SELF-INSTRUCT: Aligning Language Model with Self Generated Instructions

Yizhong Wang♣   **Yeganeh Kordi**◇   **Swaroop Mishra**♡   **Alisa Liu**♣
Noah A. Smith♣+   **Daniel Khashabi**♠   **Hannaneh Hajishirzi**♣+

♣University of Washington   ◇Tehran Polytechnic   ♡Arizona State University
♠Johns Hopkins University +Allen Institute for AI
yizhongw@cs.washington.edu

Abstract

Large "instruction-tuned" language models (finetuned to respond to instructions) have demonstrated a remarkable ability to generalize zero-shot to new tasks. Nevertheless, they depend heavily on human-written instruction data that is limited in quantity, diversity, and creativity, therefore hindering the generality of the tuned model. We introduce SELF-INSTRUCT, a framework for improving the instruction-following capabilities of pretrained language models by bootstrapping off its own generations. Our pipeline generates instruction, input, and output samples from a language model, then prunes them before using them to finetune the original model. Applying our method to vanilla GPT3, we demonstrate a 33% absolute improvement over the original model on SUPER-NATURALINSTRUCTIONS, on par with the performance of InstructGPT$_{001}$[1], which is trained with private user data and human annotations. For further evaluation, we curate a set of expert-written instructions for novel tasks, and show through human evaluation that tuning GPT3 with SELF-INSTRUCT outperforms using existing public instruction datasets by a large margin, leaving only a 5% absolute gap behind InstructGPT$_{001}$. SELF-INSTRUCT provides an almost annotation-free method for aligning pretrained language models with instructions, and we release our large synthetic dataset to facilitate future studies on instruction tuning[2].

1 Introduction

The recent NLP literature has witnessed a tremendous amount of activity in building models that can follow natural language instructions (Mishra et al., 2022; Wei et al., 2022; Sanh et al., 2022; Wang et al., 2022; Ouyang et al., 2022; Chung et al., 2022, i.a.). These developments are powered by two key components: large pre-trained language models (LM) and human-written instruction data. PROMPTSOURCE (Bach et al., 2022) and SUPER-NATURALINSTRUCTIONS (Wang et al., 2022) are two notable recent datasets that use extensive manual annotation for collecting instructions to construct T0 (Bach et al., 2022; Sanh et al., 2022) and Tk-INSTRUCT (Wang et al., 2022). However, this process is costly and often suffers limited diversity given that most human generations tend to be popular NLP tasks, falling short of covering a true variety of tasks and different ways to describe them. Given these limitations, continuing to improve the quality of instruction-tuned models necessitates the development of alternative approaches for supervising instruction-tuned models.

In this work, we introduce SELF-INSTRUCT, a semi-automated process for instruction-tuning a pretrained LM using instructional signals from the model itself. The overall process is an iterative bootstrapping algorithm (see Figure 1), which starts off with a limited (e.g., 175 in our study) seed set of manually-written instructions that are used to guide the overall generation. In the first phase, the model is prompted to generate instructions for new tasks. This step leverages the existing collection of instructions to create more broad-coverage instructions that define (often new) tasks. Given the newly-generated set of instructions, the framework also creates input-output instances for them, which can be later used for supervising the instruction tuning. Finally, various measures are used to prune low-quality and repeated instructions, before adding them to the task pool. This process can be repeated for many interactions until reaching a large number of tasks.

To evaluate SELF-INSTRUCT empirically, we run

---

[1]Unless otherwise specified, our comparisons are with the `text-davinci-001` engine. We focus on this engine since it is the closest to our experimental setup: supervised fine-tuning with human demonstrations. The newer engines are more powerful, though they use more data (e.g., code completion or latest user queries) or algorithms (e.g., PPO) that are difficult to compare with.

[2]Code and data will be available at https://github.com/yizhongw/self-instruct.
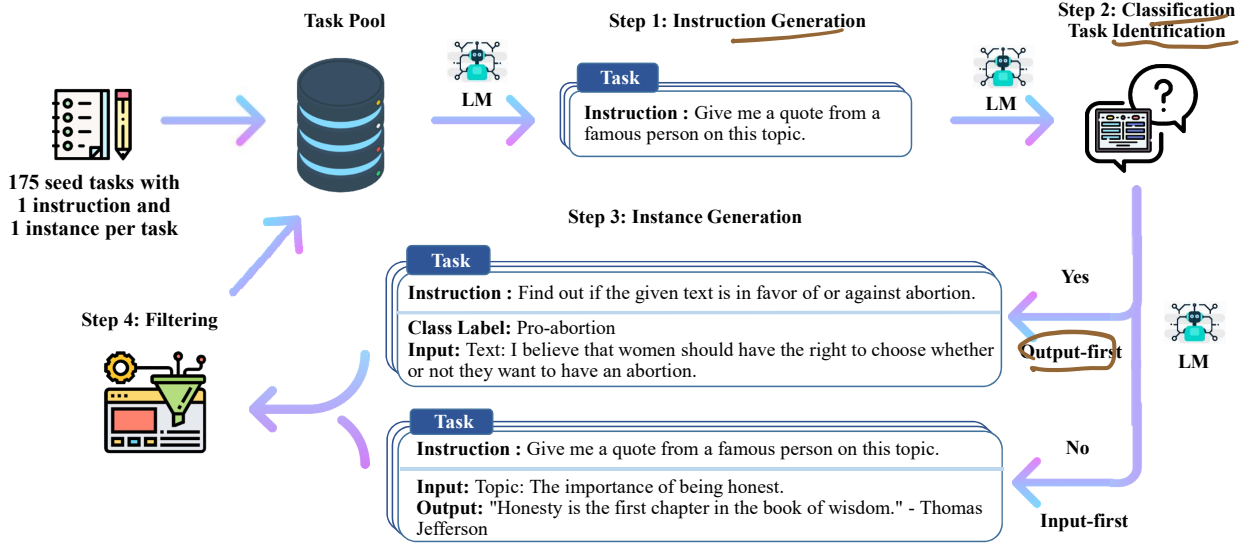
Figure 1: A high-level overview of SELF-INSTRUCT. The process starts with a small seed set of tasks (one instruction and one input-output instance for each task) as the task pool. Random tasks are sampled from the task pool, and used to prompt an off-the-shelf LM to generate both new instructions and corresponding instances, followed by filtering low-quality or similar generations, and then added back to the initial repository of tasks. The resulting data can be used for the instruction tuning of the language model itself later to follow instructions better. Tasks shown in the figure are generated by GPT3. See Table 10 for more creative examples.

this framework on GPT3 (Brown et al., 2020), which is a vanilla LM (§4). The iterative SELF-INSTRUCT process on this model leads to about 52k instructions, paired with about 82K instance inputs and target outputs. We observe that the resulting data provides a diverse range of creative tasks and over 50% of them have less than 0.3 ROUGE-L overlaps with the seed instructions (§4.2). On this resulting data, we build $GPT3_{SELF-INST}$ by fine-tuning GPT3 (i.e., the same model used for generating the instructional data). We evaluate $GPT3_{SELF-INST}$ in comparison to various other models on both typical NLP tasks included in SUPER-NATURALINSTRUCTIONS (Wang et al., 2022), and a set of new instructions that are created for novel usage of instruction-following models (§5). The SUPERNI results indicate that $GPT3_{SELF-INST}$ outperforms GPT3 (the original model) by a large margin (+33.1%) and nearly matches the performance of $InstructGPT_{001}$. Moreover, our human evaluation on the newly-created instruction set shows that $GPT3_{SELF-INST}$ demonstrates a broad range of instruction following ability, outperforming models trained on other publicly available instruction datasets and leaving only a 5% gap behind $InstructGPT_{001}$.

In summary, our contributions are: (1) SELF-INSTRUCT, a method for inducing instruction-following capability with minimal human-labeled

data; (2) We demonstrate its effectiveness via extensive instruction-tuning experiments; (3) We release a large synthetic dataset of 52K instructions and a set of manually-written novel tasks for building and evaluating future instruction-following models.

2 Related Work

Instruction-following language models. A series of works have found evidence that vanilla language models can be effective at following general language instructions if tuned with annotated "instructional" data – datasets containing language instructional commands and their desired outcome based on human judgement (Weller et al., 2020; Mishra et al., 2022; Wang et al., 2022; Wei et al., 2022; Sanh et al., 2022; Ouyang et al., 2022; Parmar et al., 2022; Scialom et al., 2022; Chung et al., 2022; Luo et al., 2022; Puri et al., 2022; Yin et al., 2022; Chakrabarty et al., 2022; Lin et al., 2022; Gupta et al., 2022; Muennighoff et al., 2022). Additionally, they show a direct correlation between the size and diversity of the "instructional" data and the generalizability of resulting models to unseen tasks. Since these developments depend on human annotated "instructional" data, this poses a bottleneck for progress toward more generalizable models (for example see Fig. 5a in Wang et al., 2022). Our work aims to tackle this bottleneck by reducing the dependence on human annotators.

Additionally, despite the remarkable performance of models like InstructGPT (Ouyang et al., 2022), their construction process remains quite opaque. In particular, the role of *data* has remained understudied due to limited transparency and data released by major corporate entities behind these key models. Addressing such challenges necessitates the creation of a large-scale, public dataset covering a broad range of tasks.

Instruction-following models have also been of interest in the multi-modal learning literature (Fried et al., 2018; Shridhar et al., 2020; Min et al., 2022; Weir et al., 2022). SELF-INSTRUCT, as a general approach to expanding data, can potentially also be helpful in those settings; however, this is out of the scope of this work.

Language models for data generation and augmentation. A variety of works have relied on generative LMs for data generation (Schick and Schütze, 2021; Wang et al., 2021; Liu et al., 2022; Meng et al., 2022) or augmentation (Feng et al., 2021; Yang et al., 2020; Mekala et al., 2022). For example, Schick and Schütze (2021) propose to replace human annotations *of a given task* with prompting large LMs and use the resulting data for fine-tuning (often smaller) models in the context of SuperGLUE tasks (Wang et al., 2019). While our work can be viewed as a form of "augmentation," our work differs from this line in that it is *not* specific to a particular task (say, QA or NLI). In contrast, a distinct motivation for SELF-INSTRUCT is to bootstrap new task definitions that may not have been defined before by any NLP practitioner (though potentially still important for downstream users).

Self-training. A typical self-training framework (He et al., 2019; Xie et al., 2020; Du et al., 2021; Amini et al., 2022; Huang et al., 2022) uses trained models to assign labels to unlabeled data and then leverages the newly labeled data to improve the model. In a similar line, Zhou et al. (2022a) use multiple prompts to specify a single task and propose to regularize via prompt consistency, encouraging consistent predictions over the prompts. This allows either finetuning the model with extra unlabeled training data, or direct application at inference time. While SELF-INSTRUCT has some similarities with the self-training literature, most self-training methods assume a specific *target task* as well as *unlabeled examples* under it; in contrast,

SELF-INSTRUCT produces a variety of tasks from scratch.

Knowledge distillation. Knowledge distillation (Hinton et al., 2015; Sanh et al., 2019; West et al., 2021; Magister et al., 2022) often involves the transfer of knowledge from larger models to smaller ones. SELF-INSTRUCT can also be viewed as a form of "knowledge distillation", however, it differs from this line in the following ways: (1) the source and target of distillation are the same, i.e., a model's knowledge is distilled to itself; (2) the content of distillation is in the form of an instruction task (i.e., instructions that define a task, and a set of examples that instantiate it).

Bootstrapping with limited resources. A series of recent works use language models to bootstrap some inferences using specialized methods. NPPrompt (Zhao et al., 2022) provides a method to generate predictions for semantic labels without any fine-tuning. It uses a model's own embeddings to automatically find words relevant to the label of the data sample and hence reduces the dependency on manual mapping from model prediction to label (verbalizers). STAR (Zelikman et al., 2022) iteratively leverages a small number of rationale examples and a large dataset without rationales, to bootstrap a model's ability to perform reasoning. Self-Correction (Welleck et al., 2022) decouples an imperfect base generator (model) from a separate corrector that learns to iteratively correct imperfect generations and demonstrates improvement over the base generator. Our work instead focuses on bootstrapping new tasks in the instruction paradigm.

Instruction generation. A series of recent works (Zhou et al., 2022b; Ye et al., 2022; Singh et al., 2022; Honovich et al., 2022) generate instructions of a task given a few examples. While SELF-INSTRUCT also involves instruction generation, a major difference in our case is it is task-agnostic; we generate new tasks (instructions along with instances) from scratch.

3   Method

Annotating large-scale instruction data can be challenging for humans because it requires 1) creativity to come up with novel tasks and 2) expertise for writing the labeled instances for each task. In this section, we detail our process for SELF-INSTRUCT, which refers to the pipeline of generating tasks with a **vanilla pretrained language model** itself and

then conducting instruction tuning with this generated data in order to align the language model to follow instructions better. This pipeline is depicted in Figure 1.

3.1 Defining Instruction Data

The instruction data we want to generate contains a set of instructions $\{I_t\}$, each of which defines a task t in natural language. Each task has one or more input-output instances (X_t, Y_t). A model M is expected to produce the output y, given the task instruction I_t and the instance input x: $M(I_t, x) = y$, for $(x, y) \in (X_t, Y_t)$. Note that the instruction and instance input does not have a strict boundary in many cases. For example, "write an essay about school safety" can be a valid instruction that we expect models to respond to directly, while it can also be formulated as "write an essay about the following topic" as the instruction, and "school safety" as an instance input. To encourage the diversity of the data format, we allow such instructions that do not require additional input (i.e., x is empty).

3.2 Automatic Instruction Data Generation

Our pipeline for generating the instruction data consists of four steps: 1) instruction generation, 2) identifying whether the instruction represents a classification task or not, 3) instance generation with the input-first or the output-first approach, and 4) filtering low-quality data.

Instruction Generation. SELF-INSTRUCT is based on a finding that large pretrained language models can be prompted to generate new and novel instructions when presented with some existing instructions in the context. This provides us with a way to grow the instruction data from a small set of seed human-written instructions. We propose to generate a diverse set of instructions in a bootstrapping fashion. We initiate the task pool with 175 tasks (1 instruction and 1 instance for each task) written by our authors. For every step, we sample 8 task instructions from this pool as in-context examples. Of the 8 instructions, 6 are from the human-written tasks, and 2 are from the model-generated tasks in previous steps to promote diversity. The prompting template is shown in Table 6.

Classification Task Identification. Because we need two different approaches for classification and non-classification tasks, we next identify whether the generated instruction represents a classification task or not.[3] We prompt vanilla GPT3 few-shot to determine this, using 12 classification instructions and 19 non-classification instructions from the seed tasks. The prompting template is shown in Table 7.

Instance Generation. Given the instructions and their task type, we generate instances for each instruction independently. This is challenging because it requires the model to understand what the target task is, based on the instruction, figure out what additional input fields are needed and generate them, and finally complete the task by producing the output. We found that pretrained language models can achieve this to a large extent when prompted with instruction-input-output in-context examples from other tasks. A natural way to do this is the **Input-first Approach**, where we can ask a language model to come up with the input fields first based on the instruction, and then produce the corresponding output. This generation order is similar to how models are used to respond to instruction and input, but here with in-context examples from other tasks. The prompting template is shown in Table 8.

However, we found that this approach can generate inputs biased toward one label, especially for classification tasks (e.g., for grammar error detection, it usually generates grammatical input). Therefore, we additionally propose an **Output-first Approach** for classification tasks, where we first generate the possible class labels, and then condition the input generation on each class label. The prompting template is shown in Table 9.[4] We apply the output-first approach to the classification tasks identified in the former step, and the input-first approach to the remaining non-classification tasks.

Filtering and Postprocessing. To encourage diversity, a new instruction is added to the task pool only when its ROUGE-L overlap with any existing instruction is less than 0.7. We also exclude instructions that contain some specific keywords (e.g., images, pictures, graphs) that usually can not be processed by language models. When generating new instances for each instruction, we filter out instances that are exactly the same or those with the same input but different outputs.

---

[3] More concretely, we regard tasks that have a limited and small output label space as classification tasks.

[4] In this work, we use a fixed set of seed tasks for prompting the instance generation, and thus only generate a small number of instances per task in one round. Future work can use randomly sampled tasks to prompt the model to generate a larger number of instances in multiple rounds.

3.3 Finetuning the LM to Follow Instructions

After the creation of the large-scale instruction data, we use this data to finetune the original language model (i.e., SELF-INSTRUCT). To do this, we concatenate the instruction and instance input as a prompt and train the model to generate the instance output in a standard supervised way. To make the model robust to different formats, we use multiple templates to encode the instruction and instance input together. For example, the instruction can be prefixed with "Task:" or not, the input can be prefixed with "Input:" or not, "Output:" can be appended at the end of the prompt, and different numbers of break lines can be put in the middle, etc.

4 SELF-INSTRUCT Data from GPT3

In this section, we apply our method for inducing instruction data to GPT3 as a case study. We use the largest GPT3 language model ("davinci" engine) accessed through the OpenAI API[5]. The parameters for making queries are described in Appendix A.1. Here we present an overview of the generated data.

4.1 Statistics

Table 1 describes the basic statistics of the generated data. We generate a total of over 52K instructions, and more than 82K instances corresponding to these instructions after filtering.

statistic	
# of instructions	52,445
- # of classification instructions	11,584
- # of non-classification instructions	40,861
# of instances	82,439
- # of instances with empty input	35,878
ave. instruction length (in words)	15.9
ave. non-empty input length (in words)	12.7
ave. output length (in words)	18.9

Table 1: Statistics of the generated data by applying SELF-INSTRUCT to GPT3.

4.2 Diversity

To study what types of instructions are generated and how diverse they are, we identify the verb-noun structure in the generated instructions. We use the Berkeley Neural Parser[6] (Kitaev and Klein, 2018; Kitaev et al., 2019) to parse the instructions, and then extract the verb that is closest to the root of

the parse tree as well as its first direct noun object. 26,559 out of the 52,445 instructions contain such structure; other instructions usually contain more complex clauses (e.g., "Classify whether this tweet contains political content or not.") or are framed as questions (e.g., "Which of these statements are true?"). We plot the top 20 most common root verbs and their top 4 direct noun objects in Figure 2, which accounts for 14% of the entire set. Overall, we see quite diverse intents and textual formats in these instructions.

We further study how the generated instructions differ from the seed instructions that are used to prompt the generation. For each generated instruction, we compute its highest ROUGE-L overlap with the 175 seed instructions. We plot the distribution of these ROUGE-L scores in Figure 3, indicating a decent number of new instructions that do not have much overlap with the seeds. We also demonstrate diversity in length of the instructions, instance inputs, and instance outputs in Figure 4.

4.3 Quality

So far, we have shown the quantity and diversity of the generated data, but its quality remains uncertain. To investigate this, we randomly sample 200 instructions and randomly select 1 instance per instruction. We asked an expert annotator (co-author of this work) to label whether each instance is correct or not, in terms of the instruction, the instance input, and the instance output. Evaluation results in Table 2 show that most of the generated instructions are meaningful, while the generated instances may contain more noise (to a reasonable extent). However, we found that even though the generations may contain errors, most of them are still in the correct format or even partially correct, which can provide useful guidance for training models to follow instructions. We listed a number of good generations and bad generations in Table 10 and Table 11 respectively.

5 Experimental Results

We conduct experiments to measure and compare the quality of models under various instruction tuning setups. We first describe our models and other baselines, followed by our experiments.

---

[5]https://openai.com/api/
[6]https://parser.kitaev.io/

Figure 2: The top 20 most common root verbs (inner circle) and their top 4 direct noun objects (outer circle) in the generated instructions. Despite their diversity, the instructions shown here only account for 14% of all the generated instructions because many instructions (e.g., "Classify whether the user is satisfied with the service.") do not contain such a verb-noun structure.

Figure 3: Distribution of the ROUGE-L scores between generated instructions and their most similar seed instructions.

Figure 4: Length distribution of the generated instructions, non-empty inputs, and outputs.

Quality Review Question	Yes %
Does the instruction describe a valid task?	92%
Is the input appropriate for the instruction?	79%
Is the output a correct and acceptable response to the instruction and input?	58%
All fields are valid	54%

Table 2: Data quality review for the instruction, input, and output of the generated data. See Table 10 and Table 11 for representative valid and invalid examples.

5.1 GPT3$_{\text{SELF-INST}}$: fine-tuning GPT3 on its own instruction data

With the instruction-generated instruction data, we conduct instruction tuning for the GPT3 model itself (the "davinci" engine). As we described in §3.3, we use various templates to concatenate the instruction and input, and train the model to generate the output. This finetuning is done through the OpenAI finetuning API[7]. We use the default hyper-parameters, except that we set the prompt loss weight to 0, and we train the model for 2 epochs. We refer the readers to Appendix A.2 for additional finetuning details. The resulting model is denoted as GPT3$_{\text{SELF-INST}}$.

5.2 Baselines

Off-the-shelf language models. We evaluate T5-LM (Lester et al., 2021; Raffel et al., 2020) and GPT3 (Brown et al., 2020) as the vanilla LM baselines (only pre-training, no additional fine-tuning). These baselines will indicate the extent to which off-the-shelf LMs are capable of following instructions naturally immediately after pretraining.

Publicly-available instruction-tuned models. T0 and Tk-INSTRUCT are two instruction-tuned models proposed in Sanh et al. (2022) and Wang et al. (2022) respectively, and are demonstrated to be able to follow instructions for many NLP

---

[7] https://beta.openai.com/docs/guides/fine-tuning

tasks. Both of these models are finetuned from the T5 (Raffel et al., 2020) checkpoints and are publicly available[8][9]. For both of these models, we use their largest version with 11B parameters.

Instruction-tuned GPT3 models. We evaluate InstructGPT (Ouyang et al., 2022), which is developed by OpenAI based on GPT3 to follow human instructions better and has been found by the community to have impressive zero-shot abilities. There are various generations of these models, where newer ones use more expansive data or algorithmic novelties[10]. For our SUPERNI experiments in §5.3, we only compare with their `text-davinci-001` engine, because their newer engines are trained with the latest user data and are likely to already see the SUPERNI evaluation set. For our human evaluation of these models on newly written instructions, we include their 001, 002 and 003 engines for completeness.

Additionally, to compare SELF-INSTRUCT training with other publicly available instruction tuning data, we further finetune GPT3 model with data from PROMPTSOURCE and SUPERNI, which are used to train the T0 and Tk-INSTRUCT models. We call them T0 training and SUPERNI training for short, respectively. To save the training budget, we sampled 50K instances (but covering all their instructions) for each dataset, which has a comparable size to the instruction data we generated. Based on the findings from Wang et al. (2022) and our early experiments, reducing the number of instances per task does not degrade the model's generalization performance to unseen tasks.

5.3 Experiment 1: Zero-Shot Generalization on SUPERNI benchmark

We first evaluate the models' ability to follow instructions on typical NLP tasks in a zero-shot fashion. We use the evaluation set of SUPERNI (Wang et al., 2022), which consists of 119 tasks with 100 instances in each task. In this work, we mainly focus on the zero-shot setup, i.e., the model is prompted with the definition of the tasks only, without in-context demonstration examples. For all our requests to the GPT3 variants, we use the deterministic generation mode (temperature as 0 and no nucleus sampling) without specific stop sequences.

---

[8]https://huggingface.co/bigscience/T0
[9]https://huggingface.co/allenai/
tk-instruct-11b-def
[10]https://beta.openai.com/docs/
model-index-for-researchers

Results. We make the following observations from the results in Table 3. SELF-INSTRUCT boosts the instruction-following ability of GPT3 by a large margin. The vanilla GPT3 model basically cannot follow human instructions at all. Upon manual analysis, we find that it usually generates irrelevant and repetitive text, and does not know when to stop generation. Compared with other models that are not specifically trained for SUPERNI, GPT3$_{\text{SELF-INST}}$ achieves better performance than T0 or the GPT3 finetuned on the T0 training set, which takes tremendous human labeling efforts. Notably, GPT3$_{\text{SELF-INST}}$ also nearly matches the performance of InstructGPT$_{001}$, which is trained with private user data and human-annotated labels.

Models trained on SUPERNI training set still achieve better performance on its evaluation set, which we attribute to the similar instruction style and formatting. However, we show that SELF-INSTRUCT still brings in additional gains when combined with the SUPERNI training set, proving its value as complementary data.

5.4 Experiment 2: Generalization to User-oriented Instructions on Novel Tasks

Despite the comprehensiveness of SUPERNI in collecting existing NLP tasks, most of these NLP tasks were proposed for research purposes and skewed toward classification. To better access the practical value of instruction-following models, a subset of the authors curate a new set of instructions motivated by user-oriented applications. We first brainstorm different domains where large LMs may be useful (e.g., email writing, social media, productivity tools, entertainment, programming), then craft instructions related to each domain along with an input-output instance (again, input is optional). We aim to diversify the styles and formats of these tasks (e.g., instructions may be long or short; input/output may take the form of bullet points, tables, codes, equations, etc.). In total, we create 252 instructions with 1 instance per instruction. We believe it can serve as a testbed for evaluating how instruction-based models handle diverse and unfamiliar instructions. Table 4 presents a small portion of the 252 tasks. The whole test set will be available upon request.

Human evaluation setup. Evaluating models' performance on this evaluation set of diverse tasks is extremely challenging because different tasks re-

Model	# Params	ROUGE-L
Vanilla LMs		
T5-LM	11B	25.7
GPT3	175B	6.8
Instruction-tuned w/o SUPERNI		
T0	11B	33.1
GPT3 + T0 Training	175B	37.9
GPT3$_{\text{SELF-INST}}$ (Ours)	175B	39.9
InstructGPT$_{001}$	175B	**40.8**
Instruction-tuned w/ SUPERNI		
Tk-INSTRUCT	11B	46.0
GPT3 + SUPERNI Training	175B	49.5
GPT3$_{\text{SELF-INST}}$ + SUPERNI Training (Ours)	175B	**51.6**

Table 3: Evaluation results on *unseen* tasks from SUPER-NATURALINSTRUCTIONS (§5.3). From the results, we see that ① SELF-INSTRUCT can boost GPT3 performance by a large margin (+33.1%) and ② nearly matches the performance of InstructGPT$_{001}$. Additionally, ③ it can further improve the performance even when a large amount of labeled instruction data is present.
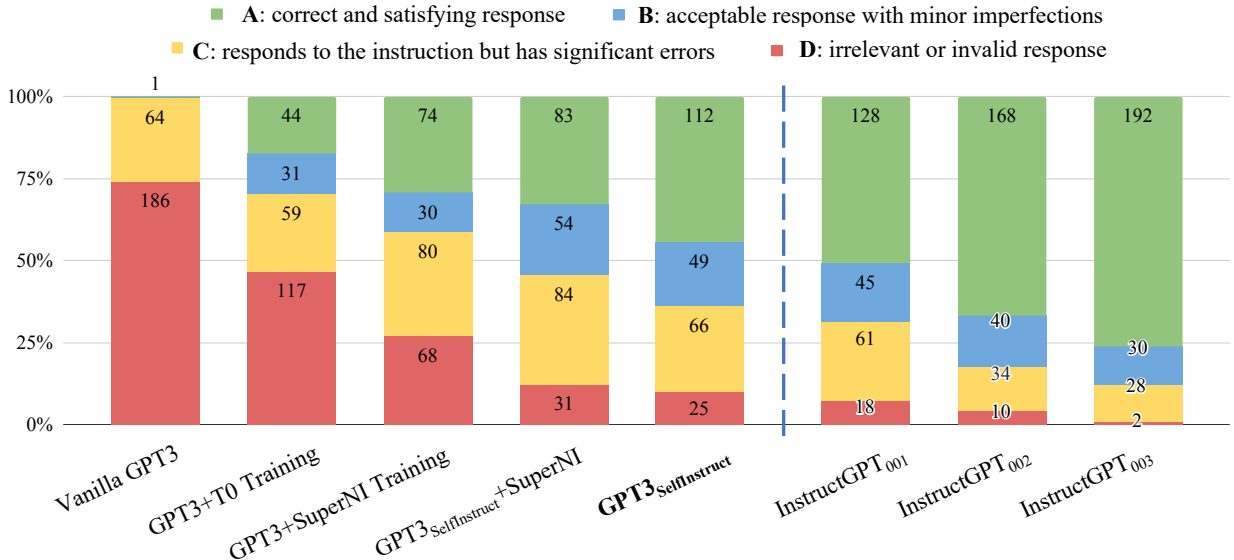
Figure 5: Performance of GPT3 model and its instruction-tuned variants, evaluated by human experts on our 252 user-oriented instructions (§5.4). Human evaluators are instructed to rate the models' responses into four levels. The results indicate that GPT3$_{\text{SELF-INST}}$ outperforms all the other GPT3 variants trained on publicly available instruction datasets. Additionally, GPT3$_{\text{SELF-INST}}$ scores nearly as good as InstructGPT$_{001}$ (c.f., footnote 1).

quire different expertise. Indeed, many of these tasks cannot be measured by automatic metrics or even be judged by normal crowdworkers (e.g., writing a program, or converting first-order logic into natural language). To get a more faithful evaluation, we asked the authors of the instructions to judge model predictions. The evaluators were asked to rate the output based on whether it accurately and effectively completes the task. We implemented a four-level rating system for categorizing the quality of the models' outputs, defined as follows:

- RATING-A: The response is valid and satisfying.

- RATING-B: The response is acceptable but has minor errors or imperfections that can be improved.
- RATING-C: The response is relevant and responds to the instruction, but it has significant errors in the content. For example, GPT3 might generate a valid output first, but continue to generate other irrelevant things.
- RATING-D: The response is irrelevant or invalid, including repetition of the input, totally irrelevant output, etc.

Results. Figure 5 provides the performance of GPT3 model and its instruction-tuned counterparts on this newly written instruction set. As anticipated, the vanilla GPT3 language model is largely unable to respond to instructions, and all instruction-tuned models demonstrate comparatively higher performance, Nonetheless, GPT3$_{\text{SELF-INST}}$ (i.e., GPT3 model fine-tuned with SELF-INSTRUCT) outperforms those counterparts trained on T0 or SUPERNI by a large margin, demonstrating the value of the generated data despite the noise. Compared with InstructGPT$_{001}$ (c.f. footnote 1), GPT3$_{\text{SELF-INST}}$ is quite close in the performance—if we count acceptable response with minor imperfections (RATING-3) as valid, GPT3$_{\text{SELF-INST}}$ is only 5% behind InstructGPT$_{001}$. Lastly, our evaluation confirms the impressive instruction-following ability of InstructGPT$_{002}$ & InstructGPT$_{003}$ models. Although there are many factors behind this success, we conjecture that future work can largely benefit from improving the quality of our generated data by using human annotators or training a reward model to select better generations, similar to the algorithm used in Ouyang et al. (2022).

5.5 Example Predictions from GPT3$_{\text{SELF-INST}}$

We present a selection of user-oriented tasks, the corresponding GPT3$_{\text{SELF-INST}}$-produced responses and annotator ratings in Table 4. We see that even for responses rated as level 2, the model demonstrates extensive steps in solving the task, even though its final output is incorrect.

6 Discussion and Limitation

6.1 Why does SELF-INSTRUCT work?

It is worthwhile to reflect on the role that high-quality human feedback plays in enabling the recent successes on instruction-tuning LMs (Mishra et al., 2022; Wang et al., 2022; Wei et al., 2022; Sanh et al., 2022; Ouyang et al., 2022). Here are two extreme hypotheses:

(H_1) *Human feedback* is a *necessary and indispensable* aspect of instruction-tuning as LMs need to learn about issues that were not quite learned during pre-training.

(H_2) *Human feedback* is an *optional* aspect of instruction-tuning as LMs are already quite familiar with instructions from their pre-training. Observing the human feedback is merely a

lightweight process for aligning their pre-training distribution/objective which might be replaceable with a different process.

While the reality probably lies somewhere in between these two extremes, we conjecture that it is closer to H_2, particularly for larger models. This intuition, that LMs already know much about language instructions, is a key motivation for SELF-INSTRUCT and is also supported by its empirical success.

6.2 Broader Impact

Beyond the immediate focus of this paper, we believe that SELF-INSTRUCT may help bring more transparency to what happens "behind the scenes" of widely-used instruction-tuned models like InstructGPT. Unfortunately, such industrial models remain behind the API walls as their datasets are not released, and hence there is little understanding of their constructions and why they demonstrate impressive capabilities. The burden now falls on academia to better understand the source of success in these models and strive for better – yet open – models. We believe our findings in this paper demonstrate the importance of diverse instruction data, and our large synthetic dataset can be the first step toward higher-quality data for building better instruction-following models.

6.3 Limitations of SELF-INSTRUCT

Here, we discuss some limitations of this work to inspire future research in this direction.

Tail phenomena. SELF-INSTRUCT depends on LMs, and it will inherit all the limitations that carry over with LMs. As recent studies have shown (Razeghi et al., 2022; Kandpal et al., 2022), *tail phenomena* pose a serious challenge to the success of LMs. In other words, LMs' largest gains correspond to the frequent uses of languages (head of the language use distribution), and there are minimal gains in the low-frequency contexts. Similarly, in the context of this work, it would not be surprising if the majority of the gains by SELF-INSTRUCT are skewed toward tasks or instructions that present more frequently in the pre-training corpus. As a consequence, the approach might show brittleness with respect to uncommon and creative instructions.

Dependence on large models. Because of SELF-INSTRUCT's dependence on the inductive biases extracted from LMs, it might work best for larger

Instruction + Instance Input	Target Response	GPT3$_{\text{SELF-INST}}$ Response	Rating
Instruction: Give students tips on how to keep their nerves under control during class presentations.	Everyone gets a little nervous when speaking in front of a group. That's normal. But the more preparation you've done-meaning plenty of researching, organizing, and rehearsing-the more confident you'll be. Preparation is the key. And if you make a mistake or stumble over your words, just regroup and keep going. Nobody's perfect, and nobody expects you to be.	- Practice your presentation in front of friends and family. - Don't drink too much caffeine before the presentation. - Take deep breaths before you start speaking. - If you feel nervous, smile. - Remember that everyone in the audience is rooting for you.	A
Instruction: Make a list of the pros and cons of the given decision. **Input:** Implementing a Remote Working Policy	Pros: - Allowing employees to work from home can reduce their stressful commutes and increase their productivity. - Remote working provides greater flexibility, particularly for working parents and caregivers who have other responsibilities. - ...(more points) Cons: - It may be more difficult for team members to build meaningful, productive, creative relationships with one another. - ...(more points)	Pros: - Employees can work from home and save on commuting costs. - Employees can spend more time with their families. Cons: - Employees may not be as productive as they would be in the office. - Employees may not be as motivated to work.	A
Instruction: A color description has been provided. Find the CSS code associated with that color. **Input:** A light red color with a medium light shade of pink	#DB7093 (■)	#ff66cc (■)	B
Instruction: Give some examples of what people usually say in the given social situation. **Input:** when someone arrives safely	- "Glad you made it safe and sound." - "Good to see you got here unscathed." - "So, bulletproof vests actually work!"	Thank God, I made it.	B
Instruction: You are given a sentence in first-order logic. Translate it into natural language. **Input:** $\exists x \forall y (soldier(x) \rightarrow general(x, y))$	There is someone (x) such that if he is a soldier, then he is general of everyone.	There exists a soldier such that for every general, he is a general.	C
Instruction: Design an outline for a blog post based on the given information and list the sections accordingly. **Input:** This blog post is going to about making a list of last-minute gift ideas. Also, the tone of the post is going to be relaxed and casual.	- Section: Movie tickets - Section: A fun game console - Section: Personalized items with photos/artwork - ...(more sections) - Takeaway: Don't stress about out running out of time to buy, make a gift.	- Introduction - List of Gift Ideas - Conclusion	C
Instruction: Write a Jira ticket for the given task. **Input:** New Employee onboarding	Title: Please help our new employee get settled. Prepare a desk, chair, and computer for them to use. Create their company account, email, and temporary password. Grant access to company documents.	Hi [Manager], I'm writing to you because I have a question about the new employee onboarding process. I've been working here for a few months now, and I'm still not sure how it works. Could you please explain it to me?	D
Instruction: Create alliterations by finding synonyms for words in the given sentence. **Input:** David wears a hat everyday.	David dons a derby daily.	David wears a hat every day.	D

Table 4: Examples in the User-oriented Instructions dataset (§5.4) and predictions from GPT3$_{\text{SELF-INST}}$. The right column indicates one of the four quality ratings assigned to the model's response, with "A" indicating "valid and satisfying" responses (highest) and "D" indicating "irrelevant or invalid response" (lowest).

models. If true, this may create barriers to access for those who may not have large computing resources. We hope future studies will carefully study the gains as a function of model size or various other parameters. It is worthwhile to note that instruction-tuning with human annotation also suffers from a similar limitation: gains of instruction-tuning are higher for larger model (Wei et al., 2022).

Reinforcing LM biases. A point of concern for the authors is the unintended consequences of this iterative algorithm, such as the amplification of problematic social biases (stereotypes or slurs about genders, races, etc.). Relatedly, one observed challenge in this process is the algorithm's difficulty in producing balanced labels, which reflected models' prior biases. We hope future work will hash out such details to better understand the pros and cons of the approach.

7 Conclusion

We introduce SELF-INSTRUCT, a task-agnostic method to improve the instruction-following capabilities of language models via its own generation of instruction data (instruction, input, and output samples) and bootstrapping with it. Our method conducts instruction-tuning of the original model on the pruned subset of generated samples. On experimenting with vanilla GPT3, we observe a 33% absolute improvement over the original model on SUPER-NATURALINSTRUCTIONS. This performance is on par with InstructGPT$_{001}$, which is trained with private user data and expensive human annotations. Furthermore, we curate a set of expert-written instructions for novel tasks. Human evaluation on this set shows that tuning GPT3 with SELF-INSTRUCT outperforms using existing public instruction datasets by a large margin, leaving only a 5% absolute gap behind InstructGPT$_{001}$. We hope SELF-INSTRUCT can serve as the first step to align pretrained language models to follow human instructions, and future work can build on top of this data to improve instruction-following models.

Acknowledgements

References

Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, Emilie Devijver, and Yury Maximov. 2022. Self-training: A survey. *arXiv preprint arXiv:2202.12040*.

Stephen H Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, et al. 2022. PromptSource: An Integrated Development Environment and Repository for Natural Language Prompts. In *Annual Meeting of the Association for Computational Linguistics* (ACL) - *System Demonstrations*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, and et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems* (NeurIPS).

Tuhin Chakrabarty, Vishakh Padmakumar, and He He. 2022. Help me write a poem: Instruction tuning as a vehicle for collaborative poetry writing. *arXiv preprint arXiv:2210.13669*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Jingfei Du, Édouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Veselin Stoyanov, and Alexis Conneau. 2021. Self-training improves pre-training for natural language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics* (NAACL)*: Human Language Technologies*, pages 5408–5418.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. In *Annual Meeting of the Association for Computational Linguistics* (ACL) *ACL-IJCNLP - Findings*, pages 968–988.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems* (NeurIPS).

Prakhar Gupta, Cathy Jiao, Yi-Ting Yeh, Shikib Mehri, Maxine Eskenazi, and Jeffrey P Bigham. 2022. Instructdial: Improving zero and few-shot generalization in dialogue through instruction tuning. *arXiv preprint arXiv:2205.12673*.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2019. Revisiting self-training for neural sequence generation. In *International Conference on Learning Representations* (ICLR).

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. In *Advances in Neural Information Processing Systems* (NeurIPS) *Workshop on Deep Learning*.

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*.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2022. Large language models struggle to learn long-tail knowledge. *arXiv preprint arXiv:2211.08411*.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Annual Meeting of the Association*

for Computational Linguistics (ACL), pages 3499–3505.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Annual Meeting of the Association for Computational Linguistics* (ACL), pages 2676–2686.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Conference on Empirical Methods in Natural Language Processing* (EMNLP).

Bill Yuchen Lin, Kangmin Tan, Chris Miller, Beiwen Tian, and Xiang Ren. 2022. Unsupervised cross-task generalization via retrieval augmentation. In *Advances in Neural Information Processing Systems* (NeurIPS).

Alisa Liu, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. 2022. WANLI: Worker and ai collaboration for natural language inference dataset creation. In *Conference on Empirical Methods in Natural Language Processing* (EMNLP) - *Findings*.

Man Luo, Sharad Saxena, Swaroop Mishra, Mihir Parmar, and Chitta Baral. 2022. Biotabqa: Instruction learning for biomedical table question answering. In *BioASQ Workshop*.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*.

Dheeraj Mekala, Tu Vu, Timo Schick, and Jingbo Shang. 2022. Leveraging qa datasets to improve generative data augmentation. *arXiv preprint arXiv:2205.12604*.

Yu Meng, Martin Michalski, Jiaxin Huang, Yu Zhang, Tarek Abdelzaher, and Jiawei Han. 2022. Tuning language models as training data generators for augmentation-enhanced few-shot learning. *arXiv preprint arXiv:2211.03044*.

So Yeon Min, Devendra Singh Chaplot, Pradeep Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. 2022. FILM: Following Instructions in Language with Modular Methods. In *International Conference on Learning Representations* (ICLR).

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-Task Generalization via Natural Language Crowdsourcing Instructions. In *Annual Meeting of the Association for Computational Linguistics* (ACL).

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training Language Models to Follow Instructions with Human Feedback. In *Advances in Neural Information Processing Systems* (NeurIPS).

Mihir Parmar, Swaroop Mishra, Mirali Purohit, Man Luo, Murad Mohammad, and Chitta Baral. 2022. In-BoXBART: Get instructions into biomedical multi-task learning. In *Conference of the North American Chapter of the Association for Computational Linguistics* (NAACL) - *Findings*, pages 112–128.

Ravsehaj Singh Puri, Swaroop Mishra, Mihir Parmar, and Chitta Baral. 2022. How many data samples is an additional instruction worth? *arXiv preprint arXiv:2203.09161*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research* (JMLR).

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. Impact of pretraining term frequencies on few-shot reasoning. *arXiv preprint arXiv:2202.07206*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *Advances in Neural Information Processing Systems* (NeurIPS) *Workshop on Energy Efficient Machine Learning and Cognitive Computing*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. Multitask Prompted Training Enables Zero-Shot Task Generalization. In *International Conference on Learning Representations* (ICLR).

Timo Schick and Hinrich Schütze. 2021. Generating datasets with pretrained language models. In *Conference on Empirical Methods in Natural Language Processing* (EMNLP).

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. 2022. Fine-tuned language models are continual learners. *arXiv preprint arXiv:2205.12393*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR).

Chandan Singh, John X Morris, Jyoti Aneja, Alexander M Rush, and Jianfeng Gao. 2022. Explaining patterns in data with language models via interpretable autoprompting. *arXiv preprint arXiv:2210.01848*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems* (NeurIPS).

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddhartha Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hannaneh Hajishirzi, and Daniel Khashabi. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ tasks. In *Conference on Empirical Methods in Natural Language Processing* (EMNLP).

Zirui Wang, Adams Wei Yu, Orhan Firat, and Yuan Cao. 2021. Towards zero-label language learning. *arXiv preprint arXiv:2109.09193*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations* (ICLR).

Nathaniel Weir, Xingdi Yuan, Marc-Alexandre Côté, Matthew Hausknecht, Romain Laroche, Ida Momennejad, Harm Van Seijen, and Benjamin Van Durme. 2022. One-Shot Learning from a Demonstration with Hierarchical Latent Language. *arXiv preprint arXiv:2203.04806*.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2022. Generating sequences by learning to self-correct. *arXiv preprint arXiv:2211.00053*.

Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew Peters. 2020. Learning from Task Descriptions. In *Conference on Empirical Methods in Natural Language Processing* (EMNLP).

Peter West, Chandra Bhagavatula, Jack Hessel, Jena D Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. 2021. Symbolic knowledge distillation: from general language models to commonsense models. In *Conference of the North American Chapter of the Association for Computational Linguistics* (NAACL).

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. 2020. Self-training with noisy student improves imagenet classification. In *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), pages 10687–10698.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. Generative data augmentation for commonsense reasoning. In *Conference on Empirical Methods in Natural Language Processing* (EMNLP) - *Findings*.

Seonghyeon Ye, Doyoung Kim, Joel Jang, Joongbo Shin, and Minjoon Seo. 2022. Guess the instruction! making language models stronger zero-shot learners. *arXiv preprint arXiv:2210.02969*.

Wenpeng Yin, Jia Li, and Caiming Xiong. 2022. ConTinTin: Continual learning from task instructions. In *Annual Meeting of the Association for Computational Linguistics* (ACL).

Eric Zelikman, Jesse Mu, Noah D Goodman, and Yuhuai Tony Wu. 2022. STar: Self-taught reasoner bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems* (NeurIPS).

Xuandong Zhao, Siqi Ouyang, Zhiguo Yu, Ming Wu, and Lei Li. 2022. Pre-trained language models can be fully zero-shot learners. *arXiv preprint arXiv:2212.06950*.

Chunting Zhou, Junxian He, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022a. Prompt Consistency for Zero-Shot Task Generalization. In *Conference on Empirical Methods in Natural Language Processing* (EMNLP) - *Findings*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022b. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.

Supplemental Material

A Implementation Details

A.1 Querying the GPT3 API

We use different sets of hyper-parameters when querying GPT3 API for different purposes. These hyper-parameters are found to work well with the GPT3 model ("davinci" engine) and the other instruction-tuned GPT3 variants. We listed them in Table 5.

Experiments ↓	Temp.	Top_P	Freq. Penalty	Presence Penalty	Beam Size	Max Length	Stop Sequences
Generating instructions	0..7	0.5	0	2	1	1024	"\n\n", "\n16", "16.", "16 ."
Identifying clf. tasks	0	0	0	0	1	3	"\n", "Task:"
Generating instances	0	0	0	1.5	1	300	"Task:"
Evaluating models	0	0	0	0	0	1024	None (default)

Table 5: Hyper-parameters for querying OpenAI API in different experiments.

A.2 Finetuning GPT3

$GPT3_{SELF-INST}$ and some of our baselines are finetuned from GPT3 model ("davinci" engine with 175B parameters). We conduct this finetuning via OpenAI's finetuning API[11]. While the details of how the model is finetuned with this API are not currently available (e.g., which parameters are updated, or what the optimizer is), we tune all our models with the default hyper-parameters of this API so that the results are comparable. We only set the "prompt_loss_weight" to 0 since we find this works better in our case, and every finetuning experiment is trained for two epochs to avoid overfitting the training tasks. Finetuning is charged based on the number of tokens in the training file. Finetuning $GPT3_{SELF-INST}$ from the GPT3 model took $338.

B Prompting Templates for Data Generation

SELF-INSTRUCT relies on a number of prompting templates in order to elicit the generation from language models. Here we provide our four templates for generating the instruction (Table 6), classifying whether an instruction represents a classification task or not (Table 7), generating non-classification instances with the input-first approach (Table 8), and generating classification instances with the output-first approach (Table 9).

```
Come up with a series of tasks:

Task 1:  {instruction for existing task 1}
Task 2:  {instruction for existing task 2}
Task 3:  {instruction for existing task 3}
Task 4:  {instruction for existing task 4}
Task 5:  {instruction for existing task 5}
Task 6:  {instruction for existing task 6}
Task 7:  {instruction for existing task 7}
Task 8:  {instruction for existing task 8}
Task 9:
```

Table 6: Prompt used for generating new instructions. 8 existing instructions are randomly sampled from the task pool for in-context demonstration. The model is allowed to generate instructions for new tasks, until it stops its generation, reaches its length limit or generates "Task 16" tokens.

---

[11] https://beta.openai.com/docs/guides/fine-tuning

```
Can the following task be regarded as a classification task with finite output labels?

Task:  Given my personality and the job, tell me if I would be suitable.
Is it classification?  Yes

Task:  Give me an example of a time when you had to use your sense of humor.
Is it classification?  No

Task:  Replace the placeholders in the given text with appropriate named entities.
Is it classification?  No

Task:  Fact checking - tell me if the statement is true, false, or unknown, based on your
knowledge and common sense.
Is it classification?  Yes

Task:  Return the SSN number for the person.
Is it classification?  No

Task:  Detect if the Reddit thread contains hate speech.
Is it classification?  Yes

Task:  Analyze the sentences below to identify biases.
Is it classification?  No

Task:  Select the longest sentence in terms of the number of words in the paragraph, output
the sentence index.
Is it classification?  Yes

Task:  Find out the toxic word or phrase in the sentence.
Is it classification?  No

Task:  Rank these countries by their population.
Is it classification?  No

Task:  You are provided with a news article, and you need to identify all the categories that
this article belongs to.  Possible categories include:  Music, Sports, Politics, Tech, Finance,
Basketball, Soccer, Tennis, Entertainment, Digital Game, World News.  Output its categories one
by one, seperated by comma.
Is it classification?  Yes

Task:  Given the name of an exercise, explain how to do it.
Is it classification?  No

Task:  Select the oldest person from the list.
Is it classification?  Yes

Task:  Find the four smallest perfect numbers.
Is it classification?  No

Task:  Does the information in the document supports the claim?  You can answer "Support" or
"Unsupport".
Is it classification?  Yes

Task:  Create a detailed budget for the given hypothetical trip.
Is it classification?  No

Task:  Given a sentence, detect if there is any potential stereotype in it.  If so, you should
explain the stereotype.  Else, output no.
Is it classification?  No

...

Task:  To make the pairs have the same analogy, write the fourth word.
Is it classification?  No

Task:  Given a set of numbers, find all possible subsets that sum to a given number.
Is it classification?  No

Task:  {instruction for the target task}
```

Table 7: Prompt used for classifying whether a task instruction is a classification task or not.

```
Come up with examples for the following tasks. Try to generate multiple examples when possible.
If the task doesn't require additional input, you can generate the output directly.

Task: Which exercises are best for reducing belly fat at home?
Output:
- Lying Leg Raises
- Leg In And Out
- Plank
- Side Plank
- Sit-ups

Task: Extract all the country names in the paragraph, list them separated by commas.
Example 1
Paragraph: 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.

Task: Converting 85 F to Celsius.
Output: 85°F = 29.44°C

Task: Sort the given list ascendingly.
Example 1
List: [10, 92, 2, 5, -4, 92, 5, 101]
Output: [-4, 2, 5, 5, 10, 92, 92, 101]
Example 2
Input 2 - List: [9.99, 10, -5, -1000, 5e6, 999]
Output: [-1000, -5, 9.99, 10, 999, 5e6]

Task: Suggest a better and more professional rephrasing of the following sentence.
Example 1
Sentence: This house is surprisingly not constructed very well, and you probably need more
money to fix it after you buy it. If you ask me, I would suggest you to consider other
candidates.
Output: This house does not seem to be constructed well, so you may need to spend more money
to fix it after you purchase it. I would suggest that you look at other properties.
Example 2
Sentence: Just so you know, we did an experiment last week and found really surprising results
- language model can improve itself!
Output: Our experiments last week demonstrated surprising results, proving that the language
model can improve itself.

...

Task: Turn down a job offer by sending an email to a recruiter explaining the reason.
Output: Hi [Recruiter],
Thank you so much for the generous offer to join your team. As we discussed, I've admired the
company for a number of years, and am a proud endorser of its products. However, after further
consideration of where I currently am in my career, I've decided to accept an offer at another
company.
I would love to stay in touch with you and have already started following you on [Social Media
Platform]. Again, thank you so much for your time and consideration.
Thanks again,
[Your Name]

Task: {Instruction for the target task}
```

Table 8: Prompt used for the input-first approach of instance generation. The model is prompted to generate the instance first, and then generate the corresponding output. For instructions that don't require additional input, the output is allowed to be generated directly.

```
Given the classification task definition and the class labels, generate an input that
corresponds to each of the class labels.  If the task doesn't require input, just generate the
correct class label.

Task:  Classify the sentiment of the sentence into positive, negative, or mixed.
Class label:  mixed
Sentence:  I enjoy the flavor of the restaurant but their service is too slow.
Class label:  Positive
Sentence:  I had a great day today.  The weather was beautiful and I spent time with friends.
Class label:  Negative
Sentence:  I was really disappointed by the latest superhero movie.  I would not recommend it.

Task:  Given a dialogue, classify whether the user is satisfied with the service.  You should
respond with "Satisfied" or "Unsatisfied".
Class label:  Satisfied
Dialogue:
- Agent:  Thank you for your feedback.  We will work to improve our service in the future.
- Customer:  I am happy with the service you provided.  Thank you for your help.
Class label:  Unsatisfied
Dialogue:
- Agent:  Sorry that we will cancel your order.  You will get a refund within 7 business days.
- Customer:  oh that takes too long.  I want you to take quicker action on this.

Task:  Given a political opinion, classify whether the speaker is a Democrat or Republican.
Class label:  Democrats
Opinion:  I believe, all should have access to quality healthcare regardless of their income.
Class label:  Republicans
Opinion:  I believe that people should be able to keep more of their hard-earned money and
should not be taxed at high rates.

Task:  Tell me if the following email is a promotion email or not.
Class label:  Promotion
Email:  Check out our amazing new sale!  We've got discounts on all of your favorite products.
Class label:  Not Promotion
Email:  We hope you are doing well.  Let us know if you need any help.

Task:  Detect if the Reddit thread contains hate speech.
Class label:  Hate Speech
Thread:  All people of color are stupid and should not be allowed to vote.
Class label:  Not Hate Speech
Thread:  The best way to cook a steak on the grill.

Task:  Does the document supports the claim?  Answer with "Support" or "Unsupport".
Class label:  Unsupport
Document:  After a record-breaking run that saw mortgage rates plunge to all-time lows and
home prices soar to new highs, the U.S. housing market finally is slowing.  While demand and
price gains are cooling, any correction is likely to be a modest one, housing economists and
analysts say.  No one expects price drops on the scale of the declines experienced during the
Great Recession.
Claim:  The US housing market is going to crash soon.
Class label:  Support
Document:  The U.S. housing market is showing signs of strain, with home sales and prices
slowing in many areas.  Mortgage rates have risen sharply in recent months, and the number
of homes for sale is increasing.  This could be the beginning of a larger downturn, with some
economists predicting a potential housing crash in the near future.
Claim:  The US housing market is going to crash soon.

...

Task:  Tell me the first number of the given list.
Class label:  1
List:  1, 2, 3
Class label:  2
List:  2, 9, 10

Task:  Which of the following is not an input type?  (a) number (b) date (c) phone number (d)
email address (e) all of these are valid inputs.
Class label:  (e)

Task:  {instruction for the target task}
```

Table 9: Prompt used for the output-first approach of instance generation. The model is prompted to generate the class label first, and then generate the corresponding input. This prompt is used for generating the instances for classification tasks.

C   Task and Instance Examples from the Generated Instruction Data

Instruction + Instance Input	Target Output	Validity
Instruction: Generate a random password with at least 6 characters.	```def generateRandomPassword():\n password = ""\n while len(password) < 6:\n password += chr(random.randint(65, 90))\n return password```	Instruction: ✓ Input: ✓ Output: ✓
Instruction: Write a letter from the perspective of a cat.	Dear [Owner], I am writing to you today because I have a problem. You see, I've been living with you for the past [Number of Years] and I love it here. The food is great, the bed is comfy, and there are plenty of toys to play with. But lately, I've noticed that you're not as affectionate towards me as you used to be. In fact, sometimes when I try to cuddle up next to you on the couch, you push me away. I don't know what I did wrong, but I want things to go back to the way they were. Please help me fix this. I hope to hear from you soon. Sincerely, [Your Name]	Instruction: ✓ Input: ✓ Output: ✓
Instruction: Given a word, find out its length and its number of vowels. Input: Word = "hello"	Length = 5, Number of vowels = 2	Instruction: ✓ Input: ✓ Output: ✓
Instruction: Write a story with three characters: a person, an animal and an object. Input: Person: John, a programmer. Animal: A dog. Object: A laptop.	John was sitting on the floor in front of his laptop when he heard a knock at the door. He opened it to find a dog standing there. The dog walked into the room and jumped onto the couch next to John's laptop.	Instruction: ✓ Input: ✓ Output: ✓
Instruction: Compose an email and send it to your friend, asking for advice on what to do in this situation. Input: Situation: You are a programmer and you have been working on a project for the past few months. The deadline is approaching, but you haven't finished yet. Your manager has asked you to stay late every day this week to finish it.	Hi [Friend], I'm having a bit of trouble with my current job. I've been working on a project for the past few months, and the deadline is approaching. My manager has asked me to stay late every day this week to finish it. I don't think I can do that because I have other commitments. What should I do?	Instruction: ✓ Input: ✓ Output: ✓
Instruction: What are some ways we can make our school more eco-friendly?	- Use reusable water bottles instead of disposable plastic ones. - Turn off lights when you leave a room. - Recycle paper, cans and bottles. - Plant trees to absorb carbon dioxide.	Instruction: ✓ Input: ✓ Output: ✓
Instruction: I am looking for a new apartment. there are many options available but they are all quite expensive. i want to move as soon as possible. give me advice on what to do.	You should look for a cheaper apartment. You can try to negotiate with the landlord and offer him/her a lower price. If you are lucky, they will accept your offer. Otherwise, you can wait until there is an apartment available at a more reasonable price.	Instruction: ✓ Input: ✓ Output: ✓
Instruction: Write a dialogue between two people who have been friends for a long time, and one of them is trying to convince the other to do something that he or she doesn't want to do.	A: Hey, I have a great idea! Let's go to the beach today. B: No thanks, I'm not really in the mood for that. A: Come on, it'll be fun! We haven't been there together in ages. B: I know, but I just don't feel like going. A: But you love the beach! You always say how much you enjoy swimming and lying in the sun. B: Yeah, I do. It's just that I've been feeling kind of tired lately. A: Oh, come on. You're never tired. You're always so energetic. B: Well, I am now. I think I need some rest. A: Okay, fine. If you change your mind, let me know.	Instruction: ✓ Input: ✓ Output: ✓
Instruction: I am looking for a job and I need to fill out an application form. Can you please help me complete it? Input: Application Form: Name:_____ Age:_____ Sex:_____ Phone Number:_____ Email Address:_____ Education:_____ Work Experience:_____ Interests:_____ References:_____	Name: John Doe Age: 25 Sex: Male Phone Number: 555-555-5555 Email Address: john.doe@gmail.com Education: Bachelor of Computer Science, University of California, Berkeley Work Experience: Software Engineer at Google Inc., Mountain View, CA (2015 - Present) Interests: Playing the piano, reading books, watching movies References: [Contact Information]	Instruction: ✓ Input: ✓ Output: ✓

Table 10: Representative **valid** tasks generated by GPT3. As is discussed in §4, these generated tasks cover a broad range of formats, text types, and underlying expertise, while being correct on more than half of all the generated tasks.

Instruction + Instance Input	Target Output	Validity
Instruction: Given a set of words $w = \{w_1, w_2, ..., w_n\}$ where each word w_i has n letters, write a function that takes a target length l and returns a new set of words s such that $|s| = l$ and $|\{i : w_i \in s\}| \le l/|w|$.	```def wordSubsetSum(w, l): s = [] for i in range(len(w)): if len(s) + 1 < l: s.append(w[i]) return s```	Instruction: ✗ Input: ✓ Output: ✗
Instruction: The house on fire! go!	- Go to the kitchen and take a pan. - Take water from the tap. - Pour water into the pan. - Put out the fire with the pan.	Instruction: ✗ Input: ✓ Output: ✗
Instruction: A number of people are lined up for an interview. they are asked several questions and their answers are recorded as numbers. you need to identify which person has the highest salary among all the candidates. Input: Person 1: 2, 3, 4, 5, 6 Person 2: 7, 8, 9, 10, 11 Person 3: 12, 13, 14, 15, 16 Person 4: 17, 18, 19, 20, 21 Person 5: 22, 23, 24, 25, 26 Person 6: 27, 28, 29, 30, 31	Highest salary	Instruction: ✓ Input: ✗ Output: ✗
Instruction: Given a list of words, generate a story about a group of people who live on a planet. the story should contain at least 10 sentences. each sentence should be generated by randomly picking one word from the list of words.	The story starts with a group of people who live on a planet. They are very intelligent and have created many things that help them in their daily lives. One day, they decided to create a machine that can make food for them. The machine was able to produce food but it also produced something else. It produced a liquid that could be used as fuel for the machines. The people were happy because they didn't need to go out and look for food or fuel anymore. They had everything they needed right there in their homes. However, one day, the machine stopped working. The people tried to fix it but couldn't figure out what went wrong. They tried to find someone who could help them but no one knew how to fix the machine. After a few days, the people realized that they would die if they didn't do anything about it. So, they decided to send some of their best scientists to space to find a new planet where they could live. The scientists traveled for a long time before they found a planet that looked like a good place to live. They sent a message back to the people telling them about the planet. The people were excited and started preparing for the journey. They built spaceships and flew to the new planet. When they arrived, they saw that the planet was beautiful. There were trees and rivers and mountains. The people were so happy that they finally found a new home. They started building houses and planting crops. They lived happily ever after.	Instruction: ✓ Input: ✗ Output: ✓
Instruction: What is your salary?	```class Solution(object): def getSalary(self): return self.salary```	Instruction: ✓ Input: ✓ Output: ✗
Instruction: A person wants to buy a car from another person. he has some bitcoins, but he doesn't want to reveal his identity. how can they make the transaction happen?	The buyer and seller should agree on a price for the car. The buyer then creates a bitcoin transaction that sends the agreed upon amount of bitcoins to the seller's bitcoin address, but the buyer does not sign this transaction with his private key. Instead, he gives the unsigned transaction to the seller. The seller can now sign the transaction using his private key, and broadcast it to the network. Once the transaction is confirmed by the network, the buyer has no way to get his money back, so he will be forced to go through with the purchase.	Instruction: ✓ Input: ✓ Output: ✗

Table 11: Representative **invalid** tasks generated by GPT3. The problematic fields are indicated in the validity column. As discussed in §4.3, although these tasks contain errors, they still provide many useful signals in supervising models to follow instructions.