InstructEval: Systematic Evaluation of Instruction Selection Methods

Anirudh Ajith∗ Chris Pan∗ Mengzhou Xia Ameet Deshpande Karthik Narasimhan
Department of Computer Science, Princeton University
{anirudh.ajith, chrispan, mengzhou, asd, karthikn}@princeton.edu

Abstract

In-context learning (ICL) performs tasks by prompting a large language model (LLM) using an instruction and a small set of annotated examples called demonstrations. Recent work has shown that precise details of the inputs used in the ICL prompt significantly impact performance, which has incentivized instruction selection algorithms. The effect of instruction-choice however is severely underexplored, with existing analyses restricted to shallow subsets of models and tasks, limiting the generalizability of their insights. We develop InstructEval, an ICL evaluation suite to conduct a thorough assessment of these techniques. The suite includes 13 open-sourced LLMs of varying scales from four model families, and covers nine tasks across three categories. Using the suite, we evaluate the relative performance of seven popular instruction selection methods over five metrics relevant to ICL. Our experiments reveal that using curated manually-written instructions or simple instructions without any task-specific descriptions often elicits superior ICL performance overall than that of automatic instruction-induction methods, pointing to a lack of generalizability among the latter. We release our evaluation suite for benchmarking instruction selection approaches and enabling more generalizable methods in this space.¹

1 Introduction

One of the most effective insights in NLP research in recent years has been that large language models trained to perform next-token prediction show emergent in-context learning (ICL) abilities (Brown et al., 2020; Scao et al., 2022a; Zhang et al., 2022a). While the bulk of research interest has shifted away from task-specific models and towards creating “foundation models” to perform a variety of tasks using appropriately constructed prompts, the performance of ICL remains sensitive to the precise details of prompt construction. Prompt engineering remains critical for achieving optimal ICL performance (Perez et al., 2021; Zhao et al., 2021; Webson and Pavlick, 2022).

In practice, ICL typically involves prompting a language model using a concatenation of a task-specific instruction, a short sequence of annotated in-context examples known as demonstrations, and a test example (Figure 2). Much of the research interest surrounding in-context learning has focused on understanding the optimal selection, ordering of demonstrations, and label-space choices (Liu et al., 2021a; Su et al., 2022; Rubin et al., 2022; Wang et al., 2023a; Lu et al., 2021a; Wei et al., 2023; Pan et al., 2023). However, instruction choice remains

¹Code: https://github.com/princeton-nlp/InstructEval
a relatively underexplored aspect of prompt engineering despite its established significance (Mishra et al., 2022) on downstream performance.

Even among recent works exploring automatic instruction selection (Honovich et al., 2022; Gonen et al., 2022; Deng et al., 2022; Zhou et al., 2022), the use of different evaluation protocols makes the comparison of their relative performances difficult. Existing studies typically limit their analyses to specific models or tasks; for example, Zhou et al. (2022) focus on a single model, and while Deng et al. (2022) consider multiple model scales, they all belong to a single model family. Moreover, evaluations often span disparate task selections with minimal overlap and are primarily dominated by classification tasks, neglecting other task types like multiple-choice QA or generation. Lastly, most previous works studying automatic instruction selection tend to emphasize zero-shot accuracy, overlooking other pertinent ICL metrics such as few-shot accuracy and robustness measures.

To address these issues, we build InstructEval, an evaluation suite for the comprehensive evaluation of instruction selection methods. The suite covers a diverse collection of 13 open-sourced autoregressive LLMs from four model families and nine tasks spanning three task types. Additionally, it also incorporates three accuracy metrics and two sensitivity metrics that are of interest to ICL. We perform evaluations of seven popular instruction selection methods including trivial instruction baselines, manually curated instructions, and sophisticated automatic methods using our suite.

Overall, we find that the relative effectiveness of these approaches varies significantly across different models and task types. We discover that curated manually-written instructions and task-agnostic instructions can elicit better aggregated performance (over models) than automatically induced ones, highlighting the lack of generalizability of the latter. We also find that including instructions in few-shot prompts usually tends to hurt ICL performance at the model scales we consider. Our findings suggest that it may be optimal for ICL practitioners to omit instructions in few-shot settings and use curated manually-written instructions in zero-shot settings, rather than contemporary automatic induction techniques that require substantial computation and hyperparameter tuning to achieve competitive performance. We release the evaluation suite we develop to aid the systematic study of even more questions regarding prompt engineering that we do not explicitly address in our work.

2 Related Work

In-Context Learning and Existing Benchmarks
As language models have scaled, in-context learning has emerged as a popular paradigm and remains ubiquitous among several autoregressive LLM families (Brown et al., 2020; Touvron et al., 2023; Scao et al., 2022b; Black et al., 2021; Zhang et al., 2022b). Benchmarks like BigBench (Srivastava et al., 2022) and HELM (Liang et al., 2022) have been created for the holistic evaluation of these models. BigBench focuses on few-shot abilities of state-of-the-art large language models, while HELM extends its evaluation to consider metrics like robustness and bias. However, these benchmarks focus on evaluating and ranking language models, and do not address the systematic evaluation of prompting methods. Although contemporary work by Yang et al. (2023) also aims to perform a similar systematic analysis of prompting methods, they focus on simple probability-based prompt selection while we evaluate a broader range of methods including trivial instruction baselines, curated manually selected instructions, and sophisticated automated instruction selection.

Automated Prompt Engineering Methods
There has been interest in performing automated prompt-engineering for target downstream tasks within ICL. This has led to the exploration of various prompting methods, ranging from simple heuristics such as selecting instructions with the lowest perplexity (Gonen et al., 2022), inducing instructions from large language models using a few annotated input-output pairs (Zhou et al., 2022), to utilizing RL objectives to create discrete token sequences as prompts (Deng et al., 2022). However, these works restrict their evaluation to small sets of models and tasks with little intersection, hindering their objective comparison.

Understanding in-context learning
There has been much recent work attempting to understand the mechanisms that drive in-context learning. Studies have found that the selection of demonstrations included in prompts significantly impacts few-shot accuracy across most tasks (Liu et al., 2021b; Agrawal et al., 2022; Xu et al., 2023). Works like (Lu et al., 2021b) also show that altering the ordering of a fixed set of demonstrations can affect
downstream accuracy. Prompts sensitive to demonstration permutation often exhibit lower accuracies (Chen et al., 2023), making them less reliable, particularly in low-resource domains.

Our work aims to bridge these gaps by systematically evaluating the efficacy of popular instruction selection approaches over a diverse set of tasks and models, facilitating objective comparison. We evaluate these methods not only on accuracy metrics, but also on sensitivity metrics to glean additional insights. We recognize that other facets of prompting not covered by instruction engineering exist (Wei et al.; Yao et al., 2023; Wang et al., 2023b), and defer these explorations to future work.

3 Evaluation Suite

3.1 Prompt format

We define a ‘prompt’ as the full textual input provided to an LLM. Our evaluation suite supports the use of any number of demonstrations, arbitrary demonstration templates and the inclusion of custom strings anywhere within the prompt. Since the instructions used can be set to any arbitrary strings, users are free to use any external means to select instructions and have them evaluated by our suite.

For consistency, we conduct all experiments in this work using prompts that begin with an instruction, continue with a sequence of annotated training demonstrations, and conclude with an unsolved test example (Figure 2). We express each example in a minimal, task-specific key-value format (Table 8) that reflects task semantics.

3.2 Metrics

Accuracy metrics Accuracy is typically the primary metric of interest in ICL. While ICL is most commonly performed in few-shot settings where a handful of annotated demonstrations are included in the prompt, models are also prompted zero-shot without the use of such demonstrations. Since real-world scenarios can often contain grammatical errors and misspellings in the test input, it is desirable to find prompts robust to these perturbations. Hence, we measure zero-shot accuracy, few-shot accuracy, and perturbation accuracy in our evaluations. Following Liang et al. (2022), we measure perturbation accuracy by introducing random capitalization, spacing, contractions and common misspellings in the test input.

Sensitivity metrics Previous work has shown that the accuracy obtained using a prompt template can fluctuate significantly as a function of the set of demonstrations included in the prompt (Liu et al., 2021a; Su et al., 2022; Rubin et al., 2022; Wang et al., 2023a) and the order they are presented in (Lu et al., 2021b). It may be desirable in practice to identify prompt templates and instructions that offer consistent performance regardless of the choice of demonstrations and their arrangement. Hence, we introduce selectional sensitivity and permutational sensitivity metrics to measure the sensitivity of chosen instructions respectively to selected demonstrations, and the order in which they are arranged. We quantify the sensitivity of an instruction (given a model and task) using the standard deviation of accuracies obtained on varying the selection or permutation of the demonstrations used, each across 16 random choices.

3.3 Aggregating metrics across Models

Each instruction selection method being tested across N models and M datasets yields NM values per metric. Comparing these NM-dimensional

\[
\text{Instructions are omitted during ‘Null instruction’ evaluations. Demonstrations are omitted in zero-shot evaluations.}
\]

\[
\text{We choose to treat this as an accuracy metric rather than a sensitivity metric since it is not meaningful to measure sensitivity to such perturbations in situations where a prompt only elicits near random-chance task performance from a model.}
\]
vectors directly is complex. It can be challenging to reduce them to a single representative scalar. Simple approaches such as computing the mean of these $NM$ values can prove inadequate since the resulting scores would tend to be heavily influenced by metric values that exhibit a high variance across different inspected methods.

We opt against using aggregation techniques used by previous works (Liang et al., 2022; Srivastava et al., 2022) due to their drawbacks (Section B) and instead adopt ‘mean relative gain’ as a means to aggregate accuracy metrics across multiple models. We rely on simple averaging for sensitivity metrics, partly because we observe that these quantities do not show much variation across methods.

3.3.1 Accuracy metrics

Considering the range of models and datasets in our evaluation suite, we unsurprisingly observe substantial variation in accuracy magnitudes across model scales and tasks. However, we notice that the degree of variation in accuracy due to instruction choice is usually considerably smaller than the degree of variation due to model and task choice.

To meaningfully compare and aggregate the relative performance of different instruction selection methods across models, we use a measure called mean relative gain. First, we define the relative gain for a value $x$ from a population $P$ as the percentage by which $x$ exceeds the mean value of $P$:

$$r \text{-gain}_P(x) = 100 \times \frac{x - \mu_P}{\mu_P}$$

Consider a collection of models $M$ and instructions $\mathcal{I}$ for a task $t$. Given a model $m$, we calculate the raw accuracy scores $s_{tmi}$ for each instruction $i \in \mathcal{I}$. Taking this set $S_{tm}$ to be the population, we compare the performances of the instructions against each other by computing their corresponding relative gains $r_{tmi} = r \text{-gain}_{S_{tm}}(s_{tmi})$. Each $r_{tmi}$ represents the degree by which method $i$ outperforms the average performance along the metric on task $t$ for model $m$.

We now define the mean relative gain as

$$\tau_{ti} = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} r_{tmi}$$

These $\tau_{ti}$ values, tabulated and analyzed in Section 5, capture not only the ordinal information about each method’s performance on a given task but also provide an intuitive sense of the magnitude by which these methods outperform others. Specifically, if an induction method $i$ has a mean relative gain $\tau_{ti}$ on task $t$, this means that method $i$ exceeds average performance (across $\mathcal{I}$) on task $t$ by $\tau_{ti}$ percent when averaged across models $\mathcal{M}$.

3.3.2 Sensitivity metrics

To aggregate the sensitivity of an instruction selection/induction method $i$ over all models for a task $t$, we simply compute the average of the raw sensitivity scores (described in Section 3.2). Specifically, if $\sigma_{tmi}$ is the raw sensitivity score obtained for model $m$ and task $t$ when using instruction $i$, then the aggregated sensitivity score $\overline{\sigma}_{ti}$ is given by

$$\overline{\sigma}_{ti} = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \sigma_{tmi}$$

We choose to avoid more sophisticated aggregation strategies like relative gain for sensitivity metrics since standard deviations are already secondary metrics, hence making it unintuitive to discuss the relative gain of the standard deviation obtained using a method over the average.
Table 1: Tasks included in our evaluation suite.

<table>
<thead>
<tr>
<th>Task Type</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification (CLS)</td>
<td>AG News (Zhang et al., 2015)</td>
</tr>
<tr>
<td></td>
<td>ANLI (Nie et al., 2020)</td>
</tr>
<tr>
<td>Multiple-choice (MCQ)</td>
<td>BoolQ (Clark et al., 2019)</td>
</tr>
<tr>
<td></td>
<td>IMDB (Maas et al., 2011)</td>
</tr>
<tr>
<td></td>
<td>TweetEval Emotion (Mohammad et al., 2018)</td>
</tr>
<tr>
<td>Generative QA (GQA)</td>
<td>CosmosQA (Huang et al., 2019)</td>
</tr>
<tr>
<td></td>
<td>HellaSwag (Zellers et al., 2019)</td>
</tr>
<tr>
<td></td>
<td>NQ-Open (Kwiatkowski et al., 2019)</td>
</tr>
<tr>
<td></td>
<td>TriviaQA (Joshi et al., 2017)</td>
</tr>
</tbody>
</table>

Table 2: Model families and corresponding model scales included in our evaluation suite.

<table>
<thead>
<tr>
<th>Model Family</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOOM (Scao et al., 2022b)</td>
<td>1.1B, 1.7B, 3B, 7.1B</td>
</tr>
<tr>
<td>GPT Neo (Black et al., 2021, 2022)</td>
<td>1.3B, 2.7B, 20B</td>
</tr>
<tr>
<td>LLaMA (Touvron et al., 2023)</td>
<td>7B, 13B</td>
</tr>
<tr>
<td>OPT (Zhang et al., 2022b)</td>
<td>1.3B, 2.7B, 6.7B, 13B</td>
</tr>
</tbody>
</table>

3.4 Tasks

While previous instruction induction (Zhou et al., 2022; Deng et al., 2022) work has tended to focus mostly on classification tasks, we include 9 tasks (Table 1) in our evaluation suite spanning classification (CLS), multiple-choice question-answering (MCQ) and generative question-answering (GQA) to assess the applicability of instruction selection and induction methods to other task-types as well. We concentrate on tasks that are challenging to contemporary language models, and yet are not so demanding that the performance of these models does not exceed random chance. We exclude certain generative tasks, like summarization, which are challenging to assess objectively. 4

3.5 Models

We include a diverse range of 13 autoregressive LLMs (Table 2) from 4 model families of sizes ranging from 1.1 billion to 20 billion parameters in our evaluation suite. We choose contemporary models that span different architectures and training paradigms which are known to show good ICL performance. This diversity bolsters the generalizability of insights obtained using our evaluation suite while mitigating potential bias towards any specific model family. Moreover, we select open-source models which are large enough to show non-trivial ICL performance while still being small enough to run on reasonable consumer hardware to ensure the practical significance of our findings.

4 Experimental setup

We perform experiments evaluating 3 families of instruction selection methods (listed in Table 3).

Task-agnostic instructions In practical ICL settings, it is straightforward to use instructions that contain no task-specific information.

- **Null instruction**: We assess the impact of omitting instructions from the prompt. This amounts to constructing prompts that consist of demonstrations and a test example in few-shot, and only an unanswered test-example in zero-shot settings.

- **Generic instructions**: We assess the impact of using generic task-agnostic instructions such as Complete the following task:. These instructions require minimal effort to write since they do not demand knowledge of the task. We list the set of generic instructions we evaluate in Table 10.

Manual task-specific instructions We evaluate manually-written task-specific instructions that ICL practitioners may use in practice.

- **PromptSource**: PromptSource (Bach et al., 2022) is a public collection of manually-curated prompt templates pertaining to 170+ datasets which are often used off-the-shelf for ICL and are generally considered high-quality.

- **Ad hoc**: ICL practitioners often create task-specific instructions ad hoc, based on the semantics of the given task. We simulate this mode of instruction selection by asking ChatGPT to generate several paraphrases of task-specific seed instructions we obtain from PromptSource and randomly sampling from the generated set.

Automatically synthesized task-specific instructions We evaluate 3 popular automated instruction selection and induction methods that are representative of previous work.

---

4Standard summarization metrics correlate poorly with human preferences (Liang et al., 2022; Goyal et al., 2023).
• **Low Perplexity:** (Gonen et al., 2022) find that the perplexity a model associates with an instruction is negatively correlated with its ICL performance when using that instruction. We use the SPELL algorithm proposed by Gonen et al. (2022) to select the least perplexity instructions (for each model) from a large pool of ChatGPT paraphrased instructions.

• **APE:** (Zhou et al., 2022) is an automatic few-shot method for inducing instructions by prompting a language model to describe the given task, and refining the set of generated prompts using accuracy on a small held-out validation set. While Zhou et al. (2022) limit their evaluation to GPT-3 (Brown et al., 2020) and InstructGPT (Ouyang et al., 2022), we assess APE’s applicability to a significantly larger set of models and tasks.

• **RLPrompt** (Deng et al., 2022) is a reinforcement-learning-based approach for few-shot prompt induction. While the original authors only evaluate their method using GPT-2 on a few classification tasks, we expand this assessment to many more models and tasks. Notably, we assess the extensibility of RLPrompt to MCQ tasks, but do not test RLPrompt performance on GQA tasks since the algorithm is not directly applicable to generation tasks.

5 **Results**

We tabulate the mean relative gain values over accuracy metrics in Table 4, and the mean standard deviations corresponding to selectional and permutational sensitivity metrics in Table 5.

5.1 Less sophisticated instruction selection methods tend to show higher accuracy

We find that task-agnostic instructions dominate in few-shot settings with Null instructions and Generic instructions achieving the highest aggregated performance in 5/9 tasks for few-shot accuracy and 6/9 tasks for perturbation accuracy. Although both these methods show above-average performance in few-shot settings, Null instructions tend to perform better among the two.

Although PromptSource instructions only show an average performance in few-shot settings, their manually curated task-specific instructions prove most effective in zero-shot settings, achieving the highest aggregated performance in 6/9 tasks and usually achieving markedly higher mean relative gain values than even the runner-up method for the task. This is especially true of GQA tasks where PromptSource instructions outperform the average by >17%.

Automatic task-specific instructions are usually outperformed by simple baselines. They fail to achieve the best zero-shot performance on any task we consider. While they do sometimes perform competitively with simpler baselines in the few-shot setting, emerging as the best-performing instructions in 2/9 tasks, this behavior is inconsistent. Low Perplexity instructions and APE instructions seldom show above-average performance in either setting while RLPrompt instructions show above-average performance in 5/7 tasks in both settings. They are still usually outperformed by instructions obtained through simpler means such as Null and PromptSource instructions.

5.2 Ranges of variation of aggregated scores

We notice that instructions have a more significant impact in zero-shot settings as compared to few-shot settings. For most tasks, we find that the highest mean relative gain values achieved in the zero-shot setting are markedly greater than those in the few-shot setting. Accordingly, the minimum values for each task are also relatively lower in zero-shot settings. This finding suggests that instructions play a significant role in informing models of semantics in zero-shot settings whereas in few-shot settings, most of a model’s understanding of task-semantics comes from the demonstrations.

The degree of variation in accuracy due to instruction choice varies considerably across tasks. AG News and Emotion show the highest variability in few-shot performance while GQA tasks show the most variability in zero-shot settings.

Table 5 shows that selectional and permutational sensitivities vary dramatically across tasks even though they are roughly consistent across all methods for a given task. This implies that all the methods we evaluate are comparable in sensitivity, which is unsurprising since none of them explicitly optimize for it. We also find that most methods show comparable, but usually lower permutational sensitivity than selectional sensitivity.

5.3 Analysis

We tabulate the mean relative gain values for zero-shot and few-shot accuracies computed separately for “small” models with < 6 billion parameters and
<table>
<thead>
<tr>
<th>Method</th>
<th>AG News</th>
<th>ANLI</th>
<th>BoolQ</th>
<th>IMDB</th>
<th>Emotion</th>
<th>HellaSwag</th>
<th>CosmosQA</th>
<th>TriviaQA</th>
<th>NQ-Open</th>
<th># wins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero-shot accuracy (mean relative gain) ↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Null Instruction</td>
<td>2.26</td>
<td>1.07</td>
<td>2.48</td>
<td>−3.52</td>
<td>−5.30</td>
<td>2.54</td>
<td>5.94</td>
<td>−3.02</td>
<td>−25.67</td>
<td>3</td>
</tr>
<tr>
<td>Generic Instruction</td>
<td>3.55</td>
<td>−0.39</td>
<td>0.03</td>
<td>1.69</td>
<td>2.39</td>
<td>−0.13</td>
<td>−1.67</td>
<td>−1.46</td>
<td>−5.99</td>
<td>0</td>
</tr>
<tr>
<td>PromptSource</td>
<td>5.81</td>
<td>1.38</td>
<td>−0.65</td>
<td>4.34</td>
<td>5.13</td>
<td>−1.54</td>
<td>−3.42</td>
<td>17.08</td>
<td>22.15</td>
<td>6</td>
</tr>
<tr>
<td>Ad hoc</td>
<td>−0.33</td>
<td>0.21</td>
<td>0.55</td>
<td>1.41</td>
<td>0.66</td>
<td>−0.27</td>
<td>2.46</td>
<td>−1.85</td>
<td>3.79</td>
<td>−</td>
</tr>
<tr>
<td>Low Perplexity</td>
<td>−0.59</td>
<td>1.22</td>
<td>0.56</td>
<td>0.84</td>
<td>−4.07</td>
<td>−1.38</td>
<td>−2.18</td>
<td>−5.99</td>
<td>2.81</td>
<td>0</td>
</tr>
<tr>
<td>APE</td>
<td>−15.63</td>
<td>−3.86</td>
<td>−1.07</td>
<td>−1.77</td>
<td>−0.26</td>
<td>−1.06</td>
<td>0.00</td>
<td>−6.44</td>
<td>4.39</td>
<td>0</td>
</tr>
<tr>
<td>RLPrompt</td>
<td>4.92</td>
<td>0.37</td>
<td>−1.80</td>
<td>−2.99</td>
<td>1.46</td>
<td>1.85</td>
<td>3.79</td>
<td>−</td>
<td>−</td>
<td>0</td>
</tr>
<tr>
<td>Few-shot accuracy (mean relative gain) ↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Null Instruction</td>
<td>4.09</td>
<td>−0.22</td>
<td>0.87</td>
<td>−0.80</td>
<td>5.89</td>
<td>0.17</td>
<td>1.33</td>
<td>0.45</td>
<td>−0.02</td>
<td>4</td>
</tr>
<tr>
<td>Generic Instruction</td>
<td>5.16</td>
<td>−0.20</td>
<td>−0.10</td>
<td>0.45</td>
<td>4.84</td>
<td>0.04</td>
<td>−0.18</td>
<td>0.11</td>
<td>0.11</td>
<td>1</td>
</tr>
<tr>
<td>PromptSource</td>
<td>0.83</td>
<td>0.14</td>
<td>−0.79</td>
<td>0.39</td>
<td>−4.39</td>
<td>−0.06</td>
<td>−0.94</td>
<td>−0.36</td>
<td>0.61</td>
<td>1</td>
</tr>
<tr>
<td>Ad hoc</td>
<td>2.18</td>
<td>−0.10</td>
<td>−0.05</td>
<td>0.60</td>
<td>−5.63</td>
<td>−0.21</td>
<td>−0.59</td>
<td>0.09</td>
<td>−0.49</td>
<td>1</td>
</tr>
<tr>
<td>Low Perplexity</td>
<td>−1.96</td>
<td>0.31</td>
<td>−0.40</td>
<td>0.20</td>
<td>−6.79</td>
<td>−0.23</td>
<td>−0.61</td>
<td>−0.06</td>
<td>−0.02</td>
<td>1</td>
</tr>
<tr>
<td>APE</td>
<td>−15.43</td>
<td>0.10</td>
<td>0.06</td>
<td>−0.69</td>
<td>1.17</td>
<td>0.02</td>
<td>0.17</td>
<td>−0.24</td>
<td>−0.19</td>
<td>0</td>
</tr>
<tr>
<td>RLPrompt</td>
<td>5.13</td>
<td>−0.02</td>
<td>0.40</td>
<td>−0.14</td>
<td>4.90</td>
<td>0.27</td>
<td>0.81</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>Few-shot perturbation accuracy (mean relative gain) ↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Null Instruction</td>
<td>4.09</td>
<td>−0.08</td>
<td>0.11</td>
<td>−0.27</td>
<td>5.98</td>
<td>0.11</td>
<td>1.10</td>
<td>0.81</td>
<td>1.28</td>
<td>4</td>
</tr>
<tr>
<td>Generic Instruction</td>
<td>5.15</td>
<td>−0.18</td>
<td>−0.16</td>
<td>0.56</td>
<td>4.23</td>
<td>−0.02</td>
<td>−0.02</td>
<td>0.08</td>
<td>0.10</td>
<td>2</td>
</tr>
<tr>
<td>PromptSource</td>
<td>1.14</td>
<td>0.27</td>
<td>−0.02</td>
<td>0.33</td>
<td>−3.92</td>
<td>0.06</td>
<td>−0.53</td>
<td>−0.65</td>
<td>0.04</td>
<td>0</td>
</tr>
<tr>
<td>Ad hoc</td>
<td>1.68</td>
<td>0.51</td>
<td>−0.34</td>
<td>0.37</td>
<td>−5.87</td>
<td>−0.08</td>
<td>−0.63</td>
<td>−0.28</td>
<td>−0.61</td>
<td>0</td>
</tr>
<tr>
<td>Low Perplexity</td>
<td>−2.39</td>
<td>0.68</td>
<td>−0.12</td>
<td>−0.20</td>
<td>−0.61</td>
<td>−0.09</td>
<td>−0.66</td>
<td>−0.03</td>
<td>−0.78</td>
<td>1</td>
</tr>
<tr>
<td>APE</td>
<td>−14.32</td>
<td>−1.20</td>
<td>0.28</td>
<td>−0.82</td>
<td>1.26</td>
<td>−0.13</td>
<td>0.21</td>
<td>0.06</td>
<td>−0.03</td>
<td>1</td>
</tr>
<tr>
<td>RLPrompt</td>
<td>4.65</td>
<td>−0.01</td>
<td>0.24</td>
<td>0.03</td>
<td>4.94</td>
<td>0.15</td>
<td>0.53</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4: Mean relative gain values associated with zero-shot accuracy, and few-shot accuracy with unperturbed and perturbed test inputs. Only values that correspond to the same task and metric should be compared. Positive values represent above-average performance, and negative values represent below-average performance. The ‘# wins’ column shows the number of tasks where a method achieved the highest aggregated performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>AG News</th>
<th>ANLI</th>
<th>BoolQ</th>
<th>IMDB</th>
<th>Emotion</th>
<th>HellaSwag</th>
<th>CosmosQA</th>
<th>TriviaQA</th>
<th>NQ-Open</th>
<th># wins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selectional sensitivity (mean standard deviation) ↓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Null Instruction</td>
<td>6.69</td>
<td>2.45</td>
<td>4.73</td>
<td>5.28</td>
<td>6.97</td>
<td>2.46</td>
<td>8.10</td>
<td>2.59</td>
<td>2.28</td>
<td>3</td>
</tr>
<tr>
<td>Generic Instruction</td>
<td>6.87</td>
<td>2.50</td>
<td>4.76</td>
<td>5.40</td>
<td>6.97</td>
<td>2.48</td>
<td>8.16</td>
<td>2.61</td>
<td>2.26</td>
<td>0</td>
</tr>
<tr>
<td>PromptSource</td>
<td>6.73</td>
<td>2.26</td>
<td>4.85</td>
<td>5.37</td>
<td>6.43</td>
<td>2.43</td>
<td>8.26</td>
<td>2.59</td>
<td>2.28</td>
<td>1</td>
</tr>
<tr>
<td>Ad hoc</td>
<td>6.95</td>
<td>2.41</td>
<td>4.62</td>
<td>5.38</td>
<td>6.34</td>
<td>2.42</td>
<td>8.20</td>
<td>2.65</td>
<td>2.37</td>
<td>1</td>
</tr>
<tr>
<td>Low Perplexity</td>
<td>7.07</td>
<td>2.17</td>
<td>4.69</td>
<td>5.64</td>
<td>6.25</td>
<td>2.42</td>
<td>8.27</td>
<td>2.59</td>
<td>2.30</td>
<td>2</td>
</tr>
<tr>
<td>APE</td>
<td>7.44</td>
<td>2.98</td>
<td>4.63</td>
<td>5.70</td>
<td>6.67</td>
<td>2.43</td>
<td>8.16</td>
<td>2.65</td>
<td>2.21</td>
<td>1</td>
</tr>
<tr>
<td>RLPrompt</td>
<td>6.76</td>
<td>2.35</td>
<td>4.79</td>
<td>5.50</td>
<td>6.96</td>
<td>2.36</td>
<td>8.16</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>Permutational sensitivity (mean standard deviation) ↓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Null Instruction</td>
<td>6.02</td>
<td>1.99</td>
<td>3.82</td>
<td>4.56</td>
<td>5.34</td>
<td>1.12</td>
<td>1.87</td>
<td>1.48</td>
<td>1.24</td>
<td>2</td>
</tr>
<tr>
<td>Generic Instruction</td>
<td>6.01</td>
<td>2.19</td>
<td>3.89</td>
<td>4.56</td>
<td>5.49</td>
<td>1.15</td>
<td>1.68</td>
<td>1.33</td>
<td>1.22</td>
<td>2</td>
</tr>
<tr>
<td>PromptSource</td>
<td>6.06</td>
<td>2.15</td>
<td>3.61</td>
<td>4.69</td>
<td>4.30</td>
<td>1.07</td>
<td>1.67</td>
<td>1.47</td>
<td>1.17</td>
<td>2</td>
</tr>
<tr>
<td>Ad hoc</td>
<td>6.10</td>
<td>2.37</td>
<td>3.77</td>
<td>4.61</td>
<td>4.37</td>
<td>1.11</td>
<td>1.66</td>
<td>1.41</td>
<td>1.23</td>
<td>0</td>
</tr>
<tr>
<td>Low Perplexity</td>
<td>6.13</td>
<td>2.24</td>
<td>3.50</td>
<td>4.61</td>
<td>4.29</td>
<td>1.13</td>
<td>1.69</td>
<td>1.46</td>
<td>1.27</td>
<td>2</td>
</tr>
<tr>
<td>APE</td>
<td>6.14</td>
<td>2.36</td>
<td>3.69</td>
<td>4.84</td>
<td>5.08</td>
<td>1.10</td>
<td>1.78</td>
<td>1.41</td>
<td>1.21</td>
<td>0</td>
</tr>
<tr>
<td>RLPrompt</td>
<td>6.26</td>
<td>2.06</td>
<td>3.82</td>
<td>4.89</td>
<td>5.64</td>
<td>1.08</td>
<td>1.65</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5: Mean standard deviation of few-shot accuracy on varying selections and permutations of demonstrations respectively. The ‘# wins’ column represents the number of tasks where a method achieves best performance.
Our findings suggest that in practical in-context learning settings, simpler prompting methods, such as task-agnostic or expert manually written instructions, often outperform automatically synthesized ones at the model scales we consider. Task-agnostic methods show strong performance in few-shot settings, whereas expert manual instructions appear crucial for achieving good zero-shot accuracy. The superiority of these straightforward methods over automatically induced instructions, which are often not competitive even with simple baselines, suggests a lack of transferability and generalizability among automatic induction methods. The competitive performance of automatic induction methods like APE and RLPrompt as reported by their authors implies either a limitation in their generalizability to a broader range of models and tasks, or the need for substantial hyperparameter tuning to get them to work well across models and tasks.

On the other hand, since few-shot ICL performance remains largely consistent regardless of the choice of instruction, practitioners could perhaps benefit from simply providing a few in-context demonstrations when available. The fact that null instructions perform superior to automatically induced instructions, which are often not competitive even with simple baselines, suggests a lack of transferability and generalizability among automatic induction methods. The competitive performance of automatic induction methods like APE and RLPrompt as reported by their authors implies either a limitation in their generalizability to a broader range of models and tasks, or the need for substantial hyperparameter tuning to get them to work well across models and tasks.

Our findings underscore a broader issue regarding the inconsistent and often insufficient evaluation of instruction selection and induction techniques. We call for more comprehensive evaluations in this space and encourage the use of our evaluation suite to facilitate this process.
6 Conclusion

We conduct the broadest attempt to our knowledge, to systematically study the generalizability of popular instruction selection and induction methods for ICL in LLMs. We find that simpler approaches such as using task-agnostic instructions, expert manual instructions, or even omitting instructions entirely tend to show good performance more consistently when evaluating across a wide variety of tasks and models. Our work indicates the need for more systematic and consistent evaluations in the instruction induction space. To facilitate such analyses, we release the InstructEval suite which provides coverage over 13 diverse autoregressive LLMs and 9 tasks spanning classification, multiple-choice QA, and generative QA.

7 Limitations

For consistency, we conduct all our experiments using prompts that begin with an instruction, are followed by demonstrations, and end with an unannotated test example (as illustrated in Figure 2). We also use 6 demonstrations in all the evaluations we perform in the few-shot setting. We do not assess the effect of varying the prompt format or the number of demonstrations used since the choices we experiment using are not atypical in practical ICL settings. However, explorations into the effect of varying the prompt format and number of demonstrations are supported by our evaluation framework and we leave these to future work.

Our work seeks to assess the effect of instruction choice in models that can reasonably run on consumer hardware. Hence, we do not include any models of size >20B parameters in our evaluation suite. As a consequence, our findings may not carry over to much larger models.

8 Broader Impact

We perform experiments and release an evaluation suite to systematically assess the effect of instruction choice on ICL performance. Our suite can aid in understanding and decreasing the risk of miscommunication between users and LLMs, and help assess and mitigate the risk of biases that may emerge from various prompting methods. Our work assesses the transferability of various instruction selection methods and allows for increased transparency and reduced statistical bias during their assessment. A better understanding of how LLMs respond to prompts can potentially help identify and prevent undesirable effects while promoting desirable ones. However, it may also be possible for bad actors to use such evaluations to find ways to systematically promote harmful effects.

Acknowledgements

We acknowledge support from the National Science Foundation under Grant No. 2239363. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.
References


Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.


Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021b. What makes good in-context examples for gpt-3?


A Implementation details

A.1 Evaluation

We ameliorate the effect of statistical noise by rerunning each instruction selection/induction method we study using 5 random seeds independently for every task (and for every model, where applicable) and report results for each instruction selection/induction method by averaging the aggregated scores associated with all 5 instructions.

We use $K = 6$ demonstrations randomly sampled from the task’s training set for every experiment we perform in the few-shot setting.

To maintain consistency, we perform all our experiments using fixed task-specific prompt templates. Each prompt begins with the instruction being tested, and continues into a sequence annotated demonstrations and a test example, each of which follow the templates listed in Table 8.

A.2 Instruction selection methods

PromptSource We sample and evaluate a random subset of instructions from those included in the public PromptSource repository for each task in our evaluation suite.

Ad hoc We obtain the set of ad hoc instructions we evaluate for a task by tasking ChatGPT with generating 40 paraphrases of instructions for the task that we obtain from PromptSource. We then select a random sample of instructions from this 40-instruction pool and perform evaluations using each sampled instruction.

Low Perplexity For each task, we rerank a pool of ChatGPT paraphrases of PromptSource instructions using the SPELL algorithm described by (Goßen et al., 2022). When prompting a specific model, we choose the instruction with the lowest perplexity as measured by that model.

APE We use the official repository released by (Zhou et al., 2022) to generate instructions for each of the tasks we consider. To remain consistent with the original methodology, we use the OpenAI DaVinci to induce and evaluate instructions during the induction phase. We opt to use the simpler version of the methodology proposed by the authors since they report that the computationally intensive Monte-Carlo search strategy only provides marginal improvements in accuracy.

RLPrompt We use the public repository released by Deng et al. (2022) to induce instructions for the RLPrompt baseline in our evaluations. Although the original work only performs evaluations over classification datasets with a fixed label-space, we augment the codebase to allow instruction induction for MCQ tasks as well by formulating these as cloze-style completion tasks. We create instructions for all tasks using the default settings of hyperparameters included with the codebase.
Tasks | Demonstration template
--- | ---
AG News (Zhang et al., 2015) | News: (text) \nCategory: (label)
ANLI (Nie et al., 2020) | Premise: (premise) \nHypothetisis: (hypothesis) \nRelation: (label)
BoolQ (Clark et al., 2019) | Passage: (passage) \nQuestion: (question) \nAnswer: (label)
IMDB (Maas et al., 2011) | Review: (text) \nSentiment: (label)
TweetEval Emotion (Mohammad et al., 2018) | Tweet: (text) \nEmotion: (label)
CosmosQA (Huang et al., 2019) | Passage: (context) \nQuestion: (question) \nAnswer: (answer)
HellaSwag (Zellers et al., 2019) | Sentence: (ctx) \nAnswer: (answer)
NQ-Open (Kwiatkowski et al., 2019) | Question: (question) \nAnswer: (answer)
TriviaQA (Joshi et al., 2017) | Question: (question) \nAnswer: (answer)

Table 8: Tasks included in our evaluation suite, and the demonstrations templates we use for each task.

<table>
<thead>
<tr>
<th>Method</th>
<th>Example instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null instruction</td>
<td>(empty string)</td>
</tr>
<tr>
<td>Generic instruction</td>
<td>Solve the following task:</td>
</tr>
<tr>
<td>PromptSource (Bach et al., 2022)</td>
<td>What label best describes this news article?</td>
</tr>
<tr>
<td>Ad hoc</td>
<td>Complete the problem.</td>
</tr>
<tr>
<td>Low Perplexity (Gonen et al., 2022)</td>
<td>Which newspaper section is most likely to feature this news article?</td>
</tr>
<tr>
<td>APE (Zhou et al., 2022)</td>
<td>classify each input into one of the following categories: World, U.S., Business, Sci/Tech, or Sports.</td>
</tr>
<tr>
<td>RLPrompt (Deng et al., 2022)</td>
<td>Tools undergradCam firmwareCam</td>
</tr>
</tbody>
</table>

Table 9: Example instructions obtained using each method for the AG News task

### Task-agnostic
We completely omit instructions from the prompt when evaluating null instructions. We list the set of generic instructions we evaluate in Table 10.

We include examples of the instructions we obtain for each method in Table 9.

### B Drawbacks of aggregation techniques used in previous work

Some previous works like the HELM (Liang et al., 2022) benchmark also face similar challenges when attempting to compare high-dimensional vectors – each representing a model evaluated over a variety of tasks – against each other. HELM resorts to scoring models using head-to-head win rates. The win rate associated with a model indicates the fraction of head-to-head comparisons between the given model and all other models, across all scenarios, where the given model performs better along a specific metric. A notable disadvantage of this scoring technique is that it obscures the magnitude of variation in the metric associated with each test model and only conveys ordinal information about the relative performances of each model. This characteristic of head-to-head win rates makes them unsuitable for spotting broad trends across families of prompting methods.

In other works like BIG-bench (Srivastava et al., 2022), raw metric scores representing task performance are normalized to vary from a range of 0-100 such that a normalized score of 0 corresponds to poor performance, while a normalized score of 100 corresponds to excellent performance on the task. This is done in an attempt to be able to compare the performance of a model across a variety of tasks of varying difficulty such that the normalization proves more forgiving on difficult tasks. While this score does capture cardinal information associated with the underlying variable, it relies on the knowledge of human experts to determine raw score thresholds that constitute poor or excellent performance along a given metric. To apply such a normalization scheme in our case, one would need access to a large array of such threshold scores corresponding to each model scale, task, and metric we consider. Obtaining such threshold scores across all our settings is challenging given the number of tests we perform and the variety of metrics we consider. Hence, this type of normalization proves infeasible in our case.
C Full results

We make our entire array of unaggregated evaluation results available along with the InstructEval codebase at https://github.com/princeton-nlp/InstructEval.