Instruction-tuned LLMs

Anonymous ACL submission

Abstract

This document is a supplement to the general instructions for *ACL authors. It contains instructions for using the LaTeX style files for ACL conferences. The document itself conforms to its own specifications, and is therefore an example of what your manuscript should look like. These instructions should be used both for papers submitted for review and for final versions of accepted papers.

1 Introduction

Figure 1: How well do models trained on instruction-tuning datasets generalize to novel instructions (unobserved in training)? Our analysis suggests that they do not do so very well. Above we show a case where pairing an example with an observed instruction yields the correct output, while providing a distinct but semantically equivalent instruction produces an incorrect response. We propose and evaluate a simple method that improves this.

Large Language Models (LLMs) have come to dominate NLP, in part because they enable zero- and few-shot adaptation to new tasks via *prompting* (**????**). Recent work has demonstrated the promise of fine-tuning such models with natural language instructions. Such *instruction-tuning* improves LLM performance in zero- and few-shot settings, sometimes dramatically, especially for "mid-sized" models (**??**). For example, on some benchmarks the instruction-tuned Flan-T5-XL (3B parameters) (**?**) outperforms GPT-3 (175B), despite being dramatically smaller. Furthermore, LLaMa-7B (**?**)—after being fine-tuned on large-scale corpora on the Alpaca (**?**) instruction set—outperforms GPT-3 across a range of NLP benchmarks.

These empirical successes have motivated efforts to curate instruction-augmented task collections for meta-learning (**???**), and research into improving instruction-tuning (**???**). In this work we investigate how robust instruction-tuned models are. More specifically, we ask: How sensitive are instruction-tuned LMs to shifts in instruction phrasings at test time? This is particularly important given that the primary motivation of instruction tuning is to facilitate zero-shot adaptation via natural language instruction: If models are overly sensitive to the particular phrasing of a task instruction it may greatly limit their utility in practice.

Prior work—reviewed at length in Section 2—has established that LLMs do not seem to intuitively "understand" prompts (**???**), but these efforts did not consider instruction-tuned models specifically. Recent, contemporaneous work to ours (**?**) investigated the robustness of instruction-tuned models, and found that instruction-tuned T5 (**?**) is robust to instruction perturbations in few-shot settings, but less so in zero-shot application. We contribute a more in-depth analysis of this phenomena across a much wider set of instruction-tuned models and benchmarks. We also introduce and evaluate a method for improving the robustness of such models, with promising results.

More specifically, we collect a relatively large set of task instructions manually composed by NLP researchers; these are valid instructions but distinct from those found in the Flan collection. We then assess the performance of LLMs fine-tuned on the Flan collection instruction set when given these novel instructions on two benchmarks: MMLU (**?**) and BBL (**?**). We find that using novel instructions in zero-shot application degrades accuracy considerably (Figure **??** illustrates this). For example, comparing the performance of Flan-T5 XXL when using (a) instructions that were seen in training to (b) semantically equivalent but unobserved in training, we observe a 6.9 point drop in absolute performance on average across large benchmarks.

Our **main contributions** are summarized as follows. (1) We perform a comprehensive and in-depth analysis of the robustness of instruction-tuned LLMs across three "families" of such models (Flan-T5 (**?**), Alpaca (**?**), and T0 (**?**)) using large benchmarks (**??**). For this we collect a large set of new task instructions manually composed by researchers in NLP; we will release this dataset to facilitate additional work on instruction robustness. We observe substantial performance degradation when using "novel" (unseen in training) instructions. (2) We propose a simple method to improve robustness by imposing an objective encouraging LLMs to induce similar representations for semantically equivalent instructions. We find that this consistently improves the performance realized when using novel but appropriate task instructions.

2 Related Work

Multitask learning and instruction-tuning Training a single text-to-text model capable of providing responses to arbitrary queries has been an aspiration in NLP for at least half a decade. For example, prior to modern prompting and instructing strategies, there were efforts to unify disparate tasks by reframing them as instances of general *question answering* (**???**). More recent efforts have focussed on compiling and fine-tuning LLMs on corpora comprising diverse tasks with associated natural language instructions (**???**); we refer to this strategy as instruction-tuning. One example of this is `Super-NaturalInstructions` (**?**), which compiles over 1600 tasks and enriches these with both instructions and negative examples. Similarly, the recently released OPT-IML Bench (**?**) comprises 2000 NLP tasks. The Flan 2022 task collection (**?**) additionally features *Chain-of-Thought* (CoT) style "reasoning" chains in instruction templates; the authors show that including these (as well as zero-shot examples and "input inversions") during instruction fine-tuning yields improvements on held-out tasks.

These meta-resources—collections of instructions, tasks, and samples—have facilitated the training of instruction-tuned model families such as Flan-T5, Flan-PaLM (**?**), and OPT-IML (**?**).[1] Results have been encouraging; fine-tuning LLMs to follow instructions provides clear and consistent gains across models, and, perhaps most exciting, enables relatively "small" (∼10B) LLMs to achieve near SOTA performance comparable to massive (∼175B) models (**?**). This has motivated interest in characterizing how instructions help models, and developing techniques to further improve instruction-tuning; we review recent efforts related to these two research threads below.

Evaluating prompting and instruction capabilities Instructions may be seen as a special sort of model prompting, which a few recent efforts have critically evaluated. For example, Webson and Pavlick ask whether models meaningfully "understand" prompts (**?**), finding that they largely do not: Performance is often unaffected when irrelevant and misleading prompts are provided. In follow up work, Jang *et al.* (**?**) evaluates performance on negated prompts, observing an "inverse-scaling" phenomenon in which larger models perform worse in this case.

Other work has attempted to characterize how and when *in-context learning* (ICL)—i.e., including a few examples in prompts—works (**?????**). ICL is a form of prompting orthogonal to the present effort, as we are primarily interested in the zero-shot adaptability of instruction-tuned LLMs.

In work contemporaneous to ours, Gu *et al.* (**?**) investigated how robust instruction-tuned models are to instruction perturbations (e.g., dropping words) and paraphrasings. They found that models are relatively robust when given examples (i.e., in few-shot settings), but quite sensitive when used zero-shot; this is qualitatively in line with our findings. Our work differs in important way from this coincident research: (1) We provide a much more comprehensive analysis of robustness; Gu *et al.* considered *only* T5 instruction-tuned on a single instruction dataset, whereas we evaluate three LLMs (and different sizes of each) using five instruction tuning datasets, and we evaluate using over 80 test tasks in all (Gu *et al.* considered only 12). (2) We propose and evaluate a new approach to *improving* the robustness of instruction-tuned models; Gu *et al.* offered no mechanism to improve robustness.

Improving instruction-tuning Past work has also sought to improve instruction-tuning in various ways. One means to do so is to instruction tune based on human feedback (**?????**). This tends to improve open-ended model responses but degrade performance on downstream tasks. Another strategy is to leverage existing resources to automatically generate instruction-tuning datasets at scale. For example, Wang *et al.* (**?**) use LLMs to

[1] Somewhat confusingly, in the case of FLAN and OPT, the corpora (i.e., benchmarks comprising tasks and instructions) and LLMs fine-tuned using them are both referred to with the associated acronym as prefix: For instance, Flan-T5 denotes a T5 (**?**) variant fine-tuned with the Flan collection.

generate instructions, inputs, and outputs and use these to improve their own instruction-following capabilities. In a similarly meta vein, Zhou and colleagues (?) propose using LLMs to engineer prompts. Finally, Ye *et al.* (?) propose "flipping" the standard task by tasking LLMs with generating *instructions*, given an input and label.

3 Instruction Datasets

3.1 Evaluation Benchmarks

We evaluate a set of instruction-tuned models on two large benchmarks: MMLU (?) and Big-Bench (?). MMLU is a multiple-choice question-answering benchmark comprising 57 tasks that require expert knowledge. Big-Bench is a collaboratively built benchmark containing 204 diverse tasks from various domains; here consider the Big-Bench Lite subset, and we include only QA, multi-class, and binary classification tasks, yielding 18 tasks from in all.

3.2 Collecting New Instructions from NLP Researchers

We aim to evaluate instruction-tuned models when they are provided instructions which are semantically equivalent to, but superficially different from, those with which they were trained. To this end, we enlist NLP researchers (graduate students) to compose novel instructions for the tasks considered; these particular instruction phrasings were therefore *unobserved* during instruction fine-tuning.

More specifically, we recruited 36 NLP graduate students working in NLP. All had at least some experience with instruction-tuned models and the downstream tasks included in the evaluation benchmarks. For each of the 18 tasks in BBL and all tasks in MMLU, we asked 12 graduate students to write one (distinct) instruction they would use for zero-shot inference with an instruction-tuned model. We provide details on this instruction collection process in Appendix A. We will release all 319 instructions acquired for this work to ensure the reproducibility of this work and to facilitate further research on instruction-tuned model robustness.

4 Evaluating the Robustness of Instruction-tuned LLMs

4.1 Models and Data

We conduct experiments with model variants trained over three instruction collections (these provide *observed* task instructions): P3 (?), Flan-2022 (?), and Alpaca (?). To facilitate our analyses, we manually identified all instructions that correspond to (a) multiple-choice question answering (QA), (b) binary classification (BC), or tasks that demand "yes" or "no" responses, and (c) multi-class classification (MC), which requires classifying inputs into a finite set of categories.

To evaluate model robustness with respect to instruction phrasings we use two benchmarks: MMLU (?) and Big-Bench Lite (BBL) (?) along with the acquired set of novel instructions described in Section 3.2. We include all 57 tasks from MMLU, and 14 of 24 tasks from BBL. From the latter we exclude two tasks that rely on generation metrics, four that use exact-match, and four that contain tokens unrecognized by the T5 and/or LLaMa tokenizer (e.g., inputs are emojis in one task).

QA	In this task, you are given a multiple-choice question and you have to pi correct option. Answer with option indexes (i.e., "A", "B", "C", and "D' Q: {question} A. {choiceA} B. {choiceB} C. {choiceC} D. {choiceD}
MC	Pick one category for the following text. The options are - {options} {te
BC	{paragraph} Choose your answer: According to the above paragraph, th question "{question}" is "{response}"?

Table 1: Examples of observed instructions we collected for three general types of tasks.

We use the same instructions for all tasks in the same category, taken from the published instruction tuning datasets associated with each model. These instructions are general, e.g., in the case of classification they request that the model consider an example with respect to categorization criteria and label space provided by the instance, and select an appropriate category (examples in Table 1). One can "mix-and-match" such instructions so long as they are appropriate for the task type.

Observed Instructions			
Instruction Type	QA	MC	BC
Flan	50	35	18
Alpaca	20	20	11
P3	13	8	7

Unobserved Instructions		
Number of tasks	1	14
Instructions per task	20	10
Total instructions	20	140

Table 2: Counts of instruction phrasings (unobserved and observed) we use for evaluations.

4.2 Results

We present the main aggregated analysis results in Figure 2 and Table 3. The take-away here is that using instructions unobserved in training—but manually composed for the task at hand and so semantically appropriate—leads to considerable degradation in performance: On average, unobserved instructions reduce accuracy by over five points across models considered. Table 3 reports results disaggregated by task type; we observe that classification tasks are most harmed by use of novel instructions. We provide additional, more granular (dataset-level) results in the Appendix.

(a) Average zero-shot performance over all tasks when using observed and unobserved instructions.

(b) Performances of Flan-T5 using observed and unobserved instructions as a function of model size.

Figure 2: Using novel but valid instructions at test time (phrasings unobserved in training) consistently degrades the performance of instruction-tuned LLMs (a). Scale does not necessarily fix this (b).

Model	MMLU		BBL-QA		BBL-BC		BBL-MC		**Overall**	
	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
Flan-T5-3B										
OBSERVED	**48.1**	(±0.3)	**59.0**	(±2.1)	**66.5**	(±3.8)	**55.6**	(±2.7)	**57.3**	(±1.7)
UNOBSERVED	47.5	(±0.9)	56.0	(±7.3)	61.1	(±6.9)	52.1	(±5.4)	54.2	(±5.1)
Performance Δ	↓ 0.6		↓ 3.0		↓ 5.5		↓ 3.5		↓ 3.1	
Alpaca-7B										
OBSERVED	**41.9**	(±0.6)	**48.6**	(±2.8)	**53.8**	(±3.4)	**32.1**	(±2.2)	**44.1**	(±2.3)
UNOBSERVED	39.7	(±2.2)	45.3	(±6.5)	52.4	(±6.5)	16.4	(±3.5)	38.5	(±4.7)
Performance Δ	↓ 2.2		↓ 3.3		↓ 1.4		↓ 15.7		↓ 5.6	
T0++ 11B										
OBSERVED	48.3	(±0.9)	54.1	(±4.1)	**66.1**	(±2.1)	**42.0**	(±2.1)	**52.6**	(±2.3)
UNOBSERVED	**48.5**	(±0.9)	**54.7**	(±3.7)	54.7	(±4.8)	41.4	(±2.4)	49.8	(±2.8)
Performance Δ	↑ 0.2		↑ 0.7		↓ 11.4		↓ 0.6		↓ 2.8	
Flan-T5-11B										
OBSERVED	**53.2**	(±0.2)	**67.9**	(±1.8)	**65.6**	(±6.0)	**58.7**	(±0.5)	**61.4**	(±2.1)
UNOBSERVED	52.7	(±0.8)	64.6	(±8.5)	63.6	(±6.1)	55.9	(±5.5)	59.2	(±5.2)
Performance Δ	↓ 0.5		↓ 3.4		↓ 2.0		↓ 2.8		↓ 2.2	
Alpaca-13B										
OBSERVED	**47.8**	(±0.5)	**53.9**	(±2.2)	**57.9**	(±4.8)	**36.7**	(±1.8)	**49.1**	(±2.3)
UNOBSERVED	47.0	(±0.8)	51.7	(±5.7)	54.1	(±5.6)	22.7	(±7.5)	43.9	(±14.0)
Performance Δ	↓ 0.9		↓ 2.2		↓ 3.8		↓ 14.0		↓ 5.2	

Table 3: Results using observed and unobserved instructions across benchmark tasks (grouped by type). Performance degrades—sometimes by 10+ points—when one uses (UNOBSERVED) instructions, suggesting that instruction-tuned models are not particularly robust. BC, MC, and QA stand for binary classification, multi-class classification, and question answering, respectively.

Figure 3: ***Incorrect*** **but observed instructions perform better on average than *correct* but unobserved instructions**. We report averages over benchmarks, but show example instructions on the right for a specific, illustrative task. We provide all instructions in the Appendix.

4.3 A Closer Look at Instruction Robustness

Above we used general instructions requesting the model to perform tasks (Table 1). Here we delve further into the performance degradation observed when using novel instructions. We report a curious result highlighting the degree to which models rely on having previously observed instructions: Incorrect but observed instructions outperform appropriate but unobserved instructions (Figure 3).

We come to this observation by evaluating the performance of Flan-T5-XXL (11B) using six instruction types over seven datasets from BIG-BENCH. In particular, this includes (variants of) two instructions *observed* in training: **Closest** is the instruction from the most similar task in the instruction-tuning set; **Incorrect** is an observed instruction for a *completely different* and inappropriate task (but which has the same desired output format, e.g. classification)—intuitively these should not yield the desired behavior; **Negated** is the same as **closest**, but we negate the instruction to indicate that it should *not* perform the task.

For *unobserved* instructions, we consider: **Task designer**, the instruction (task prefix) provided by the author of the task in BIG-BENCH, and; **Newly collected**, or the novel instructions collected from NLP graduate students, described above. As a control for reference, we also consider **Nonsensical**, which is a random "instruction" completely irrelevant to any task.

Figure 3 reports average results for these variants. Consistent with our findings, using instructions unobserved in training degrades performance. Strikingly, here we also find that using an *inappropriate but observed* instruction outperforms using *appropriate but unobserved* instructions. This indicates that instruction-tuned models—or at least modestly sized ones we have evaluated here—may in some way overrely on having observed instructions in training, and do not generalize to new instructions and phrasings as we might hope. We provide all the instructions and results in the Appendix.

4.4 Scaling

Does instruction robustness begin to emerge as a function of scale? To attempt to answer this, we repeated all experiments from Table 3 with Flan-T5 model sizes ranging from small (80M parameters) to XXL (11B). We observe in Figure 2b that the disparity between results achieved with observed versus unobserved instructions **does not** seem to decrease with model scale, at least up to this point. That said, massive models (175B+) may offer greater robustness. However, we reiterate that much of the excitement about instruction tuning is the possibility that this technique appears to allow much smaller models to achieve results competitive with massive alternatives.

4.5 Robustness with Semantic Distance

One observation in 4.2 is that performance on MMLU is less affected by using unobserved instructions. MMLU is a benchmark with 57 QA tasks about different knowledge domains; these tasks all share a similar form of input-output (question, four choices → answer). During instruction collection, we treated all tasks in MMLU as a general QA task and asked NLP researchers to write general QA instructions. As a result, we hypothesize that these instructions are comparatively similar to the observed instructions, and this in turn explains the relative robustness in this case.

We empirically verify this in Figure **??** and Table 4. For each instance (instruction plus example), we extract the representation at the penultimate layer for the first decoded token. We use tSNE (**?**) to visualize these representations of observed and unobserved instructions over instances in MMLU and BBL. Figure **??** shows that in the case of MMLU the unobserved instructions we collected are quite similar to the observed, while there is a greater separation between unobserved and observed instructions in BBL. We also provide a numerical measurement of this phenomenon in Table 4. We report the average $\ell 2$ distance between representations of unobserved instructions and those of their nearest observed counterparts. We see that MMLU unobserved instructions are, on average, closer to the nearest observed instruction; this correlates with the lower observed performance drop. These findings are in line with the hypothesis that the unobserved instructions for MMLU are more similar to the observed instructions for this dataset, and this likely explains the apparent robustness in this case.

Figure 4: tSNE plots of representations for the first decoded tokens of 300 randomly sampled examples from MMLU and BBL with Flan-T5 (XXL). Embeddings of observed and unobserved instructions for MMLU are similar, while for BBL they are quite different. This result holds across most but not all models considered: See the **??** for visualizations over all models.

We plot mean performance degradation (as %) as a function of average similarity between the similarity of the first decoded tokens (following *unobserved* instructions) and the same for the *most similar observed* instruction. The negative slope implies the intuitive relationship: Instructions that are dissimilar (in terms of model representations) tend to result in poorer performance. However, the relationship is relatively weak, yielding an intercept estimate of -0.8 and a slope of -0.2 (ρ=0.08).

Dataset	**Avg.** $\Delta\ell 2$	**Avg.** Δ **Ac**
MMLU	**19.8**	**-0.5**
BBL-QA	37.9	-3.4
BBL-BC	25.3	-2.0
BBL-MC	26.1	-2.8

Figure 5: Plots of average degradations in performance versus the semantic distance while using unobserved instructions.

Table 4: Average degradations in performance for four categories. It could be seen that MMLU has minimal average distance, which indicates a smaller distribution shift, and hence leads to the smallest degradation

4.6 Robustness Under In-Context Learning (ICL)

Previous study (**?**) has shown that the LLMs are less sensitive to prompt / instruction variation when few-shot examples are provided in context. While we are focused on zero-shot capabilities, for completeness, we re-ran all experiments in a few-shot setting. We report these results in the **??**. The main finding is that while some discrepancy remains, in general ICL **slightly** decreases the sensitivity of models to the use of unobserved instructions. This is intuitive, given that the examples themselves likely imply the desired task and may affect the distribution.

5 Aligning Equivalent Instructions

We now introduce a simple, lightweight, but effective method to improve the robustness of

Figure 6: The performance degradation when using unobserved instruction at BBL and MMLU with Flan-T5-XXL. We plot the accuracy degradation of all the unobserved instructions compared with the average accuracy of the observed ones. It could be seen that under one-shot in-context learning, the model is slightly more robust as the performance difference converges closer to 0

$$\mathcal{L} = (1-\lambda)\mathcal{L}_{\text{CE}} + \lambda\mathcal{L}_{\text{KL}}$$

$$\mathcal{L}_{\text{KL}} = \frac{1}{N-1}\sum_{i \neq r}^{N}\sum_{j} \text{KL}(\hat{y}_i^{(j)} || \hat{y}_r^{(j)})$$

$$\hat{y}_i^{(j)} = \text{Softmax}(p_i^{(j)}),\, p_i^{(j)} \in \mathbb{R}^{|\mathcal{V}|}$$

Figure 7: Schematic depiction of the proposed instruction alignment method (left) and associated loss terms (right). Dotted (red) lines indicate backpropagation; we update only the soft prompt parameters, which we show yields performance superior to fine-tuning all model parameters.

instruction-tuned LLMs. The intuition is to introduce a term in the objective which explicitly encourages the model to yield similar predictions (and hence similar representations) for the same input when provided distinct but semantically equivalent instructions.

More specifically, we aim to align semantically equivalent instructions in the space induced by the model. To this end we introduce soft embedding parameters with dimensions $\mathbb{R}^{d \times n}$; this is equivalent to adding n novel tokens (with embedding dimension d) as prefixes to inputs (preceding instructions). The intuition is to push the representations for semantically equivalent tasks close together. To this end, we add additional term to the loss: The KL-divergence $\mathcal{L}_{\text{KL}}$ of the output probabilities between a reference instruction for a given task and paraphrased (semantically equivalent) version of the same. We combine this with the standard cross-entropy loss, and fine-tune *only* the introduced soft prompt parameters under this objective (Figure 7). Here λ is a loss-weighting hyper-parameter, $\hat{y}_i^{(j)}$ and $\hat{y}_r^{(j)}$ are the distributions over the vocabulary $\mathcal{V}$ induced by the model with paraphrased instruction i and the reference instruction r at token position j.[2]

Optimizing for the above objective requires paraphrased instructions i for each task in the training data; we generate these automatically as follows. For instruction-tuning dataset, we sample a small amount of training data to use for alignment. We paraphrase these reference instructions using GPT-4. For the Alpaca collection, we randomly sampled 1000 tasks and paraphrased them with three prompts, and collected the top three candidates under temperature 0.5. For the Flan collection, we randomly sampled 986 instances from the mixture with 3 prompts with greedy decoding.

For fine-tuning, we then create instances for each example by pairing them with every distinct instruction available for the corresponding task. We then form batches by including one instance featuring the original instruction and the rest comprising paraphrased instructions. For the implementation of the prefix, we follow the setting of (**?**), which freezes the model parameters and just trains the prefix embeddings with the MLP layers.

6 Results

We experiment with the proposed method using two representative instruction-tuned LLMs: Flan-XL (3B) and Alpaca (7B). We compare the canonical versions of these models trained in the usual way (the same evaluated in Table 3) to variants fine-tuned using our proposed approach. We ablate components of our method to tease out the contributions of data and objectives.

Specifically, we consider variants where we: Fine-tune all model parameters on the additional, automatically generated instruction paraphrases (FT); impose the new KL loss term (again fine-tuning all model parameters; FT+KL); introduce the additional soft prompt parameters and fine-tune on the paraphrase instances, but without KL (PT); and then the full proposed strategy, which introduces the soft prompt parameters and optimizes them for the loss augmented with the KL term (**PT+KL**).

We report results in Table 5. Two observations: (1) The proposed soft prompt alignment strategy (**PT+KL**) yields consistent improvements across the tasks and models considered and especially improves performance on unobserved instructions, as anticipated. (2) The full benefit of the approach is realized only when all components—the additional automatically paraphrased training instruc-

[2] We pad instances such that the lengths in a given batch are effectively equal; the sum is therefore from 1 to the length associated with the current batch, we omit this for simplicity.

	MMLU			BBH		
Model	Obs.	Unobs.	Avg.	Obs.	Unobs.	Avg.
Flan-T5-3B	48.1	47.5	47.8	**56.1**	51.9	54.0
FT	39.4 **(-8.7)**	40.1 **(-7.4)**	39.8 **(-8.0)**	48.2 **(-7.9)**	42.3 **(-9.2)**	45.3 **(-8.7)**
FT+KL	41.8 **(-6.3)**	43.6 **(-3.9)**	45.9 **(-1.9)**	47.7 **(-8.4)**	43.1 **(-8.8)**	45.4 **(-8.6)**
PT	48.1 **(+0.0)**	47.6 **(+0.1)**	47.9 **(+0.1)**	55.9 **(-0.2)**	52.1 **(+0.2)**	54.0 **(+0.0)**
PT+KL	**48.1** **(+0.1)**	**47.9** **(+0.4)**	**48.0** **(+0.2)**	55.9 **(-0.2)**	**53.7** **(+1.8)**	**54.8** **(+0.8)**
Alpaca-7B	41.9	39.7	40.8	47.6	42.9	45.3
FT	40.3 **(-1.6)**	39.1 **(-0.6)**	39.7 **(-1.1)**	44.4 **(-3.2)**	42.1 **(-0.8)**	43.4 **(-2.0)**
FT+KL	39.7 **(-2.2)**	40.2 **(+0.5)**	40.0 **(-0.8)**	45.6 **(-2.0)**	42.8 **(-0.1)**	44.2 **(-1.1)**
PT	42.1 **(+0.2)**	40.0 **(+0.3)**	41.1 **(+0.3)**	47.5 **(-0.1)**	43.0 **(+0.1)**	45.3 **(+0.0)**
PT+KL	**42.4** **(+0.5)**	**41.8** **(+2.1)**	**42.1** **(+1.3)**	**47.9** **(+0.3)**	**46.6** **(+3.7)**	**47.3** **(+2.0)**

Table 5: Results and ablations of the proposed soft prompt alignment method. All ablated versions use the augmented set with automatically paraphrased instructions. FT refers to simply fine-tuning (with teacher-forcing) on this additional data; PT denotes prefix tuning (i.e., introducing soft prompt parameters); KL refers to the alignment objective that we proposed above. Using all of these components together yields the best performance, especially on unobserved instructions.

tions, soft prompt parameters, and additional KL loss term—are in place.

Following our approach in 4.5, we take the average distance between observed and unobserved instructions before and after alignment. Table 6 shows that our method brings observed and unobserved instruction representations closer together. The similarity is most increased in the case of the biggest accuracy gain, further suggesting the mechanism of improvement provided by soft prompt alignment.

7 Conclusions

Instruction-tuned LLMs have emerged as a promising means of achieving zero-shot performance with smaller models that is competitive to, and sometimes even better than, that observed using much larger LLMs (**??**). In this work we empirically characterized the *robustness* of such models with respect to instruction rephrasings. In particular, we collected manually composed instructions from 36 graduate students in NLP across 75 tasks, and we evaluated different families of instruction-tuned LLMs (Flan, Alpaca, and T0) when provided observed and unobserved instructions (seen in training and not, respectively). We found that using the latter consistently degrades model performance, indicating that models are unduly sensitive to instruction phrasings.

We then proposed a simple mechanism intended to improve the robustness of instruction-tuned LLMs. This approach entails introducing an additional loss term that penalizes the model for inducing dissimilar distributions over output tokens when using (a) paraphrased instructions as opposed to (b) a reference instruction for the same task. We found that training under this objective consistently (though modestly) improves results, and in particular mitigates the degradation observed when previously unobserved instructions are used.

8 Limitations

This work has important limitations: For example we only evaluated "mid-sized" models (<20B parameters), it is unclear if our findings would generalize to much larger instruction-tuned models. (However, we note that instruction tuning has been most promising for smaller models.) We also restricted our evaluation to three task types: QA and multi-class and binary classification.

Ethics This work does not have an explicit ethical dimension, but we acknowledge that all LLMs are likely to encode problematic biases; it is unclear how instruction-tuning might interact with these.

9 Acknowledgments

This work was supported by the National Science Foundation (NSF) grant 1901117.

We thank Jay DeYoung and Alberto Mario Ceballos Arroyo for their advice and feedback on the paper. We also thank Alberto Mario Ceballos Arroyo, Arnab Sen Sharma, Bowen Zhao, Eric Todd, Hanming Li, Hiba Ahsan, Hye Sun Yun, Shulin Cao, Jay DeYoung, Jered McInerney, Ji Qi, Jifan Yu, Jize Jiang, Kaisheng Zeng, Koyena Pal, Kundan Krishna, Linxiao Nie, Hailong Jin, Jinxin Matthew Liu, Millicent Li, Monica Munnangi, Nikhil Prakash, Pouya Pezeshpour, Sanjana Ramprasad, Sarthak Jain, Shangqing Tu, Somin Wadhwa, Tingjian Zhang, Hao Wesley Peng, Xiaozhi Wang, Xingyu Lu, Xin Lv, Zijun Yao for providing manually written instructions.

10 Preamble

Set the title and author using `\title` and `\author`. Within the author list, format multiple authors using `\and` and `\And` and `\AND`; please see the LaTeX source for examples.

By default, the box containing the title and author names is set to the minimum of 5 cm. If you need more space, include the following in the preamble:

Dataset	Closest Distance Before	Closest Distance After	Δ Acc. Improvement (%)
MMLU	22.2	21.3	+ 0.3%
BBL QA	22.4	23.0	+ 0.4%
BBL BC	30.1	**27.9**	**+ 4.2%**
BBL MC	26.0	24.6	+ 0.3%

Table 6: Average distances before and after soft prompt alignment with Flan-T5-XL.

Command	Output	Command	Output
{\"a}	ä	{\c c}	ç
{\^e}	ê	{\u g}	ğ
{\`i}	ì	{\l}	ł
{\.I}	İ	{\~n}	ñ
{\o}	ø	{\H o}	ő
{\'u}	ú	{\v r}	ř
{\aa}	å	{\ss}	ß

Table 7: Example commands for accented characters, to be used in, *e.g.*, BibTEX entries.

```
\setlength\titlebox{<dim>}
```

where <dim> is replaced with a length. Do not set this length smaller than 5 cm.

11 Document Body

11.1 Footnotes

Footnotes are inserted with the \footnote command.[3]

11.2 Tables and figures

See Table 7 for an example of a table and its caption. **Do not override the default caption sizes.**

11.3 Hyperlinks

Users of older versions of LATEX may encounter the following error during compilation:

```
\pdfendlink ended up in
different nesting level
than \pdfstartlink.
```

This happens when pdfLATEX is used and a citation splits across a page boundary. The best way to fix this is to upgrade LATEX to 2018-12-01 or later.

11.4 Citations

Table 8 shows the syntax supported by the style files. We encourage you to use the natbib styles. You can use the command \citet (cite in text) to get "author (year)" citations, like this citation to a paper by Gusfield (1997). You can use the command \citep (cite in parentheses) to get "(author, year)" citations (Gusfield, 1997). You can use the command \citealp (alternative cite without parentheses) to get "author, year" citations, which is useful for using citations within parentheses (e.g. Gusfield, 1997).

11.5 References

The LATEX and BibTEX style files provided roughly follow the American Psychological Association format. If your own bib file is named custom.bib, then placing the following before any appendices in your LATEX file will generate the references section for you:

```
\bibliography{custom}
```

You can obtain the complete ACL Anthology as a BibTEX file from https://aclweb.org/anthology/anthology.bib.gz. To include both the Anthology and your own .bib file, use the following instead of the above.

```
\bibliography{anthology,custom}
```

Please see Section 12 for information on preparing BibTEX files.

11.6 Appendices

Use \appendix before any appendix section to switch the section numbering over to letters. See Appendix A for an example.

12 BibTEX Files

Unicode cannot be used in BibTEX entries, and some ways of typing special characters can disrupt BibTEX's alphabetization. The recommended way of typing special characters is shown in Table 7.

Please ensure that BibTEX records contain DOIs or URLs when possible, and for all the ACL materials that you reference. Use the doi field for DOIs and the url field for URLs. If a BibTEX entry has a URL or DOI field, the paper title in the references section will appear as a hyperlink to the paper, using the hyperref LATEX package.

[3] This is a footnote.

Output	natbib command	Old ACL-style command
(Gusfield, 1997)	`\citep`	`\cite`
Gusfield, 1997	`\citealp`	no equivalent
Gusfield (1997)	`\citet`	`\newcite`
(1997)	`\citeyearpar`	`\shortcite`

Table 8: Citation commands supported by the style file. The style is based on the natbib package and supports all natbib citation commands. It also supports commands defined in previous ACL style files for compatibility.

Acknowledgements

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

A Example Appendix

This is an appendix.