Two Examples are Better than One: Context Regularization for Gradient-based Prompt Tuning

Hyeonmin Ha[1] **Soyoung Jung**[1] **Jinsol Park**[1]
Minjoon Seo[2] **Seung-won Hwang**[1] **Byung-Gon Chun**[1,3]
[1]Seoul National University [2]KAIST AI [3]FriendliAI
{hyeonmin.ha, sy.jung, jinsolpark, seungwonh, bgchun}@snu.ac.kr
minjoon@kaist.ac.kr

Abstract

Prompting has gained tremendous attention as an efficient method for the adaptation of large-scale language models. However, prompts often act against human intuition and report unstable performances, which has motivated methods that automatically find effective prompts. One popular approach is gradient-based search, which iteratively updates a (randomly) initialized prompt towards the optimal one with the guide of gradients. We propose a novel regularization method, CoRe, for gradient-based prompt tuning techniques, which guides a prompt to produce a task context properly. CoRe realizes two regularization effects — context attuning and context filtering — that improve prediction performance in a zero-shot in-context learning setting where a model makes inferences only with the prompt tuned by CoRe, without any demonstration examples for in-context learning. Context attuning guides the context generated by the input and the tuned prompt toward embedding the appropriate context for the task. In our theoretical analysis, regularizing the context extends to improving zero-shot in-context learning performance. Context filtering steers the prompt to select only the task-related context so that context attuning solely focuses on creating and sending the right task context. We evaluate CoRe on natural language understanding datasets and two large language models, GPT2-XL and GPT-J. Our training scheme shows performance improvements up to 11.9% on GPT2-XL, and up to 6.3% on GPT-J in zero-shot settings.

1 Introduction

A prompt is a carefully composed input to adapt a pretrained language model (LM) with in-context learning, and has gained attention as a new method for the adaptation of LMs. Radford et al. (2019) and Brown et al. (2020) showed that pretrained LMs can be transferred to various downstream NLP tasks by designing prompts to be aligned with pretraining objectives and downstream tasks. This method has a unique capability in that it does not mandatorily require training examples or parameter access for LM adaptation. Moreover, Radford et al. (2019) suggested, and Brown et al. (2020) systematically explored that providing demonstration examples in the prompt can also boost the performance of prompting.

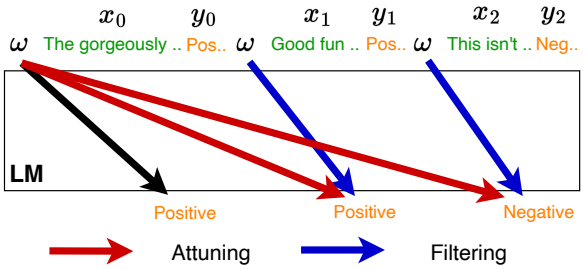

Figure 1: Abstracted illustration of the two regularizers, context attuning (red line) and context filtering (blue line), of CoRe with sentiment analysis data. The black line indicates regular gradient-based prompt tuning where an input sequence consists of one data example (x_0 and y_0) and the prompt ω_0 is optimized with respect to y_0 only.

Initially, a prompt is manually crafted by a developer with human intuition. However, because of the unstable performances and unpredictable behaviors of manual prompts (Liu et al., 2021), automatic methods for prompt tuning have been proposed. One of the most popular paradigms is optimizing prompts using stochastic gradient descent (SGD) (Lester et al., 2021; Shin et al., 2020; Liu et al., 2021; Zhong et al., 2021) — gradient-based prompt tuning, which updates prompt tokens or parameterized embeddings with a guide of gradients.

To further boost the performance of gradient-based prompt tuning methods, this paper suggests a new context regularization scheme, CoRe. CoRe introduces two regularizers: context attuning and context filtering. Context attuning guides a prompt ω and an example $\{x, y\}$ to generate an appropri-

Findings of the Association for Computational Linguistics: ACL 2023, pages 3335–3350
July 9-14, 2023

ate task context without being biased towards the example. This is realized as a concatenation of multiple examples, resulting in a single sequence including the examples, as the concatenation of the three examples in Figure 1. Context attuning then optimizes the first prompt ($\boldsymbol{\omega}$ that comes before x_0) towards minimizing the losses of the succeeding examples (y_1, y_2). Therefore, the prompt does not become biased to $\{x_0, y_0\}$, but is generalized for the appended examples as well and generates less biased task contexts.

However, context attuning alone does not show the effect of regularization because the following examples receive too diverse contexts, such as styles, topics, relationships between entities, or task information. Such diversity hinders the optimization of the regularizer towards the task context. To resolve this problem, we add another regularizer, context filtering, to adequately extract the task context from the preceding examples. Specifically, following the example in Figure 1, context filtering (blue line) optimizes prompts given preceding examples — optimization of ω_1 with respect to y_1 when $\{x_0, y_0, x_1\}$ is given.

We evaluate the effect of CoRe in a zero-shot in-context learning (ICL) setting as most prompt tuning methods have focused on the setting. In zero-shot ICL, a tuned prompt and a test input is only given without any additional demonstration examples. While ICL often assumes demonstration examples are given, for simplicity we call this setting zero-shot ICL. It can be also seen as a typical zero-shot instruction following.

Context attuning and context filtering improve the model's performance in a zero-shot in-context learning (ICL) setting even though the prompts have been trained in a few-shot ICL setting (i.e., concatenation of examples). We conjecture that this successful transfer from a few-shot setting to a zero-shot setting is possible since prompts have been tuned with the guide of CoRe to generate a more generalized task context. Such a generalized task context is effective regardless of concatenated examples, thus improving the performance in a zero-shot setting. We highlight that this transferability from few-shot settings to zero-shot settings is noteworthy and verify it on a theoretical framework based on the hidden Markov model and Bayesian inference (Xie et al., 2022).

To the best of our knowledge, this work is the first to leverage interactions between examples in a sequence for the purpose of regularization. There are several works training on multiple examples in a sequence (Min et al., 2021; Chen et al., 2022; Gao et al., 2021). They focused on improving few-shot in-context learning by maximizing the likelihood of an example given demonstration examples. Our work is distinct from the above works in two aspects. First, we target zero-shot settings. Second, we introduce a novel regularization technique — context attuning— to improve zero-shot performance by leveraging the influence between multiple examples within a sequence.

We evaluate CoRe on two large autoregressive language models, GPT2-XL (Radford et al., 2019) and GPT-J (Wang and Komatsuzaki, 2021), and 8 NLU datasets: five from SuperGLUE, SST2, AGNews, SNLI. CoRe enhances the performances of zero-shot inference up to 11.9% on GPT2-XL and 6.3% on GPT-J when applied to P-tuning (Liu et al., 2021), Prefix-tuning (Li and Liang, 2021), and Softprompt-tuning (Lester et al., 2021).

2 Background

In-context learning (ICL) is a relatively new paradigm in the adaptation of pretrained language models (LMs). ICL have been used to adapt pretrained LMs to numerous downstream tasks such as Natural Language Inference (Schick and Schütze, 2020), text classification (Gao et al., 2021; Puri and Catanzaro, 2019; Schick and Schütze, 2020), question answering (Jiang et al., 2021; Khashabi et al., 2020; Raffel et al., 2020) and many more.

In ICL, a prompt is a task specification given as part of the input and aims to guide the language model into generating the desired output according to the given task. A prompt usually consists of a task description, a template of the input with placeholders for data, and optionally demonstration examples. For example, when given an input "**Translate English to Spanish:** I love you →", the LM is expected to finish the input by generating the answer "te amo". This is different from fine-tuning, where the parameters of the LM are updated to optimize the translation performance.

Compared to fine-tuning, which is a representative LM adaptation technique, ICL is more memory-efficient when we want the LM to process various different tasks. In ICL, the LM parameters are typically frozen, and developers only have to keep at most dozens of tokens per task, which are task descriptions and input templates. However,

for fine-tuning, developers have to keep different versions of parameters per task, which can be as huge as 540 billion parameters (Chowdhery et al., 2022) in the status quo.

2.1 Manual prompt tuning

In the early stages of prompting, prompts were often handcrafted with heuristics. Radford et al. (2019) and Brown et al. (2020) demonstrate the transferability of pretrained LMs using manual prompts. These two works show that LMs can perform well on diverse NLP tasks when provided with some adequate prompts.

In particular, Brown et al. (2020) highlights in-context learning, a new training scheme where several demonstration examples are appended in front of the target input. LMs are expected to learn the task by referring to the prepended demonstrations only. Like other GPTs, GPT3 is also known to require carefully crafted prompts to get the task done adequately.

In subsequent works, manual prompts have been applied to a number of research areas such as factual probing (Petroni et al., 2019; Jiang et al., 2020a), knowledge mining (Davison et al., 2019), question answering (Khashabi et al., 2020; Raffel et al., 2020), and text classification (Puri and Catanzaro, 2019; Schick and Schütze, 2020). Despite their remarkable adaptation ability, manually crafted prompts have exhibited unstable performances. Besides, prompts that show good performance are, in fact, quite against commonsense, unable to understand why some specific prompts work well (Liu et al., 2021).

2.2 Automatic prompt tuning

Along with the unstable performances and unaccountable behavior of manual prompts, creating manual prompts involves extensive human effort in devising and evaluating numerous handcrafted templates of different patterns. Therefore, recent studies suggest techniques to automatically search for an optimal prompt.

One of the most popular methods is gradient-based methods, which keep updating prompt tokens or parameterized prompt embeddings leveraging gradients. For an example, Lester et al. (2021), OPTIPROMPT (Zhong et al., 2021), and P-tuning (Liu et al., 2021) first concatenate initialized embeddings for prompts with the embeddings of input data in a pre-defined order. They then directly optimize prompt embeddings on a given task dataset. The optimized prompt embeddings are used at inference time without any modification.

Other than gradient-based methods, Jiang et al. (2020b) proposes a mining-based method that uses the Wikipedia text corpus to extract a prompt. Gao et al. (2021) uses the pretrained T5 model (Raffel et al., 2020) to fill in the missing spans of a given input sentence to construct a prompt. Guo et al. (2021) proposed an RL method for a controlled generation of LM, and applied it to prompt generation.

3 Approach

Our key approach to improve SGD-based prompt tuning is to concatenate multiple examples and regularize a context generated by them, which transfers to improvement in zero-shot prediction. Our training scheme first concatenates multiple examples from the same task. Taking an example of concatenating three examples $(x_0, y_0, x_1, y_1, x_2, y_2)$ as depicted in Figure 1, a model is given an input $S = \{\boldsymbol{\omega}, x_0, y_0, \boldsymbol{\omega}, x_1, y_1, \boldsymbol{\omega}, x_2, y_2\}$ where $\boldsymbol{\omega}$ is the tunable prompt. Then, CoRe introduces two regularizers: context attuning and context filtering. The context attuning regularizer steers an example to send the correct task context to the succeeding examples, while the context filtering regularizer guides the succeeding examples to receive only the task context among diverse contexts from preceding examples.

We target zero-shot ICL where only a prompt and a test input are provided (i.e., no demonstration examples). During inference, a model is given $\boldsymbol{\omega}, x_i$ and expected to predict y_i where $\boldsymbol{\omega}$ has been trained with CoRe. We denote this setting as *zero-shot in-context learning* to imply that the model learns from the context provided by the tuned prompt, but we do not give any demonstration examples (zero-shot example for in-context learning). Please note that this differs from the typical zero-shot setting. Although the model still does not see any training example for making predictions, training data has been used when optimizing the prompt with CoRe.

We would like to highlight our finding that prompts tuned with CoRe in a few-shot ICL setting show improved performances in a zero-shot setting. CoRe's two regularizers – context attuning and context filtering– are applied on a concatenation of multiple examples (i.e., few-shot ICL), and the regularizers steer the prompt during train-

ing time to generate a more generalized task context for following examples (x_1, x_2) in a sequence. Such a well-generated task context (generated by $\boldsymbol{\omega}$) is propagated not only to the following examples (x_1, x_2) but also to the example (x_0) paired with the prompt. Therefore, a prompt tuned with CoRe helps the model's prediction when given an input of a single example ($S = \{\boldsymbol{\omega}, x_i\}$), which is equivalent to a zero-shot ICL setting.

Throughout this section, we use a simple form of prompt for conciseness of the explanation but without loss of generality: $\{\boldsymbol{\omega}, x_i, y_i\}$ where $\boldsymbol{\omega}$ is a tunable prompt. Our work focuses on autoregressive language models (LMs), so our method is not explored or analyzed upon autoencoding LMs such as BERT (Devlin et al., 2019).

3.1 Context attuning

Context attuning prevents a prompt from being biased to a single example for generalization on the zero-shot ICL setting, using regularization on a context. In detail, this regularizer guides a prompt $\boldsymbol{\omega}$ and an example $\{x, y\}$ to create an appropriate task context without being biased towards the example.

3.1.1 Mechanism

To realize the goal, we first put multiple examples in an input sequence so that examples can exchange the task context with one another. This is different from typical SGD-based prompt tuning where each input sequence has only one training example, and the gradients cannot flow between the examples. In the sequence, the appended examples attend to the preceding examples and are influenced by the context of preceding examples during predictions. The prompt of the first example plays a key role in context attuning— it minimizes not only the losses of the first example (black line in Figure 1) but also the losses of the succeeding examples (blue line in Figure 1). We hypothesize that such optimization inherently regularizes a task context from the prompt and prevents the prompt from being too biased to a single example, ultimately achieving better generalization ability.

We give a formal definition of the context attuning regularizer. In autoregressive LMs, as depicted in Figure 1 with the red lines, our training method optimizes as the following:

$$\boldsymbol{\omega} \longleftarrow \boldsymbol{\omega} - \epsilon \cdot \nabla_{\boldsymbol{\omega}_0} \sum_{k>0} p_{\boldsymbol{\theta}}(y_k | S_{<k}, \boldsymbol{\omega}_k, x_k), \quad (1)$$

where $S_k = \{\boldsymbol{\omega}_k, x_{i,k}, y_{i,k}\}$ is the k-th example in the sequence, $S_{<k} = \{S_j | j < k\}$, $\boldsymbol{\omega}_k$ is a trainable parameterized prompt, s is the number of concatenated examples, and $\boldsymbol{\theta}$ is the parameters of the LM. $\boldsymbol{\omega}_k (k \neq 0)$ is the same with $\boldsymbol{\omega}_0$ but regarded as a constant and not optimized. We introduce k to represent the positions of the prompts and indicate which prompts are optimized. On our method, the prediction of each example $S_{i,j}$ is conditioned on the prompt $\boldsymbol{\omega}_0$ paired with the first example, *"The gorgeously.."* in Figure 1 for an example, and the prompt is optimized for multiple examples.

We were inspired by few-shot ICL (Radford et al., 2019; Brown et al., 2020) when designing the context attuning regularizer. When predicting the answer for a new input, few-shot ICL prepends a few *demonstrations* — input-answer pairs — to the new input and condition on those demonstrations to understand the task. The success of few-shot in-context learning has shown that prepended examples provide some meaningful information (e.g., task context) to succeeding data. Few-shot ICL leverages the interaction among examples during inference; our method can be considered as leveraging the same advantage during training (prompt tuning) to achieve better generalization.

3.1.2 Theoretical analysis

We analyze how the regularizer affects context attuning using a theoretical framework from Xie et al. (2022) and verify that context attuning improves zero-shot inference. Xie et al. (2022) designed the framework to explain what enables in-context learning of LMs. They viewed an LM as Bayesian inference with a hidden Markov model (HMM), and the transitions of the HMM are parameterized by a latent concept $\mathbf{c}$, which represents a task. On their framework, our regularizer can be expanded as follows:

$$p(y_k | S_{<k}, \boldsymbol{\omega}_k, x_k) \propto \int_{\mathbf{c}} \sum_{h_k^s} p(y_k | x_k, h_k^s, \mathbf{c}) p(h_k^s | S_{<k}, \boldsymbol{\omega}_k, x_k, \mathbf{c}) \, p(S_{<k}, \boldsymbol{\omega}_k, x_k | \mathbf{c}) p(\mathbf{c}) \, d\mathbf{c}, \quad (2)$$

where h_k^s is a hidden state corresponding to x_k. Note that we omit the index of sequence i as our analysis is done on a single sequence.

We are interested in how the regularizer attunes the task context and generalizes to maximize $\sum_i p(y_{i,0} | \boldsymbol{\omega}_0, x_{i,0})$, which is zero-shot inference. On the framework, our regularizer affects the terms

that $\boldsymbol{\omega}_0$ is involved in: $p(h_k^s|S_{<k}, \boldsymbol{\omega}_k, x_k, \mathbf{c})$ and $p(S_{<k}, \boldsymbol{\omega}_k, x_k|\mathbf{c})$. We analyze the two terms to identify the effect of the cross-data regularizer on zero-shot inference.

First, optimizing $p(h_k^s|S_{<k}, \boldsymbol{\omega}_k, x_k, \mathbf{c})$ prevents the prompt from being biased towards a single data instance by optimizing the task context passed to the following examples. The term we optimize can be expanded as follows:

$$p(h_k^s|S_{<k}, \boldsymbol{\omega}_k, x_k, \mathbf{c}) = \sum_{h_0^s} p(h_k^s|h_0^s, S_{<k-1}^{-\boldsymbol{\omega}_0}, \boldsymbol{\omega}_k, x_k, \mathbf{c}) p(h_0^s|\boldsymbol{\omega}_0, x_0, \mathbf{c}), \tag{3}$$

where $S_{<k-1}^{-\boldsymbol{\omega}_0} = S_{<k-1} - \{\boldsymbol{\omega}_0\}$. Optimizing only zero-shot inference $p(y_0|\boldsymbol{\omega}, x_0)$ may cause the hidden state h_0^s to be biased to the data instance. However, with our regularization, the prompt $\boldsymbol{\omega}_0$ is tuned to pass the proper task context via the hidden state h_k^s to the following examples. To increase the probability of such hidden state, the prompt should be tuned to increase the probability $p(h_0^s|\boldsymbol{\omega}_0, x_0, \mathbf{c})$ where h_0^s is likely to transit to the proper hidden state h_k^s, where the first term of RHS of Equation 3, $p(h_k^s|h_0^s, S_{<k-1}^{-\boldsymbol{\omega}_0}, \boldsymbol{\omega}_k, x_k, \mathbf{c})$, is high. Since h_0^s is unlikely to transit the hidden states that embed the proper task context when h_0^s has already been biased to a specific data instance, the bias is naturally avoided. Finally, the unbiased hidden state h_0^s improves the average zero-shot inference performance, which is the average of Equation 2 over the dataset where $k = 0$.

Moreover, the optimization also calibrates $p(S_{<k}, \boldsymbol{\omega}_k, x_k|\mathbf{c})$ to align the input data to the optimal concept. Comparing that the original method only optimizes $p(\boldsymbol{\omega}_0, x_{i,0}|\mathbf{c})$, $S_{<k}$ contains y_0 so we can align the input to the task additionally considering the answer for the input.

3.2 Context filtering

The context filtering regularizer guides prompts to filter and receive only the task-related context (task context). With context attuning regularizer, CoRe help LMs to obtain unbiased hidden states for zero-shot inference but the context attuning regularizer alone cannot realize the regularization effect if hidden states of succeeding examples are too noisy.

Hidden states deliver various contexts, including not only task contexts but also contexts related to the style, topics, or content of preceding examples. The following examples' predictions depend on the mixed contexts. Such phenomenon has been empirically reported in the work of Liu et al. (2022) where each example has a unique set of optimal demonstrations.

We conjecture that contexts that are not directly related to the task add noise to optimization on hidden states, undermining the regularization effect of context attuning regularizer. Thus, there should be some auxiliary mechanism that selects only the set of hidden states that convey task-related optimal signal and the context filtering regularizer serves that role.

Context filtering regularizer steers a parameterized prompt to filter only the set of hidden states that convey task contexts. We maximize the likelihood of an example given prepended demonstration examples, and this is the same loss with MetaICL (Min et al., 2021) and ICT (Chen et al., 2022) but without meta-learning. Specifically, we use the following SGD step:

$$\boldsymbol{\omega} \longleftarrow \boldsymbol{\omega} - \epsilon \cdot \sum_{k>0} \nabla_{\boldsymbol{\omega}_k} p_{\boldsymbol{\theta}}(y_k|S_{<k}, \boldsymbol{\omega}_k, x_k), \tag{4}$$

as depicted in Figure 1 with the blue lines. The only required context for inference with demonstrations is the task context extracted from the preceding demonstrations because the task is the only correlation between multiple examples. Therefore, this regularizer intrinsically leads the prompt to extract a proper task context from preceding examples.

3.3 Practical objectives

Our SGD step consists of the original likelihood maximization and the two regularizers:

$$\begin{aligned}\boldsymbol{\omega} \longleftarrow \boldsymbol{\omega} &- \nabla_{\boldsymbol{\omega}_0} p_{\boldsymbol{\theta}}(y_0|\boldsymbol{\omega}_0, x_0) \\ &- \nabla_{\boldsymbol{\omega}_0} p_{\boldsymbol{\theta}}(y_1|\boldsymbol{\omega}_0, x_0, y_0, \boldsymbol{\omega}_1, x_1) \\ &- \nabla_{\boldsymbol{\omega}_1} p_{\boldsymbol{\theta}}(y_1|\boldsymbol{\omega}_0, x_0, y_0, \boldsymbol{\omega}_1, x_1),\end{aligned} \tag{5}$$

when we place two examples in a sequence. This can be implemented by simply concatenating examples and minimizing the cross entropy of all of the answers. However, when we concatenate more than two examples, we need a more complex implementation and multiple iterations for a sequence because some prompts should be regarded as constants. For efficient training, we modify the losses by allowing $\boldsymbol{\omega}_{k>0}$ to be optimized for the cross-data regularizer so that we can simply compute the required gradients in a single iteration. We describe the modified losses and the complexity of the original loss in detail in Appendix A.

4 Experimental Setup

In this section, we briefly specify the setup that our experiments are conducted on: models, datasets, and hyperparameters. For more details, please check Appendix B.

Model and Dataset We evaluate our training method on two large language models, GPT2-XL (1.5B parameters) and GPT-J (6B parameters), and seven classification datasets (CB (de Marneffe et al., 2019), RTE, WSC, WiC (Pilehvar and Camacho-Collados, 2019), COPA (Roemmele et al., 2011), BoolQ (Clark et al., 2019), MultiRC (Khashabi et al., 2018)) from SuperGLUE benchmark (Wang et al., 2019). The benchmark has English datasets from various tasks and domains such as news, blogs, or encyclopedia. We downloaded the model checkpoints from Huggingface transformers (Wolf et al., 2020) and datasets from Huggingface datasets (Lhoest et al., 2021).

Baselines We evaluate our method on three gradient-based prompt-tuning baselines: P-tuning (Liu et al., 2021), Prefix-tuning (Li and Liang, 2021), and Softprompt (Lester et al., 2021).

Input Construction for CoRe In our experiments, the input is formed by simply adding several trainable prompt embeddings in front of each element of a data instance. This way of constructing an input is highly convenient because it does not require manual human effort and additional tuning for each task. For example, a single data instance of natural language inference tasks consists of three elements: premise, hypothesis, and label. We transform the data instance into $(e_0, premise, e_1, hypothesis, e_2, label)$ where $e_i \in \mathbb{R}^{n \times h}$ is a parameterized prompt that we optimize, and h is the hidden state size. We use $n = 1$ for P-tuning and Softprompt. For Prefix-tuning, we use a different template since prefix-tuning appends prompts only at the front of a data instance. We transform the data instance into $(e_0, premise, hypothesis, label)$ where e_i is a parameterized prompt of size 5.

Sequence Size and Batch Size We introduce a special hyperparameter used in CoRe, named *sequence size*. This refers to the number of data instances concatenated for a single training example when CoRe. For example, an input for CoRe of sequence size 2 and 3 becomes $\{\boldsymbol{\omega}_0, x_0, y_0, \boldsymbol{\omega}_1, x_1, y_1\}$ and $\{\boldsymbol{\omega}_0, x_0, y_0, \boldsymbol{\omega}_1, x_1, y_1\, \boldsymbol{\omega}_2, x_2, y_2\}$, respectively. Please note that CoRe of sequence size 1 is equivalent to the standard SGD training. To make a fair comparison, we keep the size of batch size = (number of sequences in a batch) $\times$ (sequence size) to be a constant. This way, a model sees an equal amount of data for every iteration of CoRe across all sequence sizes.

Evaluation We use accuracy averaged over different random seeds. We use ten seeds for GPT2-XL experiments, and five for GPT-J experiments except for MultiRC experiments, where we use five and three seeds respectively.

5 Experimental Results

We evaluate CoRe on the setup we presented in Section 4. We first show the performance gain of our method on the three baselines. Then, we present how the performance changes according to sequence sizes, and further analyze the effect of two regularizers of CoRe with the ablation studies. In addition, we present how the similarity of concatenated examples affects performance. Finally, we present the performance of CoRe in few-shot inference settings.

5.1 Main result

We first experiment with our method on the three baselines, which use parameterized continuous prompts. Table 1 compares the zero-shot performances of prompts trained only with baseline tuning methods against prompts trained with CoRe on top of the methods. On GPT2-XL, CoRe shows improvements in accuracy compared to P-tuning, Prefix-tuning, and Softprompt, up to 11.9%. On GPT-J, CoRe also shows consistent enhancements up to 6.3% for the three methods. The absolute gains are reduced on GPT-J, compared to the gains on GPT2-XL. This is expected as it is typically hard to earn a large gain as the baseline performance increases.

Interestingly, we found that our method shows a higher accuracy gain for NLI tasks — SuperGLUE CB and RTE — on GPT2-XL across all three baselines. We can hypothesize that there are some correlations between NLI tasks and the pretraining objective, or between the tasks and the pretraining dataset of GPT2-XL. It is worth analyzing the relationship between the pretraining and downstream tasks to further improve the gains for other datasets as a future work.

We observe that there are mainly two cases where CoRe does not show performance gain. First,

Model	Method	CB	RTE	WSC	WiC	COPA	BoolQ	MultiRC
GPT2-XL	P-tuning	76.79	68.84	**64.23**	63.01	57.60	70.90	**67.64**
	+ CoRe	**81.79**	**72.13**	**64.23**	**66.91**	**58.80**	**71.75**	65.52
	Softprompt	73.57	68.66	63.65	63.90	**60.50**	68.01	**70.48**
	+ CoRe	**82.32**	**73.50**	**64.42**	**64.95**	58.90	**71.77**	69.87
	Prefix	65.90	59.35	**63.56**	53.12	**57.70**	62.56	**72.24**
	+ CoRe	**79.46**	**61.66**	64.14	**53.64**	56.40	**67.40**	69.58
GPT-J	P-tuning	94.64	79.93	65.19	69.03	**70.20**	78.36	**84.45**
	+ CoRe	**97.50**	**84.98**	**65.77**	**70.85**	67.20	**84.01**	84.12
	Softprompt	91.07	82.24	65.00	**67.43**	61.80	82.31	**82.26**
	+ CoRe	**95.36**	**83.25**	**65.19**	67.09	**62.60**	**82.50**	80.27
	Prefix	94.20	81.88	65.19	**70.50**	**66.60**	**83.38**	**84.21**
	+ CoRe	**94.64**	**82.46**	**65.77**	66.96	65.20	82.81	81.61

Table 1: Comparison of zero-shot evaluation results between three baselines and applying CoRe to the baselines across various NLU datasets. For the evaluation metric, we used averaged accuracy. We highlight the better one among a baseline and that with CoRe.

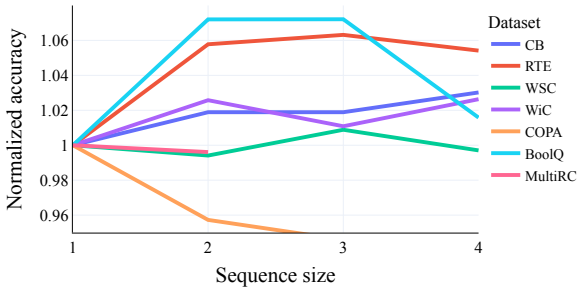

Figure 2: Zero-shot evaluation on varying sequence sizes. Accuracy for each dataset is normalized with respect to its accuracy when the sequence size is 1.

Method	Task	orig.	+ A	+ F	+ both (CoRe)
P-tuning	RTE	68.84	69.86	68.48	**72.13**
	CB	76.79	76.07	73.57	**81.79**
Softprompt	RTE	68.66	69.31	69.71	**73.14**
	CB	73.57	75.00	**75.89**	**75.89**
Prefix	RTE	59.35	55.56	53.72	**61.66**
	CB	65.90	73.39	71.07	**79.46**

Table 2: Ablations studies on the effect of context attuning regularizer (A) and context filtering regularizer (F) where sequence size is 2.

CoRe performs worse than the baseline on SuperGLUE MultiRC. We conjecture that task context does not get properly propagated since training samples of MultiRC are far longer than samples of other SuperGLUE subsets.

Another observation is that CoRe generally does not work well for Prefix-tuning on GPT-J. We hypothesize that this might have some correlation with Rotary Positional Embedding (RoPE) of GPT-J. It has not yet been explored how the mechanism of Prefix-tuning (appending prompts to all the key and values of each layer) interact with RoPE and this may undermine the proper working of CoRe.

5.2 How many examples to concat?

To see how the sequence size of CoRe affects the performance, we train prompts on GPT2-XL and GPT-J using varying sequence sizes from one to four. Figure 2 shows the result of P-tuning on GPT-J with zero shot evaluation, and Table 7, Table 8, and Table 9 in Appendix C presents the results of all of the baselines on both models with the exact numbers.

In most cases, a sequence size of 3 or 4 exhibits the best performances on GPT-J, but CoRe shows better performance with smaller sequence sizes, 2 or 3, on GPT2-XL. This shows that context attuning leads to context with less bias when given more data in a sequence, and provides effective regularization. Increasing the sequence size over a certain level results in accuracy drops. From this, we infer that too many examples in a single sequence lead to causing noise to become part of the context.

5.3 The impact of each regularizer

To see how context attuning and context filtering each contributes to the performance gain from CoRe, we conduct ablation studies with gradients involved in CoRe. We factorize the gradients as in Equation 5 with a sequence size of 2, and the gradients represent the gradient in the baseline method, context attuning, and context filtering, respectively. We then test the effect of each type of gradients.

As presented in Table 2, in general, CoRe exhibits the best performance when all three gradients (baseline, context attuning, and context filtering)

Method	Task	orig.	+ CoRe
P-tuning	RTE	0.024	**0.050**
	CB	0.261	**0.286**
Softprompt	RTE	0.006	**0.014**
	CB	0.137	**0.157**
Prefix	RTE	0.013	**0.020**
	CB	0.015	**0.037**

Table 3: Comparison of one-step generalization ratio (Liu et al., 2020) before and after applying CoRe.

Similarity	CB	RTE	WiC
max	78.93	67.22	62.79
max (10%)	76.96	70.43	64.70
standard	**81.79**	**72.13**	**65.03**
min (10%)	77.68	62.13	64.67
min	81.07	69.35	64.48

Table 4: Comparison of zero-shot performance of various sampling methods considering semantic similarity. Standard is CoRe without such sampling.

are involved. Context attuning alone has no gain or shows smaller gains than applying both regularizers. It requires a mechanism to filter unnecessary contexts and make context attuning focus on the task context, and our context filtering serves that role. There is one unexpected behavior in SoftPrompt on CB that applying the filtering gradient alone shows performance on par with CoRe. We hypothesize that filtering appropriate task context alone shows good enough performance for this specific case.

5.4 Measurement of bias caused by prompts

To showcase the regularization effect of CoRe, we compare how much prompts are biased to training data samples before and after applying CoRe. For quantifying the amount of bias, we use 'one-step generalization ratio (OSGR)' suggested by (Liu et al., 2020), which is a validation loss drop during a single training step divided by a training loss drop during the step. We calculate the training loss only on the single batch of data used in the training step, not the entire training set, to represent how much the prompt is over-fitted (biased) to the batch. The higher the OSGR, the faster the validation loss drops with respect to the training loss, meaning the smaller generalization gap between the training batch and the validation set. Note that a model sees the exact same training data in a single training step regardless of applying CoRe, as we mentioned in Section 4. In Table 3, we report results for three baselines on CB and RTE datasets, measured for 100 training steps after 3 epochs, and averaged over 10 seeds.

Despite training for the same batch of data examples, all three baselines show a smaller OSGR than the baselines with CoRe. This result implies that vanilla prompt-tuning is prone to make a prompt more biased toward data examples that the model has seen during training. CoRe produce prompts that can mitigate such bias which aligns with the performance gain shown in Table 1.

5.5 Which examples to concat?

Previous studies show that concatenating semantically similar examples improves downstream task performances. Liu et al. (2022) empirically shows that demonstration examples that are semantically similar to the target example improve in-context learning performances. Some studies also show that placing semantically similar examples in an input sequence during finetuning (Gao et al., 2021) or pretraining (Levine et al., 2021) improves downstream task performance. We experiment how semantic similarity between examples in an input sequence affects the performance of CoRe. We consider CoRe with sequence size 2.

For the experiment, we first sample *batch size*$/2$ samples for each iteration during prompt tuning. For each sampled example, we select the most similar example (*max*) or the least similar example (*min*) among the unseen examples in the epoch, and append the selected example after the sampled example. We also experiment with less extreme similarity by sampling an example to be appended from 10% most similar (*max 10%*) or 10% least similar (*min 10%*) examples of the unseen examples in the epoch. We measure the semantic similarity between examples using stsb-roberta-large model from sentence transformers (Reimers and Gurevych, 2019).

As presented in Table 4, sampling considering semantic similarity degenerates the performances of CoRe compared to the standard sampling, where demonstrations are sampled at random. This shows that CoRe also benefits from concatenating semantically similar examples — standard sampling reports better evaluation result compared to *min* and *min 10%*. However, choosing to concatenate the more similar example (*max* and *max 10%*) introduces a bias to the prompt, and the bias depresses the regularization effects of CoRe.

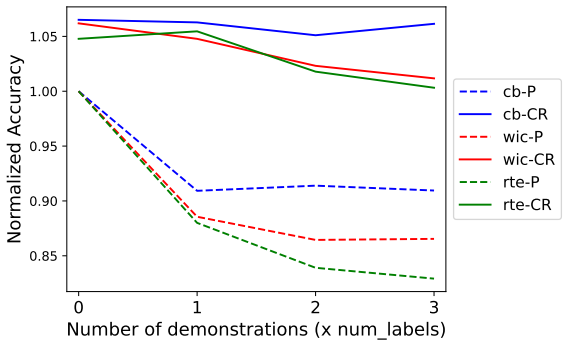

Figure 3: Comparison of few-shot in-context learning performance between P-tuning and P-tuning with CoRe, normalized by zero-shot performance of P-tuning. Number of demonstrations we use is a multiple (x axis) of the number of each dataset's classes.

5.6 Few-shot in-context learning

We also evaluate our method in a few-shot in-context learning setting, where we prepend multiple examples as demonstrations to a target example during evaluation. Although the few-shot setting is not originally our target, the objective of two regularizers — to optimize a prompt towards sending and receiving a proper task context — naturally contributes to improving the predictions prepended with demonstrations — that is, few-shot in-context learning. Figure 3 shows the result of few-shot evaluation on three different datasets with prompts from P-tuning and CoRe on P-tuning.

We observe that both P-tuning and CoRe on P-tuning report degradation in general as the number of demonstrations increases. However, CoRe on P-tuning exhibits far less accuracy drop across different numbers of demonstrations and even performance gain in the case of CB dataset. We assume that less degradation compared to P-tuning comes from two regularizers of CoRe.

6 Conclusion

CoRe is a novel gradient-based prompt tuning method that regularizes task contexts among multiple examples, which finally regularizes a prompt to improve zero-shot performance. Two regularizers of CoRe, context attuning and context filtering, regularize the prompt to create proper task context and guides the prompt to convey only the task-related context to succeeding examples on the concatenation of multiple examples. We provide a theoretical analysis for the effect of context attuning and context filtering and our experimental results back this up. Following the theoretical analysis, CoRe achieves performance gain over three different baseline prompt tuning methods in zero-shot setting up to 11.9% on GPT2-XL and 6.3% on GPT-J. It implies that CoRe can serve as an effective and memory-efficient adaptation method for hyper-scale LMs.

Limitations

Our study has three limitations:

- As reported in Section 5.1, NLI tasks significantly benefit from CoRe while other tasks marginally do, or there is no benefit at all. We suppose that such a difference comes from characteristics of a task. However, we have not yet thoroughly explored which characteristics of a task attribute the performance gain. To solve this problem, we need a novel deep learning interpretation method to probe latent contexts of LM, or a thorough analysis on relationship between the pretraining objective and downstream tasks and how prompting bridges two distinct phases. We leave these research questions as our future work.
- CoRe does not work for cases where the train dataset has too long sequence texts. CoRe requires multiple examples to be concatenated, so developers cannot benefit from CoRe if a majority of concatenated examples from their dataset exceed the maximum sequence length of an LM.
- We have not yet analyzed whether CoRe is applicable to natural language generation (NLG) tasks. NLG is undoubtedly an important pillar in natural language processing research, along with NLU, with many interesting applications. We believe that the concept of context attuning and context filtering can be of help to major challenges in NLG, for example, controlled NLG. We plan to explore CoRe on NLG tasks after this submission.

Acknowledgements

This work was supported by SNU-Naver Hyperscale AI Center.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind

Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2022. Meta-learning via language model in-context tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 719–730, Dublin, Ireland. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*.

Joe Davison, Joshua Feldman, and Alexander M Rush. 2019. Commonsense knowledge mining from pretrained models. 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 1173–1178.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Han Guo, Bowen Tan, Zhengzhong Liu, Eric P Xing, and Zhiting Hu. 2021. Text generation with efficient (soft) q-learning. *arXiv preprint arXiv:2106.07704*.

Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig. 2020a. X-factr: Multilingual factual knowledge retrieval from pretrained language models. *arXiv preprint arXiv:2010.06189*.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977.

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

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface:a challenge set for reading comprehension over multiple sentences. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*.

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.

Yoav Levine, Noam Wies, Daniel Jannai, Dan Navon, Yedid Hoshen, and Amnon Shashua. 2021. The inductive bias of in-context learning: Rethinking pretraining example design.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, 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. 2022. What

makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Jinlong Liu, Yunzhi Bai, Guoqing Jiang, Ting Chen, and Huayan Wang. 2020. Understanding why neural networks generalize well through gsnr of parameters. In *International Conference on Learning Representations*.

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

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2021. Metaicl: Learning to learn in context. *arXiv preprint arXiv:2110.15943*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Raul Puri and Bryan Catanzaro. 2019. Zero-shot text classification with generative language models. *arXiv preprint arXiv:1912.10165*.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Melissa Roemmele, Cosmin Bejan, and Andrew Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.

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 *Empirical Methods in Natural Language Processing (EMNLP)*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

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*.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [mask]: Learning vs. learning to recall. In *North American Association for Computational Linguistics (NAACL)*.

A Practical Objective

As we mentioned in Section 3.3, when we concatenate more than two examples, we need a more complex implementation and multiple iterations for a sequence because some prompts should be regarded as constants. For the implementation, we consider PyTorch-like frameworks, where a backpropagation iteration computes the gradients of a single scalar loss for listed parameters, and autoregressive LMs.

On such frameworks and LMs, CoRe requires $n-1$ backpropagation iterations, where n is the number of concatenated examples. For example, when $n=3$, CoRe needs to compute the gradient of the original likelihood G_{00}, context attuning $\{G_{01}, G_{02}\}$, and context filtering $\{G_{11}, G_{22}\}$, where $G_{ij} = \nabla_{\boldsymbol{\omega}_i} p_{\boldsymbol{\theta}}(y_j | S_{<j}, \boldsymbol{\omega}_j, x_j)$. We can

compute G_{00}, G_{01}, G_{02}, and G_{22} at the first back-propagation iteration, and G_{11} at the second, resulting in two iterations. This is not scalable because we need more iterations as the number of concatenated examples increases.

For efficient training, we modify the losses by allowing $\boldsymbol{\omega}_k (k > 0)$ to be optimized for the cross-data regularizer so that we can simply compute the required gradients in a single iteration. As a result, the final SGD step of CoRe is as follows:

$$\boldsymbol{\omega} \longleftarrow \boldsymbol{\omega} - \sum_{i<n} \sum_{j<i} G_{ij}, \tag{6}$$

where n is the number of concatenated examples.

The modified CoRe step is similar to the combination of the original CoRe step with various number of concatenated examples. If we assume $G_{ii} \approx \nabla_{\boldsymbol{\omega}_i} p_{\boldsymbol{\theta}}(y_i|\boldsymbol{\omega}_i, x_i)$, the set of the gradients of the new CoRe objective is the same as the union of the original CoRe gradients on n inputs — $\{\boldsymbol{\omega}_i, x_i, y_i | i < j\}$ where $j \leq n$ — before the modification. For example, when $n = 3$, the original CoRe gradients on $\{\boldsymbol{\omega}_0, x_0, y_0, \boldsymbol{\omega}_1, x_1, y_1, \boldsymbol{\omega}_2, x_2, y_2\}$ are the same with $\{G_{00}, G_{01}, G_{02}, G_{11}, G_{22}\}$, those on $\{\boldsymbol{\omega}_1, x_1, y_1, \boldsymbol{\omega}_2, x_2, y_2\}$ are the same with $\{G_{11}, G_{12}, G_{22}\}$, and those on $\{\boldsymbol{\omega}_2, x_2, y_2\}$ are the same with $\{G_{22}\}$ under the assumption. Therefore, the gradient sum of the three steps are as follows:

$$\sum_{i<n} \sum_{j<i} G_{ij} + G_{11} + 2G_{22} \tag{7}$$

Suppressing the magnitude of G_{11} and G_{22} by multiplying $1/2$ and $1/3$ respectively, Equation 7 become the modified CoRe gradients. Therefore, the modified CoRe gradients are the same with the summation of the original CoRe gradients on various sequence lengths with some calibrations.

B Experimental Setup Details

We search for the best set of hyperparameters (learning rate and batch size) on the search space presented in Table 5. We use SuperGLUE CB, the smallest dataset among SuperGLUE subsets, for quick hyperparameter search. For each search space of three methods and two models, we train a prompt using the baseline and the baseline with CoRe of sequence size 2.

Batch Size We use batch size 32 for all three methods at GPT2-XL and GPT-J. Note that the batch size equals (number of sequences in a batch) $\times$ (sequence size), as we described in Section 4.

Learning Rate In Table 6, we report learning rate selected for each method at GPT2-XL and GPT-J. Both the baseline and the baseline with CoRe show the best performance at the same learning rate except for Prefix-tuning at GPT-J. Therefore, for this specific case, we use the optimal learning rates for each setting ($1e{-}4$ for baseline and $2e{-}4$ for CoRe). We use the same learning rate and batch size for all datasets.

Training Steps We train prompts for 30 epochs on SuperGLUE CB, WSC, and COPA, which are small datasets, and 20 epochs on SuperGLUE RTE and WiC, which are large datasets. The size of SuperGLUE BoolQ (9K) and MultiRC (27K) are fairly larger than other SuperGLUE subsets (< 5K), so we match the number of training iterations for those two datasets to that of SuperGLUE RTE instead of setting epochs for them.

C CoRe's effect at various sequence size

Table 7, Table 8, and Table 9 present the full result of experiment in Section 5.2 on GPT2-XL and GPT-J. For SuperGLUE MultiRC, we experiment up to sequence size of 2 as its training samples are fairly longer than other SuperGLUE subsets.

D Assets used

We present assets used and their licenses in Table 10. We did our best to use models, datasets, and prompt tuning methods according to their original intended usage.

Hyperparameter	Method	GPT2-XL	GPT-J
LR	P-tuning	$\{2e{-}4, 4e{-}4, 8e{-}4\}$	$\{1e{-}4, 2e{-}4, 4e{-}4, 8e{-}4\}$
	Softprompt	$\{2e{-}3, 2e{-}2, 2e{-}1\}$	$\{2e{-}3, 1e{-}2, 5e{-}2\}$
	Prefix	$\{8e{-}6, 4e{-}5, 2e{-}4\}$	$\{2e{-}5, 1e{-}4, 2e{-}4, 4e{-}4\}$
Batch size	All	$\{32, 64\}$	$\{32, 64\}$

Table 5: Hyperparameter search space for P-tuning, Softprompt, and Prefix-tuning at GPT2-XL and GPT-J

Method	GPT2-XL	GPT-J
P-tuning	$4e{-}4$	$2e{-}4$
+ CoRe	$4e{-}4$	$2e{-}4$
Softprompt	$2e{-}2$	$1e{-}2$
+ CoRe	$2e{-}2$	$1e{-}2$
Prefix	$2e{-}4$	$1e{-}4$
+ CoRe	$2e{-}4$	$2e{-}4$

Table 6: Learning rate for P-tuning, Softprompt, and Prefix-tuning at GPT2-XL and GPT-J

		SuperGLUE						
Model	s	CB	RTE	WSC	WiC	COPA	BoolQ	MultiRC
GPT2-XL	1 (P-tuning)	76.79	68.84	**64.23**	63.01	57.60	70.90	**67.64**
	2	**81.79**	**72.13**	**64.23**	65.03	58.70	70.16	65.52
	3	77.50	63.54	64.04	**66.91**	58.30	71.02	–
	4	78.39	71.95	63.94	63.43	**58.80**	**71.75**	–
GPT-J	1 (P-tuning)	94.64	79.93	65.19	69.03	**70.20**	78.36	**84.45**
	2	96.43	84.55	64.81	70.81	67.20	**84.01**	84.12
	3	96.43	**84.98**	**65.77**	69.78	66.40	**84.01**	–
	4	**97.50**	84.26	65.00	**70.85**	62.20	79.60	–

Table 7: The performance of P-tuning ($s = 1$) and CoRe on P-tuning where s is the sequence size. We highlight the best performance among all sequence sizes.

		SuperGLUE						
Model	s	CB	RTE	WSC	WiC	COPA	BoolQ	MultiRC
GPT2-XL	1 (Softprompt)	73.57	68.66	63.65	63.90	**60.50**	68.01	70.48
	2	75.89	73.14	**64.42**	64.80	58.20	70.31	69.87
	3	78.39	71.88	63.85	64.11	57.90	**71.77**	–
	4	**82.32**	**73.50**	63.85	**64.95**	58.90	70.74	–
GPT-J	1 (Softprompt)	91.07	82.24	65.00	**67.43**	61.80	82.31	**82.26**
	2	**95.36**	**83.25**	65.00	67.09	62.00	**82.50**	80.27
	3	**95.36**	83.11	**65.19**	65.36	**62.60**	82.34	–
	4	**95.36**	82.89	64.42	66.21	61.00	81.26	–

Table 8: The performance of Softprompt ($s = 1$) and CoRe on Softprompt where s is the sequence size. We highlight the best performance among all sequence sizes.

Model	s	SuperGLUE CB	RTE	WSC	WiC	COPA	BoolQ	MultiRC
GPT2-XL	1 (Prefix)	65.89	59.35	63.56	53.12	**57.70**	62.56	**72.24**
	2	**79.46**	**61.66**	**64.14**	**53.64**	56.40	**67.40**	69.58
GPT-J	1 (Prefix)	94.20	81.88	**65.19**	**70.50**	**66.60**	**83.38**	**84.21**
	2	**94.64**	**82.46**	65.00	66.96	65.20	82.81	81.61

Table 9: The performance of Prefix-tuning ($s = 1$) and CoRe on Prefix-tuning where s is the sequence size. We highlight the best performance among all sequence sizes.

Asset	License
SuperGLUE benchmark	MIT
P-tuning	MIT
Softprompt	MIT
Prefix-tuning	Not specified
GPT-J	apache-2.0
GPT2-XL	MIT

Table 10: Assets used and their licenses.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
the section 7 after the conclusion section

☒ A2. Did you discuss any potential risks of your work?
Our work have no risk such as privacy or ethical risks.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
abstraction and section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 5

☑ B1. Did you cite the creators of artifacts you used?
Section 5

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Appendix D

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Appendix D

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
The benchmark we used is one of the most popular benchmark with 1000 citations. To the best of our knowledge, there is no such risk in the benchmark.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4

☒ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
The datasets we used are public datasets, and such statistics are also previously reported.

C ☑ Did you run computational experiments?

Section 5

☒ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
We did report the number of parameters of models used, but we did not report GPU hours as we ran experiments on various GPUs. Plus, as our method is lightweight enough, we did not consider measuring GPU hours during our experiments.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4 and Appendix B

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
We reported the average performance of multiple runs up to 10 runs.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.