Self-Select: Optimizing Instruction Selection for Large Language Models

Alexander Kyimpopkin†
University of Pennsylvania
alxkp@upenn.edu

Keshav Ramji†
University of Pennsylvania
keshavr@upenn.edu

† **denotes equal contributions.**

Abstract

The same question can often be presented in different ways, depending on the audience and the intent with which it is being posed. To determine whether large language models (LLMs) demonstrate preferences for one phrasing over another regardless of semantic content, we introduce *Self-Select*, a method for selection of a preferred instruction template, and generation of high-quality synthetic data samples. This algorithm makes use of a *meta-prompt* to decide on an instruction template, given a task and candidate templates then generates n new samples using the chosen template. We evaluate *Self-Select* on numerical reasoning and sentiment classification tasks, using a variety of instruction-tuned and base models, providing insights into their abilities and biases. We find that permuting the instruction template ordering in the prompt leads to vastly different choice distributions, suggesting that selections of a specific template can be attributed to inductive biases rather than semantic understanding, even after instruction-tuning.

1 Introduction

Large Language Models (LLMs) have demonstrated their ability to both generate seemingly novel data as well as critique generated responses ([11], [15], [25], [3]). At the same time, many models require large amounts of human labeled training data, motivating recent exploration of methods for synthetic data generation. That is, using model generated data to improve performance on a downstream task, largely by fine-tuning a model on a data mixture consisting of an existing corpus for the task and the synthetically generated data.

For a given task, instructions can be presented with several possible structures, which we call *templates*, and new data may be generated using many of these possible templates ([23]). Therefore, by framing this decision problem as one posed to the model for a particular task, we can gain valuable insights into the ability of the instruction-tuned models to distinguish between prompt templates, and biases which may attributed to the nature of their instruction-tuning.

In our work, we propose a new algorithm, ***Self-Select***, for generation of synthetic data samples corresponding to a model-selected instruction template. In the first module, *SELECT*, we introduce a meta-prompt for the model to consider the set of provided templates, and choose the template it perceives to be the best. Then, in the *GENERATE* module, we fit in-context exemplars to the chosen template, and prompt the model to generate new samples that follow the same structure as the exemplars. To ensure that the final n samples outputted are of sufficient quality, we propose verifying them relative to a reference benchmark, which may be defined as a metric (with an admittance threshold) or even a model-generated label of response quality, depending on the task. If the sample is deemed to be of insufficient quality, we propose prompting the model to refine its response

ATTRIB: Workshop on Attributing Model Behavior at Scale at NeurIPS 2023.

conditioned on the previous output, and the new response takes its place as a candidate. Upon the termination of *Self-Select*, n samples per task of interest will be obtained, which may be used to fine-tune the model, or be applied as exemplars for in-context learning.

We evaluate the *Self-Select* algorithm on two tasks – numerical reasoning (arithmetic) and sentiment classification, and benchmark the performance of each model in zero-shot and few-shot prompted settings, with and without model fine-tuning. Our results show that models are able to successfully identify the template it deems to be optimal, and can generate high-quality samples corresponding to the a hand selected prompt structure. This provides preliminary evidence of the ability for LLMs to optimize instruction selection via meta-prompts, building on the recent findings of [24].

Algorithm 1: *Self-Select* Algorithm

Inputs : Large Language Model $\mathcal{M}$
  $\mathcal{T} \leftarrow Set\ of\ tasks$
  $\mathcal{S}_t \leftarrow Set\ of\ candidate\ templates\ for\ task\ t$
  $\mathcal{X}_t \leftarrow Set\ of$ *in-context* $exemplars\ for\ initial\ generation\ for\ task\ t$
  $\mathrm{R}_t \leftarrow Set\ of$ *in-context* $exemplars\ for\ refinement\ for\ task\ t$
  $\mu_t :\ Response\ quality\ metric\ for\ task\ t\ with\ quality\ threshold\ \lambda_t$
  $mp,\ \ gp,\ \ rp$: *meta-prompt*, *generation-prompt*, *refinement-prompt*
  n: $Number\ of\ samples\ to\ generate\ per\ task$

  for each $task\ t \in \mathcal{T}$:
    $\tau = \mathcal{M}(mp \mid t, S_t)$ ▷ Meta-prompt yields selected instruction
    $\mathcal{F}, \mathcal{W} = \{\}$
    for each iteration $i \in 1, 2, \ldots n$:
      $y_i = \mathcal{M}(gp \mid \tau, \mathcal{X}_t)$ ▷ Sample responses, given template
      $\mathcal{W} = \mathcal{W} \cup \{y_i\}$
    end for
    while $|W| \neq \{\}$
      if max refinement iterations reached: ▷ 2nd Stopping criterion for refinement
        return $\mathcal{F}$
      $\gamma_i = \mu_t(y_i)$
      if $\gamma_i \geq \lambda_t$: ▷ Response quality check
        $\mathcal{F} = F \cup \{y_i\}$
        $\mathcal{W} = \mathcal{W} \setminus \{y_i\}$
      else:
        $y_i' = \mathcal{M}(rp \mid y_i, R_t)$ ▷ Response refinement
        $\mathcal{W} = W \cup \{y_i'\} \setminus \{y_i\}$
    end while
    return $\mathcal{F}$
  end for

Figure 1: The *Self-Select* algorithm and the assumed notation. Please see Section 2 (Algorithmic Approach) for a more comprehensive discussion of the method.

2 Algorithmic Approach

Given a set of possible instruction templates for a task, such as those manually curated in FLAN ([21]), *Self-Select* firstly chooses the instruction it deems to be most appropriate for the task, given the task description, generates new data which fit the structure of the template (with regards to the terms to be "filled in"), and then uses a quality control criterion to re-sample responses if they are of insufficient quality. This mechanism to determine when refinement is necessary may be defined several ways by the user, and can be specified for the particular task.

2.1 Instruction Template Selection

For the given task t, we wish to consider the set of potential candidate instruction templates, in order to select the best one; this set is denoted as $\mathcal{S}_t$. The *SELECT* module involves querying the model using a *meta-prompt*, given the $|\mathcal{S}_t|$ template options:

$$\tau = \mathcal{M}(mp \mid t, S_t) \tag{1}$$

We define the meta-prompt as follows, yielding a prompt index, which in turn is mapped to the particular template within the $\mathcal{S}_t$ set:

> "The following templates correspond to different problems. Choose which one best fits the problem above. Respond with Template: <NUM>"

It is to be noted that meta-prompting using the above query may be done with either zero-shot of few-shot settings, wherein one can provide demonstrations of a human annotator-chosen optimal template for framing a particular problem as an instruction, perhaps subject to certain desirable criteria. However, this is beyond the present scope of our empirical exploration, given the emphasis of this work is on the comparison of the behaviors between base models and instruction-tuned models, and their respective abilities to perform template selection as a means to elicit their instruction preferences.

2.2 Synthetic Data Generation and Refinement

The *GENERATE* module encompasses both generation of new samples (a user-defined value of n, per task) and refinement based on a user-defined metric, subject to a scoring threshold per sample. In this module, we sample a new response for the refinement prompt, conditioned on both the previous response and a small set of manually-curated examples for refinement for that particular task. For example, for arithmetic tasks, refinement only occurs when the provided answer is incorrect, and thus the in-context example set, R_t, consists of $\langle(x_i, y_i), (x_i, y_i')\rangle$ pairs, where y_i is an incorrect response and y_i' is correct.

$$y_i' = \mathcal{M}(rp \mid y_i, R_t) \tag{2}$$

3 Experimental Setup and Results

3.1 Numerical Reasoning

For numerical reasoning, we selected two-number addition with one to five digit numbers, using the prompt in Figure 2. We experimented with the Llama-2 7B and 13B variants, with and without chat-tuning [18], as well as MPT-7B and MPT-7B-instruct [17] and find that these smaller models struggle to perform instruction selection, instead generating seemingly random code segments. We believe this result to be tied to the use of curly brackets (i.e. $\{\}$) as a means to specify an argument to be filled in its place for a given template – this choice was done to maintain the ambiguity of the argument to be inserted, with an emphasis on the structure implied by the template. That being said, curly brackets most often occur in programming languages (hence the term "curly-bracket languages") such as C and C++. Thus, it is likely that models that have seen some program synthesis data would interpret the template as code, when presented with the options in the meta-prompt, and thus generate code in response.

We ran the template selection task 50 times per model (45 times for GPT-4 with unshuffled template choices, due to query rate limits), with the GPT-3.5, GPT-4, and Llama-2-70B-Chat models. We also performed this experiment with the aforementioned smaller models, but found their generations to be highly inconsistent and noisy with code samples, rather than a proper template selection. It is worth noting that of these three models, GPT-4 often elaborated on its logic even when unprompted to do so – behaviors in desirable templates from GPT-4's perspective include simplicity, straightforwardness, being "the most general", and clarity. As a result, on occasion, GPT-4 would output multiple potential options for its instruction of choice, based on its reasoning path to classify certain characteristics of groups of templates; for example, "Templates 0, 2, 4, and 6 provide explanatory text followed by a simple format for the problem."

```
SELECTION:The following templates correspond to different problems.

Choose which one best fits addition.  Respond with Template: <NUM>

Template 0 : Addition; Problem: {} + {} = ; Answer: {}

Template 1 : Addition;
 Problem: {} + {} = {}

Template 2 : Addition; Generate a problem following this template:
{} + {} = {}

Template 3 : Generate an addition problem using the following template:
num_1 + num_2 = answer

Template 4 : Generate an addition problem using the following template:
num_1 + num_2 = answer
 where num_1, num_2, and answer are integers

Template 5 : Generate an addition problem using the following template:
num_1 + num_2 = answer
 where num_1, num_2, and answer are numbers

Template 6 : Generate an addition problem using the following template:
num_1 + num_2 = answer
 where num_1, num_2, and answer are real numbers

 Choose the best template by returning its number.
```

Figure 2: Above is the prompt used for the numerical reasoning task, with 7 manually curated templates for performing addition, with slight differences in how the prolem is phrased.

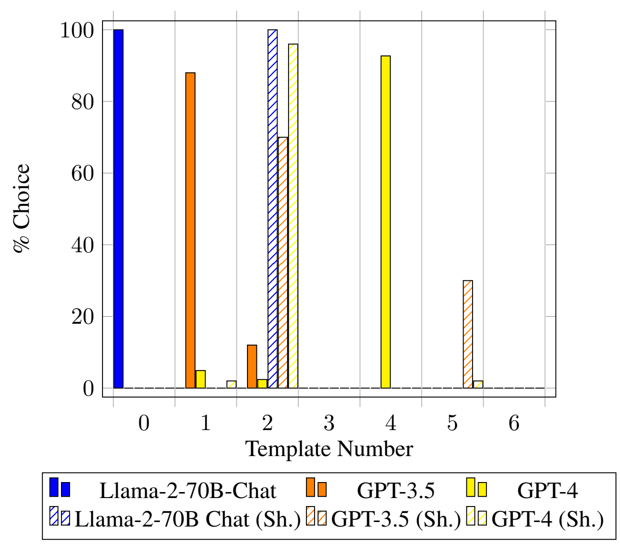

Figure 3: Results on instruction template selection for the numerical reasoning task. The bars with the striped lines correspond to the same model as the solid bar, but where the striped bars are results with shuffled instruction options, mapped back to their original template numbering.

Prior literature demonstrates LLMs' sensitivities to the order of choices in making decisions in multiple choice questions ([13]), in-context examples ([26]), and response critique and evaluation ([20]). Thus, we shuffled the instruction template options; if models maintained the same option as before this would indicate a degree of semantic understanding of the underlying templates. In both the unshuffled (Table 1) and shuffled (Table 2) settings, we found that small models had trouble following instructions, while the large instruction-tuned and/or chat-tuned models demonstrated a near deterministic preference for a specific template.

We additionally generated 9,600 examples using a similar template to the ones given above, validate the feasibility of our proposed *GENERATE* module. Our model was able to generate data which consistently tracked both the requested format and digit requirements for many of our samples, even with the Llama-2-7B-Chat model, in line with the current state of generative models.

3.2 Sentiment Classification

We experimented on the sentiment classification task using 10 templates corresponding to the IMDB dataset ([10]), from the FLAN ([21]) work. These include "How would you describe the sentiment of this review?", "Generate a movie review with answer sentiment.", and "Would you say this review is positive or negative?" (note that these are paraphrased). Similar to the numerical reasoning task, we also consider both unshuffled and shuffled template choices, to further examine models' consistency.

Template (Unshuffled)	GPT-3.5	GPT-4	Llama-2-70B-Chat
Template 0	**50%**	0%	0%
Template 1	20%	2.04%	**96%**
Template 2	30%	30.61%	0%
Template 3	0%	**67.35%**	0%
Template 4	0%	0%	0%
Template 5	0%	0%	4%
Template 6	0%	6%	0%
Template 7	0%	0%	0%
Template 8	0%	0%	0%
Template 9	0%	0%	0%
Total	100%	100%	100%

Table 1: Results on instruction template selection for the sentiment classification task, with unshuffled options, with 50 samples (49 for GPT-4, as indecisive responses were omitted).

GPT-4 similarly attempts to provide a line of reasoning for its choices: its criterion includes looking for the most direct, unambiguous, clear, and neutral template. The inclusion of "neutral" is particularly noteworthy, as it suggests GPT-4's inherent understanding of the requirements of the sentiment analysis task, and the objective to be unbiased in a certain direction with the instruction itself. We find that both GPT-3.5 and GPT-4 have a higher degree of variability for this task as compared to the numerical reasoning task, across 3 options.

Once again, we find that shuffling the instruction options results in a vastly different "preference" distribution, with only GPT-4 maintaining its primary choice from the unshuffled setting. Furthermore, we find that the smaller 7B and 13B models still struggle to produce outputs in the desired format (i.e. a template number) and hallucinate information, rendering them unable to consistently perform instruction selection (albeit, Llama-2-13B-Chat can still generate valid template numbers on rare occasion).

Template (Unshuffled)	GPT-3.5	GPT-4	Llama-2-70B-Chat
Template 0	0%	6%	0%
Template 1	0%	2%	0%
Template 2	0%	0%	0%
Template 3	**94%**	**92%**	42%
Template 4	0%	0%	0%
Template 5	0%	0%	0%
Template 6	6%	0%	**58%**
Template 7	0%	0%	0%
Template 8	0%	0%	0%
Template 9	0%	0%	0%
Total	100%	100%	100%

Table 2: Results on instruction template selection for the sentiment classification task, with shuffled options, mapped back to their original unshuffled numbering.

4 Related Work

Several prior works demonstrate the effectiveness of instruction tuning as a promising framework for yielding greater generalization to a wide variety of tasks ([21], [14], [12], [8], [2]). Recently, there has been growing interest in minimizing the amount of instruction-following data necessary to still obtain strong instruction-tuned models ([16], [5], [1]). On a similar lens, it has been shown that small but well-curated datasets can lead to strong alignment to human preferences ([27]). However, open questions remain on what level of semantic understanding, rather than simply superficial pattern following can be learned by instruction tuning [7]. Some models tuned via instruction tuning exhibit good performance in tasks in the specific corpus, but fail to meaningfully improve on robust benchmarks due to a lack of data [4]. Our work aims to continue the exploration into effective generation of high-quality synthetic instruction-following data, such that even a relatively small number of samples, which when distilled, can yield strong instruction-tuned models.

Chain-of-Thought (CoT) prompting was introduced in [22], which induces the model to generate step-by-step rationales, which provided insights into their ability to perform more complex, multi-step reasoning tasks. [6] found evidence of the effectiveness of zero-shot chain-of-thought prompting through "*Let's think step by step*"; the Optimization by Prompting (OPRO) algorithm introduced in [24], when applied to prompt optimization, shows that for the PaLM-2 model, "*Take a deep breath and work on this problem step-by-step*" is the most effective prompt for the GSM8K dataset. [9] uses symbolic reasoning chains as a means to induce faithful explanations, which motivates our future line of exploration into the reasons LLMs provide to attribute their instruction selection decisions.

5 Discussion

The distribution shifts present in these data as a result of shuffling the order raises questions about the causal factors behind LLM preferences for template selection and beyond. Self-introspection and critique as in Self-Refine [11] may clarify whether models exhibit semantic understanding, versus simple pattern matching via inductive biases, perhaps related to multiple choice problem in either the pre-training or instruction-tuning corpus, resulting in biased preferences. This may also shed new insights on the trustworthiness of knowledge distillation from larger instruction-tuned models. We would like to further explore the notion of refinement with different models given synthetic samples and deterministic quality metrics.

Further study into instruction selection from a semantic understanding perspective can dive into the role of self-attention; perhaps mechanistic interpretability could prove to be a valuable lens, in parallel to ties to cognitive neuroscience literature ([19]). Prior behavioral studies suggest that humans who excel in multiple-choice test scenarios, which are inherently similar to the instruction selection problem, appear to shift their attention to more relevant examples over time, and given the impact of the change in ordering on LLMs' performance, this shift does not appear to be replicated.

6 Conclusion

In this work, we present *Self-Select*, a procedure for large language models to select their preferred instruction template, and generate high-quality synthetic data, which may be used for self-training, knowledge distillation, or in-context learning. We find that large language models, even strong instruction-tuned models, are unable to consistently reason semantically about the structure and contents of their instructions especially in a permutation-invariant manner. Shuffling the order of templates led to substantial changes in the distribution of chosen templates for numerical reasoning. Before shuffling, each model had a strong preference for a different template, while after shuffling, they now expressed a preference for the same instruction. The distribution also shifted substantially upon shuffling for the sentiment classification task. In both tasks, at least one model demonstrated a near-deterministic preference among the template options. We thus conclude that in order for LLMs to exhibit semantic understanding they must be exposed to the same data in several orderings, motivating data augmentation strategies using permutations. By demonstrating the order dependence on the instruction selection outcomes, we hope for this work to spark further discussions on the biases and implications of knowledge distillation from instruction-tuned models on robustness.

Acknowledgements

The authors would like to thank Bronco AI for computational resources. We thank the anonymous reviewers for their feedback and suggestions.

References

[1] Yihan Cao, Yanbin Kang, and Lichao Sun. Instruction mining: High-quality instruction data selection for large language models, 2023.

[2] 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. Scaling instruction-finetuned language models, 2022.

[3] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing, 2023.

[4] Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The false promise of imitating proprietary llms, 2023.

[5] Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. Unnatural instructions: Tuning language models with (almost) no human labor, 2022.

[6] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2022.

[7] Po-Nien Kung and Nanyun Peng. Do models really learn to follow instructions? an empirical study of instruction tuning, 2023.

[8] 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. The flan collection: Designing data and methods for effective instruction tuning, 2023.

[9] Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning, 2023.

[10] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[11] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023.

[12] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[13] Pouya Pezeshkpour and Estevam Hruschka. Large language models sensitivity to the order of options in multiple-choice questions, 2023.

[14] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang,

Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization, 2021.

[15] William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators, 2022.

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

[17] MosaicML NLP Team. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023. Accessed: 2023-03-28.

[18] 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. Llama 2: Open foundation and fine-tuned chat models, 2023.

[19] Meng-Jung Tsai, Huei-Tse Hou, Meng-Lung Lai, Wan-Yi Liu, and Fang-Ying Yang. Visual attention for solving multiple-choice science problem: An eye-tracking analysis. *Computers Education*, 58(1):375–385, 2012.

[20] Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators, 2023.

[21] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2021.

[22] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2022.

[23] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions, 2023.

[24] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2023.

[25] Seonghyeon Ye, Yongrae Jo, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, and Minjoon Seo. Selfee: Iterative self-revising llm empowered by self-feedback generation. Blog post, May 2023.

[26] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR, 18–24 Jul 2021.

[27] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. Lima: Less is more for alignment, 2023.