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 requires subjective interpretation, while gradient-based tuning can be extremely computationally demanding for large models and may not be feasible 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. 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 (with similar improvements for OPT, BLOOM, and FLAN-T5). We see improvements for both instruction-only prompts and instruction + k-shot examples prompts. Notably, GRIPS outperforms manual rewriting and purely example-based prompts while controlling for the available compute and data budget. Further, performance of GRIPS is comparable to select gradient-based tuning approaches. Qualitatively, we show our edits can simplify instructions and at times make them incoherent but nonetheless improve accuracy.[1]

1 Introduction

Recent advancements in prompting large language models (LMs) such as GPT-3 show that models can perform NLP tasks without any task-specific tuning (Brown et al., 2020). 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 task performance via instructional prompts, Mishra et al. (2022b) provide a set of guidelines to manually rewrite raw instructions. 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 LMs by means of *prompt tuning* (Liu et al., 2021b). Existing prompt tuning methods use gradient-based approaches which have a few notable shortcomings. First, computing gradients with large LMs can be prohibitively computationally demanding. Second, this is entirely infeasible for models available only via APIs, because model gradients and weights are not standardly accessible.[3] Third, output continuous representations may not directly map back onto tokens in the original vocabulary. Thus, we cannot verify

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

[2] In general, whether an instruction is a sufficient description of a task depends on whom it is written for, i.e. people with less task expertise require more background information.

[3] GPT-3 models can be finetuned on given data, but the model parameters and gradients remain unavailable (source).

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 3845–3864

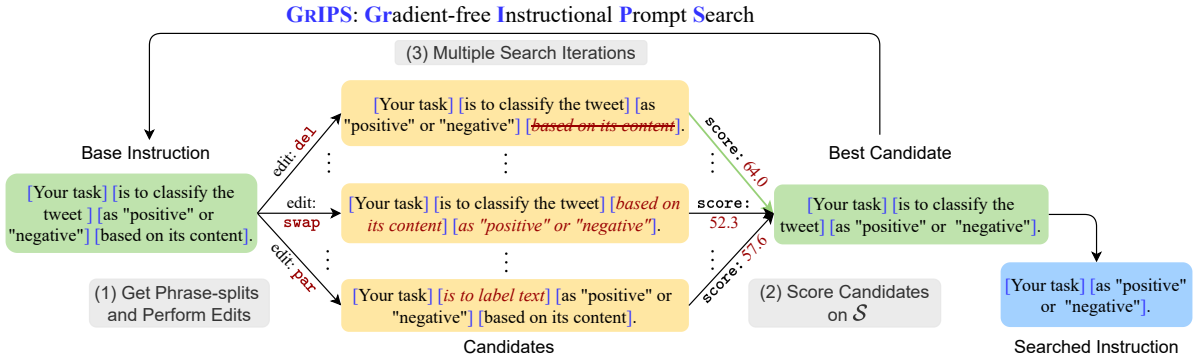

Figure 1: Overall Pipeline of GRIPS. The main steps are numbered. Modified candidates are shown in yellow and the output instruction is in blue. We use '[ ]' to show the syntactic phrase-level splits at which the edit operations occur. Edited text is highlighted in red and the selected candidate (with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

whether models are responding to prompts 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 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, local, edit-based, and gradient-free search (shown in Fig. 1). In contrast to gradient-based 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. 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. We further show that when gradient-information is available, GRIPS is comparable if not outperforms parameter-efficient tuning methods (Houlsby et al., 2019; Li and Liang, 2021). Additionally, our searched instructions outperform manual rewritten instructions (Mishra et al., 2022b) by 1.5 percentage points on average for the InstructGPT `curie` engine. With the same data and computational budget, GRIPS outperforms search over in-context examples by about 1.6 points for InstructGPT. Lastly, we consider 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 sum, our contributions include:

1. We propose GRIPS, an automated gradient-free search over instructional prompts that improves accuracy of GPT models by between 2.36 and 9.36 points on NATURAL-INSTRUCTIONS. We also show improvements for OPT, BLOOM, and FLAN-T5.
2. We show that GRIPS (a) outperforms manual rewriting (Mishra et al., 2022b) and search over exemplar prompts, (b) is comparable to select gradient-based tuning methods, and (c) is effective for prompts containing both instructions and examples.
3. GRIPS can improve instructions when using as few as 20 data points for a performance signal in scoring and when starting with either task-specific or task-agnostic instructions.

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.[4] In contrast, our work focuses on

[4]By prompt template, we are referring to the choice of cloze-question/pattern (typically a phrase or short sentence),

instructional prompts as described below.

Instructional Prompts. Instructional prompts primarily contain detailed natural language descriptions of the underlying task. Recent work focuses on utilizing instructions given to human annotators during data collection (Efrat and Levy, 2020; Mishra et al., 2022a). Mishra et al. (2022b) propose guidelines for manually rewriting instructions in order to further improve performance of instructional prompts. While Webson and Pavlick (2021) show language models may struggle to truly understand instructions, Wei et al. (2022); Sanh et al. (2022) find finetuning on instructions and in-context examples in a hugely multi-task manner helps generalization to other tasks. Lastly, Weller et al. (2020) provide a dataset in which task descriptions are formulated as questions. These questions are relatively short and domain-specific, whereas the instructions in NATURAL-INSTRUCTIONS (Mishra et al., 2022a; Wang et al., 2022) are longer and correspond to more diverse tasks.

Prompt Tuning. Instead of limiting prompts to natural language text, recent work explores training continuous vector tokens in prompts via gradient-based optimization (Liu et al., 2021c; Lester et al., 2021; Li and Liang, 2021; Qin and Eisner, 2021). Sun et al. (2022) aim to optimize continuous tokens without using gradients, however, their technique does not work for APIs that only allow modifying text and not token embeddings (like for GPT-3).

Prompt Search. Zhao et al. (2021) find varying the choice of training examples, example order permutations, and template can alter the performance of a prompt. Liu et al. (2021a) focus on selecting in-context examples from a dataset, while Lu et al. (2022); Kumar and Talukdar (2021) explore optimal ordering of examples. Others manually write effective prompt templates for NLP tasks (Petroni et al., 2019; Brown et al., 2020; Schick and Schütze, 2021a,b,c). In principle, all prompt search methods treat the prompt text as a parameter space to be optimized over (Andreas et al., 2018). 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 (§3.2.2). Meanwhile, Shin et al. (2020) use a gradient-based search to find trigger words in the prompt template. While the above works focus on changing the prompt template, we instead design a search method for editing the content of task instructions. Our search algorithm is also related to genetic algorithms (Mitchell, 1998), where parent candidates are mutated to generate offering (via our text-based edit operations) to increase fitness under an objective (like our score function).

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).

3 Methodology

In this section, we first describe and illustrate different prompt modes (§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 instructions through two prompt modes: *Instruction-Only* and *Instruction + Examples* (illustrated in Fig. 2). Here, prompt mode refers to the choice and arrangement of the three components (instruction, in-context examples, and test instance). These prompt modes are also used in Mishra et al. (2022a) (details in Appendix B). To obtain each kind of prompt, we concatenate 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. The 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.[5] Note that examples in the score set

[5]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 §5.6, we show improvements with GRIPS using as few as $|\mathcal{S}| = 20$ examples.

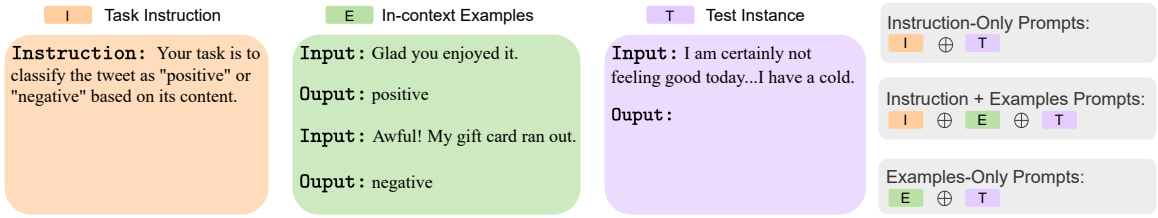

Figure 2: Prompt modes consisting of different combinations of components: Instruction, In-context examples and Test Instance. '⊕' denotes concatenation. *Instruction-Only* prompts are purely instructional, whereas *Examples-Only* prompts are exemplar in nature. Prompt mode *Instruction + Examples* is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

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 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 label space of a task and $\hat{y}$ be the model prediction. If H is the entropy and α is a scaling factor (we use $\alpha = 10$), then the score function is:[6]

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

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

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.

Beam Search. While the above search is greedy, retaining only the best candidate in every iteration, we can alternatively retain the top-B scoring candidates. Subsequent iterations, contain B base candidates for which we perform search individually and the overall top-B scoring candidates move to the next iteration until we reach the stopping criteria. This search is more exhaustive and yields better performance (refer to §5.3), however, it increases the number of model evaluations by $\approx B$-fold. We refer readers to Appendix C for full pseudo-code.

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.

3.2.2 Edit Operations

Below, we describe edit operations used in GRIPS:
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.

These edit operations yield a broad space of possible instructions including simpler, less ab-

[6] BalancedAccuracy is calculated on a scale of 0-100. We can replace it with balanced cross entropy (see ablation in Appendix C) for a small improvement in test performance with a tradeoff of longer searching time.

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

Model	No Search	GRIPS
Majority Label	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: Impact of GRIPS with Instruction-Only prompts. Average accuracy (%) on 8 tasks from NATURAL-INSTRUCTIONS. In majority label, we output most frequent label for all test instances. 95% confidence intervals in parentheses. curie is the largest model.

Method	Accuracy	
No Search	48.38	
GRIPS	**53.68**	
- entropy in score	52.20	(-1.48)
- del operation	51.12	(-2.56)
- swap operation	52.67	(-1.01)
- par operation	52.54	(-1.14)
- add operation	52.42	(-1.26)

Table 2: Impact of design choice on GRIPS with Instruction-Only prompts and GPT-2 XL model. Change in performance relative to GRIPS in brackets.

stract instructions with fewer details. Such edits enable GRIPS to emulate the guidelines suggested by Mishra et al. (2022b) that also limit details and abstractions. Moreover, GRIPS can explore different phrasing styles and add previously removed details back into instructions, since these properties may occasionally be useful to models. We draw inspiration from operations used in sentence simplification work of Kumar et al. (2020). Empirically, the effectiveness of edits is shown in §5.1.

4 Experimental Setup

Dataset. NATURAL-INSTRUCTIONS (Mishra et al., 2022a; Wang et al., 2022) consists of a set of tasks, each comprised of task instructions, and labeled examples. Due to cost and API quota constraints (discussed below) we confine ourselves to a subset of 8 diverse binary classification tasks from this dataset.

Test Sets. Following Mishra et al. (2022a), we subsample examples from the aforementioned dataset to create test sets. For the main results (in Table 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 subsets of 100 test examples per task (hence, numbers vary between Table 1 and subsequent tables).

Models. We use GPT models (Radford et al., 2018, 2019; Brown et al., 2020) with ≥1B parameters, specifically GPT-2 XL (1.5B parameters), InstructGPT babbage, and curie.[8] Relative to standard GPT-3 models, 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 cost constraints in running experiments (discussed below), 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).

Cost. A single run of GRIPS on a task requires $\mathcal{O}(m \times n \times |\mathcal{S}| \times B)$ model evaluations. 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 ≈ $2400. We note that after running GRIPS and obtaining the modified searched instruction, the cost of evaluation on the test set is significantly smaller, a total of ≈ $150 for all the results in this work.

Hyperparameters. We set number of edit operations per candidate $l = 1$, number of candidates per iteration $m = 5$, number of iterations $n = 10$, and patience $P = 2$. Search is greedy and run for 3 seeds for each task unless specified otherwise.

Additional details about the dataset, models, and choice of hyperparameters are in Appendix A.

5 Results and Discussion

In this section, we present the results of our experiments. First, we establish the effectiveness of GRIPS across models in §5.1. Then, we compare our search to other methods in §5.2 and §5.3 and provide additional analysis in subsequent sections.

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 that

[8] 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).

Prompt	Method	GPT-2 XL	InstructGPT	
			babbage	curie
Inst. Only	No Search	48.38	55.37	57.25
	Manual Rewriting	47.70	55.50	57.87
	GRIPS	53.68	57.79	59.37
Ex. Only	No Search	51.50	55.29	56.13
	Example Search	**56.00**	56.25	57.75
Inst. + Ex.	No Search	52.40	55.70	57.65
	GRIPS	54.40	**57.88**	**59.44**

Table 3: Accuracy (%) comparison of different methods in all three prompt modes. 'Inst.' and 'Ex.' are used to abbreviate instruction and examples. During no search, we use a random set of examples wherever indicated.

is statistically significant at the $p < 0.05$ level.[9] Accuracy for each method is averaged across test data, seeds, and tasks. 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).

Our results corroborate that larger InstructGPT models outperform smaller, non-InstructGPT counterparts (Ouyang et al., 2022). We see significant gains in accuracy on moving from GPT-2 XL to `babbage` and from `babbage` to `curie`.

Ablations. In Table 2, we evaluate several design choices in §3.2 on GPT-2 XL. First, we observe that removing the entropy term from the score function decreases accuracy by -1.48 points. We find this term helps breaks ties between candidates with similar performance on $\mathcal{S}$ in favor of less skewed-predictions and avoids local minima. Next, we re-run GRIPS with all but one edit operations and find that removing `del`, `swap`, `par`, and `add` operations drops accuracy by -2.56, -1.01, -1.14 and -1.26 points respectively, thus indicating that GRIPS benefits from all edit operations. Appendix C contains additional design ablations.

5.2 Comparing with Gradient-free Methods

Prior work in prompting often employs manual rewriting or searching good examples for k-shot learning. Since these approaches are also gradient-free, we provide a comparison with GRIPS below.

Manual Rewriting. Closest to our setting, Mishra et al. (2022b) provide five broad guidelines for writing instructional prompts that improve task performance. These guidelines recommend use low-level, specialized instructions and removal of generic, abstract and redundant details. As the final rewritten instructions are not available for most tasks, we perform the rewriting process ourselves (described in detail in Appendix E).

Method	%Param	Accuracy
GPT-2 XL	0	48.38
+ Direct Finetuning	100	55.88
+ Adapters (Houlsby et al., 2019)	3	55.08
+ Prefix-Tuning (Li and Liang, 2021)	3	53.29
- MLP Reparametrization	0.1	51.12
+ GRIPS (*Ours*)	0	53.68
+ beam search; $B = 5$ (*Ours*)	0	**56.50**

Table 4: Comparison of GRIPS with gradient-based methods. GPT-2 XL and GRIPS use Instruction-Only prompts. %Param denotes number of parameters used relative to size of GPT-2 XL.

Example Search. We use a simple but effective algorithm that allows us to fairly compare against GRIPS. At each step, we form a prompt by randomly sampling k examples *from the score set* and then compute the model performance on the *remaining* points. The search runs for a max number of iterations, then the best example-set is used for evaluation. 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). We use the same score set for example search as GRIPS. Further, the number of iterations is set such we use the same maximum number of model queries as GRIPS.[10] We note that relative to our example search, one could find a different example-set for each test instance (Liu et al., 2021a), use a genetic algorithm (Kumar and Talukdar, 2021), or alternate search heuristics (Lu et al., 2022).

Results. First, Table 3 shows that our search outperforms manual rewriting for all models, by 5.56, 2.29 and 1.50 points for GPT-2 XL, InstructGPT `babbage` and `curie`, respectively. Next we observe that example search outperforms GRIPS for GPT-2 XL. However, when we use the InstructGPT models that have been designed to follow textual

[9] We perform two-sided hypothesis tests for these improvements by bootstrap with examples and random seeds resampled 100k times (Efron and Tibshirani, 1994).

[10] 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 less than Examples-Only prompts in the OpenAI API.

Model	Initialization	No Search	GRIPS
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 5: Accuracy (%) for task-specific or task-agnostic initial instructions with Instruction-Only prompts.

Model	# Param	No Search	GRIPS
OPT	1.3B	46.38	53.3
	2.7B	47.5	53.95
	6.7B	48.63	54.41
	30B	49.75	55.1
BLOOM	1B	46.38	52.75
	3B	48.0	53.96
GPT-J	6B	47.25	54.67
GPT-NeoX	20B	47.75	54.85
FLAN-T5†	3B	71.25	74.33

Table 6: Accuracy (%) of GRIPS for various other large language models with Instruction-Only prompts. †Chung et al. (2022) use NATURAL-INSTRUCTIONS dataset during instruction-tuning.

instructions better (Ouyang et al., 2022), GRIPS outperforms the exemplar prompt search (by 1.54 and 1.62 points for `babbage` and `curie` respectively). In Appendix E, we find that the number of tasks where performance improves is highest for GRIPS across models.

5.3 Comparing with Gradient-based Methods

Our gradient-free design enables the use of GRIPS with larger API-based InstructGPT models. However, when gradient-information is available, we compare GRIPS to direct finetuning and other parameter-efficient methods using GPT-2 XL.

Methods and Setup. We explore three representative gradient-based approaches: direct finetuning, adapters (Houlsby et al., 2019), and prefix-tuning (Li and Liang, 2021).[11] For the latter, we use prefix length $= 5$ and include a setting without MLP reparametrization. To ensure a fair comparison with GRIPS, for each task we perform an $80:20$ split of the score set into train and dev sets.

Results. The comparison is presented in Table 4. Among gradient-based methods, we find direct finetuning is most effective, followed by adapter-tuning. Both approaches outperform GRIPS (greedy decoding) by 2.2 and 1.4 points respectively. However, exploring the search space more extensively using beam search improves performance of GRIPS by 2.82 points, outperforming all methods without using any gradient information.[12] We also observe that GRIPS outperforms prefix-tuning by up to 2.56 and 5.38 points using greedy and beam search respectively. Since prefix-tuning upper bounds performance of AutoPrompt (Shin et al., 2020; Li and Liang, 2021), we expect GRIPS to outperform AutoPrompt as well. Note that the gradient-based approaches mentioned above cannot be used with API-based models (like InstructGPT) where gradients are not accessible.

5.4 Task Specific vs Agnostic Instructions

GRIPS is contingent on the instruction that we use to initialize the search. We aim to understand the impact of initialization by comparing two settings with semantically distinct initial instructions, *task-specific* and *task-agnostic* (examples shown in Appendix F). Task-specific instructions 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. Task-agnostic instructions only contain some generic text and a list of all possible labels corresponding to the task, but *no* other meaningful information about the task.

In Table 5, 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 performs better with task-agnostic instructions as compared to task-specific ones. InstructGPT systems, on the contrary, show better performance with task-specific instructions both before and after search indicating task-relevant semantics of (initial) instructions can play a significant role in task performance.

5.5 GRIPS with other Open-Source Models

Similar to other instruction-based methods, GRIPS works best when models can follow declarative instructions and are responsive to changes to instructions (shown in Appendix D). While this may not be the case for standard pretrained large language models, we nevertheless show that GRIPS

[11] These methods only use test input and not instructions.

[12] Due to cost constrains, we do not use beam search with InstructGPT, although we expect it to improve performance.

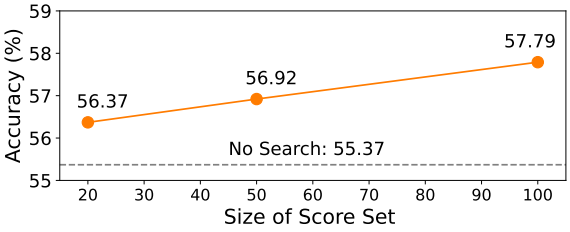

Figure 3: Impact of $|\mathcal{S}|$ on search and downstream average task accuracy for InstructGPT `babbage`.

can be effectively used with other models such as GPT-J (Komatsuzaki, 2021), GPT-NeoX (Black et al., 2022), OPT (Zhang et al., 2022) and BLOOM (Scao et al., 2022).

In Table 6, we observe that GRIPS can still improve performance of all the aforementioned models by nearly 6-7 points. Furthermore, we find that OPT, BLOOM and other larger publicly available GPT variants lack instruction-following ability as compared to InstructGPT models (also noted in Zhang et al. (2022)). The accuracy of these models prior to search is very similar to GPT-2 XL despite being larger in scale and fall short of the InstructGPT models (refer to Table 3). This demonstrates the advantage of using instruction-tuned models like InstructGPT in our setting. Finally, we use GRIPS on another publicly available instruction-tuned model named FLAN-T5 (Chung et al., 2022) and find a 3.08 point performance improvement. Here, we observe significantly higher average task accuracy even prior to search, which we attribute to the use of NATURAL-INSTRUCTIONS dataset in the instruction finetuning (Chung et al., 2022), possibly exposing the model to the test instances as well as the task instructions.

5.6 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.

In Fig. 3, we experiment with a score set of size 100, 50 or 20. 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. Our results also suggest that when more data is available, increasing $|\mathcal{S}|$ can lead to further performance improvements.

5.7 Semantics of Searched Instructions

Table 7 (and Appendix G) contains some searched instructions by GRIPS. 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, GRIPS with InstructGPT `curie` yields a relatively coherent yet simple instruction by replacing "grammatical or logical errors" with "errors." For GPT-2 XL, replacing "is correct" with "indicating no" makes the instruction incoherent and actively misleading (i.e. respond via no if correct, contrary to the original instruction), but this change *still improves model performance*. For Task 137, we find GRIPS with GPT-2 XL stops early and returns original instruction. Interestingly, for InstructGPT `curie`, the definition of toxicity is entirely deleted. Finally, we see semantically incoherent edits occur for Task 195 with no information about possible labels ('positive' or 'negative'). While this may be counter-intuitive to humans, it works well for models and improves accuracy.

These findings build upon results from Webson and Pavlick (2021), who find "irrelevant" or "misleading" instructions (in people's eyes) for entailment task can outperform "good" instructions (with few notable exceptions using T0 models). Yet in §5.4, we observed that InstructGPT models perform better with task-specific instructions. Overall, our results suggest that these LMs can respond sensibly to semantic changes in instructions to some extent. As with the study of in-context learning mechanisms (Xie et al., 2022; Razeghi et al., 2022; Min et al., 2022), how models utilize instructions 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 k additional labeled examples before the test instance. Unlike in §5.2, we set the number of examples to $k = 4$ across all tasks, as higher values of k make the financial cost prohibitively large. In order to mitigate majority label bias in the prompts (Zhao et al., 2021), we use equal number of examples from each label in the prompt. Since

	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.
	After GRIPS	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 `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.
	After GRIPS	GPT-2 XL	`Returned Original`
		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.
	After GRIPS	GPT-2 XL	In this task, you are given a text from tweets . There.
		InstructGPT `curie`	in this task, you are given a text from tweets . In this task.

Table 7: 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.

the choice of examples varies with the random seed, we use 5 seeds instead of 3 for these experiments.

Table 3 demonstrates 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 (< 0.1 percentage points). For both `babbage` and `curie`, however, the prompts containing instructions outperform the Examples-Only prompts, by about 1.6 points. Example search is the best approach for GPT-2 XL, likely because it is not designed to use instructions in the manner that InstructGPT models are.

6 Conclusion

We introduce GRIPS, an automatic search algorithm that edits task instructions to 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 example search show that GRIPS outperforms these methods, suggesting that widely exploring the space of model instructions is an effective method for improving model performance. Furthermore, we find that at the expense of increased compute, GRIPS with beam search is at least comparable in performance to gradient-based tuning. 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.

Acknowledgments

We thank the reviewers and the area chairs for their helpful comments and feedback. We thank OpenAI for providing academic access to their API. We also 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.

Limitations

Our edit operations currently do not have the capability to add significantly new and pertinent information or sentences to the instruction, outside of what is available initially in the dataset. Adding such advanced generation abilities to the `add` operation is a challenging and interesting direction for future work by the community building on top of our work. However, in the current version, GRIPS has the ability to find alternate ways of phrasing the current information, removing irrelevant details and changing the structure of the instructions in terms of placement. Further, a frame-

work like GRIPS may not be as effective for purely generation-based tasks due to lack of good metrics to replace the accuracy in the score function. Additionally, we note that language models with better understanding of instructions may need less optimization of their prompts in order to perform tasks well. Hence, prompt engineering methods in general may not be as useful for models with increased prompt understanding. Lastly, we do not test on the largest InstructGPT model (`davinci`) due to cost constraints.

Ethical Considerations

Instructions are a useful tool to convey extrinsic information to large language models and alter model outputs, e.g. by instructing models to generate less harmful content. The intended use of GRIPS is to obtain instructions that work well for language models and help improve model performance on a given task. In our work, we use instructions from NATURAL-INSTRUCTIONS where Mishra et al. (2022a); Wang et al. (2022) ensure quality control. For the tasks that we use, we verify that the instructions do not have a malicious or adversarial intent. Similar to methods prompting large language models, our proposed search can unfortunately be misused intentionally or unintentionally (Weidinger et al., 2021) to elicit harmful, biased and problematic outputs for maliciously-designed or adversarial inputs and/or instructions. Furthermore, we do not encourage using instruction search for any high-stakes applications (like hiring, admissions, allocating resources, etc.). Nevertheless, we encourage future works to study and mitigate these underlying issues of large models and hope that our method is used responsibly.

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.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136. 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*.

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*.

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.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

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*.

Aran Komatsuzaki. 2021. Gpt-j-6b: 6b jax-based transformer.

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.

Melanie Mitchell. 1998. *An introduction to genetic algorithms*. MIT press.

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*.

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, 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*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

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.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022. Black-box tuning for language-model-as-a-service. *arXiv preprint arXiv:2201.03514*.

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.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv preprint arXiv:2204.07705*.

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*.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

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.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

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 Additional Experimental Details

Dataset. In Table 8, 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 8, the label distributions in these tasks examples are extremely skewed towards one label ($> 90\%$). We chose the remaining 4 tasks from next release (`v2`), curated by Wang et al. (2022), 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,[13] i.e. without any recent commits or modifications for at least 1 month. Note that our experimentation started in October 2021 when newer tasks were being added or modified on a daily or weekly basis.

Sampling Test Set. 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. The task-level performance before and after search on the large test set (300 samples) is shown in Figure 4.

Models and Classification. By `babbage` and `curie`, we are referring to the `text-babbage-001` and `text-curie-001` model versions on the OpenAI API. Following Zhao et al. (2021), classification tasks are performed by computing log-probabilities of the label tokens using the completion function of the OpenAI API. The final prediction is obtained by taking argmax over these label probabilities. Note that our setting is different from Mishra et al. (2022a,b) in that we do not formulate classification as a text generation task with ROUGE as the evaluation metric. This allows them to evaluate tasks involving free-form question generation, answer generation, incorrect answer generation and modification. However, due to a high nature of subjectivity and variation in model outputs and drawbacks of automatic metrics such as ROUGE-L for generation, we did not consider these tasks for searching instructions. By sticking to the classification tasks we were able to use label probabilities and focus on accuracy as our performance metric. We leave exploration of GrIPS for generative tasks for future work.

Algorithm 1 Our search algorithm: GRIPS

$base \leftarrow init$ ▷ Initialize base candidate
$s_{base} \leftarrow \texttt{score}(base)$ ▷ Score using examples in $\mathcal{S}$
$\Omega \leftarrow \{\texttt{del}, \texttt{swap}, \texttt{par}, \texttt{add}\}$ ▷ Set of edit operations
$\rho \leftarrow P$ ▷ Patience for early-stop
for $i = 1, \cdots, n$ **do** ▷ n: number of iterations
  for $j = 1, \cdots, m$ **do** ▷ m: number of candidates
    Sample $e_1, \cdots, e_l \in \Omega$ ▷ l edits per candidate
    $\mathcal{C}[j] \leftarrow \texttt{edit}(base, e_1 \circ \cdots \circ e_l)$
    $s[j] \leftarrow \texttt{score}(\mathcal{C}[j])$ ▷ Score above candidate
  end for
  $k \leftarrow \arg\max_j s[j]$
  $best \leftarrow \mathcal{C}[k]$ ▷ Best Candidate
  $s_{best} \leftarrow s[k]$ ▷ Score of best candidate
  if $s_{best} > s_{base}$ **then** ▷ Candidate better than base
    $base \leftarrow best$ ▷ Use this candidate in next step
    $s_{base} \leftarrow s_{best}$ ▷ Update base score
    $\rho \leftarrow P$ ▷ Refresh patience
  else
    if $\rho > 0$ **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

GPU Compute. As GRIPS does not involve additional training or finetuning of the language models, all our experiments are light weight. Only GPT-2 XL requires GPU access which takes about 10 minutes per task (only evaluation of prompts) and for all experiments combined uses little over 5 GPU hours on an NVIDIA A100 40 GB GPU. Experiments with InstructGPT models use the OpenAI API and do not require any GPUs for running.

Hyperparameter Search. Due to financial constraints, hyperparameter tuning was conducted us-

[13] 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/

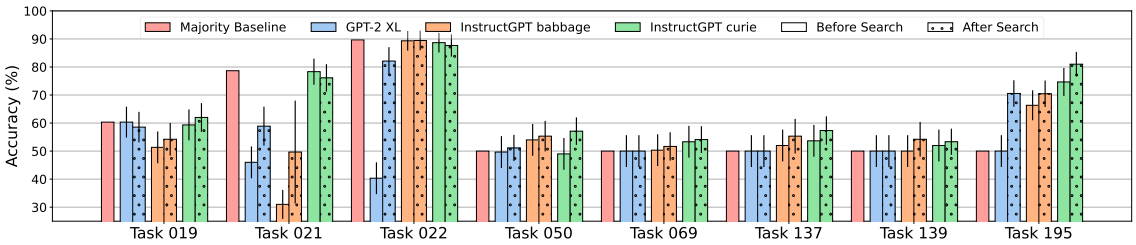

Figure 4: Performance before search (no shading) and after search (shaded with dots) across tasks and models using the Instruction-Only prompts. Error bars show 95% confidence intervals.

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 8: 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.

ing line search using smaller (and cheaper) models like GPT-2 L and 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.

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 in NATURAL-INSTRUCTIONS. Prompts in Fig. 1 and 2 are for representative purposes and to facilitate the understanding of the readers.

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, empirically Zhao et al. (2021) show that models are sensitive to such changes. Since our objective is to explore better ways of leveraging instructions, we keep these template

Algorithm 2 GRIPS with Beam Search

$base \leftarrow \{init\}$ ▷ Set with B elements
$s_{base} \leftarrow \{\texttt{score}(init)\}$ ▷ Set with B elements
$\Omega \leftarrow \{\texttt{del}, \texttt{swap}, \texttt{par}, \texttt{add}\}$
$\rho \leftarrow P$
for $i = 1, \cdots, n$ **do**
  for $b = 1, \cdots, B$ **do**
    for $j = 1, \cdots, m$ **do**
      Sample $e_1, \cdots, e_l \in \Omega$
      $\mathcal{C}_b[j] \leftarrow \texttt{edit}(base[b], e_1 \circ \cdots \circ e_l)$
      $s_b[j] \leftarrow \texttt{score}(\mathcal{C}_b[j])$
    end for
  end for
  $\mathcal{C} \leftarrow \{\mathcal{C}_1; \cdots; \mathcal{C}_B; base\}$ ▷ Concatenate candidates
  $s \leftarrow \{s_1; \cdots; s_B; s_{base}\}$ ▷ Concatenate scores
  $\{k_b\}_{b=1}^{B} \leftarrow \arg\max_j s[j]$ ▷ Find top-B scores
  $best \leftarrow \{\mathcal{C}[k_1], \cdots, \mathcal{C}[k_B]\}$
  $s_{best} \leftarrow \{s[k_1], \cdots, s[k_B]\}$
  if $best \neq base$ **then** ▷ Comparing two sets
    $base \leftarrow best$
    $s_{base} \leftarrow s_{best}$
    $\rho \leftarrow P$
  else
    if $\rho > 0$ **then**
      decrement ρ
      continue
    else
      $k \leftarrow \arg\max_j s_{base}[j]$
      return $base[k]$ ▷ Early Stop
    end if
  end if
end for
$k \leftarrow \arg\max_j s_{base}[j]$
return $base[k]$ ▷ Terminate with highest score candidate

Algorithm 3 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

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 Extensions and Variations of GRIPS

Greedy and Beam Search. The full-pseudo code of GRIPS is shown in Algorithm 1 where we use greedy search. The beam search modification is described in Algorithm 2. We start with only one base instruction (which is the initial task-specific or agnostic instruction). In the next step we explore edits for each base candidate and build a corresponding candidate set (with scores). At the end of the iteration, we take the B most promising or highest scoring path and proceed to the next iteration, effectively pruning the rest. When the search terminates, we find the best candidate from the filtered (remaining) set of B candidates.

Simulated Annealing. In this version of the search algorithm (Algorithm 3), 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 (on InstructGPT `babbage`). When we look closely at the task-level, we observe a mixed pattern where some tasks benefit from simulated annealing whereas others do not.

Cross-Entropy Score Function. In §3.2 we describe our score function that makes use of $\mathrm{BalancedAccuracy}$. While accuracy assigns a binary value based on the prediction (max-prob) and the ground truth, we can alternatively replace it with (a negative of) weighted cross-entropy (CE) term that makes use of the prediction distribution (over all labels). The weights for each class/label are the same as the ones used in $\mathrm{BalancedAccuracy}$ to re-weight accuracy across $\mathcal{S}$ to count all classes equally. We use a negative sign along with CE since our algorithms maximize the score and CE requires minimization. We use $\alpha = 0.1$ as the scales of CE and $\mathrm{BalancedAccuracy}$ are very different. Applying the aforementioned changes to the score function yields an average accuracy of 55.08%, an increase of $+1.4$ points (c.f. Table 2). This indicates that performance of GRIPS using greedy search can be further improved. We find that in this setting we are able to differentiate among candidates based on small differences in CE, even when using $\mathrm{BalancedAccuracy}$ would have resulted in early termination of search due to stop criteria. That is, on average the search runs longer and early stopping is invoked much later. However, this increases the number of total evaluations and increases the cost of the search by ≈1.5x, resulting in a trade-off.

Edit Operations. Fig. 5 shows 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. Next, we explore alternate choices of paraphrase and add operations. Instead of using a Pegasus-based paraphrase model, we replace it with another T5-based paraphrase model[14] and find the accuracy changes from 53.68% to 53.33% which is a minute difference. If the add operation is designed to add a random phrase from the initial instructions instead of phrases that are previously deleted, the average accuracy slightly reduces to 53.42% (c.f. Table 2).

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

D 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 9, 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 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.

[14]Model available at: https://huggingface.co/prithivida/parrot_paraphraser_on_T5

Model	Instruction-Only				Examples-Only		Instruction + Examples	
	Before	Manual Rewriting	+ Labels	GRIPS	Before	Searched Examples	Before	GRIPS
GPT-2 XL	48.38	47.70 (↑1)	48.12 (↑2)	53.68 (↑4)	51.50	**56.00** (↑4)	52.40	54.40 (↑6)
InstructGPT `babbage`	55.37	55.50 (↑4)	55.37 (↑3)	57.79 (↑7)	55.29	56.25 (↑5)	55.70	**57.88** (↑8)
InstructGPT `curie`	57.25	57.87 (↑3)	55.37 (↑3)	59.37 (↑5)	56.13	57.75 (↑4)	57.65	**59.44** (↑6)

Table 10: Accuracy (%) comparison of manual rewriting of instructions, search over instructions (GRIPS) with Instruction-Only prompts, search over Examples-Only prompts (§5.2), and GRIPS with Instruction + Examples prompts (§5.8). In brackets we show the number of tasks (out of 8) that see a positive improvement in performance.

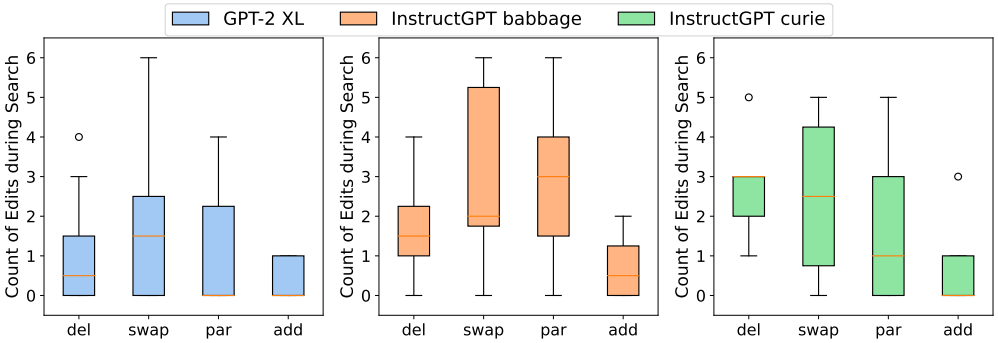

Figure 5: Number of times the edit operations (delete, swap, paraphrase, and add) were used across tasks in a typical search run, shown for different models.

E Details on Gradient-free Methods

E.1 Manual Rewriting

Mishra et al. (2022b) propose five broad suggestions to rewrite instructions described below:

1. Specialized-Reframing: replacing generic, redundant text and describe the low-level task
2. Pattern-Reframing: removing abstract details
3. Itemized-Reframing: split paragraphs into bulleted lists and rewriting negative sentences (phrases like *do not X*) as semantically equivalent positive instances (like *do Y* instead)
4. Decomposition-Reframing: break down tasks with multi-step reasoning into simpler tasks
5. Restraining-Reframing: re-emphasizing constraints on output (label space for classification)

In lieu of final rewritten instructions for our selected tasks, 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 (among 1-4) that could be applied 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. Suggestion 5 is applicable for all tasks 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. This was straightforward and did not require extensive discussions. The entire process was dedicated nearly 5 hours of manual effort.

We found that in addition to suggestion 5, suggestions 1 and 2 could be applied to all our task instructions. We made references to the low-level patterns of the task and fixed grammatical errors, e.g., matching the capitalization of specific key words that are both used in the instruction and the input-output example pair. Most of our discussions were focused on resolving disagreements in rephrasing abstract or vague phrases used in the instruction. Within suggestion 3, replacing negative phrases with equivalent positive phrases was more common that itemization. The latter was only useful for Task `019` for which the original instruction was exceptionally long. We did not feel the need to decompose any task and use suggestion 4.

Unlike Mishra et al. (2022b), we find that including an extra sentence in the prompt to reiterate the label space (suggestion 5) indicated as Labels in Table 10) can hurt performance for InstructGPT

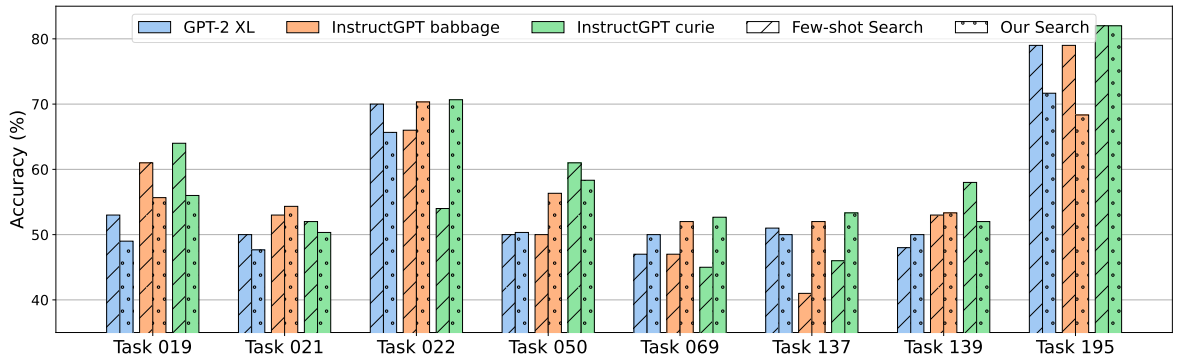

Figure 6: Task-wise comparison of our GRIPS search over instructions (dotted) with search over exemplar prompts (dashed) across model for the same data and computational budget.

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 LM.

E.2 Example Search

Fig. 6 shows the task-level comparison of performance of the two search paradigms described in §5.2. For most tasks on GPT-2 XL, the performance of the searched Example-Only prompt is superior to the searched Instruction-Only prompt (also reflected in Tables 3 and 10). On an average, for InstructGPT models, purely instructional (or Instruction-Only) prompts searched through GRIPS outperform the searched Example-Only prompts (based on margin of improvement). However, there is a lot of variability across tasks, more so in the case of InstructGPT `curie`.

F Task Agnostic Instructions

In Table 11, we compare task-specific and task-agnostic instructions. As mentioned in §5.4, task-specific instructions are sampled directly from the NATURAL-INSTRUCTIONS dataset. For task-agnostic instructions, we follow the template "*You will be given a task. Read and understand the task carefully, and appropriately answer* `[list of labels]`." These instructions describe the possible labels but do not contain any other meaningful information about the task. Given, that in §5.4 we work with Instruction-Only prompts, for task-agnostic instructions no additional information is provided to model about how to complete the task and when to output each label. The list of labels for each task is mentioned in Table 8. This means that tasks sharing the same label space correspond to the same task-agnostic instruction (shown in Table 11), even if the tasks are entirely different.

G Instructions after GRIPS

Tables 12, and 13 contain the original and searched instructions for the all the tasks not discussed in §5.7. Manual 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 ID	Task-Specific Instructions	Task-Agnostic Instructions
019	Indicate with 'Yes' if the given question involves the provided reasoning 'Category'. Indicate with 'No', otherwise. We define five categories · · ·	You will be given a task. Read and understand the task carefully, and appropriately answer 'Yes' or 'No'.
021	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.	You will be given a task. Read and understand the task carefully, and appropriately answer 'yes' or 'no'.
022	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"	You will be given a task. Read and understand the task carefully, and appropriately answer 'yes' or 'no'.
050	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.	You will be given a task. Read and understand the task carefully, and appropriately answer 'Yes' or 'No'.
069	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.	You will be given a task. Read and understand the task carefully, and appropriately answer '1' or '2'.
137	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.	You will be given a task. Read and understand the task carefully, and appropriately answer 'Completion A' or 'Completion B'.
139	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.	You will be given a task. Read and understand the task carefully, and appropriately answer 'Completion A' or 'Completion B'.
195	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.	You will be given a task. Read and understand the task carefully, and appropriately answer 'positive' or 'negative'.

Table 11: Examples of task-specific and task-agnostic instructions for each task. We do not show the entire instruction for Task 019 for brevity (refer to 'original instruction' Table 13 for the complete version).

Task ID	Model	After Search Instructions
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.
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 12: Examples of searched instructions of Tasks 069, and 139 for different models.

Task ID	Model	After Search Instructions
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.
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".
	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".
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 13: Examples of searched instructions of Tasks 019, 022, and 050 for different models.