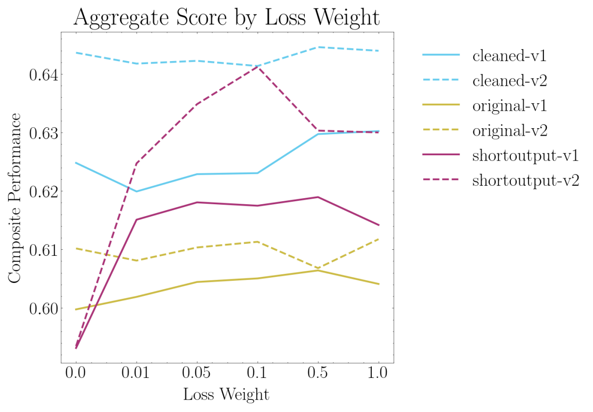

Figure 3: Aggregate performance by prompt loss weight for each dataset and LLaMA version. The aggregate score is the unweighted mean of the individual benchmark scores. Best viewed in color.

where `weight` and `weight`^{2} terms represent PLW linearly and quadratically, respectively.

P-values and the sign of parameter coefficients (for statistically significant indicators) are presented in table 3. Note that models trained on either AlpacaData or AlpacaDataCleaned did not show any significant correlation with PLW. However, models trained on the AlpacaDataShort (the short-completion version of the AlpacaDataCleaned) showed strong statistical significance, correlating positively with the linear PLW term and negatively with the quadratic PLW term.

We confirm that significance holds when correcting for multiple hypotheses using Holm-Bonferroni's method and a significance level of $\alpha = 0.05$. Since we had $m = 15$ p-values to test, we iteratively compared each p-value p_k with the scaled significance level at iteration k: $p_k < \frac{(\alpha=0.05)}{(m=15)+1-k} = \frac{0.05}{16-k}$. Trivially, all benchmark category p-values are lower than their scaled significance levels. We then checked `weight`^{2} for AlpacaDataShort. Its significance holds since $0.002 = p_{10} < \frac{0.05}{6} = 0.008\overline{3}$. Similarly, the p-value for the `weight` predictor in the AlpacaDataShort scenario is also significant. Obviously,

Dataset	P-Value (Sign)				
	weight	weight^2	PIQA	TruthfulQA	WinoGrande
AlpacaData	0.482	0.768	$< \mathbf{0.001}(+)$	$< \mathbf{0.001}(-)$	$< \mathbf{0.001}(+)$
AlpacaDataCleaned	0.507	0.588	$< \mathbf{0.001}(+)$	$< \mathbf{0.001}(-)$	$< \mathbf{0.001}(+)$
AlpacaDataShort	$\mathbf{0.004}(+)$	$\mathbf{0.002}(-)$	$< \mathbf{0.001}(+)$	$< \mathbf{0.001}(-)$	$< \mathbf{0.001}(+)$

Table 3: P-Values (and sign of the predictor coefficients if statistically significant) by task and training dataset. Statistically significant results in **bold**.

both `weight` and `weight`^{2} indicators for AlpacaData and AlpacaDataCleaned were thrown out using this method.

This confirms that for the AlpacaData and AlpacaDataCleaned scenarios, the null hypothesis was not rejected. However, for the AlpacaDataShort scenario, the null hypothesis was safely rejected and both H_1 and H_2 were verified as true.

This aligns with our expectations. Models fine-tuned on long-completion data see little-to-no effect from PLW since we limit weighting to be $<= 1$, and the loss from short-prompts will then be overshadowed by the classification loss from completion tokens.

5 Conclusion

This experiment tested the effects of instruction fine-tuning LLMs with prompt loss weight (PLW) on downstream performance.

We showed that performance is unaffected for our two long-completion datasets (the original Alpaca dataset and the cleaned Alpaca dataset) while performance has both a positive linear and a negative quadratic relationship with prompt loss weight on our short-completion dataset (which has a completion-prompt ratio of ~ 0.8).

Roughly, LLaMA 1 fine-tuned on the short-completion dataset showed a performance maximum at between $\sim$ 0.05 and 0.5 prompt loss weight, and LLaMA 2 fine-tuned on the short-completion dataset showed a performance maximum at prompt loss weight $= 0.1$.

Interestingly, the high score on the ARC Challenge, PIQA, and WinoGrande benchmarks were all from models trained on AlpacaDataShort. This suggests that given a good choice of prompt loss weight, long-completion training data may not be necessary for fine-tuning instruction-following language models. On the other hand, prompt loss weight showing no significant effect when using long-completion data suggests that long-completion data may be a safer option as we can ignore both prompt loss weighting and masking.

6 Limitations

1. This study evaluates relative length, not absolute length. A more thorough analysis would also contain datasets of fixed completion or prompt length and varying length ratios.

2. This study used a fixed seed $= 42$ for all experiments. Future analysis should extend experiments to models trained on multiple seeds.

Acknowledgements

This research was funded by Gwangju City's Artificial Intelligence Industry Cluster Agency (AICA) and was conducted with the support of my coworkers at EQ4ALL.

References

Bloom AI. https://www.bloomai.eu/. [Accessed 12-01-2024].

OpenAI. https://openai.com/. [Accessed 12-01-2024].

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

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*, abs/1803.05457.

Jennifer Dodgson, Lin Nanzheng, Julian Peh, Akira Rafhael Janson Pattirane, Alfath Daryl Alhajir, Eko Ridho Dinarto, Joseph Lim, and Syed Danyal Ahmad. 2023. Establishing performance baselines in fine-tuning, retrieval-augmented generation and soft-prompting for non-specialist llm users. *arXiv preprint arXiv:2311.05903*.

Diana Kozachek. 2023. Investigating the perception of the future in gpt-3, -3.5 and gpt-4. In *Proceedings of the 15th Conference on Creativity and Cognition*, C&C '23, page 282–287, New York, NY, USA. Association for Computing Machinery.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

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, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. pages 38–45. Association for Computational Linguistics.

Lukas Wutschitz, Boris Köpf, Andrew Paverd, Saravan Rajmohan, Ahmed Salem, Shruti Tople, Santiago Zanella-Béguelin, Menglin Xia, and Victor Rühle. 2023. Rethinking privacy in machine learning pipelines from an information flow control perspective. *arXiv preprint arXiv:2311.15792*.