Ensemble-Instruct: Generating Instruction-Tuning Data with a Heterogeneous Mixture of LMs

Young-Suk Lee, Md Arafat Sultan, Yousef El-Kurdi, Tahira Naseem
Asim Munawar, Radu Florian, Salim Roukos, Ramón Fernandez Astudillo
{ysuklee,yousefelk,tnaseem,raduf,roukos}@us.ibm.com
{arafat.sultan,asim,ramon.astudillo}@ibm.com
IBM Research AI

Abstract
Using in-context learning (ICL) for data generation, techniques such as Self-Instruct (Wang et al., 2023) or the follow-up Alpaca (Taori et al., 2023) can train strong conversational agents with only a small amount of human supervision. One limitation of these approaches is that they resort to very large language models (around 175B parameters) that are also proprietary and non-public. Here we explore the application of such techniques to language models that are much smaller (around 10B–40B parameters) and have permissive licenses. We find the Self-Instruct approach to be less effective at these sizes and propose new ICL methods that draw on two main ideas: (a) Categorization and simplification of the ICL templates to make prompt learning easier for the LM, and (b) Ensembling over multiple LM outputs to help select high-quality synthetic examples. Our algorithm leverages the 175 Self-Instruct seed tasks and employs separate pipelines for instructions that require an input and instructions that do not. Empirical investigations with different LMs show that: (1) Our proposed method yields higher-quality instruction-tuning data than Self-Instruct, (2) It improves performances of both vanilla and instruction-tuned LMs by significant margins, and (3) Smaller instruction-tuned LMs generate more useful outputs than their larger un-tuned counterparts. Our codebase is available at https://github.com/IBM/ensemble-instruct.

1 Introduction
Instruction-tuned language models have demonstrated strong zero-shot generalization capabilities to new tasks (Chung et al., 2022a; Wei et al., 2021; Ouyang et al., 2022; Mishra et al., 2022; Wang et al., 2022; Longpre et al., 2023), creating interest in large-scale automatic synthesis of instruction-tuning data (Honovich et al., 2022; Wang et al., 2023; Xu et al., 2032; Sun et al., 2023a; Xu et al., 2023). In this context, Self-Instruct (Wang et al., 2023) showed that a small number of expert-annotated seed examples, coupled with in-context learning (ICL) with a base model, can be used to generate an instruction-tuning dataset to efficiently instruct that same base model. While this method yielded strong results and multiple follow-up works, most techniques resort to very large LMs (around 175B parameters) (Wang et al., 2023; Taori et al., 2023), available only through closed-access APIs, or have restricted model access.

In this paper, we present Ensemble-Instruct, a novel algorithm enabling high-quality instruction-tuning data generation with smaller LMs (40B parameters or less), that are also fully accessible and have permissive usage licenses. We show that, when using smaller models as generators, Self-Instruct struggles to produce text of adequate quality, adversely affecting the utility of the generated data and downstream model performance. Staying within the ICL framework and using the Self-Instruct seed tasks, Ensemble-Instruct explores two main ideas to solve this problem: (1) Categorizing and simplifying the ICL prompts to ease the few-shot learning process, and (2) Ensembling over multiple LM outputs to improve both accuracy and diversity of the generated data.

A standard instruction-tuning sample exemplifies a task comprising: (a) an instruction that describes the action to be performed, (b) an optional input on which the action is performed, and (c) the output of the action. Similar to Self-Instruct, we generate samples in two stages: instruction generation and instance generation, where an instance comprises an input (optional) and an output. Unlike Self-Instruct, Ensemble-Instruct seeks to simplify the problem for the generating LM by first categorizing the examples into two types—those with an input and those without—and then employing separate pipelines for the two that leverage their own unique and simplified prompts (§2.1). Further, it ensembles over the outputs of different LMs in
two complementary ways: (1) including examples generated by a heterogeneous collection of LMs in the final set to increase diversity, and (2) majority voting followed by filtering low-consensus examples to improve accuracy (§2.4).

To understand the effects of our proposed methods, we run an extensive evaluation of different models for instruction generation. This includes vanilla language models (T5) \texttt{UL} \texttt{2-20} (Tay et al., 2022), \texttt{FALCON-40} (Penedo et al., 2023), the instruction-tuned models \texttt{FLAN-T5-11} (Chung et al., 2022b) and \texttt{FLAN-UL2-20} (Tay et al., 2022) and the chat-tuned\footnote{https://huggingface.co/togethercomputer/GPT-NeoX-Chat-Base-20B} version of GPT-NeoX-20B (Black et al., 2022). As base models to fine-tune with our generated data, we use the vanilla LM Pythia-1.4B (Biderman et al., 2023) for ablation analysis, MPT-7B\footnote{https://www.mosaicml.com/blog/mpt-7b}, a decoder only LM similar to LLaMA (Touvron et al., 2023) as well as GPT-JT-6B\footnote{https://huggingface.co/togethercomputer/GPT-JT-6B-v1}, an instructed version of GPT-J (Wang and Komatsuzaki, 2021) trained on Chain of Thought and Natural instruction datasets among others. All chosen models are open-source and have permissive licenses (Apache-2).

We evaluate the models fine-tuned on the data generated by Ensemble-Instruct on the the SuperNatural Instructions (SuperNI) test set (Wang et al., 2022) and 252 user-oriented tasks from Wang et al. (2023). Our contributions can be summarized as follows:

- We propose a technique for generating high-quality instruction-tuning data with 40B-parameter or smaller LMs that are openly accessible, with non-restrictive licenses.

- We outperform Self-Instruct training of GPT3 (175B) with a far smaller base model (MPT-7B). The technique also improves the performance of instruction-tuned GPT-JT-6B.

- Ablation studies demonstrate the importance of the individual components of our technique.

- We release the synthetic instruction-tuning dataset of about 45k samples along with our ICL templates and codebase.

\section{2 Ensemble-Instruct}

\begin{algorithm}[h]
\begin{algorithmic}[1]
\State $o_{\text{best}} \leftarrow \text{None}$
\State $Rs \leftarrow \emptyset$
\For{$(i,j)$ in \{1, 2, (1, 3), (2, 3)\}}
\State $Rs \leftarrow Rs \cup \text{RougeL}(o_i, o_j)$
\EndFor
\If{$\min(Rs) > t$}
\State $i, j \leftarrow \text{argmax}(Rs)$
\State $o_{\text{best}} \leftarrow o_i$
\EndIf
\State \Return $o_{\text{best}}$
\end{algorithmic}
\caption{Output Ensembling}
\end{algorithm}

A high-level overview of Ensemble-Instruct is given in Figure 1. The algorithm has three main components: (i) Categorization of tasks and their associated prompts, (ii) Generation of instructions followed by instances, where an instance comprises an input (optional) and an output, and (iii) Ensemble of outputs from multiple LMs.

\subsection{2.1 Categorization of Tasks and Prompts}
We divide the tasks, i.e. the instruction-tuning samples, into two categories: those where the instruction needs an input to be meaningful (type A) and those where it does not (type B). Examples of tasks from these two types can be seen in Figures 1 and 2. Among the seed tasks of Wang et al. (2023), 125 belong to type A and 50 to type B. For each category, we employ a dedicated pipeline that (a) uses ICL demonstrations only of that type, and (b) tailors the number of demonstrations to the difficulty of the type, at different stages of generation.

\subsection{2.2 Instruction Generation}
For type A tasks, we use 24 ICL demonstrations during instruction generation. Out of those, 20 are randomly sampled from the 125 seed tasks of the same type, and 4 are sampled from instructions previously generated by the model itself. For type B tasks, we use 10 ICL demonstrations, of which 8 are sampled from the 50 type B seed tasks and 2 from previously generated synthetic instructions. Further, we adopt the approach of Wang et al. (2023) of adding a new instruction to the set only if its Rouge-L (Lin, 2004) score with every existing instruction is less than 0.7.
2.3 Instance Generation

During instance generation, we use 18 ICL demonstrations for type A tasks and 15 for type B tasks, randomly selected from the seed tasks. Figure 2 shows examples of type A and type B tasks, and the prompts used for instance generation.

2.4 Output Ensembling

The instruction and instance generation steps should in principle complete the process of synthesizing an instruction-tuning sample (Wang et al., 2023). However, samples generated by small LMs can be inaccurate, which prompts us to design a final step of output ensembling. Instead of simply accepting the already generated example, we use an additional set of LMs to predict new outputs, given either the generated instruction-input pair (type A) or the instruction (type B).

The final output is derived by applying the greedy consensus Algorithm 1 to the outputs generated by the different LMs. The algorithm computes the Rouge-L score between all three pairs of outputs. If the lowest Rouge-L is above a threshold $t$, it returns the first element of the pair with the highest Rouge-L score. This can be seen as a greedy version of Minimum Bayesian Risk decoding (Goel and Byrne, 2000) with additional thresholding. The minimum threshold $t$ is set to 0.01 across all tasks. It is important to note that if the above process does not select any of the three outputs, the example is filtered out.

3 Analysis of Instruction Tuning Dataset

We generate multiple instruction-tuning datasets using a heterogeneous set of LMs. Table 1 shows the labels of our synthetic datasets according to the LMs used in different stages of generation. Table 2 summarizes the set of LMs we use for generation.
### 3.1 Instance vs. Output Generation

As shown in Table 1, we use a distinct set of LMs for instruction and instance generation on one hand and output generation for ensembling on the other. The motivations are two-fold: (1) We observed that only relatively large decoder only models with 20B parameters or more are capable of generating input-output instances (type A). Therefore, we use decoder only models including FALCON, GPT-NEOXT-CHAT for input-output instance generation. (2) Instruction-tuned models are capable of generating high quality zero-shot outputs. Therefore, we use instruction-tuned models including FLAN-UL2, FLAN-T5-XXL, GPT-NEOXT-CHAT for additional output generation for ensembling. We found that vanilla LMs UL2, FALCON lag behind instruction-tuned models for output generation, as shown in EO-FALCON-LM of Table 4.

Table 3 reports the number of valid instance generations, as well as samples accepted by the ensemble Algorithm 1, using FLAN-UL2 and FLAN-T5-XXL as additional outputs. We show results for 100 random samples using different models (FALCON, FLAN-UL2, GPT-NEOXT-CHAT) to generate instruction and type A instances using the same prompt and examples. Instructed models struggle to generate valid instances and in particular FLAN-UL2 generates no valid instance for the 100 samples. Although not shown in the table, most LMs are capable of generating type B instructions and instances, indicating that instructions and instances that do not require an input is an easier task than those requiring an input.

### 3.2 Small LM Dataset Comparison

We instruction-tune Pythia-1.4B-deduped with different datasets and evaluate them on the 119 tasks of the SUPERNI test set. For validation, we use 10,589 samples from 106 SUPERNI training tasks. Note that the validation and test sets have zero task overlap. We instruction-tune the model for 5 to 7 epochs and select the checkpoint with the highest validation Rouge-L score for evaluation. Performances of these tuned models on the test set are shown in Table 4, where M-SELF-INST denotes the algorithm and ICL templates of Wang et al. (2023) applied to {UL2, NEOX}, and F-SELF-INST, the algorithm and ICL templates of Wang et al. (2023) applied to FALCON. We also show the performance of PYTHIA-1.4B-DEDUPED fine-tuned with two ex-

---

<table>
<thead>
<tr>
<th>Model</th>
<th># params</th>
<th>LM type</th>
<th>Rouge-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALCON</td>
<td>40B</td>
<td>causal</td>
<td>12.7</td>
</tr>
<tr>
<td>UL2</td>
<td>20B</td>
<td>seq2seq</td>
<td>10.4</td>
</tr>
<tr>
<td>GPT-NEOXT-CHAT</td>
<td>20B</td>
<td>causal</td>
<td>6.6</td>
</tr>
<tr>
<td>FLAN-UL2</td>
<td>20B</td>
<td>seq2seq</td>
<td>77.5</td>
</tr>
<tr>
<td>FLAN-T5-XXL</td>
<td>11B</td>
<td>seq2seq</td>
<td>73.0</td>
</tr>
</tbody>
</table>

Table 2: LMs used for instruction-tuning data generation. seq2seq denotes sequence-to-sequence and causal denotes decoder-only. GPT-NEOXT-CHAT is tuned on the OIG dataset\(^4\). FLAN-UL2 and FLAN-T5-XXL are tuned on FLAN collections. Both OIG and FLAN include SUPERNI data. Instruction-tuned models are denoted by \(^\dagger\). Zero-shot performance of each model on the SUPERNI test set is provided in Rouge-L.

---


---

Table 1: Labels of our synthetic tuning datasets according to the LMs used for generating instructions, instances and additional outputs for ensembling. Datasets with outputs from a single LM and an ensemble of LMs are prefixed with -SO and -EO, respectively. The rest of each label specifies the models that were used at different stages of the process. If additional outputs were generated using instruction-tuned LMs for ensembling, the dataset is suffixed with -LM. If vanilla LMs were used for the same purpose, we use the suffix -LM. With instruction-tuned LMs, we generate the output zero-shot; for vanilla LMs, we use few-shot ICL.

<table>
<thead>
<tr>
<th>Label</th>
<th>Instruction</th>
<th>Instances</th>
<th>Additional Outputs for Ensembling</th>
</tr>
</thead>
<tbody>
<tr>
<td>SO-FALCON</td>
<td>FALCON</td>
<td>FALCON</td>
<td>–</td>
</tr>
<tr>
<td>SO-{UL2, NEOX}</td>
<td>UL2, GPT-NEOXT-CHAT</td>
<td>UL2, GPT-NEOXT-CHAT</td>
<td>–</td>
</tr>
<tr>
<td>EO-FALCON-LM</td>
<td>FALCON</td>
<td>FALCON</td>
<td>–</td>
</tr>
<tr>
<td>EO-FALCON-ILM</td>
<td>FALCON</td>
<td>FALCON</td>
<td>–</td>
</tr>
<tr>
<td>EO-{UL2, NEOX}-ILM</td>
<td>UL2, GPT-NEOXT-CHAT</td>
<td>UL2, GPT-NEOXT-CHAT</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 3: Number of valid type A instructions and instances generated by different models for 100 samples as well and number (and percentage) of samples filtered by Algorithm 1. All models share the same prompt and examples.
Table 4: Efficacy of synthetic instruction tuning datasets measured by the performance of PYTHIA-1.4B-DEDUPED tuned models on the SUPERNI test set. Dataset labels are described in Table 1. ALPACA and SELF-INST are external synthetic datasets for further comparisons. M-SELF-INST denotes the algorithm and ICL templates of Wang et al. (2023) applied to \{UL2, NEOX\}. F-SELF-INST denotes the algorithm and ICL templates of Wang et al. (2023) applied to FALCON. All training sets include the 175 seed tasks and the learning rate is 1e-5.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># samples</th>
<th>Rouge-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZERO-SHOT BASELINE</td>
<td>0</td>
<td>9.8</td>
</tr>
<tr>
<td>ALPACA</td>
<td>51,760</td>
<td>33.4</td>
</tr>
<tr>
<td>SELF-INST</td>
<td>82,612</td>
<td>34.4</td>
</tr>
<tr>
<td>M-SELF-INST</td>
<td>24,984</td>
<td>28.5</td>
</tr>
<tr>
<td>SO-{UL2, NEOX}</td>
<td>25,660</td>
<td>33.6</td>
</tr>
<tr>
<td>EO-{UL2, NEOX}-ILM</td>
<td>18,218</td>
<td>38.3</td>
</tr>
<tr>
<td>F-SELF-INST</td>
<td>38,624</td>
<td>25.6</td>
</tr>
<tr>
<td>SO-FALCON</td>
<td>30,537</td>
<td>34.4</td>
</tr>
<tr>
<td>EO-FALCON-LM</td>
<td>26,503</td>
<td>32.9</td>
</tr>
<tr>
<td>EO-FALCON-ILM</td>
<td>26,701</td>
<td>37.1</td>
</tr>
</tbody>
</table>

Table 5: Manual evaluation of synthetic instruction tuning data quality on 140 randomly selected samples.

<table>
<thead>
<tr>
<th>Instance Type</th>
<th>criteria</th>
<th>output</th>
<th>input-output</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOOD</td>
<td>77</td>
<td>22</td>
<td></td>
<td>99 (70.7%)</td>
</tr>
<tr>
<td>BAD</td>
<td>14</td>
<td>15</td>
<td></td>
<td>29 (20.7%)</td>
</tr>
<tr>
<td>MAYBE</td>
<td>9</td>
<td>3</td>
<td></td>
<td>12 (8.6%)</td>
</tr>
<tr>
<td>total</td>
<td>100</td>
<td>40</td>
<td></td>
<td>140</td>
</tr>
</tbody>
</table>

Manual evaluation results are shown in Table 5, which was carried out by one of the authors. We find that examples containing only an instruction and an output (type B) are generally of higher quality (77% GOOD) than those also containing an input (type A) (55% GOOD). This difference in quality is reflective of the relative difficulty of generating them by smaller models, i.e. it is easier to generate output-only instances, as suggested in §3.1. Out of the 24,809 M-SELF-INST examples in Table 4 (after excluding the 175 seed tasks), 20,752 (83.6%) are of type B, further demonstrating that it is easier to generate output-only instances. Ensemble-Instruct pipeline avoids such unbalanced generation by first categorizing the tasks and then leveraging separate sets of simplified prompts for each. Each of our data sets generated with Ensemble-Instruct is an almost even split between instructions with and without an input.

Figure 3 shows some synthetic examples before and after output ensembling, depicting a few different ways in which ensembling improves the quality of the generated output. Regarding the effect of ensembling, observations show that it is particularly effective in selecting accurate output when it is short, e.g. classification tasks, via exact match. For longer outputs from generation tasks, e.g. summarization, the algorithm often filters out non-sensical outputs with hallucinations.
**Instance Generation with Both an Input and an Output:**

Generate examples for the following instructions. The instruction requires input and output instances. And you have to generate both input and output.

**Instruction:** Extract all the country names in the paragraph, list them separated by commas.
**Input:** Dr. No is the sixth novel by the English author Ian Fleming to feature his British Secret Service agent James Bond. Written at Fleming’s Goldeneye estate in Jamaica, it was first published in the United Kingdom by Jonathan Cape in 1958. In the novel Bond looks into the disappearance in Jamaica of two fellow MI6 operatives who had been investigating Doctor No. Bond travels to No’s Caribbean island and meets Honeychile Rider, who is there to collect shells. They are captured and taken to a luxurious facility carved into a mountain. The character of Doctor No, the son of a German missionary and a Chinese woman, was influenced by Sax Rohmer’s Fu Manchu stories. Dr. No was the first of Fleming’s novels to face widespread negative reviews in Britain, but it was received more favourably in the United States.

**Output:** English, British, Jamaica, the United Kingdom, German, Chinese, Britain, the United States. [EoS]

**Instruction:** Sort the given input ascendingly.
**Input:** [10, 92, 2, 5, -4, 92, 5, 101]
**Output:** [-4, 2, 5, 5, 10, 92, 92, 101] [EoS]

**Instruction:** In this task, you need to compare the meaning of the two sentences and tell if they are the same. Output yes or no.
**Input:** Sentence 1: The teacher is speaking to the class. Sentence 2: The teacher is speaking to the students.
**Output:** yes [EoS]

**Instruction:** Which exercises are best for reducing belly fat at home?
**Output:** Lying Leg Raises, Leg In And Out, Plank, Side Plank, Sit-ups [EoS]

**Instruction:** Is working part-time a good idea for teenagers between 15 and 19 years old?
**Output:** There are both pros and cons to having a part-time job as a teenager. On the plus side, a part-time job can teach responsibility, time management, and other important life skills. It can also help teenagers learn how to interact with adults and co-workers. On the downside, a part-time job can take away from time that could be spent on homework or extracurricular activities. It can also be difficult to find a job that works around a school schedule. [EoS]

**Instruction:**

**Instance Generation with Only an Output:**

Generate examples for the instructions. The instruction does not require input and generate the output directly.

**Instruction:** Which exercises are best for reducing belly fat at home?
**Output:** Lying Leg Raises, Leg In And Out, Plank, Side Plank, Sit-ups [EoS]

**Instruction:** Is working part-time a good idea for teenagers between 15 and 19 years old?
**Output:** There are both pros and cons to having a part-time job as a teenager. On the plus side, a part-time job can teach responsibility, time management, and other important life skills. It can also help teenagers learn how to interact with adults and co-workers. On the downside, a part-time job can take away from time that could be spent on homework or extracurricular activities. It can also be difficult to find a job that works around a school schedule. [EoS]

Figure 2: Example ICL templates for instance generation. The top 3 examples are for instances with both an input and an output; the bottom 3 are for output-only instances. We append the special token [EoS] to the end of each example, clearly demarcating example boundaries, to make sure generation stops after the final output token.

### 4 Experimental Results

We carry out automatic evaluations using Rouge-L on the evaluation sets in Table 6. Both 119 SUPERNI tasks and 252 user-oriented tasks are identical to those evaluated in Wang et al. (2023).\(^8\)

<table>
<thead>
<tr>
<th>Evaluation Data</th>
<th># tasks</th>
<th># samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUPERNI</td>
<td>119</td>
<td>11,810</td>
</tr>
<tr>
<td>User-Oriented</td>
<td>252</td>
<td>252</td>
</tr>
</tbody>
</table>

Table 6: Evaluation datasets for automatic evaluations using Rouge-L. None of the tasks in the evaluation are seen during training.

\(^8\)We refer the reader to §4.4 of Wang et al. (2023) on the characteristics of the 252 user-oriented test set and §A.1 on
We set aside 106 tasks (10,589 samples) from the SuperNI 756 training tasks as the validation data set. For SuperNI instruction tuning, we exclude the validation set from training to simulate evaluation on unseen tasks.

We fine-tune 2 base LMs on the instruction tuning data generated by the current technique: (1) a vanilla LM, MPT-7B, and (2) an instruction tuned LM, GPT-JT-6B. To fine-tune these models, we adopt QLoRA (Dettmers et al., 2023), which enables us to train both LMs with a single A100 GPU (40GB memory) within 24 hours. We also carried out full fine-tuning of MPT-7B for 2 data sets, EO-\{UL2,NEOX\}−ILM and SUPERNI with 2 A100 GPUs (80GB memory). The results are shown in Tables 7 and 8 for the SUPERNI test set, and in Table 9 for the 252 user-oriented test set.

In Table 7, MPT-7B fine-tuned on our synthetic data generated from vanilla LMs (SD I) outperforms both T0 and GPT3_{SELF-INST} despite the fact that the latter are fine-tuned on over 80K samples whereas MPT-7B is fine-tuned only on around 30K samples. MPT-7B fine-tuned on our synthetic data generated from instruction-tuned models (SD II) outperform the data generated using vanilla LMs (SD I) by up to 3 points. Full fine-tuning outperforms QLoRA fine-tuning by 1.4 on EO-\{UL2,NEOX\}−ILM (46.8 vs. 45.4). Full fine-tuning again outperforms QLoRA fine-tuning by 2.2 on SuperNI training (50.4 vs. 48.2). MPT-7B fine-tuned on the combination of two synthetic data sets EO-\{UL2,NEOX ∪ FALCON\}−ILM and the SuperNI training set improves the Rouge-L score over SuperNI training only by 2.2 points (from 48.2 to 50.4). We see a similar pattern in Table 8 for the instruction-tuned base LM GPT-JT-6B. The fact that our synthetically generated data significantly improve the performance of the instruction-tuned LM suggests that our technique generates data sufficiently different from the instruction tuning data incorporated into the base LM training.

In Table 9, we note that both base models, MPT-7B and GPT-JT-6B, perform worse on the user-oriented data set than on the SuperNI test set: 10.6 vs. 16.6 with MPT-7B and 6.2 vs. 10.4 with GPT-JT-6B. Fine-tuning these models on about 45K samples of the synthetic data provides a significant

---

input: 1, 2, 23, 50, 1, 2, 23, 50, 1, 6, 22
output-before: 1, 2, 23, 50, 1, 2, 23, 23
output-after: 50

input: The feature of our product is that it has a rounded body shape, which is convenient to carry around.
output-before: Our product has a round design, making it lightweight and portable.
output-after: The product has a rounded body shape.

input: When you have a happy life and you are happy, you are happy for yourself. When you’re happy for yourself, you are happy for everybody.
output-before: I felt really happy when I won the game
output-after: yes
output-before: no
input: a red rubber ball
output-after: The product has a rounded body shape.
output-before: Our product has a round design, making it lightweight and portable.
input: The feature of our product is that it has a rounded body shape, which is convenient to carry around.
output-after: oranges

Figure 3: Instruction tuning dataset examples before and after output ensembling. Ensembling generally improves different aspects of output quality, including correctness and adherence to the specifics of the question. We observe a side effect of shorter outputs being preferred over longer ones in generation tasks even if in some cases that makes the output less accurate, as shown in the last example.
Table 7: Evaluation results on the SuperNI test set. SD I denotes synthetic data generated from only vanilla LMs, and SD II, synthetic data generated from the combination of vanilla and instruction-tuned LMs. Superscript\textsuperscript{ff} denotes full fine-tuning. Superscript\textsuperscript{qlora}, QLoRA fine-tuning. Learning rate is set to 1e-6 for full fine-tuning and 5e-5 for QLoRA tuning. EO-COMBO-ILM denotes EO-\{UL2,NEOX,FALCON\}. Combination of synthetic data EO-COMBO-ILM and SuperNI training set improves over SuperNI training set by 2.2 points, from 48.2 to 50.4. Instruction tuning with SD II output-performs instruction tuning with SD I. For instruction tuning with SuperNI, we subsample 100 instances from each of the 650 training tasks.

Table 8: Results of (instruction-tuned base LM) GPT-JT-6B fine-tuned on synthetic data. EO-COMBO-ILM denotes EO-\{UL2,NEOX,FALCON\}. All models are fine-tuned with QLoRA with learning rate 5e-5. boost to the Rouge-L scores, from 10.6 to 22.1 for MPT-7B, and from 6.2 to 21.5 for GPT-JT-6B. This suggests that the synthetic data we generate capture the characteristics of user-oriented instructions to a certain degree. Consistent with the results noted in Table 4 for the SuperNI test set, the data generated by our technique is more effective than the data generated using Self-Instruct (M-SELF-INST, F-SELF-INST) on the user oriented data set as well.

In Table 10, we show experimental results with other much larger models to illustrate the scalability of the proposed Ensemble-Instruct to any black-box models. Regardless of the base model sizes, ranging from 6B to 40B, fine-tuning the base model with the synthetic data EO-\{UL2,NEOX,FALCON\}-ILM improves the Rouge-L score significantly. The fine-tuned model performances seem to correlate well with the base model’s parameter sizes, i.e. 43.1 for the smallest GPT-JT-6B, 49.9 for the largest FALCON-40B and all other model sizes and scores in between. In particular, the experimental results on FALCON-40B indicates that Ensemble-Instruct is not an instance of model distillation in the sense that the synthetic data generated from FALCON-40B and smaller models signifi-
Table 10: Fine-tuning results on large models demonstrating the scalability of the Ensemble-Instruct technique to any black-box models. Zero-shot and fine-tuned model scores are Rouge-L on SUPERNI test set. Performance improvement of FALCON-40B after fine-tuning, compared with its zero-shot performance indicates that Ensemble-Instruct is not an instance of model distillation. All models are fine-tuned with EO-\{UL2, NEOX ∪ FALCON\}-ILM in Table 7.

<table>
<thead>
<tr>
<th>Model-ParamSize</th>
<th>zero-shot</th>
<th>fine-tuned</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-JT-6B</td>
<td>10.4</td>
<td>43.1</td>
</tr>
<tr>
<td>MPT-7B</td>
<td>16.6</td>
<td>46.4</td>
</tr>
<tr>
<td>OPEN-LLAMA-13B</td>
<td>11.9</td>
<td>46.7</td>
</tr>
<tr>
<td>MPT-30B</td>
<td>12.2</td>
<td>49.5</td>
</tr>
<tr>
<td>FALCON-40B</td>
<td>12.7</td>
<td>49.9</td>
</tr>
</tbody>
</table>

Table 10: Fine-tuning results on large models demonstrating the scalability of the Ensemble-Instruct technique to any black-box models. Zero-shot and fine-tuned model scores are Rouge-L on SUPERNI test set. Performance improvement of FALCON-40B after fine-tuning, compared with its zero-shot performance indicates that Ensemble-Instruct is not an instance of model distillation. All models are fine-tuned with EO-\{UL2, NEOX ∪ FALCON\}-ILM in Table 7.

Table 10: Fine-tuning results on large models demonstrating the scalability of the Ensemble-Instruct technique to any black-box models. Zero-shot and fine-tuned model scores are Rouge-L on SUPERNI test set. Performance improvement of FALCON-40B after fine-tuning, compared with its zero-shot performance indicates that Ensemble-Instruct is not an instance of model distillation. All models are fine-tuned with EO-\{UL2, NEOX ∪ FALCON\}-ILM in Table 7.

The use of a reference metric, such as Rouge-L, to ensemble the outputs of multiple language distributions is a common technique in Minimum Bayesian Risk decoding, with applications to speech-to-text (Goel and Byrne, 2000), machine translation (Kumar and Byrne, 2004), language modeling (Suzgun et al., 2022) and parsing (Lee et al., 2022), among others. Here we use a similar technique in the context of instruction generation. To the best of our knowledge, this is the first application of such an approach to instruction-tuning data generation.

Jiang et al. (2023) proposes LLM-Blender, an ensembling framework to improve the generation qualities by leveraging the diverse strengths of multiple language models. While we utilize the output ensemble in the context of synthetic data generation with Rouge-L as the reference metric, LLM-Blender focuses on improving model output qualities using PairRanker and GenFuser, both approaches capitalize on the efficacy of ensembling as a way of improving output qualities.

Also related to this work are approaches directly distilling from ChatGPT or GPT-4 (OpenAI, 2023) without specific instruction strategies, such as Vicuna\(^\text{10}\), which distills ChatGPT, Baize (Xu et al., 2023), distilling conversations and Orca (Mukherjee et al., 2023), which uses a large amount of ChatGPT and GPT-4 outputs and combines FLAN tasks, system prompts and machine-generated explanations sampled from these models. The strength of these approaches seems to rely more on the amount and quality of teacher samples available than on the inductive biases of the self-instructing technique and still rely on proprietary models with non-permissive licenses.

6 Conclusion

We present a novel technique to generate instruction-tuning data through ICL, following the recent Self-Instruct work (Wang et al., 2023). Unlike Self-Instruct, we propose techniques that explicitly avoid the use of proprietary language models like GTP-3, ChatGPT or GPT-4. We show that when using smaller models, Self-Instruct becomes less performant. To overcome this, we draw on two main ideas: (a) Categorization and simplification of ICL templates to make prompt learning easier, and (b) Ensembling over multiple LM outputs to select high-quality examples. These ideas allow us to outperform training with Self-Instruct while utilizing the same seed tasks. The resulting synthetic data enables base models like MPT-7B to outperform GPT-3, a far larger model with 175B parameters. The results of this work also encourage the departure from closed-access models for advancing instruction generation algorithms.

---

\(^\text{10}\)https://lmsys.org/blog/2023-03-30-vicuna/
7 Limitations

Due to time and resource constraints, some parts of the experimental setup are not ideal. All model outputs were collected from an internal API serving models from HuggingFace\footnote{https://huggingface.co/}. Due to limitations of this API, different number of samples were collected for each model which may have introduced noise in the performance estimates. We report the exact number of samples used for training along with the results. Note that for cases using ensembling one has to take into account that there is an additional filtering process that removes samples. We provide approximate rates for ensembling filtering in Table 3. For the small user-oriented test set containing 252 tasks, automatic evaluation is arguably not ideal. Proper human evaluation would provide a clearer signal but this requires of significant time investment and resources. The method employs a set of various LMs, and therefore the generated synthetic data can be susceptible to the limitations of such LMs, particularly the biases inherent in the training data which may be harmful leading to synthetic data with hate, abuse and social stereotypes.

References


