SPC: Soft Prompt Construction for Cross Domain Generalization

Wenbo Zhao[1], **Arpit Gupta**[1], **Tagyoung Chung**[1] and **Jing Huang**[2]
Alexa AI, Amazon, USA
[1]{wenbzhao,guparpit,tagyoung}@amazon.com
[2]jinghuang.zhu@gmail.com

Abstract

Recent advances in prompt tuning have proven effective as a new language modeling paradigm for various natural language understanding tasks. However, it is challenging to adapt the soft prompt embeddings to different domains or generalize to low-data settings when learning soft prompts itself is unstable, task-specific, and bias-prone. This paper proposes a principled learning framework—soft prompt construction (SPC)—to facilitate learning domain-adaptable soft prompts. Derived from the SPC framework is a simple loss that can plug into various models and tuning approaches to improve their cross-domain performance. We show SPC can improve upon SOTA for contextual query rewriting, summarization, and paraphrase detection by up to 5%, 19%, and 16%, respectively.

1 Introduction

Prompting has emerged as a new paradigm for tuning language models (LMs), drawing increasing research attention. It shows impressive utility in stimulating an LM' hidden knowledge without fine-tuning it (Petroni et al., 2019). Simply adding some descriptive words to the input can steer a general-purpose LM to a specific task in a zero-shot fashion. Table 1 shows some examples of prompting. The prompt "in other words" is added after the conversation as additional input, turning the LM into a contextual query rewriter. Any natural language understanding task can be reformulated into a generation task with proper prompting (Liu et al., 2021b; Raffel et al., 2020). As shown in Table 1, the dialogue state tracking task is converted to generating the slot value given the prompted input. As a result, a natural language understanding task turns into a prompt design task. More sophisticated prompting techniques are derived to better adapt the LM, such as in-context learning (Xie et al., 2021), chain-of-thought (Wei et al., 2022). Researchers hypothesize that these prompting techniques make the LM understand the task instructions and do implicit task adaptation (Xie et al., 2021; Wang et al., 2022a).

Albeit the impressive utility of these prompting techniques, they have some limitations that encourage further research. One major drawback is that the prompt templates significantly impact model performance but require handcrafting, which is why there is increasing prompt engineering and sharing of "good" templates (Zhou et al., 2022). Chosen unwisely, a prompt template would not work or even hurt the model inference (Liu and Chilton, 2022; Wang et al., 2022b). Further, one has to design prompts for each task, making it inefficient and less robust to transfer across models or tasks (Sanh et al., 2022). The other drawback of prompting is that it works best for large models (Reynolds and McDonell, 2021). However, training and serving large LMs are known to be costly.

These limitations drive us to study new prompting methods that require little manual effort and work for large and small models. Soft prompting is one of these approaches. Unlike prompting that adds words to the input, soft prompting adds embeddings to the model, such as injecting embeddings to the token embedding layer or other model layers (Asai et al., 2022). Such embeddings, or *soft prompts*, can be automatically learned. Many studies have shown that soft prompting works well for smaller-sized models and can be as effective as fine-tuning large LMs with orders of magnitude fewer trainable parameters (Lester et al., 2021; Li and Liang, 2021; Liu et al., 2021c; Su et al., 2022).

Soft prompting is a promising approach to further improve cross-domain generalization and task adaptation for all-sized models. However, some barriers need to be removed. First, learning soft prompts lacks guidance, and traditional learning objectives (such as cross-entropy) often lead to unstable training, sub-optimality, and unstable performance. Further, the unstable performance is am-

Task	Input	Prompted Input	Output
Contextual Query Rewrite	User: Show me the latest movie by Tom Hanks. Agent: Here is Finch by Tom Hanks. User: Who directed it?	[Input] In other words,	Who directed Finch?
Summarization	Las Vegas police ...	[Input] TL;DR:	The victim ...
Dialogue State Tracking	I am looking for a gastropub.	[Input] What is ${restaurant.food}?	gastropub
Paraphrase Detection	Useless math. Algebraic topology.	Sentence 1 [S1] and sentence 2 [S2] are	equivalent

Table 1: Prompting examples.

plified in cross-domain settings. Soft prompts are often biased by a specific domain and hence less effective for other domains. In low-data settings, it is particularly difficult to learn soft prompts with consistent cross-domain performance (Su et al., 2022; Lester et al., 2021; Liu et al., 2021b; Chen et al., 2022a).

Our objective is to derive a principled paradigm for learning domain-adaptable soft prompts, for which we propose the soft prompt construction (SPC) framework. Using this SPC framework, we further derive a simple loss: the SPC loss. The loss supervises the optimization of soft prompts without re-tuning the entire model. The learned soft prompts demonstrate superior cross-domain generalization than other fine-tuning and prompt tuning approaches. Our contributions are the following:

(1) We propose an SPC framework that learns soft prompts in a zero/few-shot manner and demonstrates superior cross-domain generalization in extensive experiments on diverse tasks.

(2) We propose an SPC loss that can simply plug into various language models and boost their in/cross-domain performance.

(3) Our SPC framework builds upon the simplest soft prompt tuning approach, requiring orders of magnitude fewer trainable parameters than finetuning or other soft prompt tuning approaches.

2 Related Work

Prompting adds token-level templates to the model input, often designed manually (Petroni et al., 2019; Liu et al., 2021b). AutoPrompt automates prompt design via gradient-based search (Shin et al., 2020). Other approaches like (Gao et al., 2021; Sun et al., 2021) may use LMs to generate the prompts. Two prompting techniques are particularly useful for large LMs (LLMs): in-context learning (ICL) and chain-of-thought (CoT). ICL uses a few examples to demonstrate the task, and the LLM can perform the task in a few-shot manner (Xie et al., 2021). CoT instructs an LLM to reason step-by-step to perform a task (Wei et al., 2022). CoT can be seen as a special case of ICL as it often provides examples to demonstrate step-by-step reasoning, such as ReAct (Yao et al., 2022), self-consistency (Wang et al., 2022a). Token-based prompts' performance is highly dependent on prompt selection and constraint on natural language words.

On the other hand, soft prompt tuning uses embedding-level prompts and is optimized by gradients. The prompt embeddings do not necessarily represent natural language words and can generalize to different domains. DART (Zhang et al., 2021) treats input token embeddings as prompt templates. Qin and Eisner (2021) optimize soft prompts via perturbing a prompt mixture. PrefixTuning (Li and Liang, 2021) prepends prompt embeddings to the model layers and uses additional networks to re-parameterize them. P-Tuning v2 (Liu et al., 2021c) extends PrefixTuning and achieves performance comparable to fine-tuning. Such works often use deep prompts (adding prompt embeddings to all model layers) or reparameterization, introducing a large number of new parameters (Li and Liang, 2021; Jiang and Wang, 2022). Lester et al. (2021) add task-specific soft prompts to the model input and obtain performance comparable to fine-tuning large LMs, especially on low-resource domain adaptation tasks.

However, soft prompts' performance is unstable and sensitive to initialization (Qin and Eisner, 2021). Learning soft prompts across tasks is challenging, especially in low-data scenarios (Gu et al., 2022). To this end, Chen et al. (2021) construct soft prompts via knowledge injection and implicit relation constraint. Gu et al. (2022) pretrain soft prompts on varied tasks to improve generalization but require prompts to be added during pre-training. Qin et al. (2021) study the generalization of soft prompts via (nonlinear) subspace decomposition. However, Su et al. (2022) find that the prompt dis-

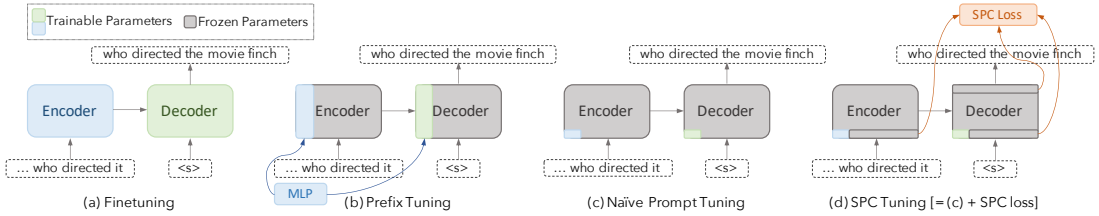


Figure 1: Model diagrams for (a) fine-tuning, (b) PrefixTuning, (c) naive prompt tuning, and (d) SPC + prompt tuning. SPC loss can plug into *any* tuning methods and language models. SPC prompt embeddings are shown as prefixes, but they can be placed *anywhere*.

tance cannot be appropriately quantified by Euclidean or cosine distances, implying a nonlinear and (potentially) non-metrizable prompt embedding space. Hence, constructing the prompt embedding space via decomposition is challenging.

3 Soft Prompt Construction

Using the examples in Table 1, prompting requires us to design different templates for different tasks, such as "in other words" for contextual query rewriting and "TL;DR" for summarization. Such handcrafting process is inefficient, does not generalize, and the performance is unstable. We aim to replace these prompt tokens with soft prompt embeddings and automatically learn domain-adaptable soft prompts. Instead of designing prompt templates, we devise a method to learn prompt embeddings that are universal to all tasks. Such prompt embeddings aim to align the model s.t. `[Input] + [Prompt] yields [Output]`; e.g., `[Input] In other words -> [Output]` $\Rightarrow$ `[input token embeddings][prompt embeddings] -> [output embeddings]`. We present a soft prompt construction (SPC) framework for learning cross-domain generalizable soft prompts.

Consider the data on cross-domain tasks $\mathbb{D} = \mathbb{X} \times \mathbb{Y}$ consisting of the input set $\mathbb{X}$ and output set $\mathbb{Y}$. For text generation tasks, for example, each element in $\mathbb{X}$ or $\mathbb{Y}$ represents a sequence of tokens. Given the generality of the tasks and data structures, we omit the dimensionality in the notations when there is no confusion. The objective of a language model is to maximize the expected log-likelihood

$$\max \mathbb{E}_{F_{\mathrm{y}}} \left[\log f_\theta(\mathrm{y} \mid x)\right] \tag{1}$$

where $x \in \mathbb{X}$, F_{y} is the probability measure w.r.t. random variable y (note we use different fonts for a random variable y and a sample y), and f_θ is the likelihood function with model parameter θ and the conditional density $f(\mathrm{y} \mid x)$. To help this objective, prompting is introduced and prompts p are added to the input $[p, x]$

$$\max \mathbb{E}_{F_{\mathrm{y}}} \left[\log \int_p f_\theta(\mathrm{y} \mid p, x) d\nu_{\mathrm{p}}\right] \tag{2}$$

where ν_{p} is the probability measure of the prompt, and we will omit this for notional simplicity when there is no confusion. Let $L = \log \int_p f_\theta(\mathrm{y} \mid p, x)$. We have

$$L = \log \int_p f_\theta(\mathrm{y} \mid p) f_\theta(p \mid x) \tag{3}$$

$$\geq \int_p \left[\log f_\theta(\mathrm{y} \mid p) + \log f_\theta(p \mid x)\right] \tag{4}$$

where from (3) to (4) we apply Jensen's inequality. To maximize (2), we need to find a maximizer of the lower-bound (4). Taking the partial derivative of (4) w.r.t. p and setting it to zero, one of the feasible maximizers p^* is obtained by (5)

$$p^* = x \oplus y \tag{5}$$

where $\oplus$ means direct sum. Indeed, if we plug (5) into (4), it achieves the maximal (3). When using soft prompts $[p, x] \overset{f_\theta}{\mapsto} y \Rightarrow [e_p, e_x] \overset{f_\theta}{\mapsto} e_y$ where e_x, e_y, e_p are embeddings of x, y, and p, relation (5) naturally extends to the embedding representations

$$e_p^* = e_x \oplus e_y \tag{6}$$

We propose to enforce such a relation using an SPC loss

$$l_{\mathrm{spc}} = \|W[W_x e_x; W_y e_y] - W_p e_p\|_2 \tag{7}$$

where $[\cdot;\cdot]$ is embedding concatenation, Ws are linear transformations that will be learned during model training. The SPC loss is directly added to the original loss (e.g., cross-entropy) to supervise the training.

For those interested, we also provide an algebraic view for our methodology in Appendix B.

Implementation details A backbone language model is used and its parameters are *frozen*. The prompt embeddings e_p have fixed length and are concatenated to the input token embeddings e_x, effectively increasing the input length. Prompt length and concatenation are explained in Section 4. Output embeddings e_y are taken from the last output layer. During training, the SPC loss (7) is directly added to the original loss (e.g., cross-entropy loss). The prompt embeddings and the linear weights of the SPC loss are learned. At inference time, the linear weights are discarded, and only the learned prompt embeddings are concatenated to the input token embeddings. *Only one soft prompt is learned, and it is universal for all domains and tasks.* An illustration is shown in Figure 1. Code 1 shows the code for computing the SPC loss.

4 Experiments

We design experiments to validate the main claims of our SPC framework: (1) SPC loss can be plugged into any model or tuning method, and (2) improve the zero/few-shot in/cross-domain generalization. We will explain our selected tasks, data, model, implementation, and results.

4.1 Task description

We select standard text generation and classification tasks: (1) contextual query rewriting (CQR), (2) abstractive summarization, and (3) paraphrase detection.

CQR refers to rewriting a contextual query in dialogue into a non-contextual query such that any missing information from the context is completed. CQR plays a crucial role in conversation systems, and the rewritten query can be used for downstream tasks such as question answering, information retrieval, and reading comprehension. CQR can be addressed by rule-based approaches, retrieval-based approaches (Fan et al., 2021), pointer-based models (Rastogi et al., 2019; Elgohary et al., 2019; Quan et al., 2019), generation-based models (Anantha et al., 2021; Yu et al., 2020; Liu et al., 2021a; Regan et al., 2019). We combine the end-to-end generation-based approach with prompt tuning to exploit both. Yilmaz and Toraman (2021) have attempted a soft prompt tuning approach for query rewriting. **Abstractive summarization** refers to generating a summary of a given text. Unlike extractive summarization, which extracts phrases from the original text, abstractive summarization may generate new phrases (Lin and Ng, 2019). The difficulty is to balance the semantic cohesion between the text and the new generation. Such problem is addressed by attention methods (Narayan et al., 2018; Gehrmann et al., 2018), extraction-abstraction-based methods (Chen and Bansal, 2018), pointer-generator-based models (See et al., 2017), generation-based methods (Zhang et al., 2020). Only a few use a prompt tuning approach (Li and Liang, 2021; Liu et al., 2022). **Paraphrase detection** refers to telling if two sentences mean the same (Yin and Schütze, 2015). Prompt tuning approaches have proven effective for this task (Lester et al., 2021; Gu et al., 2022).

Zero/few-shot cross-domain settings We train with all training data for in-domain tasks and evaluate on the test data. For cross-domain experiments, we adopt a few-shot one-to-one domain-transfer setting: first train with the source domain training data and then train with a single batch (16 samples in our settings) from the target domain training data. No target domain validation data is used. For the summarization task, we select five domains: sport, business, politics, entertainment, and technology. For the paraphrase detection task, we evaluate the few-shot cross-domain transfer between QQP and MRPC in both directions.

Zero/few-shot cross-task settings We conduct zero/few-shot cross-task transfer evaluation between CQR and summarization: take the trained model from the CQR task, evaluate on the summarization task, and vice-versa.

4.2 Data and metrics

The QReCC dataset for the CQR task consists of 14K open-domain dialogues with 80K (query, rewrite, answer) triples (Anantha et al., 2021). The XSum dataset for the abstractive summarization task consists of 227K (document, summary) pairs. The documents are collected from BBC news articles covering topics like sport, business, politics, entertainment, and technology. The summaries are of one-sentence length (Narayan et al., 2018). For both tasks, the original data split is used. Rouge score (Lin, 2004) is used as a standard evaluation metric.

Following Lester et al. (2021), we use the QQP (Shankar et al., 2017) and MRPC (Dolan and Brockett, 2005) dataset from the GLUE (Wang

et al., 2019) benchmark for the paraphrase detection task. QQP consists of 795K question pairs from Quora. Since there is no label for QQP's test set, we use the validation set for testing and use 10% of the training set for validation. MRPC consists of 5.8K sentence pairs from online news articles. The original data split is used. Following Lester et al. (2021) and Raffel et al. (2020), we format the QQP question pairs as `[question 1: <question 1 text> question 2: <question 2 text>]`. For MRPC, we format the sentence pairs as `[sentence 1: <sentence 1 text> sentence 2: <sentence 2 text>]`. We convert the binary class labels (1, 0) to "same" and "different" to better situate the task in a generative setting. Only exact matches with the label word are considered correct. [1] F1 score is used as a standard metric.

4.3 Model and baselines

We test the utility of the SPC loss by plugging it into fine-tuning and prompt tuning approaches and comparing the before/after performance. Admittedly, many prompt tuning work surfaces at the time of this manuscript (Liu et al., 2021c; Chen et al., 2022b; Gu et al., 2022). However, it is not our intention to build a better mousetrap but to present a new problem—that of learning domain-adaptable soft prompts—that naturally leads to a new learning paradigm. With this consideration, we select fine-tuning, PrefixTuning (Li and Liang, 2021), and (naive) prompt tuning (Lester et al., 2021) as our baselines because they are SOTA for respective tasks, most widely compared, and do not involve any additional components/tricks. Comparing these clean baselines present a clearer picture of SPC's benefits. The baselines are:

(1) **Fine-tuning** the BART model (Lewis et al., 2020).

(2) **PrefixTuning** (Li and Liang, 2021), a strong baseline commonly compared by other work. It uses the pretrained BART model as the backbone and prepends prompt embeddings to the encoder and decoder layers. The prefix embeddings are reparameterized by feedforward neural networks (see Figure 1). We also implement a variant of PrefixTuning—only prepending prompt embeddings at the decoder layers.

(3) **(Naive) prompt tuning** (Lester et al., 2021), the simplest soft prompt tuning. It prepends prompt embeddings to the input layer (not all layers, see Figure 1). We use the same BART backbone model for CQR and summarization tasks and add prompt embeddings to decoder input. For the paraphrase detection task, we use the same pretrained and LM-adapted T5 models (Lester et al., 2021; Raffel et al., 2020) and add prompt embeddings to encoder input.

4.4 Training and implementation

CQR, summarization We use the pretrained `BART-large` model (Lewis et al., 2019) for fine-tuning. For PrefixTuning and SPC, we freeze the BART model and only tune the prompt-related parameters. PrefixTuning requires training prefix embeddings and the reparameterization networks. SPC requires training prefix embeddings and the weights in the loss equation (7).

Paraphrase detection The pretrained and LM-adapted T5 models are used (Lester et al., 2021; Raffel et al., 2020). The `T5-small` model has 77M parameters, and the `T5-XL` model has 3B parameters.

All training and evaluation use comparable configurations and hyperparameters (e.g., batch size, learning rate, training epochs, prompt length, generation length, beam size). We find the optimal set of hyperparameters through searching, as shown in Table 8. The implementation is based on the HuggingFace's Transformers library (Wolf et al., 2020). We also re-implement the PrefixTuning model (Li and Liang, 2021). We use the metrics code that comes with the HuggingFace datasets. Training is done on the AWS cluster with eight NVIDIA V100 or A100 GPUs. More details on hyper-parameters and computation costs are shown in Appendix A.

4.5 In-domain results

CQR Table 2 shows the results on the CQR task using fine-tuning, PrefixTuning, and prompt tuning with/without SPC loss. For each prompt tuning approach, we compare random prompt initialization (*rand*) and BART word embedding initialization (*bart*). Results show that the SPC loss plugin (*row 8–12*) uniformly increases performance over fine-tuning, PrefixTuning, and prompt tuning (*row 1–7*),

[1] During inference, the generation length is 2 (including a start token). We observe that the T5 model can generate the exact "same/different" tokens with rare exceptions, so no words in vocabulary are masked off.

#	Approach	Rouge-2	Rouge-L	Gain/Loss (+/-%)
1	Finetune	74.93	**83.29**	
2	$\text{Prefix}_{\text{both,rand}}$	37.14	53.57	
3	$\text{Prefix}_{\text{both,bart}}$	55.10	68.94	
4	$\text{Prefix}_{\text{dec,rand}}$	69.68	79.13	
5	$\text{Prefix}_{\text{dec,bart}}$	72.08	81.13	
6	$\text{Prompt}_{\text{rand}}$	68.39	78.16	
7	$\text{Prompt}_{\text{bart}}$	67.13	77.34	
8	SPC+Finetune	**75.40**	83.24	+0.6/-0.06
9	SPC+$\text{Prefix}_{\text{dec,rand}}$	**72.77**	**81.60**	+4.5/+3.1
10	SPC+$\text{Prefix}_{\text{dec,bart}}$	**73.33**	**81.97**	+1.7/+1.0
11	SPC+$\text{Prompt}_{\text{rand}}$	**70.86**	**80.33**	+3.6/+2.8
12	SPC+$\text{Prompt}_{\text{bart}}$	**72.84**	**81.67**	+8.5/+5.6

Table 2: Before/after SPC results on contextual query rewriting task with fine-tuning (Finetune), PrefixTuing (Prefix), and prompt tuning (Prompt).

Approach	Rouge-2	Rouge-L	Gain/Loss (+/-%)
Finetune	20.66	35.48	
$\text{Prefix}_{\text{both,rand}}$	10.49	24.57	
$\text{Prefix}_{\text{both,bart}}$	12.47	26.63	
$\text{Prefix}_{\text{dec,rand}}$	**19.39**	**34.30**	
$\text{Prefix}_{\text{dec,bart}}$	18.81	**33.79**	
$\text{Prompt}_{\text{rand}}$	16.33	31.08	
$\text{Prompt}_{\text{bart}}$	16.66	31.63	
SPC+Finetune	**20.82**	**35.55**	+0.8/+0.2
SPC+$\text{Prefix}_{\text{dec,rand}}$	18.66	33.62	-3.8/-2.0
SPC+$\text{Prefix}_{\text{dec,bart}}$	**18.88**	33.76	+0.4/-0.09
SPC+$\text{Prompt}_{\text{rand}}$	**17.02**	**32.15**	+4.2/+3.4
SPC+$\text{Prompt}_{\text{bart}}$	**17.91**	**32.91**	+7.5/+4.0

Table 3: Before/after SPC results on abstractive summarization task with fine-tuning (Finetune), PrefixTuing (Prefix), and prompt tuning (Prompt).

with the gain ranging from 0.6%–8.5%. For PrefixTuning (*row 2–5*), prompting only at the decoder layers (*row 5*) is the best among its variations. SPC loss also significantly reduces sensitivity to prompt initialization.

Summarization Table 3 shows the results on the summarization task. We observe similar trends to CQR. [2] The SPC loss increases performance over fine-tuning, PrefixTuning, and prompt tuning, with the gain ranging from 0.2%–7.5%. SPC loss also significantly reduces sensitivity to prompt initialization.

4.6 Cross-domain results

Given the performance boost of SPC on top of other tuning approaches demonstrated in the last section, we choose to use the **simplest and most parameter-efficient** SPC loss + prompt tuning (*SPC+Prompt)* to demonstrate SPC's utility in cross-domain settings.

Summarization Table 4 presents the results for both in-domain and few-shot (16) cross-domain summarization tasks obtained with fine-tuning, PrefixTuning at the decoder with BART initialization, and SPC+Prompt with BART initialization. For in-domain, fine-tuning outperforms the prompt tuning methods. This aligns with our expectations, given that fine-tuning tunes the entire model, thereby utilizing a significantly larger number of trainable parameters compared to prompt tuning approaches. Remarkably, despite having far fewer trainable parameters, SPC+Prompt manages to achieve performance that is either comparable to or better than that of PrefixTuning. In some domains, it even matches the performance of fine-tuning. Furthermore, when plugged into fine-tuning, SPC has the potential to enhance its performance even further (as shown in Table 3). Most impressively, despite less sufficient in-domain training, SPC+Prompt surpasses other state-of-the-art methods in cross-domain performance. This greatly improves cross-domain adaptation.

Paraphrase detection Table 5 shows the results on the paraphrase detection task for both in-domain and few(16)-shot cross-domain between QQP and MRPC. SPC+Prompt initializes prompt embeddings with the first prompt-length token embeddings of the T5 model. SPC+Prompt outperforms prompt tuning with large margins, validating the cross-domain generalization of the soft prompts learned by SPC.

Cross-task Table 6 shows the results for zero/few-shot cross-task experiments between CQR and summarization. SPC+Prompt is at par or better in all settings. Prompt tuning approaches perform better than fine-tuning in low-resource settings, consistent with the findings by Chen et al. (2022a).

4.7 Analysis and discussion

Number of parameters PrefixTuning has around 7% trainable parameters (including prefix embeddings and reparameterization networks) compared to fine-tuning. Our SPC method drastically reduces the number of trainable parameters by using 0.07% parameters (including prompt embeddings and SPC loss parameters) as shown in Table 7. This ratio

[2] Our implementation does not achieve the performance reported in (Li and Liang, 2021). This may arise from different experiment configurations: compared to (Li and Liang, 2021), we do not use BPE tokens and length normalization, and we use smaller batch sizes and only 10% validation data.

Domain	Approach	Sport	Business	Politics	Entertainment	Technology
Sport	Finetune	23.89, 37.83	9.79, 23.75	11.34, 24.35	15.02, 29.16	7.69, 22.13
	Prefixtune	18.83, 33.10	13.16, 26.71	14.49, 28.14	19.02, 32.58	**12.73, 28.03**
	SPC+Prompt	18.38, 33.00	**14.04, 28.25**	**15.03, 28.68**	**20.53, 34.73**	11.38, 27.06
Business	Finetune	14.99, 29.06	15.79, 30.65	14.07, 27.44	18.28, 32.71	11.08, 25.96
	Prefixtune	15.03, 29.98	14.52, 28.80	15.12, 28.30	19.64, 32.63	12.00, 26.27
	SPC+Prompt	**16.51, 30.92**	14.70, 29.31	**15.52, 28.84**	**20.61, 34.17**	**12.25, 27.54**
Politics	Finetune	11.65, 24.98	10.49, 24.24	16.35, 29.20	16.03, 29.38	8.12, 21.93
	Prefixtune	16.04, 30.10	13.42, 27.61	15.57, 28.84	19.98, 33.63	11.52, 26.42
	SPC+Prompt	**16.50, 30.97**	**13.92, 28.15**	15.78, 29.36	**20.60, 34.80**	**12.62, 28.04**
Entertainment	Finetune	14.60, 28.67	13.01, 27.31	**15.62, 28.82**	23.62, 37.76	10.31, 25.28
	Prefixtune	16.59, 31.02	**14.07**, 28.42	15.23, 28.31	21.09, 35.26	11.56, 26.33
	SPC+Prompt	**16.60, 31.11**	**14.07, 28.50**	15.17, 28.05	20.88, 35.24	**12.15, 27.56**
Technology	Finetune	15.08, 28.81	12.90, 27.17	14.41, 27.82	18.95, 32.50	11.30, 26.15
	Prefixtune	**16.28, 30.66**	**14.16, 28.55**	**15.09, 28.52**	**32.56, 42.25**	12.00, 26.90
	SPC+Prompt	15.49, 29.74	13.22, 27.13	13.65, 26.55	19.38, 33.06	11.56, 26.31

Table 4: Few-shot cross-domain results on summarization task with fine-tuning (Finetune), PrefixTuing (Prefix), and SPC+prompt tuning (SPC+Prompt). SPC+Prompt uses significantly fewer trainable parameters. Reported as (Rouge-2, Rouge-L) pairs. **Bold numbers** represent the best of the three approaches for each domain–domain pair. Underlined numbers represent *in-domain* full-data results.

Source Domain	Target Domain	Approach	T5-small (F1)	T5-xl (F1)
QQP	QQP	Prompttune	69.93	81.26
		SPC+Prompt	**73.68**	**81.42**
	MRPC	Prompttune	68.49	70.45
		SPC+Prompt	**79.32**	**77.78**
MRPC	MRPC	Prompttune	79.89	79.87
		SPC+Prompt	**81.22**	**81.29**
	QQP	Prompttune	53.13	54.02
		SPC+Prompt	**63.64**	**60.87**

Table 5: In-domain & 16-shot cross-domain results on paraphrase detection task with prompt tuning (Prompttune) and SPC+prompt tuning (SPC+Prompt).

Source Task	Target Task	Approach	Zero-Shot	Few-Shot
QReCC	XSum	Finetune	**2.29, 11.79**	6.15, 18.56
		Prefixtune	2.01, 11.05	10.26, 24.08
		SPC+Prompt	2.14, 11.13	**13.98, 28.39**
XSum	QReCC	Finetune	9.33, 19.66	42.29, 54.22
		Prefixtune	10.61, 19.77	33.20, 45.72
		SPC+Prompt	**19.33, 31.69**	**43.26, 54.96**

Table 6: Zero/Few-shot cross-task results with fine-tuning (Finetune), PrefixTuing (Prefix), and SPC+prompt tuning (SPC+Prompt). Reported as (Rouge-2, Rouge-L) pairs.

becomes negligible as the LM goes larger (e.g., T5-XL, GPT-3). We do not show SPC results with reparameterization as its effect is inconsistent across tasks (Liu et al., 2021c).

Ablation study Table 2, 3, and Figure 2 show the effectiveness of our SPC framework in comparison with baselines. The SPC loss significantly improves the performance of fine-tuning, PrefixTuning, and prompt tuning.

Approach	Total Parameters	Trainable Parameters (#/%)
Finetuning	406,291,456	406,291,456 / 100%
PrefixTuning	447,507,008	41,215,552 / 9.21%
$\text{PrefixTuning}_{\text{dec}}$	426,899,232	20,607,776 / 4.83%
Prompt Tuning	406,393,856	102,400 / 0.025%
SPC+Prompt	406,589,225	297,769 / 0.073%

Table 7: Trainable parameter count w.r.t. BART model and fixed prompt length 100.

Prompt length Prompt length significantly affects the performance of prompt tuning approaches (Li and Liang, 2021; Liu et al., 2021c), as shown in Figure 3. Their relationship is nonlinear. A longer prompt (hence more parameters) does not translate to better performance, suggesting prompt tuning's high variance that cannot be stabilized by using more parameters or larger models (Chen et al., 2022a). Generally, simpler tasks require a shorter prompt (Liu et al., 2021c).

Prompt initialization Prompt tuning methods generally benefit from "meaningful" prompt initialization, such as initializing from model word embeddings as shown in Table 2 and 3. Our SPC approach is less sensitive to the initialization method than PrefixTuning (Chen et al., 2022a), and the SPC loss helps stabilize performance.

Prompt position Our experiments suggest that generation tasks favor prompting after the original input, i.e., at the decoder. For the text classification

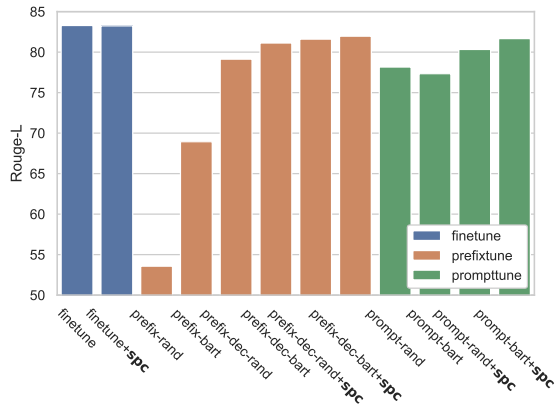
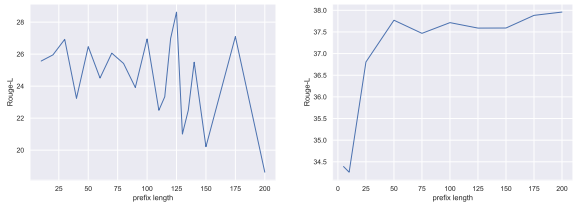


Figure 2: Ablation study on CQR task. Y-axis is the Rouge-L score and X-axis represents model variants.



(a) QReCC (b) XSum

Figure 3: Prompt length *v.s.* SPC performance on (a) QReCC and (b) XSum. Results obtained with (a) 10% validation and 25% training data; (b) 10% validation and 20% training data.

task (paraphrase detection), we prompt before the original input as suggested by Liu et al. (2021c); Lester et al. (2021). However, in our experience prompting after the original input still performs better for most tasks.

Interpretability There is an ongoing debate on whether soft prompts are interpretable. Webson and Pavlick (2021) find little evidence supporting language models understanding the prompts' meaning. Other works suggest that soft prompts form semantically close clusters (Lester et al., 2021; Zhang et al., 2021) and task-specific clusters (Su et al., 2022). However, the prompted embedding space may be highly nonlinear and potentially non-metrizable; hence, the clusters induced by Euclidean or cosine distance may not imply semantic similarity (Su et al., 2022). Instead, the clusters may be illusions arising from certain initialization strategies or the clustering tendency in high dimensions—namely, the probability for high-dimensional vectors to be uniformly distributed across dimensions is low. Our experiments find that the learned soft prompts do not convert to meaningful phrases. However, carefully controlled experiments are required to exclude confounding factors, and we leave this exploration to future work.

5 Conclusions

This paper proposes a soft prompt construction (SPC) framework to learn domain-adaptable soft prompts. The derived SPC loss is model/task-agnostic and can plug into other models and tuning methods to improve their cross-domain performance. Experiments on three tasks (contextual query rewriting, abstractive summarization, paraphrase detection) demonstrate that SPC improves significantly over strong SOTA: PrefixTuning (Li and Liang, 2021), prompt tuning (Lester et al., 2021), and fine-tuning in zero/few-shot cross-domain/task settings across different LMs (BART, T5 family). Various properties of SPC are also analyzed. This study paves the way for loss-guided domain-adaptable prompt learning.

Limitations

The proposed SPC framework is model and task agnostic and can scale to different models and tasks. It is adaptable to domain/task shifts and generalizes well with low data. However, the prompt length has to be tuned for each task, and there is no principled way of determining the optimal prompt length. This work does not explore the interpretation of soft prompts or prompt ensemble. Future work can investigate other loss forms derived from our SPC framework, explore the combination of fine-tuning's strong in-domain performance and SPC's cross-domain performance, and apply SPC and other parameter-efficient tuning approaches to large LMs.

Ethical Statement

We hereby state that our study adheres to the ACL Code of Ethics and Professional Conduct. We do not see an immediate negative impact or potential misuse of the proposed SPC method. Should our method fail in real-world applications, it will not cause potential harm to vulnerable populations. We examine the dataset and trained models and do not see bias incurred by our method. Our study does not use demographic or identity characteristics from users. No data is or will be collected from users. Our method does not increase energy and carbon costs but will potentially reduce them due to its data- and parameter-efficient training.

References

Raviteja Anantha, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi. 2021. Open-domain question answering goes conversational via question rewriting. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 520–534.

Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. 2022. Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6655–6672.

Michael Barr and Charles Wells. 1990. *Category theory for computing science*, volume 1. Prentice Hall New York.

Guanzheng Chen, Fangyu Liu, Zaiqiao Meng, and Shangsong Liang. 2022a. Revisiting parameter-efficient tuning: Are we really there yet? *arXiv preprint arXiv:2202.07962*.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2021. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. *arXiv preprint arXiv:2104.07650*.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022b. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web Conference 2022*, pages 2778–2788.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. 2019. Can you unpack that? learning to rewrite questions-in-context. 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 5918–5924.

Xing Fan, Eunah Cho, Xiaojiang Huang, and Chenlei Guo. 2021. Search based self-learning query rewrite system in conversational ai. In *2nd International Workshop on Data-Efficient Machine Learning (DeMaL)*.

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.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. Ppt: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8423.

Yuxin Jiang and Wei Wang. 2022. Deep continuous prompt for contrastive learning of sentence embeddings. *arXiv preprint arXiv:2203.06875*.

Serge Lang, FW Gehring, and KA Ribet. 2002. *Introduction to differentiable manifolds*, volume 32. Springer.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

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.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Hui Lin and Vincent Ng. 2019. Abstractive summarization: A survey of the state of the art. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9815–9822.

Hang Liu, Meng Chen, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2021a. Conversational query rewriting with self-supervised learning. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7628–7632. IEEE.

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

Vivian Liu and Lydia B Chilton. 2022. Design guidelines for prompt engineering text-to-image generative models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–23.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021c. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Xiaochen Liu, Yu Bai, Jiawei Li, Yinan Hu, and Yang Gao. 2022. Psp: Pre-trained soft prompts for few-shot abstractive summarization. *arXiv preprint arXiv:2204.04413*.

William S Massey. 2019. *A basic course in algebraic topology*, volume 127. Springer.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.

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

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

Yujia Qin, Xiaozhi Wang, Yusheng Su, Yankai Lin, Ning Ding, Zhiyuan Liu, Juanzi Li, Lei Hou, Peng Li, Maosong Sun, et al. 2021. Exploring low-dimensional intrinsic task subspace via prompt tuning. *arXiv preprint arXiv:2110.07867*.

Jun Quan, Deyi Xiong, Bonnie Webber, and Changjian Hu. 2019. Gecor: An end-to-end generative ellipsis and co-reference resolution model for task-oriented dialogue. 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 4547–4557.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Pushpendre Rastogi, AI Alexa, Arpit Gupta, and Tongfei Chen. 2019. Scaling multi-domain dialogue state tracking via query reformulation. *NAACL HLT 2019*, pages 97–105.

Michael Regan, Pushpendre Rastogi, Arpit Gupta, and Lambert Mathias. 2019. A dataset for resolving referring expressions in spoken dialogue via contextual query rewrites (cqr). *arXiv preprint arXiv:1903.11783*.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2022. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Iyer Shankar, Dandekar Nikhil, and Csernai Kornel. 2017. First quora dataset release: question pairs (2017). *URL https://www. quora. com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs*.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235.

Edwin H Spanier. 1989. *Algebraic topology*. Springer Science & Business Media.

Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Huadong Wang, Kaiyue Wen, Zhiyuan Liu, Peng Li, Juanzi Li, et al. 2022. On transferability of prompt tuning for natural language processing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3949–3969.

Yi Sun, Yu Zheng, Chao Hao, and Hangping Qiu. 2021. Nsp-bert: A prompt-based zero-shot learner through an original pre-training task–next sentence prediction. *arXiv preprint arXiv:2109.03564*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022a. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yihan Wang, Si Si, Daliang Li, Michal Lukasik, Felix Yu, Cho-Jui Hsieh, Inderjit S Dhillon, and Sanjiv Kumar. 2022b. Preserving in-context learning ability in large language model fine-tuning. *arXiv preprint arXiv:2211.00635*.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Charles A Weibel. 1995. *An introduction to homological algebra*. 38. Cambridge university press.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 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.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Eyup Halit Yilmaz and Cagri Toraman. 2021. Conqx: Semantic expansion of spoken queries for intent detection based on conditioned text generation. *arXiv preprint arXiv:2109.00729*.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911.

Shi Yu, Jiahua Liu, Jingqin Yang, Chenyan Xiong, Paul Bennett, Jianfeng Gao, and Zhiyuan Liu. 2020. Few-shot generative conversational query rewriting. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 1933–1936.

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

Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. 2021. Differentiable prompt makes pre-trained language models better few-shot learners. *arXiv preprint arXiv:2108.13161*.

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

A Additional Experiment Details

```
# Using PyTorch as demonstration
freeze_params(model)
model.train()

# Initialize prompt embedding
prompt_embedder = torch.nn.Embedding(
    prompt_length, embedding_dim)
prompt_embedder.weight = torch.nn.
    Parameter(model.shared.weight[:
    prompt_length, :].clone().detach())
prompt_embedding = prompt_embedder(range
    (prompt_length))

# Add prompt embedding to model input
    and run forward pass
inputs_embeds = torch.cat((
    prompt_embedding.clone(),
    input_embeds), dim=1)
outputs = model.forward(input_embeds)

# Get embeddings
E_x = outputs.encoder_hidden_states[0]
    # shape (batch_size, sequence_length
    , embedding_dim)
E_y = outputs.decoder_hidden_states[-1]
E_p = outputs.decoder_hidden_states[0]
    or outputs.encoder_hidden_states
    [0][:, <prompt indices>, :]

# Transform embeddings
E_x = W_x(E_x).squeeze(-1)
E_y = W_y(E_y).squeeze(-1)
E_xy = torch.cat((E_x, E_y), dim=1)
E_xy = W(E_xy)
E_p = W_p(E_p).squeeze(-1)

# Compute loss
spc_loss = (E_xy - E_p).pow(2).sum(dim
    =1).sqrt().mean()
loss = outputs.loss + spc_loss
loss.backward()
```

Code 1: Pseudo-code for SPC loss.

Table 8 shows the hyperparameter search space. The prompt lengths we used for CQR, summarization, and paraphrase detection are 200, 200, and 120. To conduct a fair comparison, we use the same batch size and number of epochs for all model variants on the same task, regardless of tuning approaches. The (batch size, training epochs) for CQR, summarization, and paraphrase detection tasks are: (16, 50), (16, 50), (16, 200).

Fine-tuning tends to converge faster than parameter-efficient approaches. For example, fine-tuning (with/without SPC) reaches high validation performance with fewer epochs and slowly improves after that. Parameter-efficient approaches (PrefixTuning, prompt tuning, with/without SPC) take more epochs to reach the same performance. However, we do not use early stopping and use the best checkpoint for evaluation after finishing all epochs.

Hyperparameter	Search Space
prompt position	{encoder, decoder, both}
prompt length	{1, 5, 10, 20, 30, ..., 200}
prompt initialization	{random, model token embedding}
max/min generation length	(vary with tasks)
beam size	{1, 2, 3, 4, 5, 6}
length penalty	{1}
batch size	{8, 16}
learning rate	{5e-5, 1e-5, 1e-4, 1e-3, 1e-2}
learning rate scheduler	{none, linear}
warmup steps	{none, 10%}
epochs	full-data: $\leq$ 200; few-shot: $\leq$ 1000

Table 8: Hyperparameter search space.

The exact number of epochs for stabilizing performance varies with each setting and task. However, we can compare their *effective FLOP* (= number of trainable parameters $\times$ number of epochs to stabilization). For example, for the CQR task, the effective FLOP for fine-tuning, PrefixTuning, and SPC+Prompt are: 406M params $\times$ 20 epochs = 8120M FLOP, 41M params $\times$ 40 epochs = 1640M FLOP, 0.3M params $\times$ 45 epochs = 13.5M FLOP. Hence, even with more epochs, SPC+Prompt has a drastically smaller effective FLOP.

B Methodology: Algebraic Perspective

The soft prompt tuning problem aims to generate the target conditioned on the prompted input, which is achieved by simply tuning the prompt embeddings rather than designing the prompt tokens. We aim to improve soft prompts' cross-domain generalizability by proposing a domain-adaptable learning paradigm.

We examine the three spaces that play a crucial role here: general task (or input) space, prompted space, and specific task (or output) space. Denote the tuple (input space, prompted space, output space) as the *three-spaces*. The input space is the general task space because it contains the task information without referring to a specific task or target. For example, an LM cannot tell if text input is for rewriting or summarization without fine-tuning on the specific task (see Table 1). However, prompts in the prompted space provide the context to specify the task and steer the LM to predict targets in the specific task space (Reynolds and McDonell, 2021). For instance, Table 1 shows how a prompted input, which combines input and prompt, contains both the original text and task-specific information. Nonetheless, the hand-designed prompts are not

scalable to other tasks as they require manual labor, and traditional soft prompt tuning approaches do not generalize to new domains.

So how do we learn domain-generalizable soft prompts? We take an algebraic approach and resort to the category theory. Category theory is fundamental to other (modern) math streams and applied to other fields, such as computer science. It studies the universal properties of mathematical objects, their structures, and the relations between them (Barr and Wells, 1990). Using category theory, we can derive the universal properties of the three-spaces and their relations, and such properties will be natural to domains, i.e., they will be domain-invariant.

First, let us formalize the three-spaces into categories. The input space, prompted space, and output space carry the canonical structure of sets, so they are categories of sets (Barr and Wells, 1990): $\mathbf{Set}_i$, $\mathbf{Set}_p$, $\mathbf{Set}_o$. Within the category, the objects are sets of tokens (i.e., tokens of inputs, prompts, and outputs), and the morphisms are functions between sets. We introduce the functor: structure-preserving map between categories $F : \mathbf{Set}_i \to \mathbf{Set}_o$, such that for any object $x \in \mathbf{Set}_i$, $F(x) \in \mathbf{Set}_o$, and for any morphism $f : x \to y \in \mathbf{Set}_i$, $F(f) : F(x) \to F(y) \in \mathbf{Set}_o$. Similarly, we introduce functors $G : \mathbf{Set}_p \to \mathbf{Set}_o$ and $H : \mathbf{Set}_i \to \mathbf{Set}_p$. A commutative diagram 4 shows such relations.

Next, we push the category of sets to the category of vector spaces—that is, we move to the corresponding embedding spaces of the three-spaces. Embedding spaces naturally carry the structure of vector spaces, so they are categories of vector spaces: $\mathbf{Vect}_i$, $\mathbf{Vect}_p$, $\mathbf{Vect}_o$ (they are actually categories of topological vector spaces as the embeddings carry manifold structure, but we omit this structure for simplicity) (Lang et al., 2002). For notional convenience, we omit the underlying field of real numbers for the vector space. The objects in the $\mathbf{Vect}$ category are vector spaces, and the morphisms are linear maps between vector spaces. Similarly, we have functors $\bar{F} : \mathbf{Vect}_i \to \mathbf{Vect}_o$, $\bar{G} : \mathbf{Vect}_p \to \mathbf{Vect}_o$, $\bar{H} : \mathbf{Vect}_i \to \mathbf{Vect}_p$. The map from the category of sets to the category of vector spaces is also a functor: the "embedding" functor.

To make $\mathbf{Vect}_p$ domain-invariant, we must find invariance between functorial maps. We resort to homology: algebraic objects that are topologically invariant (Weibel, 1995). Denote the category of chain complexes over $\mathbf{Vect}$ as $\partial\mathbf{Vect}$ whose objects are chain complexes in $\mathbf{Vect}$ and morphisms are chain maps. There is a homology functor (Spanier, 1989): $\mathring{H} : \partial\mathbf{Vect} \to \mathbf{Vect}$ that maps from chain complexes to their homology (modules). However, computing the homology is non-trivial. To simplify the problem, we introduce a short exact sequence—simplest form of chain complexes and can be used to construct longer chain complexes (Massey, 2019; Weibel, 1995)—in $\mathbf{Vect}$: $\mathcal{K} \equiv \mathbf{0} \to \boldsymbol{E}_i \xrightarrow{h} \boldsymbol{E}_p \xrightarrow{g} \boldsymbol{E}_o \to \mathbf{0}$ where $\boldsymbol{E}_i$, $\boldsymbol{E}_p$, $\boldsymbol{E}_o$ are the input, prompted, output embedding spaces, and $\mathbf{0}$ is the zero object (exercise for readers to check such exact sequence exists). This indicates $\mathcal{K}$'s homology $\mathring{H}(\mathcal{K}) \equiv \frac{\ker(g)}{\mathrm{im}(h)} = \mathbf{0}_{\mathrm{id}}$ is trivial and $\boldsymbol{E}_p$ is acyclic and invariant. Further, there exists a morphism $h^{-1} : \boldsymbol{E}_p \to \boldsymbol{E}_i$ such that $h^{-1} \circ h = \mathrm{id}_{\boldsymbol{E}_i}$. By the splitting lemma (Weibel, 1995), we have $\boldsymbol{E}_p \cong \boldsymbol{E}_i \oplus \boldsymbol{E}_o$. Hence, we can construct the prompted embedding space $\boldsymbol{E}_p$ from the direct sum of input embedding space $\boldsymbol{E}_i$ and output embedding space $\boldsymbol{E}_o$, hence the name "soft prompt construction" (SPC). There are more than one approaches to enforce such a domain-invariant property. We choose the most straightforward one: the SPC loss (7).

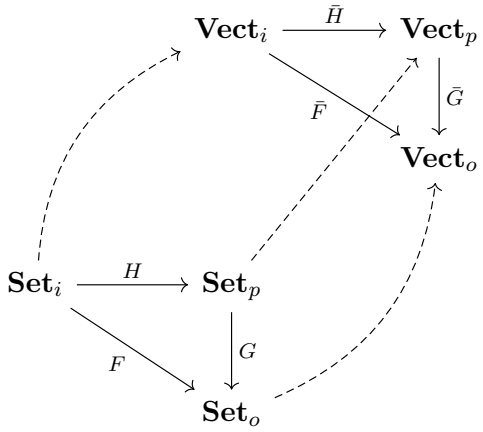


Figure 4: Commutative diagram between categories.