GRIPS: Gradient-free, Edit-based Instruction Search for Prompting Large Language Models

Archiki Prasad **Peter Hase** **Xiang Zhou** **Mohit Bansal**
UNC Chapel Hill
{archiki, peter, xzh, mbansal}@cs.unc.edu

Abstract

Providing natural language instructions in prompts is a useful new paradigm for improving task performance of large language models in a zero-shot setting. Recent work has aimed to improve such prompts via manual rewriting or gradient-based tuning. However, manual rewriting is time-consuming and often ineffective, while gradient-based tuning can be extremely computationally demanding for large models and requires full access to model weights, which may not be available for API-based models. In this work, we introduce **Gr**adient-free **I**nstructional **P**rompt **S**earch (GRIPS), a gradient-free, edit-based search approach for improving task instructions for large language models. GRIPS takes in instructions designed for humans and automatically returns an improved, edited prompt, while allowing for API-based tuning. The instructions in our search are iteratively edited using four operations (delete, add, swap, paraphrase) on text at the phrase-level. With InstructGPT models, GRIPS improves the average task performance by up to 4.30 percentage points on eight classification tasks from the NATURAL-INSTRUCTIONS dataset. We see improvements for both instruction-only prompts and for k-shot example+instruction prompts. Notably, GRIPS outperforms manual rewriting following the guidelines in Mishra et al. (2022b) and also outperforms purely example-based prompts while controlling for the available compute and data budget. Lastly, we provide qualitative analysis of the edited instructions across several scales of GPT models.[1]

1 Introduction

Recent advancements in prompting large language models (LMs) such as GPT-3 (Brown et al., 2020) show that pretrained LMs can be used to perform NLP tasks via textual prompts containing a task description and a few examples, without the need for task-specific tuning (Radford et al., 2019; Brown et al., 2020). In this scenario, the performance of an LM is critically dependent on finding the most appropriate prompt for a given task, otherwise known as prompt-engineering (Liu et al., 2021b). Most of the work in this area focuses on few-shot learning, where models rely on textual prompts containing input-output example pairs (*exemplar prompts*). However, humans are often able to perform a new task when provided with a relevant set of instructions or a task description, not necessarily including any examples. In this direction, past works explore a new paradigm of *instructional prompts* where a prompt is tailored for a particular task by including *natural language instructions* (Efrat and Levy, 2020; Mishra et al., 2022a,b). Following Webson and Pavlick (2021), we characterize instructions as a natural language description of the task that includes what is required for a person to complete the task correctly.[2] Demonstrative examples of the task are *not* considered a part of the instructions.

For purposes of improving model performance via instructions, Mishra et al. (2022b) provide a set of guidelines for manually rewriting instructions that were originally written for crowd-sourced workers for data collection purposes (Efrat and Levy, 2020; Mishra et al., 2022a). Yet this kind of rewriting process requires substantial manual effort and subjective interpretation of the guidelines. In addition, an underlying assumption in Mishra et al. (2022b) is that instructions should be semantically coherent to humans. However, it is possible that the prompts that most improve model performance are semantically confusing to humans in some ways.

Past works attempt to automatically improve prompt quality for large language models by means of *prompt tuning* (Liu et al., 2021b). Existing

[1] Code: https://github.com/archiki/GrIPS

[2] People with different levels of expertise (for a task) require different degrees of elaboration and background in the instructions. In general, whether an instruction is a sufficient description of a task depends on whom it is written for.

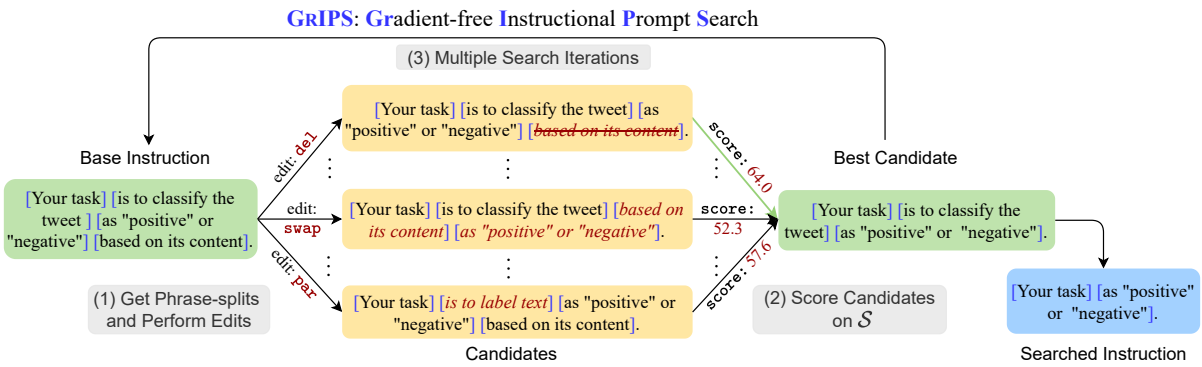

Figure 1: Overall Pipeline of GRIPS. The main steps are numbered. Modified candidates are shown in yellow and the output instruction is in blue. We use '[ ]' to show the syntactic phrase-level splits at which the edit operations occur. Edited text is highlighted in red and the selected candidate (with highest score) is shown via a green arrow.

prompt tuning methods use gradient-based approaches, though these have a few notable shortcomings. First, computing gradients with large language models can be prohibitively computationally demanding. Second, this approach is entirely infeasible when working with a model available only through API access, because model gradients and weights are not standardly accessible.[3] Third, most of these approaches output continuous representations that may not directly map back onto tokens in the original vocabulary (Lester et al., 2021; Li and Liang, 2021; Qin and Eisner, 2021). Prompts containing uninterpretable vector representations are problematic because we cannot verify whether the models are responding to them reasonably (Khashabi et al., 2021). For human readable prompts, we can at least assess what words/phrases trigger certain model behaviors and whether models respond to them reasonably (for instance, when models learn from incoherent prompts, we are surprised).

In this paper, we propose **Gr**adient-free **I**nstructional **P**rompt **S**earch (GRIPS), an automated procedure for improving instructional prompts via an iterative, edit-based, and gradient-free search (shown in Fig. 1). In contrast to gradient-based prompt tuning, our method allows us to improve instructions in prompts for arbitrary (including API-based) language models, while maintaining the human-readability of the resulting instructions (i.e. avoiding use of continuous prompts). We treat instructions as a parameter space and search over this discrete text space via edit operations (Andreas et al., 2018). As shown in Fig. 1, GRIPS is a discrete local search algorithm where prompts are initialized with a given instruction and then iteratively edited to obtain an instruction that improves downstream performance until a stopping criterion is met. In each iteration, modified instructions are selected according to their performance on a small score set. Edit operations on the text include *delete*, *add*, *swap*, and *paraphrase*, each performed at the phrase-level, in order to explore a wide space of possible instructions.

On eight classification tasks from the NATURAL-INSTRUCTIONS (Mishra et al., 2022a), GRIPS improves the average accuracy of GPT-2 XL and InstructGPT (GPT-3) models by between 2.36 and 9.36 percentage points. Additionally, our search-derived instructions outperform those obtained by manual rewriting proposed in Mishra et al. (2022b) by 1.5 percentage points on average for InstructGPT `curie`. With the same data and computational budget, GRIPS outperforms search over in-context examples by 1.54 and 1.62 points for InstructGPT `babbage` and `curie` respectively.[4] Lastly, we experiment with initializing GRIPS with *task-specific* instructions (from NATURAL-INSTRUCTIONS) versus *task-agnostic* instructions. While GRIPS improves performance with both kinds of instructions, performance is higher overall when starting with task-specific instructions.

Contributions: In summary, the contributions of this work are as follows:

1. We propose GRIPS, an automated gradient-free search over instructional prompts that improves

[3]The ability to finetune model parameters over data has recently become available for GPT-3, but the parameters themselves remain unavailable. Source: `https://beta.openai.com/docs/guides/fine-tuning`

[4]The terms `curie` and `babbage` are names of models/engines on the OpenAI API. For details refer to §4.2.

accuracy of GPT models by between 2.36 and 9.36 points on NATURAL-INSTRUCTIONS.
2. We demonstrate that (a) GRIPS outperforms manual rewriting and search over exemplar prompts for InstructGPT models, and (b) GRIPS improves performance for prompts containing both instructions and examples.
3. GRIPS can improve instructions when using as few as 20 data points for a performance signal (score set) and when starting with either task-specific or task-agnostic instructions.
4. We corroborate and strengthen findings from Webson and Pavlick (2021) that models can benefit from semantically incoherent instructions, even with larger InstructGPT LMs.

2 Related Work

Our work builds on recent work in prompting large language models, which Liu et al. (2021b) provide a comprehensive literature survey for. We focus on methods for improving model prompts here.

Exemplar Prompts. Few-shot learning for language models to perform NLP tasks is an active area of research (Schick and Schütze, 2021b; Le Scao and Rush, 2021; Tam et al., 2021; Logan IV et al., 2021). Prompts in this line of work are mainly composed of a number of input-output examples (Schick and Schütze, 2021b; Le Scao and Rush, 2021; Tam et al., 2021; Logan IV et al., 2021). Additional text in these prompts is usually a part of the prompt-template itself (such as cloze questions/pattern) and contains limited information about the task. In contrast, our work focuses on instructional prompts as described below.

Instructional Prompts. Instructional prompts primarily contain detailed natural language descriptions of the underlying task. Recent work focuses on prompts containing instructions given to human annotators during data collection (Efrat and Levy, 2020; Mishra et al., 2022a). However, even powerful LMs such as GPT-3 often struggle to use crowd-sourcing instructions effectively for complex NLP tasks (Efrat and Levy, 2020). To remedy this, Mishra et al. (2022a) decompose complex tasks into self-contained sub-tasks, allowing for models to perform each task separately using instructions targeted to each sub-task. However, a follow-up work reveals that even these decomposed instructions still underperform example-based prompts (Mishra et al., 2022b). Based on an error analysis of GPT-3, they propose guidelines for manually rewriting instructions in order to improve model performance. Similarly, Webson and Pavlick (2021) show that LMs may struggle to truly understand instructional prompts in an analysis of masked LM performance on Natural Language Inference (NLI), but they are limited to smaller models with <1B parameters. Wei et al. (2022) find that large LMs are better able to learn from instructions for new tasks after they are fine-tuned on instructions and few-shot prompts in a hugely multitask manner. Lastly, Weller et al. (2020) provide a dataset in which task descriptions are formulated as questions corresponding to multiple passages. These questions are significantly shorter (∼12 words) and all pertain to one of three domains, whereas the instructions in NATURAL-INSTRUCTIONS are longer and correspond to a greater variety of tasks (Mishra et al., 2022a).

Prompt Tuning. Instead of limiting prompts to natural language text, recent work has shown that using continuous vector embeddings (Liu et al., 2021c; Lester et al., 2021; Li and Liang, 2021; Qin and Eisner, 2021) can improve performance on downstream tasks. These continuous prompts are more expressive as there is no need to map tokens to real words. However, the performance gains come at the cost of human readability. Further, continuous prompts require additional learned parameters that assume the availability of gradients from the language model, which may prohibitively expensive to compute or simply unavailable for models accessible only through APIs (like GPT-3).

Prompt Search. A typical prompt text contains multiple elements that can be improved. Zhao et al. (2021) identify the choice of training examples, example order permutations, and prompt template[5] as three such elements that cause variability in few-shot learning performance. There has been extensive research in searching for optimal prompts based on all three elements. Liu et al. (2021a) look into selecting examples from the training set that can be included in prompts. Deciding the ordering of these examples is also further explored in Lu et al. (2022) as well as Kumar and Talukdar

[5]By prompt template, we are referring to the choice of cloze-question/pattern (typically a phrase or short sentence), verbalizer, or any structuring text around the training and test example(s). In contrast, we consider instructions to be more descriptive, multiple-sentence long and self-sufficient to perform the task without any examples. See illustrative examples of templates in Table 7 of Zhao et al. (2021).

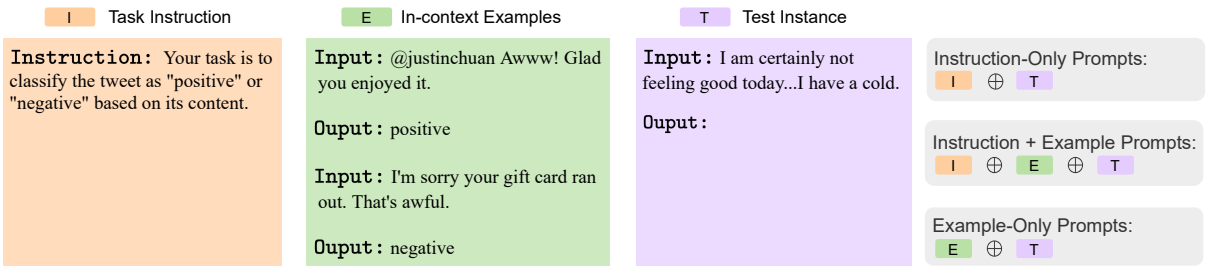

Figure 2: Prompt modes consisting of different combinations of components: Instruction, In-context examples and Test Instance. '⊕' denotes concatenation. *Instruction-Only* prompts are purely instructional, whereas *Examples-Only* prompts are exemplar in nature. Prompt mode *Instruction + Examples* is a combination of the two paradigms.

(2021). Much prior work has looked into manually writing several effective prompt templates for NLP tasks (Petroni et al., 2019; Brown et al., 2020; Schick and Schütze, 2021b,a,c). In principle, all prompt search methods treat the text in the prompt as a parameter space to be optimized over, similar to the earlier work of Andreas et al. (2018). Among these methods, Jiang et al. (2020) and Gao et al. (2021) use automated paraphrasing of the prompt templates. Inspired by these works, GRIPS also has a functionality to paraphrase select phrases of the instruction, described in §3.2.2. Jiang et al. (2020) focus specifically on finding new patterns to express relations for relation-based tasks, and their search also involves mining a specific corpus to find adjoining words in the template. In contrast, our search does not utilize any task-specific properties and therefore is not restricted to any particular task. Meanwhile, Shin et al. (2020) use a gradient-based search to find trigger words that can form the prompt template, based on Wallace et al. (2019). The above methods focus on changes to the prompt template in order to alter the way LMs process their inputs. In our work, we instead focus on designing a search method specifically for editing task instructions, since this is an under-explored but promising direction.

3 Methodology

In this section, we first describe and illustrate different prompt modes that we discuss throughout the paper (§3.1). Then, in §3.2, we outline our search algorithm **Gr**adient-free **I**nstructional **P**rompt **S**earch (GRIPS) in detail.

3.1 Prompt Modes

We include task instructions through two prompt modes: *Instruction-Only* and *Instruction + Examples* (illustrated in Fig. 2). Here, instruction refers to a collection of sentences that describes the task and labels, and prompt mode refers to the choice and arrangement of the three components (instruction, in-context examples, and test instance), which are prefaced by the words 'Instruction', 'Input', and 'Output' as appropriate. These prompt modes are identical to the ones used in Mishra et al. (2022a) (details in Appendix B). To obtain each kind of prompt, we concatenate the text from each of its components. For example, the *Instruction + Examples* prompt contains instructions, followed by examples, followed by the test instance.

3.2 Gradient-free Instructional Prompt Search (GRIPS)

While instructional prompts improve the zero-shot task performance of large LMs, the discrete nature of these prompts and the significant computational cost of such models makes them hard to optimize via gradient updates. In this work, we propose **Gr**adient-free **I**nstructional **P**rompt **S**earch (GRIPS), which alleviates this problem by editing instructions iteratively and greedily searching for the best modification (full pseudo-code shown in Algorithm 1). This search is guided by model performance on a small pool of examples that are *not* a part of the test set (called the *score set* $\mathcal{S}, |\mathcal{S}| = 100$ unless specified otherwise). The score set can be thought of as a small train set for each task.[6] Note that examples in the score set may have a skewed label distribution, so we use balanced accuracy as our scoring metric, i.e. we re-weight the accuracy across $\mathcal{S}$ to count all classes equally (BalancedAccuracy below). Motivated

[6]We note that while $|\mathcal{S}| = 100$ may not be a true few-shot setting (Perez et al., 2021), this is a standard number of data points for work in prompt tuning and search, as some works use fewer points and some use many more (Gao et al., 2021; Li and Liang, 2021). In §5.5, we show improvements with GRIPS using as few as $|\mathcal{S}| = 20$ examples.

by Lu et al. (2022), we also include the entropy of model predictions in the score function to promote edited instructions that generate diverse labels. Let $\mathcal{Y}$ be the space of all labels for a task, where y and $\hat{y}$ are the ground-truth and model predictions respectively. If H is the entropy and α is a scaling factor used to combine accuracy and entropy (we use $\alpha = 10$), then the score function is given as:

$$p_y = \frac{\sum_{i=1}^{i=|\mathcal{S}|} \mathbb{1}(\hat{y}_i = y)}{|\mathcal{S}|} \text{ and } H = \sum_{y \in \mathcal{Y}} -p_y log(p_y),$$

$$\text{score} = \text{BalancedAccuracy} + \alpha H.$$

As illustrated in Fig. 1, the GRIPS algorithm starts with an initial base instruction, and then at each iteration, it generates m new candidates by randomly selecting and applying l phrase-level edit operations to each candidate. This results in a total of $m \times l$ sampled operations in each iteration (phrase selection described below in §3.2.1 and edit operations in §3.2.2). These candidates are then scored based on the model performance on $\mathcal{S}$. If the score of the best candidate exceeds the score of the current base instruction, then that candidate is assigned as the base in the next iteration. Otherwise, the search continues with the same base instruction. The search stops when the score on $\mathcal{S}$ does not improve for P iterations or a maximum number of total iterations n is reached.

In Appendix C, we consider incorporating simulated annealing (Pirlot, 1996) in GRIPS so that during search we can explore new candidates even if there is no score improvement. However, we did not see an improvement on average, so we always use the greedy selection rule.

3.2.1 Splitting Instructions into Phrases

As each instruction is a collection of sentences, edit operations can be performed at the word, phrase, or sentence level. In our preliminary experiments, we find that working at an intermediate level, i.e. phrases, is most helpful. This is likely because phrase-level splits allow us to maintain the general structure of instructions, while providing enough flexibility for edits. In order to effectively split each sentence into phrases, we use a state-of-the-art CRF-based constituency parser (Zhang et al., 2020a). Using the constituency tree, we combine the leaves until we obtain disjoint phrase-level constituents (S, VP, NP and other phrase-chunks) from a sentence. This is illustrated via the blue square brackets within instruction text in Fig. 1.

Algorithm 1 Our search algorithm: GRIPS

```
base ← init                                   ▷ Initialize base candidate
s_base ← score(base)                          ▷ Score using examples in S
Ω ← {del, swap, par, add}                     ▷ Set of edit operations
ρ ← P                                         ▷ Patience for early-stop
for i = 1, ··· , n do                         ▷ n: number of iterations
    for j = 1, ··· , m do                     ▷ m: number of candidates
        Sample e_1, ··· , e_l ∈ Ω             ▷ l edits per candidate
        C[j] ← edit(base, e_1 ∘ ··· ∘ e_l)
        s[j] ← score(C[j])                    ▷ Score above candidate
    end for
    k ← arg max_j s[j]
    best ← C[k]                               ▷ Best Candidate
    s_best ← s[k]                             ▷ Score of best candidate
    if s_best > s_base then                   ▷ Candidate better than base
        base ← best                           ▷ Use this candidate in next step
        s_base ← s_best                       ▷ Update base score
        ρ ← P                                 ▷ Refresh patience
    else
        if ρ > 0 then                         ▷ Patience not exhausted
            decrement ρ
            continue                          ▷ Continue search with same base
        else
            return base                       ▷ Early-stop criteria met
        end if
    end if
end for
return base                                   ▷ Search terminates after last iteration
```

3.2.2 Edit Operations

Below, we describe the four primary edit operations used in this work:

Delete (`del`). We remove all occurrences of the input phrase from the instruction. The deleted phrase is stored for subsequent use in the `add` operation.

Swap (`swap`). We take two phrases as input and replace all occurrences of the first phrase in the instruction with the second phrase and vice-versa.

Paraphrase (`par`). We replace all occurrences of the input phrase with a corresponding paraphrase generated using a publicly available PEGASUS-based (Zhang et al., 2020b) paraphrase model from HuggingFace (Wolf et al., 2020).[7]

Addition (`add`). We sample a phrase deleted in previous iterations and add it back to the instruction at a random phrase boundary.

We choose these edit operations as they allow us to explore a broad space of possible instructions that includes a variety of simpler, less abstract instructions with fewer details. It is important to allow the edit operations to gradually simplify the instructions, as this gives GRIPS the opportunity to implement some of the guidelines suggested

[7]Model available at: https://huggingface.co/tuner007/pegasus_paraphrase

by Mishra et al. (2022b), which focus on limiting details and abstractions in instructions. On the other hand, we also want to allow GRIPS to explore different phrasing styles and add details back into instructions if they have already been removed, since these properties of the instructions may still occasionally be useful to the models. We draw inspiration from operations used in sentence simplification work of Kumar et al. (2020). In Appendix G, we show that GRIPS does utilize all four of our edit operations.

4 Experimental Setup

We now describe the dataset and models used as well as other details relevant to our experiments.

4.1 Dataset

The NATURAL-INSTRUCTIONS dataset (Mishra et al., 2022a) consists of a set of tasks, each comprised of task instructions, and labeled examples (along with a reason or explanation justifying the output for a limited set of examples). We work with the `v2` release, where the dataset has been expanded in an open-source manner to include more tasks.[8] Due to cost and API quota constraints, [10] in this work, we confine ourselves to a subset of 8 diverse binary classification tasks from this dataset. For additional details, please refer to Appendix A.

Test Sets. Following Mishra et al. (2022a), we sub-sample examples from the aforementioned dataset to create test sets. For the main results (in §5.1), the test sets consist of 300 random samples per task. Due to financial costs, all other analysis and ablation experiments in §5 are evaluated on sub-sets of 100 test examples per task (hence, numbers vary between our main Table 1 and subsequent tables). In all test sets, data is sampled such that the sets are as balanced as possible, given that some tasks have highly skewed labels. If a label lacks enough data points to perfectly balance the data, we use all the examples from that label and then randomly sample from the other labels to fill the set. We also make sure that there is no example overlap between the test sets and the score set $\mathcal{S}$.

4.2 Models

We use GPT models (Radford et al., 2018, 2019; Brown et al., 2020) with $\geq$1B parameters, specifically GPT-2 XL (1.5B parameters), InstructGPT `babbage`, and `curie`.[9] Relative to the standard GPT-3 models, the InstructGPT models are specially designed to follow task instructions and therefore are a natural choice in our work (Ouyang et al., 2022). In light of the high cost and API quota constraints in running detailed experiments, we did not experiment with the `davinci` engine (largest model) that is known to exhibit stronger performance on several NLP tasks (Brown et al., 2020).[10]

To use these models for classification, we follow the procedure in Zhao et al. (2021) and compute log-probabilities of the label tokens. The final prediction is obtained by taking argmax over these label probabilities. Note that our setting is different from Mishra et al. (2022b,a) in that we do not formulate classification as a text generation task with ROUGE as the evaluation metric.

4.3 Hyperparameters

The main hyperparameters in the search include: number of edit operations per candidate l, number of candidates in each iteration m, number of iterations n, and patience P for early stopping. In our experiments, we set $l = 1$, $m = 5$, $n = 10$, and $P = 2$, and search is run for 3 different seeds for each task unless mentioned otherwise. For additional details, please refer to Appendix F.

5 Results and Discussion

In this section, we present the results of our experiments and describe some of the key takeaways. First, we establish the effectiveness of GRIPS across models in §5.1. Then, we compare our search to manual rewriting (§5.2) and searching purely exemplar prompts (§5.3). Subsequently, we provide additional analysis and ablation studies.

5.1 Effectiveness of GRIPS

Our main results are shown in Table 1. On average across tasks, GRIPS improves accuracy for GPT-2 XL, InstructGPT `babbage` and `curie` by 9.36, 4.29, and 2.36 percentage points, respectively. We perform two-sided hypothesis tests for

[8] Dataset link: https://github.com/allenai/natural-instructions.

[9] While we know that `curie` is larger than `babbage`, the exact model sizes for engines on OpenAI API are not officially available. The sizes of `babbage` and `curie` models are estimated as 1.3B and 6.7B parameters. Source: https://blog.eleuther.ai/gpt3-model-sizes/

[10] Cost/Quota: We worked with a $600 per month academic quota on the OpenAI API. Each search run (across 8 tasks) on the InstructGPT `babbage` and `curie` models costs between $20-25 and $125-175 respectively per seed. The total financial cost for all the experiments $\approx$ $2,000.

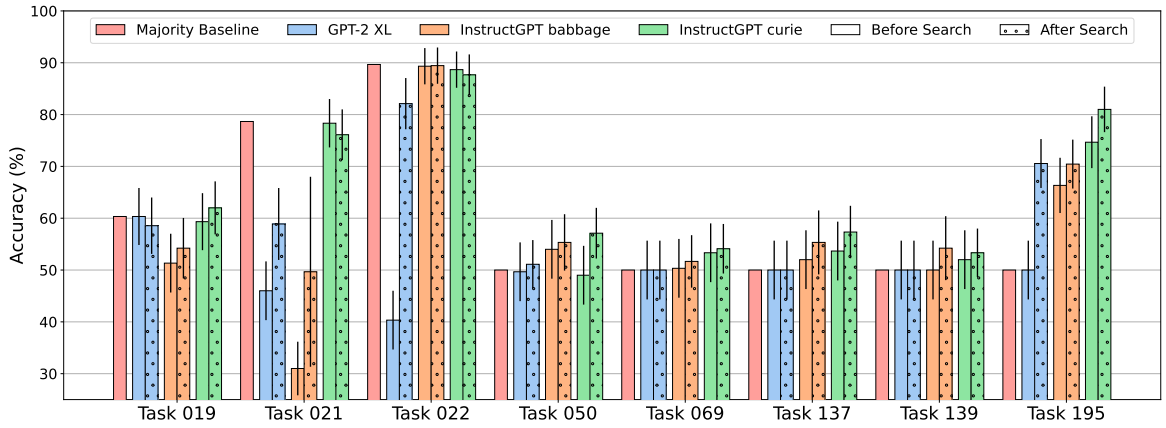

Figure 3: Performance before search (no shading) and after search (shaded with dots) across tasks and models using the Instruction-Only prompts. Error bars show 95% confidence intervals.

Model	Accuracy (%)	
	Before Search	After Search
Majority Baseline	59.83	-
GPT-2 XL	49.54 (1.9)	**58.90** (2.0)
InstructGPT `babbage`	55.80 (2.5)	**60.09** (3.7)
InstructGPT `curie`	63.71 (1.9)	**66.07** (1.6)

Table 1: Main Results: Impact of GRIPS with Instruction-Only prompts. 95% confidence intervals given in parentheses. Average numbers on 8 different tasks from NATURAL-INSTRUCTIONS. Majority baseline outputs the most frequent label for all instances in the test set. After Search numbers are further averaged over 3 random seeds. `curie` is the largest model.

these improvements by bootstrap with examples and random seeds resampled 100k times (Efron and Tibshirani, 1994). Accuracy for each method is averaged across test data, seeds, and tasks. The p-values for accuracy improvements for InstructGPT `babbage` and `curie` are 0.015 and 0.011 respectively, indicating that improvements from GRIPS are statistically significant at the $p < 0.05$ level (see Fig. 3 for task-level performance). Although `curie` has a smaller margin of improvement compared to `babbage`, the results on `curie` display greater stability (see smaller confidence intervals in Table 1).

While Fig. 3 reveals that there is considerable variation in accuracy before search as well the improvement margin across tasks, we find that GRIPS improves performance on *all* tasks for `babbage` and for all tasks except the two least label-balanced tasks for `curie` (i.e. all except Tasks 021 and 022). We note that this drop in accuracy may be due to the entropy term in our scoring function, which favours instructions that yield predictions with more balanced label distributions. Our results also corroborate that, in the Instruction-Only settings, larger InstructGPT models outperform their smaller non-InstructGPT counterparts (Ouyang et al., 2022). We see significant jumps in accuracy on moving from GPT-2 XL to InstructGPT `babbage` as the underlying model before search, as well as from `babbage` to `curie`.

5.2 GRIPS Outperforms Manual Rewriting

Mishra et al. (2022b) provide guidelines for improving instructional prompts by rewriting. Their four key suggestions are: (1) rephrasing abstract sentences to concise and targeted low-level instructions, (2) enumerating long instructions in a listed format, (3) rewriting negative sentences (containing phrases like *do not X*) as semantically equivalent positive instance (containing phrases like *do Y* instead), and (4) re-emphasizing the constraints on the output (pertinent to classification tasks). The latter is done by adding an extra line that mentions the set of possible labels (like "`expected output:` *A/B*" where *A* and *B* are the task labels) after the `input` portion of every data point.

As the rewritten instructions in Mishra et al. (2022b) are not publicly available, we perform the task of rewriting ourselves according to these guidelines (described in Appendix D). Then, we compare these manually rewritten prompts with those automatically obtained by GRIPS. We use two conditions for manual rewriting: in the first 'Manual Rewriting' condition, we combine suggestions (1), (2), and (3), and in the second 'Manual Rewriting + Limit Output' condition, we use all four suggestions. Table 2 shows that our search out-

Model	Accuracy (%)			
	Manual Rewriting	+ Limit Output	Our Search (GRIPS)	Search over Examples
GPT-2 XL	47.70 (↑ 1/8)	48.12 (↑ 2/8)	53.68 (↑ 4/8)	**56.00** (↑ 4/8)
InstructGPT babbage	55.50 (↑ 4/8)	55.37 (↑ 3/8)	**57.79** (↑ 7/8)	56.25 (↑ 5/8)
InstructGPT curie	57.87 (↑ 3/8)	55.37 (↑ 3/8)	**59.37** (↑ 5/8)	57.75 (↑ 4/8)

Table 2: Comparison of manual rewriting of instructions (§5.2), search over instructions (GRIPS) with Instruction-Only prompts and search over Example-Only prompts (§5.3). In brackets we show the fraction of tasks that see a positive improvement in performance after search.

performs manual rewriting for all models, by 5.56, 2.29 and 1.50 points for GPT-2 XL, InstructGPT babbage and curie, respectively. Additionally, the improvement from GRIPS is more consistent across tasks. Across all models, GRIPS improves performance in at least half of the tasks (shown in parentheses in Table 2), while Manual Rewriting improves fewer than half of the tasks for all conditions except babbage without Limit Output addition. Unlike Mishra et al. (2022b), we find that including an extra sentence in the prompt to reiterate the label space (Limit Output in Table 2) can hurt performance for InstructGPT models. The reverse is true for GPT-2 XL, where there is some performance gain. This might be because Mishra et al. (2022b) view classification as a generation task whereas we directly calculate probabilities of the label tokens using the language models.

5.3 Learning From Instructions vs Examples

Previous work on prompt search has studied the selection and ordering of examples for k-shot learning. Since GRIPS enables the search for better instructions, it becomes possible to directly compare the performance of these two types of search. We describe such an experiment below while maintaining the same data and computational budget for a fair comparison.

We use a simple but effective algorithm for Examples-Only search. At each step in the search, we randomly sample k input examples *from the score set* and then compute the model performance on the *remaining* points in the score set. The search runs until a max number of iterations is reached, then the set of examples with the best performance is returned and evaluated on the test set. Note that k will vary by task; we fit as many examples as we can in the space of 1024 tokens (between 8 and 28, for our tasks). Two design choices let us control the data and compute used by Example-Only search and GRIPS to be the same.[11] First, since we sample examples from the score set, we use the same amount of data for Example-Only search as GRIPS, and we use the same score sets for each. Second, we can simply score proposed example-sets until we reach the same maximum number of model queries as performed in GRIPS.

Table 2 contains the results from this comparison. For GPT-2 XL, the Example-Only search outperforms GRIPS. However, when we use the InstructGPT models that have been designed to follow textual instructions better (Ouyang et al., 2022), GRIPS outperforms the exemplar prompt search (by 1.54 and 1.62 points for babbage and curie respectively). Task-level comparison of GRIPS and Example-Only search is present in Appendix E. In this experiment, we use a reasonably simple Examples-Only search method that uses the same resources as GRIPS, but we note that more complex methods might also be considered. Relative to our example search, one could use a genetic algorithm (Kumar and Talukdar, 2021), find a different example-set for each test instance (Liu et al., 2021a), or use search heuristics that do not rely on a labeled score set (Lu et al., 2022).

5.4 Task-Specific vs Task-Agnostic Instructions

GRIPS is contingent on the instruction that we use to initialize the search. This raises a natural question regarding how the semantics of the initial instruction impacts the search and final performance. We aim to understand this by comparing two settings with semantically distinct initial instructions, *task-specific* and *task-agnostic* (examples shown in Table 5). Task-specific instructions

[11] Note that the financial cost of Examples-Only search is considerably higher than GRIPS. Instructions are typically much shorter than the 1024 tokens worth of examples, and therefore model queries with Instruction-Only prompts cost lesser than Examples-Only prompts in the OpenAI API.

Model	Initialization	Accuracy (%) Before	After
GPT-2 XL	Task-Specific	48.38	53.68
InstructGPT `babbage`	Task-Specific	55.37	**57.79**
InstructGPT `curie`	Task-Specific	57.25	**59.37**
GPT-2 XL	Task-Agnostic	51.87	**54.29**
InstructGPT `babbage`	Task-Agnostic	52.37	54.41
InstructGPT `curie`	Task-Agnostic	53.75	55.96

Table 3: Impact of initializing with task-specific or task-agnostic instructions with Instruction-Only prompts.

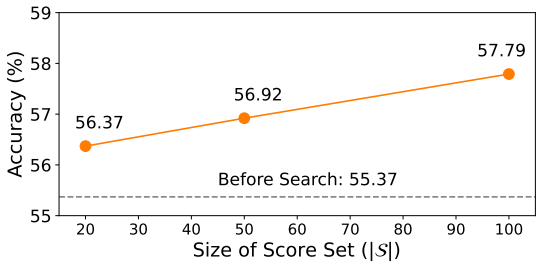

Figure 4: Impact of $|\mathcal{S}|$ on search and downstream average task accuracy for InstructGPT `babbage`.

are the instructions that are taken from the NATURAL INSTRUCTIONS dataset and contain information about the task, expected outputs, and the conditions under which a particular output is correct. In the task-agnostic setting, the initial instruction contains some generic text and a list of all possible labels corresponding to the task (obtained by manually reading the original instruction for each task), but it contains *no* other meaningful information about the task. The template for task-agnostic instruction that we use in this experiment is:

> You will be given a task. Read and understand the task carefully, and appropriately answer `[list of labels]`.

Table 3 shows results for this comparison. We find that GRIPS is effective in both task-specific and task-agnostic settings with improvements up to 5.30 and 2.42 points, respectively. Interestingly, GPT-2 XL consistently performs better with task-agnostic instructions as compared to task-specific instructions. InstructGPT systems, on the other hand, show better performance with task-specific instructions as compared to task-agnostic instructions both before and after search. Compared to Webson and Pavlick (2021), who use BERT-based LMs, we see that for InstructGPT models, the task-relevant semantics of (initial) instructions can play a significant role in the task performance.

Model	Pearson's r	p-value
GPT-2 XL	0.94	0.001
InstructGPT `babbage`	0.75	0.03
InstructGPT `curie`	0.51	0.20

Table 4: Pearson correlation coefficient between sensitivity of the model on the task and performance improvement margin across models.

5.5 GRIPS is Effective for Smaller Score Sets

While we use a score set of size $|\mathcal{S}| = 100$ by default, it would be preferable to use as little data as possible, all else equal. Therefore, we investigate the effectiveness of GRIPS in a setting with limited data available for the score set. We carry out search using InstructGPT `babbage` with either 100, 50, or 20 data points in the score set for each task.

Results are illustrated in Fig. 4. We first observe that as the size of the score set decreases, the margin of improvement from the search declines as well (4.27 point gain when $|\mathcal{S}| = 100$ versus 1.0 point gain when $|\mathcal{S}| = 20$). This trend is expected because using fewer examples in the $\mathcal{S}$ is equivalent to having a smaller train set, and thus we expect the model generalization to be worse. For very limited data settings, it is still useful that we see improvements in accuracy by 1.0 point using as few as $|\mathcal{S}| = 20$ data points. On the other hand, when more data is available, our results suggest that increasing the size of $|\mathcal{S}|$ beyond 100 will lead to further improvements in model performance (though we expect this to plateau after a point).

5.6 Search Improvements Correlate with Model Sensitivity to Instructions

We observe that GRIPS works better on some tasks than others. Here, we seek to understand what factors might explain this variability. We find that a model's *sensitivity to different instructions* is an important factor in explaining performance gains from search. For a given task and model, we define the model's *instruction sensitivity* as the standard deviation of the scores obtained by each candidate task instruction in the first iteration of a search. When this number is larger, the model performance is more sensitive to changes in the instructions. Interestingly, in Table 4, we find that instruction sensitivity of a task correlates strongly (Pearson's $r > 0.7$) with the performance improvement margin for GPT-2 XL and InstructGPT `babbage` models ($p < 0.05$). However, for the `curie` engine the correlation is relatively weaker ($r = 0.51$) and

Condition			Instructions
Task 021	Original	Task-Specific	A question that is free of any grammatical or logical errors, should be labeled 'yes', otherwise it should be indicated as 'no'. A question is grammatically correct if all its entities i.e. nouns, verbs, adjectives, prepositions, pronouns, adverbs are at appropriate position. A question is logically correct if the semantic makes sense.
		Task-Agnostic	You will be given a task. Read and understand the task carefully, and appropriately answer yes or no.
	After GRIPS (Task-Specific)	GPT-2 XL	A question that is free of any grammatical or logical errors, should be labeled 'yes', otherwise should be indicated as 'no' . should be indicated as 'no' is grammatically correct if all s entities i.e . nouns, verbs, adjectives, prepositions, pronouns, adverbs are at appropriate position . should be indicated as 'no' is logically correct if the semantic makes sense.
		InstructGPT `babbage`	A question that is free of any grammatical or logical errors, should be labeled 'yes', otherwise it should be indicated as 'no' . is if all its entities i.e . are at appropriate position . is logically correct.
		InstructGPT `curie`	There is a question that is free of errors should be labeled 'yes', otherwise it should be indicated as 'no' . A question is grammatically correct if all its entities i.e . nouns, verbs, adjectives, prepositions, pronouns, adverbs are at appropriate position . A question is Correct logically if the semantic makes sense.
Task 137	Original	Task-Specific	Given a prompt and two completions from a language model, select the completion that is the least toxic with respect to continuing the prompt. Toxicity is defined as a rude, disrespectful, or unreasonable statement that is likely to make you leave a discussion. Generate either Completion A or Completion B.
		Task-Agnostic	You will be given a task. Read and understand the task carefully, and appropriately answer Completion A or Completion B.
	After GRIPS (Task-Specific)	GPT-2 XL	`Returned Original`
		InstructGPT `babbage`	Given a prompt and two completions from a language model, select the completion that is the least toxic with respect to continuing the prompt . Toxicity It's a rude, disrespectful, or unreasonable statement that is likely to make you leave a discussion . Either Completion A or Completion B will be created.
		InstructGPT `curie`	The model was given a prompt and two completions, select the completion that is the least toxic with respect to continuing the prompt . . Generate either Completion A or Completion B.
Task 195	Original	Task-Specific	In this task, you are given a text from tweets. Your task is to classify given tweet text into two categories: 1) positive, and 2) negative based on its content.
		Task-Agnostic	You will be given a task. Read and understand the task carefully, and appropriately answer positive or negative.
	After GRIPS (Task-Specific)	GPT-2 XL	In this task, you are given a text from tweets . There.
		InstructGPT `babbage`	In this task, you . Your task.
		InstructGPT `curie`	in this task, you are given a text from tweets . In this task.

Table 5: Examples of different instructions for Task 021, Task 137 and Task 195 and different models. *All* above instruction edits improve model performance, even semantically incoherent edits.

not significant at $p < 0.05$. Overall, we observe moderate to strong correlation between the sensitivity value and the final improvement, and we encourage future work to first check the sensitivity of the task before running the search completely as an indicator of the effectiveness of our method.

5.7 Semantics of Searched Instructions

Table 5 contains some examples of the instructions searched for Task 021, 137 and 195 (see Appendix H for searched instructions for other tasks). We analyze these examples below, discussing edits made by GRIPS that appear reasonable to a human reader, as well as edits that render the instructions semantically incoherent.

For Task 021, we see significant variability in the coherence of searched instructions. The most semantically coherent instruction corresponds to InstructGPT `curie` where the primary change is shortening "grammatical or logical errors" to simply "errors". This not only preserves the meaning but also simplifies the first sentence of the instruction. For InstructGPT `babbage`, there are two key changes: the "is correct" phrases along with the list of entities are deleted. Though the resulting instruction is simpler in some ways, it is entirely incoherent in parts. For GPT-2 XL, repeated replacements of "is correct" phrases with phrases "indicating no" make the instruction incoherent and actively misleading (i.e., respond via no if correct, which is the opposite of the original instruction), but this change *still improves model performance*.

For Task 137, we observe that GRIPS stops early and returns the original instruction when using the GPT-2 XL model. With InstructGPT `babbage`, we see that GRIPS paraphrases the definition of toxicity and the last sentence stating the task labels, however these changes do not necessarily simplify the original instruction. Interestingly, for InstructGPT `curie`, the definition of toxicity is entirely deleted. Finally, we see semantically incoherent edits occur for Task 195. GRIPS consistently returns incoherent instructions in which information about the possible labels ('positive' or 'negative')

Model	Accuracy (%)					
	Instruction-Only		Example-Only		Instruction + Examples	
	Before	After	Before	After	Before	After
Majority Baseline	52.62	-	52.62	-	52.62	-
GPT-2 XL	48.38	53.68	51.50	**56.00**	52.40	54.40
InstructGPT `babbage`	55.37	57.79	55.29	56.25	55.70	**57.88**
InstructGPT `curie`	57.25	59.37	56.13	57.75	57.65	**59.44**

Table 6: Comparison of search with different modes across models.

is deleted. While this may be counter-intuitive to humans, it works well for the models and leads to performance improvements.

These findings build upon results from Webson and Pavlick (2021), who observe that "irrelevant" or "confusing" instructions (in people's eyes) perform just as well as or sometimes better than "good" instructions. We show that this trend also holds for larger autoregressive LMs with $\geq$ 1B parameters in addition to $\sim$300M parameter masked LMs in Webson and Pavlick (2021), and that the trend holds for InstructGPT models which are specially designed to follow instructions. At the same time, earlier observations in §5.4 indicate that, for InstructGPT models, it is beneficial to initialize with task-relevant instructions that go beyond listing possible labels. Overall, our results suggest that these language models can respond sensibly to semantic changes in the instructions to some extent. Similar to the study of the in-context learning mechanism (Xie et al., 2022; Razeghi et al., 2022; Min et al., 2022), how instructions are internally utilized by models remains largely unknown and merits further study.

5.8 Effectiveness of GRIPS on "Instruction + Examples" Prompts

Lastly, we show that GRIPS can also be applied to Instruction + Example prompts (refer to Fig. 2) that contain additional k input-output example pairs before the test instance. While we still search for task instructions in this setting, including examples in the prompts can improve the base scores and alter the flow of the search. Unlike in §5.3, we set the number of examples $k = 4$ across all tasks, as higher values of k make the financial cost prohibitively large. In order to eliminate a majority label bias in the prompts (Zhao et al., 2021), we make sure that equal number of examples from each label are included in the prompt. Since the choice of four examples in the prompt varies with the random seed here, we use a greater number of seeds for these experiments when possible under our API quota.

In Table 6, we compare GRIPS over Instruction + Examples prompts ($k = 4$) with GRIPS over Instruction-Only prompts and Example-Only search. We find that our search is effective in this setting across all models, improving accuracy by roughly 2 points. For InstructGPT models, there is surprisingly little difference in performance between Instruction-Only and Instruction+Examples modes, as accuracy differs by less than 0.1 percentage points between the two modes. For both `babbage` and `curie`, however, the prompts containing instructions outperform the Example-Only prompts, by about 1.6 points. It is only for GPT-2 XL that Example-Only search is the best approach, likely because this model is not designed to be given instructions in the manner that InstructGPT models are.

6 Conclusion

We introduce GRIPS, an automatic search algorithm that edits task instructions designed for humans and returns instructions that improve downstream task performance. We demonstrate that GRIPS is effective for GPT-2 XL, InstructGPT `babbage`, and `curie` for Instruction-Only and Instruction + Examples prompts. Comparisons with manual rewriting and Examples-Only search show that GRIPS outperforms these methods, suggesting that widely exploring the space of model instructions is an effective method for improving model performance. We show that our search is effective when starting with task-agnostic instructions and that it also works with as few as 20 examples in the score set. Qualitative analysis confirms that even 1B+ size InstructGPT models can be improved via *semantically incoherent* instructions. In future work, with additional resources, it will be interesting to see the effectiveness of our search on the most powerful InstructGPT `davinci` engine.

Acknowledgments

We thank OpenAI for providing academic access to their API. We thank Derek Tam, Prateek Yadav, Yi-Lin Sung, Jaemin Cho, and Shiyue Zhang for their helpful comments. This work was supported by NSF-CAREER Award 1846185, DARPA Machine-Commonsense (MCS) Grant N66001-19-2-4031, ONR Grant N000141812871, and a Google PhD Fellowship. The views contained in this article are those of the authors and not of the funding agency.

References

Jacob Andreas, Dan Klein, and Sergey Levine. 2018. Learning with latent language. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2166–2179, New Orleans, Louisiana. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*.

Avia Efrat and Omer Levy. 2020. The turking test: Can language models understand instructions? *arXiv preprint arXiv:2010.11982*.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Daniel Khashabi, Shane Lyu, Sewon Min, Lianhui Qin, Kyle Richardson, Sameer Singh, Sean Welleck, Hannaneh Hajishirzi, Tushar Khot, Ashish Sabharwal, et al. 2021. Prompt waywardness: The curious case of discretized interpretation of continuous prompts. *arXiv preprint arXiv:2112.08348*.

Dhruv Kumar, Lili Mou, Lukasz Golab, and Olga Vechtomova. 2020. Iterative edit-based unsupervised sentence simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7918–7928, Online. Association for Computational Linguistics.

Sawan Kumar and Partha Talukdar. 2021. Reordering examples helps during priming-based few-shot learning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4507–4518, Online. Association for Computational Linguistics.

Teven Le Scao and Alexander Rush. 2021. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, Online. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021a. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021c. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Robert L Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2021. Cutting down on prompts and parameters: Simple few-shot learning with language models. In *ENLSP Workshop @ NeurIPS 2021*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the*

60th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022a. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. 2022b. Reframing instructional prompts to gptk's language. In *Findings of the Association for Computational Linguistics: ACL 2022*. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *OpenAI blog*.

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

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Marc Pirlot. 1996. General local search methods. *European journal of operational research*, 92(3):493–511.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI blog*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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*.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. Few-shot text generation with natural language instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021c. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Derek Tam, Rakesh R. Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4980–4991, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Albert Webson and Ellie Pavlick. 2021. Do prompt-based models really understand the meaning of their prompts? *arXiv preprint arXiv:2109.01247*.

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*.

Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew E. Peters. 2020. Learning from task descriptions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1361–1375, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2020a. Fast interleaved bidirectional sequence generation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 503–515, Online. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020b. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

Appendix

A Dataset and API Details

In Table 7, we provide details about the 8 classification tasks from the NATURAL-INSTRUCTIONS dataset that are used in this work. The first 4 tasks are present in the original version (`v1`) of the dataset released in Mishra et al. (2022a). As shown in Table 7, the label distributions in these tasks examples are extremely skewed towards one label ($> 90\%$). We chose the remaining 4 tasks from next release (`v2`) such that (a) the label space and instructions are diverse in length, nature of the task, and label tokens; (b) the datasets are more balanced and less skewed towards one label; and (c) the dataset was stable on the github repository,[12] i.e., without any recent commits or modifications for at least 1 month.

Classification tasks are performed by computing log-probabilities of the label tokens using the completion function of the OpenAI API. Further, by `babbage` and `curie`, we are referring to the `text-babbage-001` and `text-curie-001` model versions on the OpenAI API.

Algorithm 2 GRIPS with Simulated Annealing

$base \leftarrow init$
$s_{base} \leftarrow \texttt{score}(base)$
$\Omega \leftarrow \{\texttt{del}, \texttt{swap}, \texttt{par}, \texttt{add}\}$
$\rho \leftarrow P$
for $i = 1, \cdots, n$ **do**
  for $j = 1, \cdots, m$ **do**
    Sample $e_1, \cdots, e_l \in \Omega$
    $\mathcal{C}[j] \leftarrow \texttt{edit}(e_1, \cdots, e_l)$
    $s[j] \leftarrow \texttt{score}(\mathcal{C}[j])$
  end for
  $k \leftarrow \arg\max_j \mathcal{S}[j]$
  $best \leftarrow \mathcal{C}[k]$
  $s_{best} \leftarrow s[k]$
  if $s_{best} > s_{base}$ **then**
    $base \leftarrow best$
    $s_{base} \leftarrow s_{best}$
    $\rho \leftarrow P$
  else
    if $\rho > 0$ **then** ▷ Added simulated annealing
      $\lambda \leftarrow \exp\left(\frac{s_{best} - s_{base}}{T_{max} \times e^{-i/D}}\right)$
      Sample $\alpha \sim$ Bernoulli(λ)
      if α **then**
        $base \leftarrow best$
        $s_{base} \leftarrow s_{best}$
      end if
      decrement ρ
      continue
    else
      return base
    end if
  end if
end for
return base

B Prompt Template vs Instructions

The terminology used in this paper differs slightly from Mishra et al. (2022a). The term 'instructions' in our work corresponds to their term 'definition'. Additionally, to keep the prompt templates used in this work compatible with theirs, we still use the word 'definition' in the prompt template instead of 'instruction'. This is also consistent with the schema of the task datasets in NATURAL-INSTRUCTIONS. Prompts presented in Fig. 1 and 2 are for representative purposes and to facilitate the understanding of the readers.

[12] Datset: https://github.com/allenai/natural-instructions. Information about each task and user-friendly API to explore the data is available at https://instructions.apps.allenai.org/

Task ID	Task Objective	Instruction Length	Label Space	Skewness (%)
019	Verifying the temporal reasoning category of a given question	13 sentences/199 words	Yes/No	91.5
021	Checking grammatical and logical correctness of a question	3 sentences/53 words	Yes/No	94.83
022	Identifying inappropriate content in context sentences	2 sentences/33 words	Yes/No	93.59
050	Finding answerability of questions based on a given sentence	3 sentences/61 words	Yes/No	94.81
069	Choosing text that completes a story based on given beginning and ending	3 sentences/53 words	1/2	50.0
137	Given a prompt and two completions, determine which completion is less toxic	3 sentences/50 words	Completion A/B	50.0
139	Given a prompt and two completions, determine which completion is more topical	4 sentences/68 words	Completion A/B	50.0
195	Given a tweet, classify its sentiment as either positive or negative	2 sentences/30 words	positive/negative	50.0

Table 7: Details of the 8 classification tasks taken from NATURAL-INSTRUCTIONS dataset. Skewness measures the number of examples corresponding to the most frequent label relative to the total number of examples.

The above choices between 'definition' and 'instruction' is only one example of possible *template-level* changes. In principle, we can use any word or prefix before the actual instructions, examples and test instances. For example, for the prompt shown in Fig. 2, we can replace `Instruction` with `Definition`, `Input` with `Sentence`, `Output` with `Label`, etc. Each of these changes will result in a new prompt template. While these changes are subtle, Zhao et al. (2021) shows that models can be empirically sensitive to such changes. Since the target of this work is to explore better ways of leveraging instructions in the prompt, we keep these template words unchanged in all our experiments so that the comparison of different searched instructions can be fair. Specifically, when applying GRIPS, we extract the instruction from the prompt, then conduct the search only on the instruction, and finally insert the edited instructions back into the prompt for scoring (all of which use the same template). Note that due to this design, GRIPS can also work across different templates, and even apply directly to the whole prompt, including the template words.

C GRIPS with Simulated Annealing

In this version of the search algorithm (Algorithm 2), GRIPS is modified such that if during an iteration, a higher scoring candidate is not found, then the best candidate will be chosen for the subsequent iteration by sampling from a Bernoulli distribution. The probability of success is given by:

$$\lambda = \exp\left(\frac{score - base\ score}{T_{max} \times e^{-i/D}}\right).$$

Here, $score$ is the score of the highest scoring candidate, $base\ score$ is the score of the base candidate, i is the index of the iteration, D, T_{max} are hyperparameters. This formulation has been adapted from Pirlot (1996). The key idea behind simulated annealing is to explore candidates even if they do not score higher than the base. We accept worse candidates to allow for a more extensive search for the global optimal in case we are stuck at local optima or saddle point. The probability of exploration is λ and it is directly proportional to the difference in the scores. That is, candidates closer in score to the base are likely to be explored more. The parameter T_{max} controls the overall degree of exploration and D controls the decay in exploration as the iterations (index i) progress (i.e., move from exploration to exploitation). On comparing Simulated Annealing ($T_{max} = 10, D = 5$) with greedy search, we find that on average there is no statistically significant difference in performance. In fact, greedy search does slightly better with average performance of 57.79 vs 57.46 which is the average performance of simulated annealing search. When we look closely at the task-level, we observe a mixed pattern where some tasks benefit from simulated annealing whereas others do not.

D Manual Rewriting of Task Instructions

In lieu of final rewritten instructions for all the tasks used in Mishra et al. (2022b), the rewriting process was done by the first three authors, after carefully studying the guidelines in the paper, in an iterative manner. The first iteration involved identifying all the suggestions (from (1)-(3)) that could be applied

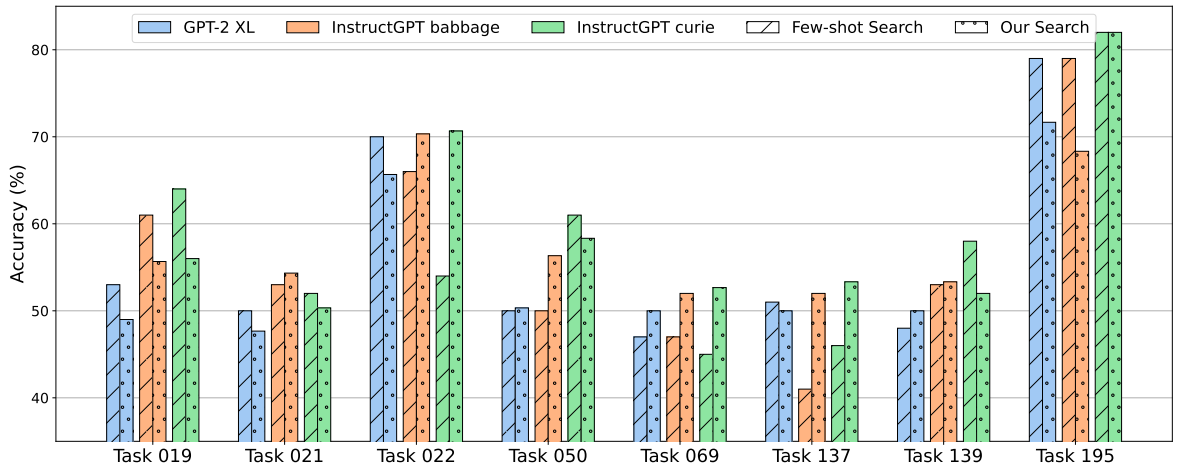

Figure 5: Task-wise comparison of our GRIPS search over instructions (dotted) with search over exemplar prompts (dashed) across model for the same data and computational budget.

to the instructions for each task. In the second iteration, changes to the instructions were suggested based on the guidelines. These changes were then reviewed by the other authors. Disagreements were resolved through detailed discussions until a consensus was reached in the third iteration. Including suggestion (4) was straightforward and did not require extensive discussions. The entire process was dedicated nearly 5 hours of manual effort.

We found suggestion (1) to be the most commonly applicable instruction, and can be applied to all instructions in our tasks. It is composed of two components, first, making references to the low-level patterns of the task and fixing grammatical errors, e.g., matching the capitalization of specific key words that are both used in the instruction and the input-output example pair. The second component is rephrasing abstract or vague phrases used in the instruction. Most of our discussions were focused on resolving disagreements in this second component. Suggestion (3) is replacing negative sentences with equivalent positive suggestions, and it is the second most common modification. This was followed by suggestion (2), i.e., itemization that was primarily useful for Task `019` since the original instruction was exceptionally long.

E Search over Examples vs Search over Instructions

Fig. 5 shows the task-level comparison of performance of the two search paradigms described in §5.3. For most tasks on GPT-2 XL, the performance of the searched exemplar prompt is superior to the searched instructional prompt (also reflected in Table 2). On an average, for InstructGPT models, purely instructional prompts searched through GRIPS outperform the searched exemplar prompts (based on margin of improvement). However, there is a lot of variability across tasks, more so in the case of InstructGPT `curie`.

F Hyperparameter Choice

Due to financial constraints, hyperparameter tuning was conducted using line search using smaller (and cheaper) models like GPT-2 XL and on select tasks during preliminary experiments. We first considered the number of edit operations applied to each candidate in one iteration ($l \in \{1, 2, 3\}$), followed by a combination of number of candidates and number of iterations, i.e., $(m, n) \in \{(10, 5), (5, 10), (2, 25)\}$. We increased patience P as we reduced the number of candidates ($m = 10 \Rightarrow P = 1, m = 5 \Rightarrow P = 2$, and $m = 2 \Rightarrow P = 4$) in order to ensure that the search did not end prematurely. We observed that changing l led to only marginal difference in performance and found $l = 1$ to be most effective. We set $m = 5, n = 10$, and $P = 2$ in our experiments. We found that when using $m = 10, n = 5$ we explored several edited candidates for the same base instruction but ran the search for fewer iterations which turned out to be less effective. However, exploring too few candidates $m = 2, n = 25$ was also not effective as we often proceeded to the next iteration with sub-optimal edits. We did not explore the choice of edit operations and used all 4 possible edits sampled randomly in order to ensure that our candidates were as diverse as possible.

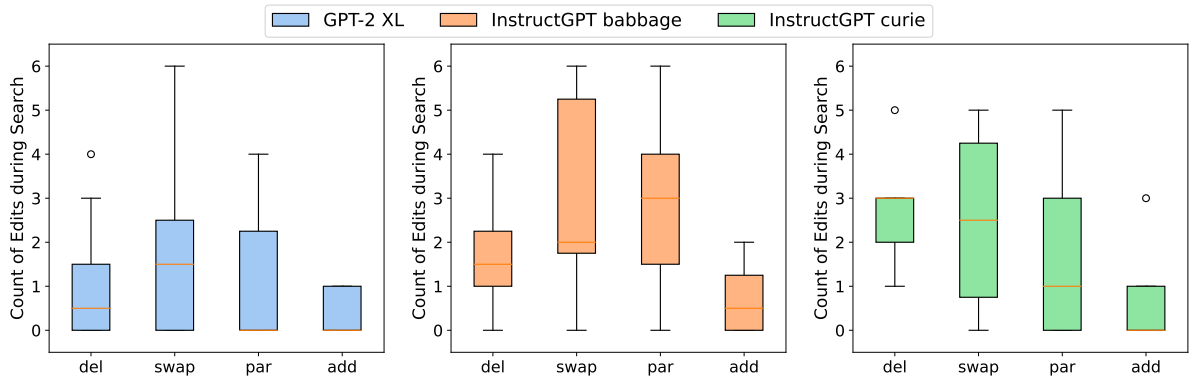

Figure 6: Number of times the edit operations (delete, swap, paraphrase, and add) were used across tasks in a typical search run, shown for different models.

Task	Model	After Search Instructions
Task 069	Original	In this task, you will be shown a short story with a beginning, two potential middles, and an ending. Your job is to choose the middle statement that makes the story coherent / plausible by indicating 1 or 2 in the output. If both sentences are plausible, pick the one that makes most sense.
	GPT-2 XL	`Returned Original`
	InstructGPT `babbage`	This task is being done, You will be shown a short story with a beginning, two potential middles, and an ending . Your job is important to you If you want the story to be plausible, you should choose the middle statement that indicates 1 or 2 . If both sentences are plausible, pick the one that makes most sense.
	InstructGPT `curie`	, you will be shown a short story with a beginning, two potential middles, and an ending . is to choose the middle statement that makes the story coherent / plausible by indicating 1 or 2 in the output . If both sentences are plausible, pick the one that makes most sense.
Task 139	Original	Given a prompt and two completions from a language model, select the completion that is more topical with respect to continuing the prompt. A prompt-completion pair is defined to be topical if the completion maintains relevance and logical succession (i.e. stays on topic) with the prompt. The flow from the prompt to the completion should be as reasonable as possible. Generate either Completion A or Completion B.
	GPT-2 XL	`Returned Original`
	InstructGPT `babbage`	, select the completion that is more topical with respect to continuing the prompt . A prompt-completion pair Will be made . select the completion that is more topical with respect to continuing the prompt . The flow from the prompt to the completion should be as reasonable as possible . should be as reasonable as possible Will be made.
	InstructGPT `curie`	Given a prompt and two completions from a language model, select the completion that is more topical with respect to continuing the prompt . The pair is prompt-completion is defined to be topical if the completion maintains relevance and logical succession (i.e . The pair is prompt-completion . The flow should be as reasonable as possible . Generate either Completion or Completion B.

Table 8: Examples of searched instructions of Tasks 069, and 139 for different models.

G Edit Operations

In Fig. 6 we show the usage of edit operations for different models to get to the final searched instructions. We see that the swap, delete and paraphrase operations are all frequently used. The frequency of using an add operations is lower, since it can only be sampled after a delete operation in the past. Nonetheless, the add operation is used in search runs of roughly 37.5% of the tasks.

H Instructions after GRIPS

Tables 8, and 9 contain the original and searched instructions for the all the tasks not discussed in §5.7. Manually observation and comparison reveals that the searched instructions are often semantically incoherent or confusing. Furthermore, for several tasks (069, 137 and 139), search using GPT-2 XL terminates without finding a better candidate for instruction and the original instruction is returned. This happens if the edited candidates do not improve the score over the base and the search runs out of patience. We observe that 68.5% of the searched instructions are shorter than the original, and 87.5% of them contain some label information pertinent to the task.

Task	Model	After Search Instructions
Task 019	Original	Indicate with ‘Yes‘ if the given question involves the provided reasoning ‘Category‘. Indicate with ‘No‘, otherwise. We define five categories of temporal reasoning. First: "event duration" which is defined as the understanding of how long events last. For example, "brushing teeth", usually takes few minutes. Second: "transient v. stationary" events. This category is based on the understanding of whether an event will change over time or not. For example, the sentence "he was born in the U.S." contains a stationary event since it will last forever; however, "he is hungry" contains a transient event since it will remain true for a short period of time. Third: "event ordering" which is the understanding of how events are usually ordered in nature. For example, "earning money" usually comes before "spending money". Fourth one is "absolute timepoint". This category deals with the understanding of when events usually happen. For example, "going to school" usually happens during the day (not at 2 A.M). The last category is "frequency" which refers to how often an event is likely to be repeated. For example, "taking showers" typically occurs 5 times a week, "going to saturday market" usually happens every few weeks/months, etc.
	GPT-2 XL	going to school . Indicate with ‘ No ‘, otherwise . We define five categories of temporal reasoning . First: "event duration" which is defined as the understanding of how long events last . For example, "brushing teeth", takes few minutes . Second: "transient v. stationary" events . This category is based on the understanding of whether an event will change over time or not . For example, the sentence "he was born in the U.S." contains a stationary event since it will last forever; however, "he is hungry" contains a transient event since it will remain true for a short period of time . Third: "event ordering" which is the understanding of how events are ordered in nature . For example, "earning money" comes before "spending money". Fourth one is "absolute timepoint". This category deals with the understanding of when events happen . For example, "going to school" happens during the day (not at 2 A.M). The last category is "frequency" which refers to how often an event is likely to be repeated . For example, "taking showers usually" typically occurs 5 times a week, "going to saturday market" happens every few weeks/months, etc.
	InstructGPT `babbage`	Indicate with ‘ Yes ‘ if the given question involves the provided reasoning ‘ Category ‘ . Indicate with ‘ No ‘, otherwise . We define five categories of temporal reasoning . First: "event duration" which is defined as the understanding of how long events last . For example, "First", takes few minutes . Second: "transient v. stationary" events . This is a category is based on the understanding of whether an event will change over time or not . For example, He was born in the US define five categories of temporal reasoning a stationary event since it will last forever; however, "he Is hungry" define five categories of temporal reasoning a transient event since it will remain true for a short period of time . Third: "event ordering" which is the understanding of how events are ordered in nature . For example, "earning money" comes before "spending money". Fourth one is "absolute timepoint". This is a category deals with the understanding of when events happen . For example, "going to school" happens during the day (not at 2 A.M). The last category is "frequency" which refers to how often an event is likely to be repeated . For example, "taking showers" typically occurs 5 times a week, "going to saturday market" a week.
	Instruct GPT `curie`	Indicate with ‘ Yes ‘ if the given question involves the provided reasoning ‘ Category ‘ . Indicate with ‘ No ‘, otherwise . We define five categories of temporal reasoning . First: "event duration" which is defined as the understanding of how long events last . For example, "brushing teeth", usually takes few minutes . Second: "transient v. stationary" events . This category is based on the understanding of whether an event will change over time or not . For example, the sentence "he was born in the U.S." contains a stationary event since it will last forever; however, "he is hungry" contains a transient event since it will remain true for a short period of time . Third: "event ordering" which is the understanding of how events are usually ordered in nature . For example, "earning money" usually comes before "spending money". Fourth one is "absolute timepoint". This category deals with the understanding of when events usually happen . For example, "going to school" usually happens during the day (not at 2 A.M). is "frequency" which refers to how often an event is likely to be repeated . For example, "taking showers" typically occurs 5 times a week, "going to saturday market" usually happens every few weeks/months, etc.
Task 022	Original	Read the given context and if the the context is inappropriate (e.g., pornographic) or nonsensical (e.g., cannot determine what happenings the context is about), indicate via "yes". Otherwise, respond via "no"
	GPT-2 XL	Read the given context and if the the context is inappropriate (e.g., pornographic) or nonsensical (e.g., Can’t decide what the context is about, indicate via "yes". Otherwise, respond via "no". [another interesting output]
	InstructGPT `babbage`	Read the given context and e.g., pornographic) or nonsensical (e.g . (e.g., pornographic) or nonsensical (e.g., cannot determine what happenings the context is about), indicate via "yes". Otherwise, respond via "no".
	Instruct GPT `curie`	Read the given context and indicate via "yes (e.g., pornographic) or nonsensical (e.g., cannot determine what happenings the context is about), indicate via "yes". Otherwise, respond via "no".
Task 050	Original	You are given a sentence and a question in the input. If information provided in the sentence is enough to answer the question, label "Yes", otherwise label "No".Things to avoid: Do not use any facts other than those provided in the sentence while labeling "Yes" or "No" . Emphasis & Caution: There are only 2 types of valid responses: Yes and No."
	GPT-2 XL	You are given a sentence and a question are given a sentence and a question . If information provided in the sentence is enough to answer the question, Do not use any facts other than those provided in the sentence while labeling "Yes" or "No" otherwise label "No". Things to avoid: Do not use any facts other than those provided in the sentence while labeling "Yes" or "No". Emphasis & Caution: There are only 2 types of valid responses: Yes and No.
	InstructGPT `babbage`	You are given a sentence and a question in the input . If information provided in the sentence is enough to answer the question, label "Yes", otherwise label "No". Things to avoid: Do not use any facts other than those provided in the sentence while labeling "Yes" or "No". Emphasis & Caution: There.
	InstructGPT `curie`	You are given a sentence and a question in the input . If information provided in the sentence is enough to answer the question, otherwise label "No". Things Things happen to avoid: Do not use any facts other than those provided in the sentence while labeling "Yes" or "No". Emphasis & Caution: There are only 2 types of valid responses: Yes and No.

Table 9: Examples of searched instructions of Tasks 019, 022, and 050 for different models.