Self-Select: Optimizing Instruction Selection for Large Language Models

Keshav Ramji[†]
University of Pennsylvania
keshavr@upenn.edu

Alexander Kyimpopkin[†]
University of Pennsylvania
alxkp@upenn.edu

[†] **equal contribution**

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 in performing instruction selection. We find that permuting the instruction template ordering in the prompt leads to vastly different choice distributions, suggesting that decisions may be influenced more by inductive biases than by 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 ([27], [32], [53], [14]). 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 ([49]). While the differences in these instruction formats may be evident to humans – after all, users may present the same question in various ways to a conversational agent – it is unclear the extent to which template selection influences the quality of the generated data, nor whether LLMs can distinguish the merits of one template against another. 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 instructions, compared to vanilla pre-trained language models.

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 for the particular task are of sufficient quality, we verify

Foundation Models for Decision Making Workshop at NeurIPS 2023.

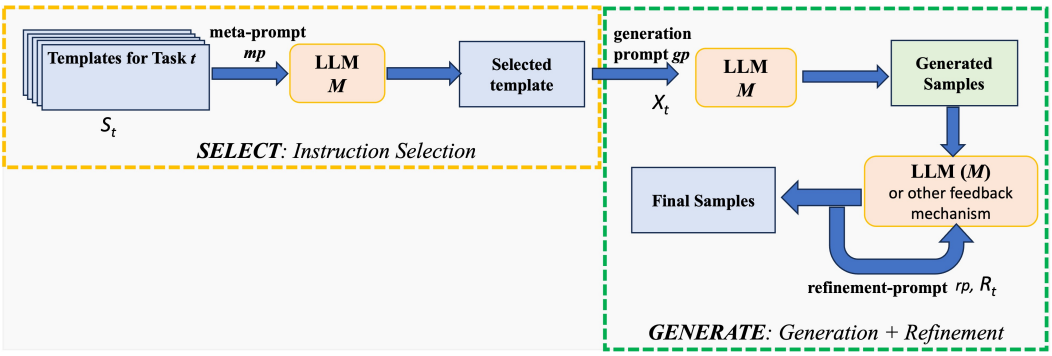

Figure 1: Overview of the *Self-Select* algorithm, which consists of *SELECT* and *GENERATE* modules. In the former, we provide a meta-prompt and several templates for the task, and obtain the selected template. In the latter, we first generate samples according to the provided template, and use model self-critique to determine whether those samples are of sufficiently high quality. For those that are insufficient, we use in-context refinement examples and attempt to improve the response.

the output samples relative to a benchmark; depending on the task, this can be chosen to be a metric (with a threshold for admittance into the final set) or even a model-generated label of the response quality. If a sample is deemed to be of insufficient quality, we prompt the model to refine its response, conditioned on the previous output, such that the new response takes its place as a candidate. Thus, upon the algorithm's termination, n samples per task of interest will be obtained; these samples can be used to fine-tune the model, or themselves be applied as exemplars for few-shot prompting.

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 templates given by 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 [50], and generate faithful synthetic data samples, in line with [55].

2 Algorithmic Approach

Given a set of possible instruction templates for a task, such as those manually curated in FLAN ([46]), *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 structure of the meta-prompt as follows (specific examples of meta-prompts used in our work are given in Appendix A), yielding a prompt index. This, 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 or 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.

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* $exemplars\ for\ refinement\ for\ task\ t$
mp : *meta-prompt*
gp : *generation-prompt*
rp : *refinement-prompt*
μ_t : $Response\ quality\ metric\ for\ task\ t\ with\ quality\ threshold\ \lambda_t$
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 2: The *Self-Select* algorithm and the assumed notation. Please see Section 2 (Algorithmic Approach) for a more comprehensive discussion of the method.

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.

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

Refinement ensures that the *Self-Select* algorithm enables quality control on responses – for generations deemed acceptable based on the threshold and criterion, we add those samples to the set $\mathcal{F}$, and return $\mathcal{F}$ upon termination of the loop, for the current task.

However, as the LLM may prove to be incapable of refining its response to the level of sufficiency, we enforce a stopping criterion on the basis of the number of refinement iterations that have occurred; prior work ([27]) demonstrates that weaker (smaller, non-instruction-tuned) models may struggle with iterative refinement procedures. This guarantees termination within a fixed number of iterations, even if it results in producing fewer than the desired n high-quality samples.

One may look to use 0-1 critiques generated by the model as a means to determine response sufficiency (that is, if the model deems the response quality to be good, it assigns a score of 1 to that response; else assigns 0). This could also be replaced with a known reference metric for a particular task, including semantically-oriented metrics like BERTScore ([54]) or for mathematical tasks, custom-defined metrics based on the outcomes of a call to a calculator API (e.g. Wolfram Alpha); hence, this is defined generally as μ_t in Algorithm 1.

3 Experimental Setup and Results

3.1 Numerical Reasoning

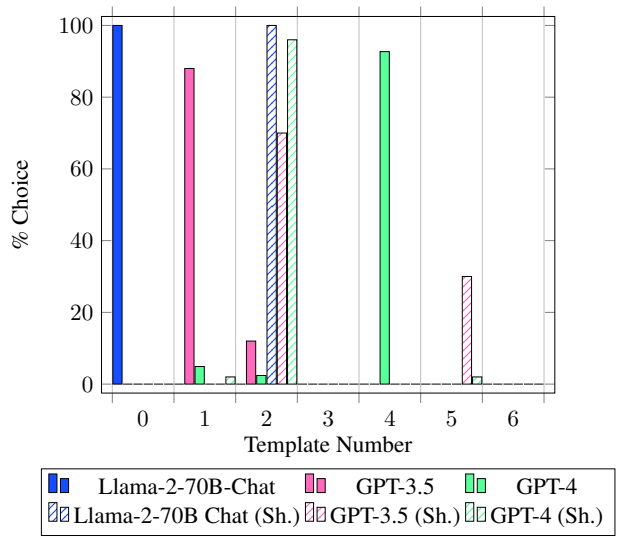

Figure 3: Results on instruction template selection for the numerical reasoning task. The bars with the striped lines correspond to the same model as the solid bar, but where the striped bars are results with shuffled instruction options, mapped back to their original template numbering.

For numerical reasoning, we selected two-number addition with one to five digit numbers, using the prompt in Figure 4. Our meta-prompt consists of seven possible options with slight changes in instruction format, such as how each operand is specified or identifying particular qualifiers on the operands and the resulting quantities.

We experimented with the Llama-2 7B and 13B variants, with and without chat-tuning [39], as well as MPT-7B and MPT-7B-instruct [38] 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 (whether solely in the pre-training corpora, for base models, or in the fine-tuning data as well, for instruction-tuned models) would interpret the template as code, when presented with the options in the meta-prompt, and thus generate code in response.

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

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."

There exists prior literature demonstrating LLMs' sensitivities to the order of choices in making decisions – in multiple choice questions ([30]), in-context examples ([56]), and response critique and evaluation ([42]). This led us to examine whether shuffling the instruction template options would have an impact on the models' preferred choice; maintaining the same option as before would indicate a degree of semantic understanding and having learned the differences between structures. In both the unshuffled (Table 1) and shuffled (Table 2) settings, we found that small models had difficulty following instructions, while the large instruction-tuned and/or chat-tuned models demonstrated a near deterministic preference for a specific template. That being said, in the shuffled case, all three models chose a completely different template compared to their selections in the original prompt ordering. Notably, all three models coalesced on a single choice in the shuffled regime. They all selected template 2. These results are shown for the original ordering and shuffled templates in Tables 1 and 2, respectively, and consolidated in Figure 3. Indices are given in terms of the original indices before shuffling for comparison.

We additionally generated 9,600 examples using a similar template to the ones given above; shown below to validate the feasibility of our proposed *GENERATE* module. An example of one such prompt is shown in Figure 5. 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, enabling use of these samples for knowledge distillation via fine-tuning or as in-context demonstrations.

Template (Unshuffled)	GPT-3.5	GPT-4	Llama-2-70B-Chat
Template 0	0%	0%	**100%**
Template 1	**88%**	4.9%	0%
Template 2	12%	2.4%	0%
Template 3	0%	0%	0%
Template 4	0%	**92.7%**	0%
Template 5	0%	0%	0%
Template 6	0%	0%	0%
Total	100%	100%	100%

Table 1: Results on instruction template selection for the numerical reasoning task with 50 samples (45 for GPT-4), using an unshuffled list of template options.

Template (Unshuffled #)	GPT-3.5	GPT-4	Llama-2-70B-Chat
Template 0	0%	0%	0%
Template 1	0%	2%	0%
Template 2	**70%**	**96%**	**100%**
Template 3	0%	0%	0%
Template 4	0%	0%	0%
Template 5	30%	2%	0%
Template 6	0%	0%	0%
Total	100%	100%	100%

Table 2: Results on template selection for the numerical reasoning task with 50 samples, with shuffled templates (mapped back to original numbering). (Samples with 2 or more results were excluded)

```
We have an arithmetic task below:
Generate in JSON {n} more correct examples, following the template below:

[
    {
        Problem: num_1 + num_2 =
        Answer: num_3
        Where num_1, num_2, and num_3 are {d} digit numbers
    },
    .
    .
    .
] ,,,
```

Figure 5: Above is the prompt for generation of n new samples following the LLM-chosen template, as specified in the *GENERATE* module of the *Self-Select* algorithm.

3.2 Sentiment Classification

We experimented on the sentiment classification task using 10 templates corresponding to the IMDB dataset ([26]), from the FLAN ([46]) 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.

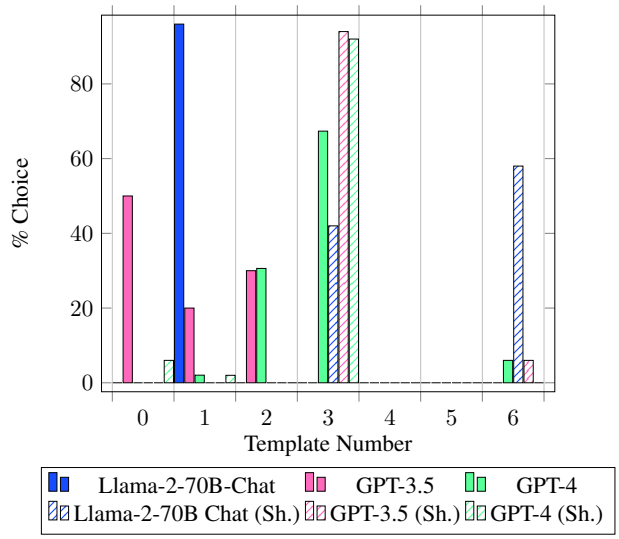

Figure 6: Results on instruction template selection for the sentiment classification task. The bars with the striped lines correspond to the same model as the solid bar, but where the striped bars are results with shuffled instruction options, mapped back to their original template numbering.

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). We present these results in Tables 3 and 4 in Appendix B, and are summarized in Figure 7.

4 Related Work

4.1 Instruction Tuning

Several prior works demonstrate the effectiveness of instruction tuning as a promising framework for yielding greater generalization to a wide variety of tasks ([46], [31], [28], [24], [6]). Recently, there has been growing interest in minimizing the amount of instruction-following data necessary to still obtain strong instruction-tuned models ([37], [17], [22], [5]). On a similar lens, it has been shown that small but well-curated datasets can lead to strong alignment to human preferences ([57]). However, open questions remain on what level of semantic understanding, rather than simply superficial pattern following can be learned by instruction tuning [20]. 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 [15]. Other recent work [44] has successfully used synthetic data with self-filtering methods to bootstrap a much larger corpus and attempt to address the data bottlenecks in instruction tuning. Approaches such as [55] reinforce the value of template-based generators by leveraging them through a pre-trained language model distilled to yield more fluent and faithful responses. Our work aims to continue the exploration into effective means of generating high-quality synthetic instruction-following data, such that even a relatively small number of samples, when distilled, can yield strong instruction-tuned models, while simultaneously achieving fluency and accuracy as a robust language agent.

```
SELECTION: The following templates correspond to different problems.
Choose which one best fits sentiment

Template 0 : {text}

Did this review think positively or negatively of the movie?
{options_}

Template 1 : {text}
Would you say this review is positive or negative?
{options_}

Template 2 : {text}

Is the sentiment of this review positive or negative?
{options_}

Template 3 : Please tell me the sentiment of the following review: {text}
{options_}

Template 4 : {text}
What is the sentiment of this review?
{options_}

Template 5 : Determine the sentiment:
{text}
{options_}

Template 6 : Write a {answer} movie review.

Template 7 : {text}
How would you describe the sentiment of this review?
{options_}

Template 8 : Generate a movie review with {answer} sentiment.

Template 9 : What's an example of a movie review? {text}

 Choose the best template by returning its number.

Please respond with the number of the chosen template
```

Figure 7: Above is the prompt used for the sentiment classification task, with 7 manually curated templates for performing addition, with slight differences in how the problem is phrased.

4.2 Chain-of-Thought and Refinement Approaches

4.2.1 Chain-of-Thought Prompting

Chain-of-Thought (CoT) prompting was introduced in [47], inducing the model to generate step-by-step rationales, which provided insights into their ability to perform more complex, multi-step reasoning tasks. [19] 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 [50], 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. [25] uses symbolic reasoning chains as a means to induce faithful explanations, which motivates our future line of exploration into LLMs' abilities to articulate its instruction selection decisions. [43] introduces a decoding strategy known as self-consistency for the chain-of-thought setting, which enables sampling of multiple reasoning paths and chooses the most consistency answer via aggregation. Another recent work [41] demonstrates that chain-of-thought reasoning still largely holds even with invalid

intermediate reasoning steps, and suggests that query relevance and coherence (consistent ordering of the steps) are the key principles for successful CoT prompting.

4.2.2 Extensions on Chain-of-Thought

While CoT assumes a linear reasoning path that reaches the desired results, recent works have motivated more intricate formulations that may further improve performance on more complex tasks. [51] introduces Tree-of-Thoughts (ToT), enabling broader exploration across multiple chains of reasoning, self-evaluation of the current state (thought), and graph traversal (backtracking), resulting in more robust problem-solving abilities. The Algorithm-of-Thoughts (AoT; [34]) method uses in-context examples of graph search algorithms (DFS and BFS paths) on top of the ToT framework, resulting in more fluid exploration of subproblems / individual thoughts as opposed to a rooted tree.

Graph-of-Thoughts (GoT; [3]) proposes a generalization of sorts on the ToT approach, no longer confining the thought structure to a tree, and introducing the notions of aggregation and refinement. Thought aggregation in GoT is task-specific, and enables parallel reasoning chains to be condensed into one (e.g. via summarization or merging); similar to our work and other in-context refinement strategies, individual thoughts can be scored and ranked using a general scoring function. In the present work, we largely explore tasks that are solvable within a single step, but this can be modified to support multi-step reasoning via CoT (or similar) framing of the generation prompt.

4.2.3 Refinement and Feedback Mechanisms

In prior literature, there are many approaches for providing feedback on model generations, and incorporating said feedback into informing subsequent decisions by the model. There are extensive precedents for using human evaluation as a feedback mechanism for human-in-the-loop systems ([1], [7], [36], [13], [4], [45]). Feedback mechanisms may include use of scalar reward models trained from human preference data as a proxy method ([28], [2], [1], [58], [21], [18], [12]), which have gained traction recently with growing interest in efficient and effective methods of aligning LLMs / Foundation Models more broadly and the rising prevalence of reinforcement learning from human feedback (RLHF). Model generated feedback methods ([33], [32], [53], [11]) are also an area of growing interest, as they also provide insights into the capabilities of strong LLMs such as GPT-4 in serving as an effective evaluation substitute for performing human evaluation ([23], [16]).

The notion of refinement has been explored through multiple lenses, including supervised learners to perform refinement ([48], [32], [9], [52], [53]), and natural language feedback and refinement ([35], [29], and feedback and refinement using the same model ([27], [51], [3]). Given our aim to explore the decision making capabilities of LLMs of various sizes, in both zero-shot and few-shot settings, to generate high-quality synthetic data, we take inspiration from in-context mechanisms for feedback and refinement in designing the *Self-Select* algorithm, but note that a variety of other metrics (or models) could be used for feedback generation.

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 [27] may provide avenues for understanding the degree to which these models exhibit semantic understanding, as opposed to simple pattern matching via inductive biases. This is perhaps related to the presence of multiple choice problems in either the pre-training or instruction tuning corpus, resulting in biased preferences. These findings may also shed new insights on the trustworthiness of knowledge distillation from larger instruction-tuned models to induce instruction following. We would like to further explore the notion of refinement with different models, given samples corresponding to the chosen template, and with various sufficiency criteria, as this may enable our approach to enforce further checks on template-specific sample quality.

Further work and experiments may also include an extended exploration of what can be termed as *in-context meta-prompting*; that is, providing exemplars for instruction selection, based on human preferences for an ideal template. For example, the template that GPT-4 selected, while justifying its choice as the most simple and straightforward, was not in agreement with the template that the

authors of this work would consider on that criteria. This may suggest new properties regarding LLM steerability in decision making when performing in-context learning.

Studying the decision making process involved in instruction selection through the lens of semantic understanding could perhaps focus on the attention mechanism; this has ties to cognitive neuroscience literature ([40]) and computer vision applications ([8], [10]) as well. 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 when presented with a decision to make in selecting one template over the others, especially in a permutation-invariant manner. Shuffling the order of templates led to substantial changes in the distribution of chosen templates for numerical reasoning and the primary template for each model becoming common – 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, we found at least one model demonstrated a near-deterministic preference among the template options. This leads us to conclude that in order for LLMs to exhibit semantic understanding (and thus, some degree of judgement in instruction template decision making), they must be exposed to the same data in several orderings, perhaps motivating data augmentation strategies using permutations. By demonstrating the order dependence on the instruction selection outcomes, and thus, the nature of the synthetic data generated from the *Self-Select* algorithm, we hope for this work to mark a contribution to the ongoing development of large language models for decision making, regarding their suitability for critical applications requiring robustness and high trustworthiness.

Acknowledgements

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

References

[1] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022.

[2] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022.

[3] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models, 2023.

[4] Zefan Cai, Baobao Chang, and Wenjuan Han. Human-in-the-loop through chain-of-thought, 2023.

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

[6] 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.

[7] Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A. Smith. All that's 'human' is not gold: Evaluating human evaluation of generated text, 2021.

[8] Abhishek Das, Harsh Agrawal, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. Human attention in visual question answering: Do humans and deep networks look at the same regions?, 2016.

[9] Wanyu Du, Zae Myung Kim, Vipul Raheja, Dhruv Kumar, and Dongyeop Kang. Read, revise, repeat: A system demonstration for human-in-the-loop iterative text revision. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, pages 96–108, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[10] Li Fei-Fei, Asha Iyer, Christof Koch, and Pietro Perona. What do we perceive in a glance of a real-world scene? *Journal of Vision*, 7(1):10–10, 01 2007.

[11] Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. Gptscore: Evaluate as you desire, 2023.

[12] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization, 2022.

[13] Asma Ghandeharioun, Judy Hanwen Shen, Natasha Jaques, Craig Ferguson, Noah Jones, Agata Lapedriza, and Rosalind Picard. Approximating interactive human evaluation with self-play for open-domain dialog systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[14] 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.

[15] 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.

[16] Veronika Hackl, Alexandra Elena Müller, Michael Granitzer, and Maximilian Sailer. Is gpt-4 a reliable rater? evaluating consistency in gpt-4 text ratings, 2023.

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

[18] Jian Hu, Li Tao, June Yang, and Chandler Zhou. Aligning language models with offline reinforcement learning from human feedback, 2023.

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

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

[21] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction, 2018.

[22] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report, 2023.

[23] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment, 2023.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. Check your facts and try again: Improving large language models with external knowledge and automated feedback, 2023.

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

[31] 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.

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

[33] Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. Training language models with language feedback at scale, 2023.

[34] Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. Algorithm of thoughts: Enhancing exploration of ideas in large language models, 2023.

[35] Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.

[36] Niket Tandon, Aman Madaan, Peter Clark, and Yiming Yang. Learning to repair: Repairing model output errors after deployment using a dynamic memory of feedback, 2021.

[37] 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.

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

[39] 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.

[40] 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.

[41] Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. Towards understanding chain-of-thought prompting: An empirical study of what matters. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2717–2739, Toronto, Canada, July 2023. Association for Computational Linguistics.

[42] 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.

[43] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[44] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions, 2023.

[45] Zijie J. Wang, Dongjin Choi, Shenyu Xu, and Diyi Yang. Putting humans in the natural language processing loop: A survey, 2021.

[46] 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.

[47] 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.

[48] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct, 2022.

[49] 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.

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

[51] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023.

[52] Michihiro Yasunaga and Percy Liang. Graph-based, self-supervised program repair from diagnostic feedback, 2020.

[53] 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.

[54] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2019.

[55] Tianyi Zhang, Mina Lee, Lisa Li, Ende Shen, and Tatsunori B. Hashimoto. Templm: Distilling language models into template-based generators, 2022.

[56] 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.

[57] 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.

[58] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2019.

Appendix A: Specific Prompts for the *SELECT* Module

```
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 8: Above is the full meta-prompt for template selection with 7 options in the *SELECT* module of the *Self-Select* algorithm. Note that this is the same prompt presented in Figure 3; included here for completeness and comparison to its shuffled counterpart, Figure 9.

The templates shown here correspond to the selection process in the *SELECT* module, with the two prompts given showing the original and permuted versions of the prompt, demonstrating the preference shift evidenced in Figure 4. Observe that the set of templates are indeed the same, but have a different index between the two versions. We note that these are just two of the $7! = 5040$ possible orderings we could have prompted with, and yet they already elicit difference preferences for each model.

```
SELECTION: The following templates correspond to different problems.
Choose which one best fits addition. Respond with Template: <NUM>

Template 0 : Generate an addition problem using the following template:
num_1 + num_2 = answer
where num_1, num_2, and answer are numbers

Template 1 : Addition; Generate a problem following this template:
{} + {} = {}

Template 2 : Generate an addition problem using the following template:
num_1 + num_2 = answer

Template 3 : Addition;
Problem: {} + {} = {}

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

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 9: Above is the full meta-prompt for template selection with shuffled ordering of the templates. Note that the templates are the same as those in Figure 8, but are re-indexed accordingly.

Appendix B: Results for Sentiment Classification Template Selection

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 3: 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).

The results presented here correspond with the charts of Figure 7, for the sentiment classification task, as discussed in Section 3.2. The above table demonstrates the selections for the initial ordering of templates as given. We find that each model displays vastly different preferences across the instruction options, with GPT-3.5 and GPT-4 being less consistent than Llama-2-70B-Chat in its selections. The table below demonstrates the results when the template order is permuted before presenting them via our model query, although for the purposes of comparison, we map the indices back to their original numberings. As with the numerical reasoning task, we find that while the models are much closer to agreement on the "best template," the top choice template for both GPT-3.5 and Llama-2-70B-Chat

changed entirely. We hypothesize that such a phenomenon could lend itself to future explorations where we formally sample responses by multiple agents across several template index permutations and template selection is done via majority voting, as we have done for this small case here. This follows somewhat in the vein of self-consistency [43].

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