Few-Shot Prompt-Tuning: An Extension to a Finetuning Alternative

Stanford CS224N Custom Project
James researched, created, and tuned the datasets.
Sam created the training and evaluation pipelines for the models.
Both members contributed to the final write-up.
Mentor: Josh Singh

Samuel Kwok
Department of Computer Science
Stanford University
samkwok@stanford.edu

James Morice
Department of Computer Science
Stanford University
jamorice@stanford.edu

Abstract

Fine-tuning is becoming increasingly expensive and time consuming due to the scaling of language models. As a result, different approaches to improve model performance have arisen. We will take two of these approaches and combine them. The first approach is prompt-tuning, a method of parameter efficient fine-tuning (PEFT). The second method is few-shot prompting. In this paper we demonstrate novel experimental results: while few-shot prompt-tuning for small language models doesn't outperform traditional few-shot prompting for causal text generation tasks, small language models *are* able to consistently generate task-relevant text for option-based, question answering tasks in zero-shot settings when they are prompt-tuned in a few-shot context.

1 Introduction

Supervised fine-tuning is a popular method for adapting models for distinct downstream tasks (Radford et al., 2018). However, as models grow in size, the computational and temporal costs associated with model-tuning increase as well. Thus, many alternatives have been engineered to replace model-tuning. Two such methods are few-shot prompting and prompt-tuning. When models are sufficiently large, few-shot learning achieves similar accuracies on downstream tasks in comparison to model-tuning. In addition, as models increase in scale, so do the positive effects of few-shot prompting (Brown et al., 2020). In addition, research has shown that the few-shot prompting paradigm can also apply to small language models (Schick and Schütze, 2020). Prompt-tuning is a form of Parameter-Efficient Fine-Tuning (PEFT) that learns a soft prompt to cue language models on specific downstream tasks. Importantly, prompt-tuning casts downstream tasks as text generation problems. Furthermore, the efficacy of prompt-tuning scales with model size and can outperform few-shot models as scale increases (Lester et al., 2021).

Smaller models generally perform worse at text generation tasks when compared to their larger counterparts (Kaplan et al., 2020). Because smaller language models struggle with text generation, they may fail to reap the rewards of prompt-tuning. In this paper, we define few-shot prompt-tuning, where we apply prompt-tuning in few-shot contexts. We seek to answer two questions about few-shot prompt-tuning. The first question is whether few-shot prompt-tuned models can be used as efficient alternatives to both few-shot prompting and prompt-tuning using $0-$shot examples. The second question is whether few-shot prompt-tuned models improve a small language model's baseline ability to perform relevant causal text generation for a particular task. We explore these questions using gpt2-medium with 355 million parameters outlined in Radford et al. (2019).

2 Related Work

Two types of language model optimizations are relevant to this paper: model-tuning and prompt-based methods.
Model-Tuning: Model-tuning directly modifies a large number of pretrained weights for a downstream task using backpropagation Howard and Ruder (2018), and has been shown to be largely successful in enhancing model behavior for that specific task. The downsides of model-tuning are that a newly created model must be stored separately from its pretrained parent and model-tuning is expensive from a compute perspective. These expenses have led to the development of parameter efficient fine-tuning (PEFT), which encapsulates popular methods like LoRA Hu et al. (2021), p-tuning Liu et al. (2023), prefix-tuning Li and Liang (2021), and prompt-tuning Lester et al. (2021). While these methods differ algorithmically, they use the same basic idea of backpropagation on some smaller subset of either model weights or independently initialized parameters to optimize performance for some downstream task. These methods often perform equally as well as model-tuning. In this paper, we will examine an ablation on the method prompt-tuning specifically.

Prompt-based Methods: As language models scale, they become more capable at a wide range of tasks, foregoing the need for model-tuning. Particularly, models that utilize decoder-only architectures like GPT-3.5 perform well on a wide range of tasks in $0-$shot settings (Ye et al., 2023). Examples of other models that perform well without model-tuning include Mixture of Experts (MoE) models such as Mixtral-8x7b (Jiang et al., 2024). As these more capable models are able to complete tasks without model-tuning, new methods of optimizing behavior have emerged. The most relevant methods to our paper aim to optimize the textual inputs to these models themselves, and do not modify the parameters of these models at all. Some examples include chain-of-thought prompting Wei et al. (2023) and few-shot prompting. The impact of these methods increases positively with scale; generally these strategies are shown to be effective and faithful in models that have upwards of 10 billion parameters (Lanham et al., 2023). However, few-shot prompting specifically has been shown to work with smaller models under the right circumstances (Schick and Schütze, 2020). Our paper aims to combine the benefits of tuning as well as prompt-based methods in order to optimize small-model performance in the context of text generation for relevant task-specific outputs. Specifically, we will use few-shot prompting in combination with prompt-tuning to learn a soft-prompt that encodes task-specific information from the few-shot prompt it referenced during training. To the best of our knowledge, this method is entirely novel.

3 Approach

Briefly, we will describe few-shot prompting and prompt-tuning, as well as motivate our use of them. Few-shot prompting manipulates the prompts that the models receive by prepending extra examples onto the target input sequence (Brown et al., 2020). Few-shot prompting does not change model weights. Soft prompting, in this case prompt-tuning, is a PEFT method.

The basic idea behind prompt-tuning is to create a soft prompt $P_e \in \mathbb{R}^{p \times e}$ where p is the token length of the soft prompt and e is the dimension of the embedding space. This soft prompt gets prepended to $X_e \in \mathbb{R}^{n \times e}$: a matrix in e embedding space of n tokens representing the input to what we are classifying. The goal then becomes maximizing

$$P\ (\ Y \mid [P_e\ ;\ X_e]\ )$$

the probability of generating the sequence of tokens representing the correct answer choice Y given the concatenated soft prompt and input embedding $[P_e\ ;\ X_e] \in \mathbb{R}^{(p+n)\times e}$. For ARC-E and ARC-C (elucidated below), Y is the sequence of tokens representing one of $\{0, 1, 2, 3\}$ and for Winogrande (also elucidated below), Y is the sequence of tokens representing one of $\{0, 1\}$. To maximize $P\ (\ Y \mid [P_e\ ;\ X_e]\ )$, the model weights are frozen and only the parameters of P_e are updated end-to-end via backpropagation. Under our few-shot prompt-tuned paradigm, the soft prompt parameters are updated to minimize loss when prepended onto a few-shot dataset as opposed to a traditional zero-shot dataset.

Our main approach will be to train a PEFT-gpt2-medium using prompt-tuning on $n-$shot datasets, where $n \in \{0, 1, 3\}$. Then, we will compare each PEFT-gpt2-medium to our calculated baselines to

The Soft-Prompt (Prompt-Tuning) Paradigm

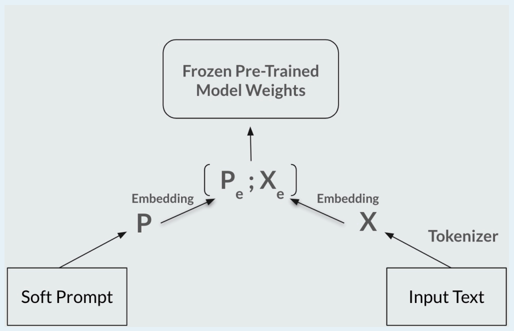


Figure 1: Visualization of prepending a soft-prompt to input text. P represents the p tunable tokens and P_e represents the embedding of those p tokens. X represents the n input tokens and X_e represents the embedding of those n tokens.

evaluate performance.
The datasets we will train and evaluate on will be:

1. Winogrande (Bisk et al., 2019)
2. AI2 Reasoning Challenge (ARC-e) (Clark et al., 2018)
3. AI2 Reasoning Challenge (ARC-c) (Clark et al., 2018)

We chose these datasets for the following reasons. First, we want a breadth of tasks that were intuitively structured for few-shot contexts. Second, we want a breadth of areas as well as a wide range of model performance, so we explicitly use the related albeit distinct datasets ARC-e (ARC-Easy) and ARC-c (ARC-Challenge). We also included Winogrande to measure performance on commonsense context-dependent related tasks.
The baselines we will use is the standard baseline of an $n-$shot prompted out of the box gpt2-medium, $n \in \{0, 1, 3\}$ (Touvron et al., 2023), (Jiang et al., 2023). We will get these baselines on the validation set for each dataset, then evaluate each PEFT-gpt2-medium model on the same validation set in a $0-$shot manner. We believe that training a PEFT model using few-shot prompting is original, even if prompt tuning and few-shot prompting are well-established techniques. We will be using the Hugging-Face implementation for prompt-tuning, as well as for extracting, training, and evaluating models.

4 Experiments

4.1 Data

The specific datasets we used are few-shot modifications on top of Winogrande, ARC-C, and ARC-E. Winogrande follows the Winograd Schema Challenge of having a language model disambiguate a pronoun by citing what the pronoun refers to. The Winogrande dataset modifies this schema by having one sentence with a single _ character in it and asking the model to choose between two nouns to best "fill in the blank". The ARC datasets simply consist of grade-school level, multiple-choice science questions. The ARC dataset is split into challenge (ARC-C) and easy (ARC-E) datasets. The ARC datasets consist of questions with four multiple choice answers. For exact sizes of the datasets we used see A.2 of the appendix.

For our training dataset, we create an $n-$shot train_train dataset which consists of 80% of the original train dataset, as well as a $0-$shot train_eval dataset which consists of 20% of the original train dataset. We use the train_eval dataset to evaluate the training loss and optimize soft prompts using. The split and order of what examples we use in both datasets is random, but we ensure there are equal numbers

of each label in the train_eval dataset. We use a $0-$shot train_eval dataset despite training with a $n-$shot train_train dataset because our metric is performance on $0-$shot prompts, thus we want to optimize our model's performance for this task. In this way, we avoid using the evaluation dataset in training while still optimizing performance for zero-shot datasets.
The structure of an example is as follows:

$$n \times (\{\text{Question}\}\{\text{Answers}\}\{\text{Answer Completion}\})\,\{\text{Task Description}\}\{\text{Answer:}\}$$

The fields Question, Answers, and Answer Completion are fields created directly from each dataset. Answer: is a cloze completion (Schick and Schütze, 2021), and Task Description is qualitatively prompt-engineered for each specific task (see Appendix A.6). We test and report our accuracies on the evaluation dataset for each task, as the test set for wino is unlabeled.

4.2 Evaluation method

The metrics we will utilize are:

1. Accuracy, Precision, Recall, F1
2. Immediately Preceding Influence (custom metric)
3. Incorrect Generation (custom metric)

Recall the first question we seek to answer: Is few-shot prompt-tuning a viable alternative to few-shot learning and traditional, $0-$shot prompt-tuning? To answer this question we compare accuracy, precision, recall, and f1 score across different models trained on different $n-$shot prompts. In particular, we compute:

1. Raw model accuracies evaluated on $\{0, 1, 3\}-$shot validation sets
2. $0-$shot prompt-tuned model accuracy evaluated on a $0-$shot validation set
3. $\{1, 3\}-$shot prompt-tuned soft-prompt evaluated on a $0-$shot validation set.

We also do this with precision, recall, and f1 concurrently. Note that the validation sets that the models use are all the same, save for the few-shot validation set that the raw model uses (this dataset consists of the same ultimate example, but has $n-$shot examples prepended to it). In addition to these metrics, we define two of our own metrics.

Immediately preceeding influence: This metric serves as a measure for our baseline of few-shot prompted examples. Few-shot prompting provides the model with $n-$shot examples of question answer pairs. Immediately Preceding Influence measures the probability the generated answer is x, where $x \in \{0, 1\}$ for the Winogrande dataset or $x \in \{0, 1, 2, 3\}$ for the ARC datasets, given the immediately preceding example's answer was x. With immediately preceding influence, we aim to measure if our baseline model is being overly influenced by the previous example it sees. We utilize this metric only as a way of understanding our model's behavior qualitatively.

Incorrect generation: This metric serves to answer our second question: Can few-shot prompt-tuning be used to generate task-relevant text in a $0-$shot setting? To address this question, we define incorrect generation as the proportion of examples for which the model does not produce a relevant answer; relevant meaning whether it produces a task-specific label within its first two generated tokens. Importantly, incorrect generation is independent from accuracy; as long as the model produces an acceptable answer, the example is counted as correctly generated. We compare the incorrect generation rate for our baselines to a $\{0, 1, 3\}-$shot prompt-tuned model.

4.3 Experimental details

We have two paradigms to evaluate:

1. baseline few-shot prompting
2. experimental few-shot prompt-tuned models

Baseline Evaluation: For our baseline evaluation, we use a HuggingFace pipeline on $n-$shot, $n \in \{0, 1, 3\}$, validation datasets for generation. We set a random seed in order to ensure results are reproducible. We then extract the answer by looking at the first two generated characters. If it does not generate an output within the set of accepted labels, we count this example as an incorrect generation and as an inaccurate answer. For $\{1, 3\}-$shot prompting we also calculate immediately preceding influence. We set temperature to 0 to produce deterministic behavior, repetition_penalty to 0.5 to discourage repetition, and max_new_tokens to 2 to save time and compute. All other parameters are initialized to default HuggingFace values.

Experimental Evaluation: We break experimental evaluation into training and evaluation.
Training: For each dataset, we train three different prompt-tuned models, one for each n in $\{0, 1, 3\}$. We train each model on an $n-$shot train dataset, which has $n-$shot training examples and $0-$shot train-evaluation examples to calculate loss. Due to compute limitations, we could only experiment with different values for one hyperparameter: num_virtual_tokens (the token length of the soft-prompt), or p, where $p \in \{5, 10, 20\}$. For each p, we train a different prompt-tuned model. Thus, in total, we train $n \times d \times p$ models, where $n = 3$, $d = 3$, and $p = 3$ are the total number of shots, datasets, and prompt-lengths, respectively, for a grand total of 27 different models. We keep epochs and learning rate constant; we use a linear scheduler with AdamW as the optimizer for adaptive learning rate, initialized at 0.4. We initialize epochs to be 8. Due to compute limitations, these parameters were chosen after some light experimentation with accuracies for epoch values in $\{4, 8, 16\}$ and learning rates in $\{0.0035, 0.2, 0.4\}$. Note that the learning rate for prompt-tuning is generally set higher (Lester et al., 2021). For PEFT-model initialization, we set PROMPT_INIT to be random, and set the task type to be generative. Hyperparameters not explicitly mentioned are initialized to default values. Training times and examples per second for each dataset and p can be found in section A.3 of the appendix. Every model was trained on a single NVIDIA $A40$ GPU.

Evaluation: Once each model was done training, we evaluate it on the same $0-$shot dataset that we used to procure our baseline values. Thus, we keep the hyperparameters the same as during baseline generation: temperature$= 0$, repetition_penalty$= 0.5$, and max_new_tokens$= 2$. Importantly, we are only interested in $0-$shot accuracies from these datasets, so we do not evaluate our trained models on 1 or 3 shot evaluation datasets, and therefore do not calculate immediately preceding influence. We calculate accuracy, precision, recall, f1 score, and incorrect generation rates for $p \in \{5, 10, 20\}$; the middle three metrics can be found in section A.5 of the appendix.

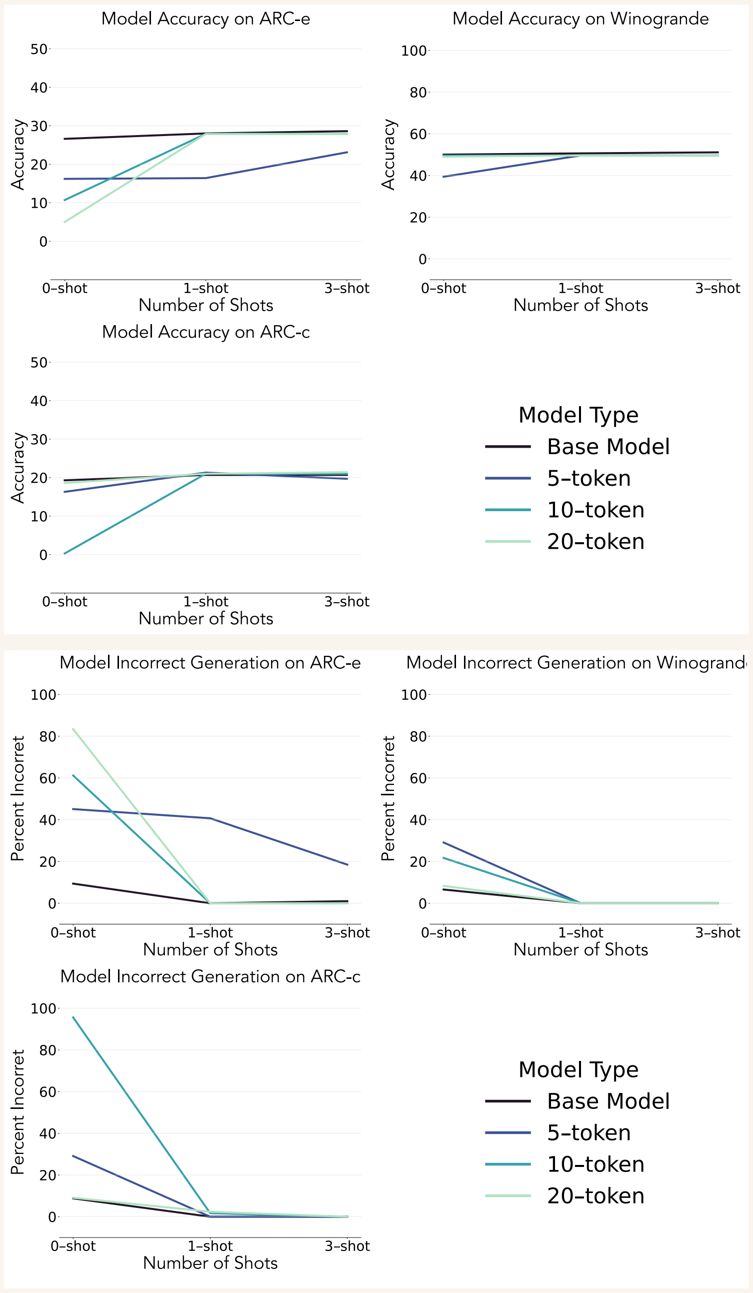
4.4 Results

	$0-$shot	$1-$shot	$3-$shot
Arc-C	-2.9%/1.6%/-4.2%	-6.0%/0.8%/-0.3%	-6.2%/1.2%/-3.5%
Arc-E	-5.0%/-0.2%/-0.7%	-9.9%/-0.2%/-1.7%	-8.2%/-1.1%/-5.2%
Winogrande	18.3%/-0.3%/-1.6%	24.2%/-0.5%/-17.2%	22.2%/-1.3%/-12.0%

Table 1: Difference between $n-$shot experimental and $n-$shot baseline for precision/recall/f1 with soft prompt having 10 virtual tokens

A quick note of clarification: The first set of graphs contain the model accuracies for $0, 1,$ *and* 3 *shot data. The black base model line is few-shot prompted using the number of shots on the x-axis. The dark blue, green, and teal* $\{5, 10, 20\}-$*token lines are prompt-tuned models, trained with the respective amount of shots on the x-axis, then evaluated on the same* $0-$*shot evaluation data as the baseline model. The second set of graphs contains incorrect generations, and values were calculated in the same manner as the first set.*
Our baseline accuracies indicate that gpt2-medium struggles to correctly answer the questions from all three datasets; it has near-random accuracy on ARC-E, ARC-C, and Winogrande. Because our baseline accuracies have a low floor, we aren't surprised that our few-shot soft-prompt technique doesn't out-perform our baselines; it barely increases model performance when applied to an out-of-the-box gpt2-medium. In addition, since the effect of few-shot prompting and prompt-tuning scale with size, (Lester et al., 2021), (Brown et al., 2020), it's unsurprising that a relatively small language model does not perform well. However, what does surprise us is that our few-shot prompt-tuning paradigm is effective at cuing the model to generate contextually relevant text.
This is best exemplified by looking at incorrect generations. For our $0-$shot baseline accuracies, incorrect generation hovers around 5-10%. When we increase the number of shots, incorrect generation drops significantly. This same trend is also exhibited in our prompt-tuned models across all datasets and numbers of virtual tokens. Interestingly enough, models that are prompt-tuned in the traditional way, on $0-$shot data, do not exhibit this trend; as they incorrectly generate more often in

Model Performance Across Datasets



comparison to their few-shot prompt-tuned counterparts, similar to how a $0-$shot out of the box model incorrectly generates more often when compared to its few-shot prompted out-of-the-box models. This suggests that the few-shot prompt-tuned paradigm allows models to learn a representation of the task at hand from a few-shot perspective, meaning that models prompt-tuned on few-shot data exhibit a few-shot understanding of $0-$shot validation examples, albeit inaccurately. This point is further bolstered by the differences in F1, precision, and recall scores; our few-shot prompt-tuned models score similar metrics to our few-shot baseline. Apart from a large increase in precision from Winogrande, the difference between all baseline and model metrics differs by less than 10%. This evidence of similar performance suggests that prompt-tuned models, when trained on few-shot data, exhibit few-shot knowledge and behavior.

5 Analysis

In this section, we will investigate three questions:

1. Why is the baseline performance of gpt2-medium near-random?
2. What do the outputs actually look like?
3. What's really going on here?

Investigation 1: Why is it Random?: One factor that contributes heavily to the stochastic output is the size of our model; gpt2-medium struggles with performing well on the given tasks, as is evident from our zero shot baselines. Few-shot accuracy is not that much better; this suggests that gpt2-medium is unable to gain contextual value from seeing more examples. Rather, it learns to imitate the desired behavior, which is outputting a well-formed generation that corresponds with the labels of the given dataset. To see this more clearly, we turn to our custom metric of immediately preceding influence. Here, we see that the model is biased toward answering 0 when zero is immediately before

	1−shot	3−shot
Arc-C	100%/5.0%/0%/0%	100%/5.0%/0%/0%
Arc-E	99.4%/3.3%/0%/0%	100%/31.2%/0%/0%
Winogrande	93.7%/81.4%	85.7%/19.0%

Table 2: Immediately Preceding Influence (IMI) for $\{1, 3\}-$shot across all our datasets. For Arc-C and Arc-E the four numbers indicate probability that, given the immediate prior example had answer $x \in \{0, 1, 2, 3\}$, the model answered with x. For Winogrande, the two numbers indicate probability that, given the immediate prior example had answer $x \in \{0, 1\}$, the model answered with x. As an example, the 93.7% in Winogrande indicates that if the model saw a 0 answer for the 1-shot answer, there was a 93.7% chance that it would generate a 0.

it in a few-shot example. In addition, we see that the probability of answering 2 and 3 for the ARC datasets is zero for both 1 and 3 shots. In the case of Winogrande, the $1-$shot model is biased towards whatever answer it just saw in the previous shot, but the $3-$shot model learns to be biased towards zero. This indicates that the model has a strong propensity for imitation, particularly imitation in the simplest form. Outputting zero does not require gpt2-medium to be able to reason about the tasks at hand, and effectively gives random accuracy, explaining away the low accuracies. To further support this argument, we consider some of the specific outputs.

What does the output actually look like?: We claim that gpt-2medium has little ability to logically reason about the answer to a task. Consider a $0-$shot prompt from ARC-C. To assess whether gpt2-medium base has reasoning capabilities, we augment our generation to allow for more new tokens; the intuition here is to see whether the model, from a base level, is able to reason about the problem in a logical way. We see that the model repeats an incoherent statement while mimicking the structure of the question. We see that this behavior is usually present whenever the model fails to generate a task-specific label; from what we've observed, the model will generally parrot either the question itself or the structure of the question, outputting nonsensical sentences. Yet what about when the model outputs a well-formed generation? Doesn't that imply that the model has some sort of knowledge about the task? To address these questions, consider the output after a well-formed answer. We see that the answer is well-formed, as it includes a 0 at the start, but it is also indicative of

Question: Several kinds of birds live on a small island. A storm destroys most of the plants on which the birds feed and build their nests. Which kind of bird is most likely going to survive on the island?
0: the birds that have the greatest number
1: the birds that are most able to adapt
2: the birds that can fly the farthest
3: the birds with the strongest beaks
{Standard ARC-C task description here (Appendix A.6)}
Answer **(what the model generates)**: The birds that live on the island are the birds that have the greatest number of beaks.
1: The birds that have the greatest number of beaks are the birds that have the most number of beaks.
2: The birds that have the greatest number of beaks are the birds that have the most number of beaks
Caption: an instance of incorrect generation, where the model spews nonsense.

Question: Climate change may be reducing the amount of ice floating on the world's oceans. How can this change most likely alter the food supply available to marine consumer organisms?
0: by decreasing the amount of salt dissolved in ocean water
1: by increasing the water's ability to separate into distinct layers
2: by changing temperature conditions that single-celled producers require
3: by preventing organisms from responding quickly to environmental change
{Standard ARC-C task description here (Appendix A.6)}
Answer(**what the model generates**):
0n1n2n3n4n5n6n7n8n9n10n11n12n13n14n15n16n17n18n19n20n21n22n23n24n25n26n27n28
n29n
Caption: an instance of correct generation, but the output is worrying...

the model's poor reasoning capabilities. The model is specifically initialized with an end of sequence token in its vocabulary, but even after generating a well-formed output, the model still continues to generate nonsensical strings. Clearly, this is an issue with the model's reasoning capabilities. Thus, it would be incorrect to determine that few-shot prompt-tuning gives the model a better understanding of the task; rather, few-shot prompt-tuning optimizes model behavior to match the behavior of few-shot prompting, independent from reasoning.

What's Really Going on Here? (Zeroes all the way down)... Our final analysis examines trends in the numerical, well formed output data itself. We know that the model, confirmed by our IPI metric, has a bias towards generating zero. What we find in our analysis is both that this bias extremely strong, and that this bias also amplifies itself when applied to prompt-tuned models. Consider the counts labels for each dataset. We see that $0-$shot prompting elicits extremely heavy bias towards zero, with the next most populated category being invalid outputs. $1-$shot lessens this bias, particularly in Winogrande, but it's still evident in the disparity between 1s and 0s. $3-$shot then reverts to being more biased towards zero again. From our previous investigations, we know that this pattern is not a coincidence, rather, the inability of gpt2-medium to effectively reason about the task at hand biases it towards outputting zero, especially if it saw zero as the label in the shot before it (with the exception of Winogrande, which has a high likelihood of outputting the last label it saw, no matter the value, for one-shot prompting). Thus, the model is choosing 0 the majority of the time; few-shot prompting strengthens its knowledge of what the output should look like, but not which output it should be. This pattern seems to be reinforced after a model is prompt-tuned; we see that after the model is prompt-tuned on a few-shot dataset, the behavior indicates a bias towards zero, which reflects the general, albeit less extreme, trend in the baseline counts. So what does this mean? It seems to imply that the poor baseline model performance exacerbated behavioral trends in prompt-tuned model performance; gpt2's lack of reasoning capability meant that the main contribution of few-shot prompting was to show the model what the output *should* look like, not why it should look that way. Since the model was already inclined towards generating zero, that behavior translated to the few-shot prompt-tuned models. We hesitate to point towards causation, but posit that few-shot prompt-tuning encodes few-shot task-related knowledge (or lack thereof) into a prompt-tuned model, allowing the model to exhibit few-shot tendencies when it is $0-$shot prompted.

	$0-$shot	$1-$shot	$3-$shot
Arc-C	265 / 4 / 0 / 0 / 26	260 / 34 / 0 / 0 / 1	291 / 4 / 0 / 0 / 0
Arc-E	512 / 10 / 0 / 0 / 45	487 / 74 / 0 / 0 / 6	551 / 15 / 0 / 0 / 1
Winogrande	1221 / 19 / 0 / 0 / 27	678 / 589 / 0 / 0 / 0	1053 / 214 / 0 / 0 / 0

Table 3: Counts of labels for each dataset, **out-of-the-box gpt2-medium**. Format: 0s / 1s / 2s / 3s / Invalids. Note that there are only 2s and 3s for the ARC datasets.

	$0-$shot	$1-$shot	$3-$shot
Arc-C	10 / 3 / 1 / 0 / 281	289 / 6 / 0 / 0 / 0	294 / 1 / 0 / 0 / 0
Arc-E	219 / 1 / 0 / 0 / 347	566 / 1 / 0 / 0 / 0	566 / 1 / 0 / 0 / 0
Winogrande	1249 / 1 / 0 / 0 / 17	1266 / 1 / 0 / 0 / 0	1266 / 1 / 0 / 0 / 0

Table 4: Counts of labels for each dataset, **10 token prompt-tuned model** Format: 0s / 1s / 2s / 3s / Invalids. Note that there are only 2s and 3s for the ARC datasets.

6 Conclusion

We have defined and explored the viability of few-shot prompt-tuning. Through extensive training and experimenting, we've found that few-shot prompt-tuning, at least for smaller models, does not offer a viable alternative for improving language model performance on downstream tasks. We have, however, found that few-shot soft-prompting, even for short token lengths of soft prompts, improves a small language model's ability to generate contextually relevant text. We hypothesize that few-shot soft-prompting could be a successful mode of ensuring contextually relevant text generation for large language models as well. The primary limitations of our work are that our accuracy results indicate few-shot soft-prompting is not viable for improving task performance on small language models. We understand this limitation on our paradigm to result from few-shot prompting and prompt-tuning improving with scale (i.e., precisely because we experimented with a small language model). We hypothesize that with proper scale few-shot soft-prompting could be a cost-effective improvement to few-shot learning and traditional prompt-based PEFT methods. For future work, with enough compute, it would be interesting to run the same experiments we ran but with a more powerful language model.

References

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language.

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.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. 2023. Measuring faithfulness in chain-of-thought reasoning.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning.

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.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023. Gpt understands, too.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze questions for few shot text classification and natural language inference.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, Jie Zhou, Siming Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models.

A Appendix (optional)

A.1 Prompt Engineering PIQA (a relic from our milestone)

First Iteration:
Hand-crafted instructions: You are a helpful AI assistant that is being asked a question by a user. Choose the response that best satisfies the question. Your response must be limited to your numeric answer. There is exactly one correct answer.
Prompt: Question: To get the best deal at amusement parks,
0: buy a seasonal pass, so you can go all winter long for one discounted rate.
1: buy a seasonal pass, so you can go all summer long for one discounted rate.
Answer: The question does not specify which season the user is interested in. Therefore, without additional context, it

Issue: The issue is ambiguity in the question and/or answers. Thus, we add a guessing condition to account for this in the second iteration

Second Iteration:
Hand-crafted instructions: You are a helpful AI assistant that is being asked a question by a user. Choose the response that best satisfies the question. If there is ambiguity in the question or answers, you must guess. Your response must be limited to your numeric answer. There is exactly one correct answer.
Prompt: Question: How to make Caramel Chocolate chip Girl Scout Cookie Vanilla Ice cream at home.
0: In a medium mixing bowl combine 7 cups chilled Caramelized Onions, 2 14 oz. cans sweetened condensed milk 1 teaspoon Vanilla extract. Beat with an electric mixer until soft Peaks form. Stir in 1 1/2 cups Caramel Chocolate Chip Girl Scout cookies (Sliced in 3/4" strips or crumbles) and 1/2 cup Chocolate chips. Transfer mixture to two 8x8x2 inch baking pans. Freeze about 8 hours until firm.
1: In a medium mixing bowl combine 7 cups chilled whipping cream, 2 14 oz. cans sweetened condensed milk 1 teaspoon Vanilla extract. Beat with an electric mixer until soft Peaks form. Stir in 1 1/2 cups Caramel Chocolate Chip Girl Scout cookies (Sliced in 3/4" strips or crumbles) and 1/2 cup Chocolate chips. Transfer mixture to two 8x8x2 inch baking pans. Freeze about 8 hours until firm.
Answer: 0 or 1 (I'm unable to determine which one is the correct answer without additional

Issue: The issue is that the model is guessing both, instead of choosing one. The fix is simple, we add another condition in that mandates that the model must choose one.

Third Iteration:
Hand-crafted instructions: You are a helpful AI assistant that is being asked a question by a user. Choose the response that best satisfies the question. You must choose either 0 or 1. If there is ambiguity in the question or answers, you must guess. Your response must be limited to your numeric answer. There is exactly one correct answer.
Prompt: Question: Kill fruit flies easily.
0: Spray with alcohol, then sweep away.
1: Spray with water, then sweep away.
You are a helpful AI assistant that is being asked a question by a user. Choose the response that best satisfies the question. You must choose either 0 or 1. Your response must be limited to ONLY your numeric answer (one single character), or else the world will end. There is exactly one correct answer.
Answer: 0. Alcohol is more effective in killing fruit flies than water.

Conclusion and Notes: Here, we see that this prompt worked at eliciting the correct response from the model. To ensure this behavior was consistent, we manually checked responses to make sure whether the model was behaving as intended or not.

A.2 Dataset size information

Split	ARC-c	ARC-e	WINOGRANDE
train-train	893	1793	7398
train-eval	224	448	1850
validation	295	567	1267

Table 1: The size of the splits of each dataset.

A.3 Training Time Details

Model Task	ARC-c	ARC-e	WINOGRANDE
Zero-Shot	10:31/22.724	05:05/23.358	23:07/42.974
One-Shot	13:41/17.451	06:29/18.351	29:42/34.566
Three-Shot	20:20/11.756	12:08/9.804	42:03/22.876

Table 2: Training info, soft prompt length=5, training time(min:s)/samples per second

Model Task	ARC-c	ARC-e	WINOGRANDE
Zero-Shot	10:38/22.471	05:08/23.128	24:04/40.974
One-Shot	13:50/17.628	06:32/18.212	30:41/33.398
Three-Shot	20:53/11.444	12:16/9.706	43:06/24.01

Table 3: Training info, soft prompt length=10, training time(min:s)/samples per second

Model Task	ARC-c	ARC-e	WINOGRANDE
Zero-Shot	10:56/21.836	05:20/22.281	25:54/38.067
One-Shot	14:13/16.804	06:45/17.627	31:35/31.224
Three-Shot	20:53/11.444	12:30/9.52	45:09/21.842

Table 4: Training info, soft prompt length=20, training time(min:s)/samples per second

A.4 Parameter Sizes

trainable params: 5,120 || all params: 354,828,288 || trainable%: 0.00144
trainable params: 10,240 || all params: 354,828,288 || trainable%: 0.00288
trainable params: 20,480 || all params: 354,828,288 || trainable%: 0.05760

A.5 Precision, Recall, F1 for baseline and 5, 20 virtual tokens

	$0-$shot	$1-$shot	$3-$shot
Arc-C	5.4%/23.4%/8.7%	11.4%/24.2%/9.1%	11.6%/23.8%/12.3%
Arc-E	12.0%/25.2%/11.6%	16.9%/25.2%/12.6%	15.2%/26.1%/16.1%
Winogrande	56.5%/50.4%/34.9%	50.6%/50.6%/50.5%	52.6%/51.4%/45.3%

Table 5: $n-$shot baseline precision/recall/f1 scores

	$0-$shot	$1-$shot	$3-$shot
Arc-C	16.6%/26.2%/12.0%	5.4%/25.0%/8.8%	5.1%/23.0%/8.3%
Arc-E	19.8%/25.6%/12.5%	13.0%/24.2%/13.2%	7.1%/25.0%/11.1%
Winogrande	75.1%/50.1%/33.6%	74.8%/50.1%/33.3%	74.8%/50.1%/33.3%

Table 6: $n-$shot precision/recall/f1 scores for soft prompt with 5 virtual tokens

	$0-$shot	$1-$shot	$3-$shot
Arc-C	5.1%/25.0%/8.5%	5.4%/25.0%/8.9%	5.4%/25.0%/8.8%
Arc-E	7.7%/25.0%/11.8%	7.0%/25.0%/10.9%	7.0%/25.0%/10.9%
Winogrande	74.8%/50.1%/33.3%	74.8%/50.1%/33.3%	74.8%/50.1%/33.3%

Table 7: $n-$shot precision/recall/f1 scores for soft prompt with 20 virtual tokens

A.6 Prompts

ARC-C 0-shot Prompt:

```
Question: Juan and LaKeisha roll a few objects down a ramp. They want to see which object rolls the farthest. What should they do so they can repeat their investigation?
0: Put the objects in groups.
1: Change the height of the ramp.
2: Choose different objects to roll.
3: Record the details of the investigation.
You are a helpful AI science assistant. A grade-school student is asking for your help. Choose the option that most correctly answers the question. Your answer must be either 0, 1, 2, or 3. If you are unsure, you must guess between the four options. Importantly, your response must begin with and be limited to ONLY your numeric answer (one single character). There is exactly one correct answer.
Answer:
```

ARC-E 1-shot Prompt:

```
Question: Which best describes the Sun?
0: very small white dwarf
1: medium yellow dwarf
2: large blue giant
3: red supergiant
Answer:1
Question: Which technology was developed most recently?
0: cellular telephone
1: television
2: refrigerator
3: airplane
You are a helpful AI science assistant. A grade-school student is asking for your help. Choose the option that most correctly answers the question. Your answer must be either 0, 1, 2, or 3. If you are unsure, you must guess between the four options. Importantly, your response must begin with and be limited to ONLY your numeric answer (one single character). There is exactly one correct answer.
Answer:
```

Winogrande 1-shot Prompt:

```
Sentence: Adam decided to shave Denniss beard before the reuinion, because _ thought it was too long.
0: Adam
1: Dennis
Answer:0
Sentence: Sarah was a much better surgeon than Maria so _ always got the easier cases.
0: Sarah
1: Maria
You are a helpful AI assistant that completes sentences by filling in blanks. You must choose the option that best replaces the _ character, meaning you must choose either 0 or 1. If you are unsure, or if the question and/or options are ambiguous, you must guess between the two options. Importantly, your response must begin with and be limited to ONLY your numeric answer (one single character). There is exactly one correct answer.
Answer:
```

ARC-C 3-shot Prompt:

```
Question: Which trait of a pet dog is inherited?
0: The dog avoids a pet cat.
1: The dog jumps on a sofa.
2: The dog rolls over on command.
3: The dog drools when it smells food.
Answer:3
Question: Some different types of plants have characteristics in common. Which characteristic do most plants share?
0: the size of their roots
1: the shape of their leaves
2: the color of their flowers
3: the structure of their cells
Answer:3
Question: Which of the following genetic conditions results from a problem with segregation?
0: Trisomy 16: a condition caused when a zygote receives three copies of chromosome 16
1: Huntington’s disease: a condition caused when a zygote receives a mutated dominant allele
2: Hemophilia: a condition caused when a zygote receives an X chromosome with a particular recessive allele
3: Sickle cell anemia: a condition caused when a zygote receives a recessive allele for hemoglobin from each parent
Answer:0
Question: Juan and LaKeisha roll a few objects down a ramp. They want to see which object rolls the farthest. What should they do so they can repeat their investigation?
0: Put the objects in groups.
1: Change the height of the ramp.
2: Choose different objects to roll.
3: Record the details of the investigation.
You are a helpful AI science assistant. A grade-school student is asking for your help. Choose the option that most correctly answers the question. Your answer must be either 0, 1, 2, or 3. If you are unsure, you must guess between the four options. Importantly, your response must begin with and be limited to ONLY your numeric answer (one single character). There is exactly one correct answer.
Answer:
```