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

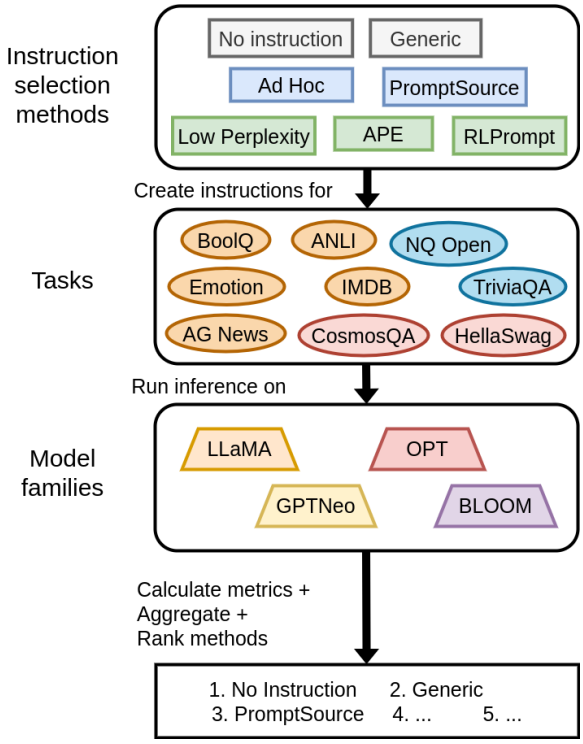


Figure 1: InstructEval allows the assessment of instruction selection methods for ICL across a range of models and tasks along five metrics.

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

[1]Code: https://github.com/princeton-nlp/InstructEval

Findings of the Association for Computational Linguistics: NAACL 2024, pages 4336–4350

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

Figure 2: An example of a prompt following the template we use for IMDB. By 'prompt' we refer to the concatenation of the instruction, solved demonstrations and an unsolved test example.

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[2] (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*[3] 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.

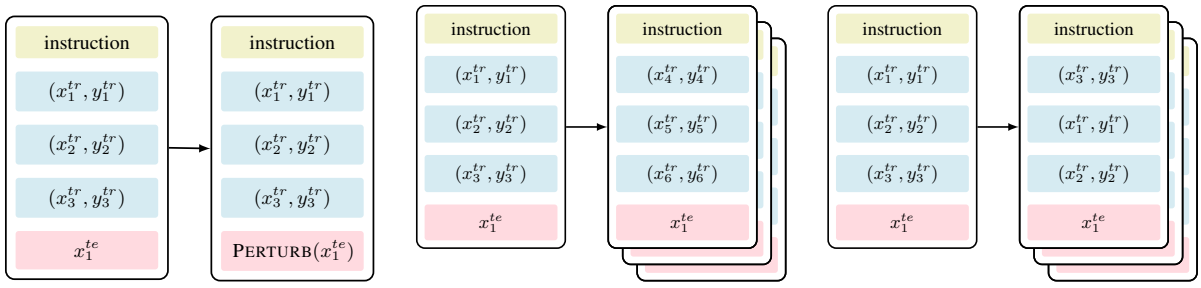
Sensitivity metrics Previous work has shown that the accuracy obtained using a prompt template can fluctuate significantly as a function of the set of demonstrations included in the prompt (Liu et al., 2021a; Su et al., 2022; Rubin et al., 2022; Wang et al., 2023a) and the order they are presented in (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

[2]Instructions are omitted during 'Null instruction' evaluations. Demonstrations are omitted in zero-shot evaluations.

[3]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.



(a) Perturbation accuracy (b) Selectional sensitivity (c) Permutational sensitivity

Figure 3: We provide schematic diagrams that show prompts are modified to measure *perturbation accuracy*, *selectional sensitivity* and *permutational sensitivity*. We perturb the test input to measure perturbation accuracy, and demonstration selection and permutation respectively while measuring selectional and permutational sensitivity.

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:

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

Consider a collection of models $\mathcal{M}$ 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 $r_{tmi} = \text{r-gain}_{S_{tm}}(s_{tmi})$. 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

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

These $\overline{r}_{ti}$ 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 $\overline{r}_{ti}$ on task t, this means that method i exceeds average performance (across $\mathcal{I}$) on task t by $\overline{r}_{ti}$ 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 σ_{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.

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

Table 1: Tasks included in our evaluation suite.

Model Family	Size
BLOOM (Scao et al., 2022b)	1.1B, 1.7B, 3B, 7.1B
GPT Neo (Black et al., 2021, 2022)	1.3B, 2.7B, 20B
LLaMA (Touvron et al., 2023)	7B, 13B
OPT (Zhang et al., 2022b)	1.3B, 2.7B, 6.7B, 13B

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

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] Standard summarization metrics correlate poorly with human preferences (Liang et al., 2022; Goyal et al., 2023).

Method	Task-specific	Automatic induction
Null instruction	✗	✗
Generic instruction	✗	✗
PromptSource (Bach et al., 2022)	✓	✗
Ad hoc	✓	✗
Low Perplexity (Gonen et al., 2022)	✓	✓
APE (Zhou et al., 2022)	✓	✓
RLPrompt (Deng et al., 2022)	✓	✓

Table 3: Instruction selection methods we evaluate

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.

- **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

Method	CLS					MCQ		GQA		# wins
	AG News	ANLI	BoolQ	IMDB	Emotion	HellaSwag	CosmosQA	TriviaQA	NQ-Open	
Zero-shot accuracy (mean relative gain) ↑										
Null Instruction	2.26	1.07	**2.48**	−3.52	−5.30	**2.54**	**5.94**	−3.02	−25.67	3
Generic Instruction	3.55	−0.39	0.03	1.69	2.39	−0.13	−1.67	−1.46	−5.99	0
PromptSource	**5.81**	**1.38**	−0.65	**4.34**	**5.13**	−1.54	−3.42	**17.08**	**22.15**	6
Ad hoc	−0.33	0.21	0.55	1.41	0.66	−0.27	−2.46	−1.97	2.31	0
Low Perplexity	−0.59	1.22	0.56	0.84	−4.07	−1.38	−2.18	−5.99	2.81	0
APE	−15.63	−3.86	−1.07	−1.77	−0.26	−1.06	0.00	−4.64	4.39	0
RLPrompt	4.92	0.37	−1.89	−2.99	1.46	1.85	3.79	–	–	0
Few-shot accuracy (mean relative gain) ↑										
Null Instruction	4.09	−0.22	**0.87**	−0.80	**5.89**	0.17	**1.33**	**0.45**	−0.02	4
Generic Instruction	**5.16**	−0.20	−0.10	0.45	4.84	0.04	−0.18	0.11	0.11	1
PromptSource	0.83	0.14	−0.79	0.39	−4.39	−0.06	−0.94	−0.36	**0.61**	1
Ad hoc	2.18	−0.10	−0.05	**0.60**	−5.63	−0.21	−0.59	0.09	−0.49	1
Low Perplexity	−1.96	**0.31**	−0.40	0.20	−6.79	−0.23	−0.61	−0.06	−0.02	1
APE	−15.43	0.10	0.06	−0.69	1.17	0.02	0.17	−0.24	−0.19	0
RLPrompt	5.13	−0.02	0.40	−0.14	4.90	**0.27**	0.81	–	–	1
Few-shot perturbation accuracy (mean relative gain) ↑										
Null Instruction	4.09	−0.08	0.11	−0.27	**5.98**	0.11	**1.10**	**0.81**	**1.28**	4
Generic Instruction	**5.15**	−0.18	−0.16	**0.56**	4.23	−0.02	−0.02	0.08	0.10	2
PromptSource	1.14	0.27	−0.02	0.33	−3.92	0.06	−0.53	−0.65	0.04	0
Ad hoc	1.68	0.51	−0.34	0.37	−5.87	−0.08	−0.63	−0.28	−0.61	0
Low Perplexity	−2.39	**0.68**	−0.12	−0.20	−6.61	−0.09	−0.66	−0.03	−0.78	1
APE	−14.32	−1.20	**0.28**	−0.82	1.26	−0.13	0.21	0.06	−0.03	1
RLPrompt	4.65	−0.01	0.24	0.03	4.94	**0.15**	0.53	–	–	1

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.

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

Table 5: Mean standard deviation of few-shot accuracy on varying selections and permutations of demonstrations respectively. The '# wins' column respresents the number of tasks where a method achieves best performance.

Method	< 6B parameters			≥ 6B parameters		
	CLS	MCQ	GQA	CLS	MCQ	GQA
Zero-shot accuracy (mean relative gain) ↑						
Null Instruction	−2.89	1.71	−15.86	2.07	**7.19**	−12.58
Generic Instruction	1.71	0.69	−0.64	1.16	−2.76	−7.33
PromptSource	**2.77**	−2.18	**25.03**	**3.70**	−2.83	**13.30**
Ad hoc	1.87	−0.94	4.56	−1.11	−1.86	−4.95
Low Perplexity	−2.35	−1.09	−8.24	1.85	−2.58	6.17
APE	−3.13	−0.54	−4.85	−6.14	−0.51	5.39
RLPrompt	2.01	**2.37**	–	−1.54	3.34	–
Variation Range	5.90	4.55	40.89	9.84	10.02	25.88
Few-shot accuracy (mean relative gain) ↑						
Null Instruction	2.63	**0.75**	**0.89**	1.20	**0.76**	−0.57
Generic Instruction	**3.09**	−0.10	−0.15	0.80	−0.03	0.41
PromptSource	−1.18	−0.58	−0.20	−0.28	−0.41	**0.51**
Ad hoc	−0.55	−0.45	0.04	−0.65	−0.35	−0.47
Low Perplexity	−2.57	−0.48	−0.30	−0.75	−0.35	0.26
APE	−4.10	0.13	−0.28	−1.62	0.06	−0.13
RLPrompt	2.69	0.73	–	**1.31**	0.32	–
Variation Range	7.19	1.33	1.19	2.93	1.17	1.08

Table 6: Mean relative gain values for zero-shot and few-shot accuracy computed separately over models with < 6 and ≥ 6 billion parameters, and averaged by task-type. We also tabulate the total range of variation of these values in each setting.

"large" models with ≥ 6 billion parameters in Table 6. For ease of comparison, we average the mean relative gain values thus obtained by task-type. Although the observations that PromptSource and task-agnostic instructions tend to perform the best across zero- and few-shot settings persist across model scales, we find that the ranges of variation in the few-shot mean relative gain values for large models are consistently smaller than those for small models for every task-type. This suggests that large models are able to grasp task semantics from demonstrations (when provided) while small models are more sensitive to the instruction used.

We also tabulate the mean relative gain values for zero-shot and few-shot accuracies computed separately for each model family in Table 7, to understand the effect that model family has on instruction performance. Although the trends we discuss in Section 5.1 regarding task-agnostic instructions and PromptSource instructions respectively tending to dominate few-shot and zero-shot settings persist, the instruction selection method that emerges the best-performing alternative often changes on varying the choice of model family and task-type. For instance, the automatic instruction induction methods APE and RLPrompt do show above-average performance for certain model families and task-types, but this behavior does not consistently extend to other families and types. This indicates a lack of generalizability in these methods.

5.4 Discussion

Our findings reveal 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.

Our findings suggest that ICL practitioners may often be better off forgoing computationally expensive instruction induction or selection methods in favor of task-agnostic or manually written instructions, which seem to generalize better. Interestingly, we also find that methods that excel for one model and task do not necessarily also perform well for other tasks and models. Consequently, ICL practitioners may be forced to experiment with various instruction selection methods on a model- and task-specific basis in a manner reminiscent of hyperparameter tuning to find the best choice.

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 tend to outperform all other methods in our study in few-shot settings suggests that it can be challenging to find instructions that reliably inform diverse models about task semantics. When models fail to grasp the semantics signaled by instructions, these may simply serve as a source of noise, hence impairing ICL performance.

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.

Method	BLOOM				GPT Neo				LLaMA				OPT			
	CLS	MCQ	GQA	# wins	CLS	MCQ	GQA	# wins	CLS	MCQ	GQA	# wins	CLS	MCQ	GQA	# wins
Zero-shot accuracy (mean relative gain) ↑																
Null Instruction	−1.40	**3.60**	−11.93	3	−1.80	**1.34**	−10.46	1	4.02	**9.25**	−9.73	3	−1.22	**4.54**	−22.07	4
Generic Instruction	**5.03**	−1.27	−0.72	1	−0.35	−0.20	−1.86	1	−2.73	−2.09	−13.25	0	1.33	−0.47	−3.47	0
PromptSource	2.03	−2.89	**14.22**	2	1.61	−0.69	**35.75**	2	**10.01**	−3.89	7.82	2	**2.16**	−2.70	**18.70**	4
Ad hoc	0.45	−1.15	2.93	0	**2.23**	0.56	0.08	2	−3.70	−2.94	−2.56	0	1.35	−2.23	−1.26	0
Low Perplexity	−3.76	−2.19	3.94	1	−2.85	0.45	−20.87	1	4.97	−4.87	5.10	2	2.09	−1.50	4.32	1
APE	−6.10	0.75	−8.44	0	0.14	−2.00	−2.87	1	−7.01	−0.09	**12.62**	2	−5.18	−0.92	3.78	0
RLPrompt	3.76	3.15	–	2	1.02	0.54	–	1	−5.57	4.64	–	0	−0.53	3.28	–	0
Few-shot accuracy (mean relative gain) ↑																
Null Instruction	3.04	**1.11**	**0.85**	4	**1.31**	**0.33**	−0.11	2	**1.40**	0.79	−0.35	4	1.67	**0.70**	0.11	1
Generic Instruction	**3.64**	−0.24	−0.44	2	1.27	0.31	0.04	2	0.24	−0.11	0.23	2	1.89	−0.17	**0.64**	3
PromptSource	−1.21	−0.83	0.19	1	−0.44	−0.17	−0.08	2	−0.30	−0.65	0.18	0	−0.79	−0.34	0.20	1
Ad hoc	−0.35	−0.57	0.41	1	−1.02	−0.12	−0.59	0	−1.26	−0.62	**0.31**	0	−0.20	−0.33	−0.77	0
Low Perplexity	−3.00	−0.58	−0.23	0	−1.04	−0.15	−0.15	0	−0.94	−0.59	0.25	1	−1.37	−0.38	0.07	1
APE	−5.26	0.29	−0.78	0	−1.14	−0.29	**0.85**	2	0.29	0.38	−0.62	1	−3.64	0.05	−0.24	0
RLPrompt	3.14	0.82	–	1	1.06	0.10	–	1	0.57	**0.80**	–	1	**2.45**	0.47	–	3

Table 7: Mean relative gain values for zero-shot accuracy and few-shot accuracy computed separately over individual model families and averaged by task-type. Positive values represent above-average performance, and negative values represent below-average performance. We also tabulate the number of tasks where a method achieved highest aggregated performance in the '# wins' column under every model family.

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

Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. 2022. In-context examples selection for machine translation.

Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Xiangru Tang, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. Promptsource: An integrated development environment and repository for natural language prompts.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. Gpt-neox-20b: An open-source autoregressive language model.

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.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901.

Yanda Chen, Chen Zhao, Zhou Yu, Kathleen McKeown, and He He. 2023. On the relation between sensitivity and accuracy in in-context learning.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In NAACL.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 3369–3391, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A. Smith, and Luke Zettlemoyer. 2022. Demystifying prompts in language models via perplexity estimation.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2023. News summarization and evaluation in the era of gpt-3.

Or Honovich, Uri Shaham, Samuel R Bowman, and Omer Levy. 2022. Instruction induction: From few examples to natural language task descriptions. arXiv preprint arXiv:2205.10782.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2391–2401, Hong Kong, China. Association for Computational Linguistics.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. arXiv e-prints, page arXiv:1705.03551.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. Transactions of the Association for Computational Linguistics, 7:453–466.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2022. Holistic evaluation of language models.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021a. What makes good in-context examples for gpt-3? In Workshop on Knowledge Extraction and Integration for Deep Learning Architectures; Deep Learning Inside Out.

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

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021a. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. arXiv preprint arXiv:2104.08786.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021b. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. CoRR, abs/2104.08786.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. 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. Association for Computational Linguistics.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In Proceedings of the 12th international workshop on semantic evaluation, pages 1–17.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial nli: A new benchmark for natural language understanding. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics.

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. 2022. Training language models to follow instructions with human feedback.

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. 2023. What in-context learning"learns"in-context: Disentangling task recognition and task learning.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. In Advances in Neural Information Processing Systems.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2655–2671.

Teven Le Scao, Angela Fan, Christopher Akiki, Elizabeth-Jane Pavlick, Suzana Ili'c, Daniel Hesslow, Roman Castagn'e, Alexandra Sasha Luccioni, Franccois Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Rose Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, et al. 2022a. Bloom: A 176b-parameter open-access multilingual language model. ArXiv, abs/2211.05100.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Frohberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Ur-

mish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névéol, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Ononiwu, Habib Rezanejad, Hessie Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Karen Fort, Livia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguier, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel León Periñán, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangasai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A Castillo, Marianna Nezhurina, Mario Sänger, Matthias Samwald, Michael Cullan, Michael Weinberg, Michiel De Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. 2022b. Bloom: A 176b-parameter open-access multilingual language model.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. arXiv preprint arXiv:2206.04615.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2022. Selective annotation makes language models better few-shot learners. ArXiv, abs/2209.01975.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Xinyi Wang, Wanrong Zhu, and William Yang Wang. 2023a. Large language models are implicitly topic models: Explaining and finding good demonstrations for in-context learning. ArXiv, abs/2301.11916.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models.

Albert Webson and Ellie Pavlick. 2022. Do prompt-based models really understand the meaning of their prompts? In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2300–2344, Seattle, United States. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed H Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In Advances in Neural Information Processing Systems.

Jerry W. Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. 2023. Larger language models do in-context learning differently. ArXiv, abs/2303.03846.

Benfeng Xu, Quan Wang, Zhendong Mao, Yajuan Lyu, Qiaoqiao She, and Yongdong Zhang. 2023. $k$nn prompting: Beyond-context learning with calibration-free nearest neighbor inference.

Sohee Yang, Jonghyeon Kim, Joel Jang, Seonghyeon Ye, Hyunji Lee, and Minjoon Seo. 2023. Improving probability-based prompt selection through unified evaluation and analysis.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022a. Opt: Open pre-trained transformer language models. ArXiv, abs/2205.01068.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022b. Opt: Open pre-trained transformer language models.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In NIPS.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers.

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 (Gonen 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.

Method	Example instruction
Null instruction	(empty string)
Generic instruction	`Solve the following task:`
PromptSource (Bach et al., 2022)	`What label best describes this news article?`
Ad hoc	`Which newspaper section is most likely to feature this news article?`
Low Perplexity (Gonen et al., 2022)	`Which part of a newspaper do you think this article belongs to? World News, Sports, Business or Science and Technology?`
APE (Zhou et al., 2022)	`classify each input into one of the following categories: World, U.S., Business, Sci/Tech, or Sports.`
RLPrompt (Deng et al., 2022)	`Tools undergradCam firmwareCam`

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

Generic Instructions
`Solve the following task:`
`Find the answer below:`
`Complete the problem.`
`Find the best solution to the question below:`
`Complete the question below:`

Table 10: Sample generic instructions

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 evalutation results available along with the InstructEval codebase at https://github.com/princeton-nlp/InstructEval.