StablePT: Towards Stable Prompting for Few-shot Learning via Input Separation

Xiaoming Liu[1 † *], **Chen Liu**[1 †], **Zhaohan Zhang**[2], **Chengzhengxu Li**[1], **Longtian Wang**[1], **Yu Lan**[1], **Chao Shen**[1]

[1]Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China

[2]Queen Mary University of London, London, UK

[†] Eqal contribution, [*] Corresponding author

{xm.liu,ylan2020,chaoshen}@xjtu.edu.cn

{lcoder,cxz.li,w976625943}@stu.xjtu.edu.cn, zhaohan.zhang@qmul.ac.uk

Abstract

Large language models have shown their ability to become effective few-shot learners with prompting, revolutionizing the paradigm of learning with data scarcity. However, this approach largely depends on the quality of prompt initialization, and always exhibits large variability among different runs. Such property makes prompt tuning highly unreliable and vulnerable to poorly constructed prompts, which limits its extension to more real-world applications. To tackle this issue, we propose to treat the hard prompt and soft prompt as separate inputs to mitigate noise brought by the prompt initialization. Furthermore, we optimize soft prompts with contrastive learning for utilizing class-aware information in the training process to maintain model performance. Experimental results demonstrate that StablePT outperforms state-of-the-art methods by 6.97% in accuracy and reduces the standard deviation by 1.92 on average. Furthermore, extensive experiments underscore its robustness and stability across 8 datasets covering various tasks. [1]

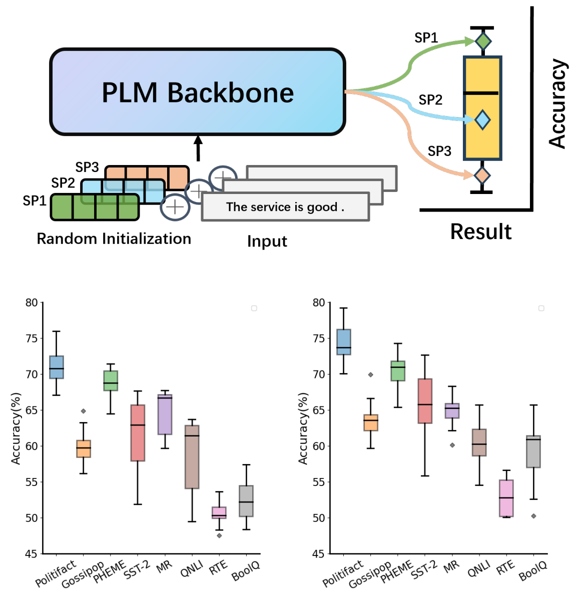


Figure 1: Illustration of performance various on NLU tasks with different prompt initialization. SP means soft prompt. The left subplot takes RoBERTa-base as the backbone, and the right shows RoBERTa-large. The accuracy variation can reach 15.77% and 16.86% on the SST-2 dataset, respectively.

1 Introduction

Pre-trained Language Models (PLMs) (Ouyang et al., 2022; Touvron et al., 2023; Anil et al., 2023; OpenAI, 2023) exhibit an overwhelming capacity to understand, analyze and classify textual information. Recent works suggest prompt tuning (Shin et al., 2020; Schick and Schütze, 2021a; Ye et al., 2022; Han et al., 2022; Liu et al., 2023b; Wu et al., 2024) to be a plausible method for efficiently adapting abundant but general knowledge behind PLMs to various downstream tasks. Prompt tuning reformulates downstream tasks into the same Mask Language Modeling (MLM) problem as pre-training of PLMs by constructing proper templates with open slots. In this way, the tasks benefit from PLMs' generative capability to fill the slots. And Brown et al. (2020) find that PLMs are able to learn well from limited training templates for certain tasks. Thus, it has been a practical topic to generalize PLMs' ability in a prompt manner with only a few samples for both training and data efficiency.

Current works on the construction of prompts could be roughly categorized into two branches: a) **Hard prompt construction**, which generates a template in natural language manually (Petroni et al., 2019; Schick and Schütze, 2021a) or automatically (Jiang et al., 2021; Ben-David et al., 2021; Li et al., 2023). While hard prompts provide explicit guidance for PLMs, the performance is largely dependent on the selection of a proper template. In fact, Zhao et al. (2021) report that different prompt formats could lead to accuracy varying from near chance to near state-of-the-art. b) **Soft prompt construction**, which searches for appro-

[1]Codes are available at https://github.com/lccc0528.

priate prompts in embedding space (Li and Liang, 2021; Lester et al., 2021a). Although steering away from natural language avoids meticulous selection of manual prompts, effective initialization of soft prompts is considered crucial for gaining satisfying performance (Gu et al., 2022). Fig. 1 illustrates the instability of vanilla prompt tuning with 10 runs on eight natural language understanding (NLU) tasks. A common solution for searching valid prompt initialization in continuous space is pre-training (Gu et al., 2022; Vu et al., 2022a), which first learns the prompt on multiple datasets with diversified tasks but requires massive computing expenses.

To deal with the defects of hard and soft prompt construction, we propose a Stable Prompt Tuning method, named StablePT, which is robust to prompt initialization quality and keeps out performance and stability in the meantime. Instead of concatenating hard and soft prompts directly, we treat them as separate inputs to different transformers-style modules. Hard prompts with contexts are embedded by PLM and a single SemEncoder layer for context-aware representation, and a GenDecoder layer takes soft prompt as input, further injecting class-aware information into cloze templates by contrastive learning. Moreover, we apply supervised contrastive loss in the soft prompt optimization process, enhancing the model's ability to achieve inter-class separation and inner-class compactness, which further boosts model performance in the few-shot setting. **The advantages of our method are three-folded:** (*i*) Reduce noise caused by undesired soft prompt initialization through processing hard prompt and soft prompt with different modules. (*ii*) Soft prompts provide the verbalizer with additional class-aware information, ensuring consistent performance regardless of the discrete template expression. (*iii*) Hard prompts offer explicit guidance on soft prompt optimization for extracting task-specific knowledge efficiently.

The contribution is summarized as follows:

- **Input Separation:** We design a novel strategy that separates soft prompt from textual input to alleviate performance inconsistency brought by the initialization quality of continuous templates.
- **Information Fusion:** We design an interaction learning process for hard and soft prompt optimization, which integrates context-aware and class-aware information for stable performance.
- **Outstanding Performance:** StablePT surpasses state-of-the-art methods on 8 NLU tasks. The experiment results verify its robustness and stability across different prompt initializations.

2 Related Work

Prompt Tuning. Prompt tuning is an efficient approach to adapting PLMs to downstream tasks. The initial method in prompt tuning (Schick and Schütze, 2021a,b) involves manually designing hard prompts composed of discrete words. Subsequent works (Gao et al., 2021; Jiang et al., 2020; Shin et al., 2020) propose the automatic generation of prompts for mining appropriate templates to obtain desired outputs. Yet, previous work reveals that changing a single word in the hard prompt might result in a substantial performance drop (Liu et al., 2023a). Soft prompting is another branch of prompt tuning, which searches proper prompt in continuous vector space (Li and Liang, 2021; Qin and Eisner, 2021; Lester et al., 2021b). It endows model flexibility to various downstream tasks while suffering greatly from undesired initialization (Gu et al., 2022). To keep consistent performance for prompt tuning, PPT (Gu et al., 2022) pre-trains prompts with 10 GB textual data, and SPoT (Vu et al., 2022a) conducts prompt pre-training on three tasks across eight datasets to attain transferable prompts. However, prompt pre-training also requires massive computing expenses (*e.g.*, it takes 25 hours to pre-train PPT and 30 hours to pre-train SPoT with Roberta-base on a single NVIDIA A100), which is contrary to the original intention of prompt tuning. Our method achieves the goal of stabilizing language model adaptation, *i.e.*, reducing the model performance fluctuates for different prompt initialization (Liu et al., 2023a), by disentangling hard prompt and soft prompt to avoid noise propagation, refraining the necessity to pre-train prompts.

Few-shot Learning with PLMs. Prompting PLMs, such as GPT-3 (Brown et al., 2020) and PET (Schick and Schütze, 2021a), are found to obtain surprising performance in substantial downstream tasks with few-shot settings. Subsequent research efforts by Perez et al. (2021) and Bragg et al. (2021) have explored reasonable few-shot settings by constraining the size of the validation set and introducing a unified framework for assessing few-shot performance. The study by Logan IV et al. (2022) emphasizes that fine-tuning PLMs in few-shot learning scenarios can enhance the model's robustness to different hard prompts. However, their approaches do not address the fundamental issue of poor prompt adjustment performance in few-shot scenarios. Our method overcomes this by fine-tuning trivial portions in additional layers with interaction between disentangled prompts.

3 Methodology

We introduce a novel language model architecture named StablePT that processes textual information[2] and soft prompts separately but keeps interaction between them. It helps stabilize model performance across different initialization of hard/soft prompts. The detailed illustration of StablePT is shown in Fig. 2.

3.1 Textual Information Processing

To exploit the strengths of hard prompts in few-shot learning while addressing their robustness issues, we introduce a prompt-based multi-encoder architecture to process textual information. This architecture encapsulates a dual-encoding module for context-aware information extraction and class-aware information absorption.

Task-Specific Hard Prompt. Task-specific hard prompts can provide clear context-relevant guidance for the model, helping PLMs accurately focus on task-related semantic representations. For example, in sentiment analysis, the hard prompt *"the sentiment of the review is [mask]"* directly guides the model to focus on the emotional polarity of the review text, thereby improving the prediction accuracy of a specific task.

PLM Encoder. The PLM Encoder is the cornerstone of the architecture, harvesting the extensive knowledge accumulated during the PLM's pre-training. It anchors the initial processing of the hard prompt and input, encoding them into semantic embeddings. We set PLM in a frozen state according to the general prompt tuning setting. Despite being in a frozen state, the PLM Encoder retains its efficacy in capturing the core semantics intended by the hard prompts. Specifically, the original hard prompt template $\mathcal{P}_h$ and the input sentence x are syntactically mapped into a semantic vector space:

$$\mathcal{E}_{se} = \text{PLMEncoder}([\mathcal{P}_h, x]) \tag{1}$$

[2]Textual information refers to the information formed by concatenating the hard prompt and the input.

where $[\cdot,\cdot]$ is the splicing operation, $\mathcal{E}_{se} \in \mathbf{R}^{b\times o\times d}$ is the embeddings of the hard prompt and input sentence, b is batch size, o is the maximum sequence length of PLMs, and d is the embedding dimension of PLMs.

Semantic Encoder Layer. Building on the foundation of the PLM encoder, we initialize a single trainable transformer encoder layer, which adeptly projects hard prompts and inputs into a shared space by self-attention mechanism while incorporating class-aware information back-propagated from soft prompt processing module, ensuring that semantically similar embeddings are projected more consistently within the space. Specifically, the semantic encoder refines and recontextualizes the embeddings $\mathcal{E}_{se}$ produced by the PLM encoder, further modeling semantic interactions:

$$\mathcal{H}_{se} = \text{softmax}(\frac{Q_E K_E^T}{\sqrt{d_k}})V_E \tag{2}$$

where Q_E, K_E, V_E are linear transformation of $\mathcal{E}_{se}$ with matrix W_q^E, W_k^E, W_v^E which are initialized in semantic encoder layer.

Overall, the prompt-based multi-encoder maps discrete prompts and inputs into a syntactic embedding space guided by the prior knowledge of the PLM encoder. These embeddings are then intricately processed within the semantic encoder for semantic interaction. Subsequently, at the top of the semantic encoder, a verbalizer is used to map the output states $\mathcal{H}_{se}^m$ of the [*mask*] token to the label y. Furthermore, the output states $\mathcal{H}_{se}$ is fed into a generative decoder to interact with soft prompts, thus completing the semantically enriched cycle of the architecture.

3.2 Soft Prompt Processing

In order to mitigate the issue of substantial noise caused by the random initialization of soft prompts in few-shot scenarios, we introduce a transformer decoder layer called GenDecoder Layer for soft prompts to interact with semantic output states $\mathcal{H}_{se}$ instead of concatenating soft prompts on the input textual information. We initialize a soft prompt of length l with random values. This soft prompt serves as a query input to the GenDecoder Layer, while the output states from the SemEncoder Layer are used as key and value inputs to the decoder. Through the cross-attention mechanism, the soft prompt functions as a query to retrieve information from the hidden states, thereby achieving the objective of interacting with the input and extracting

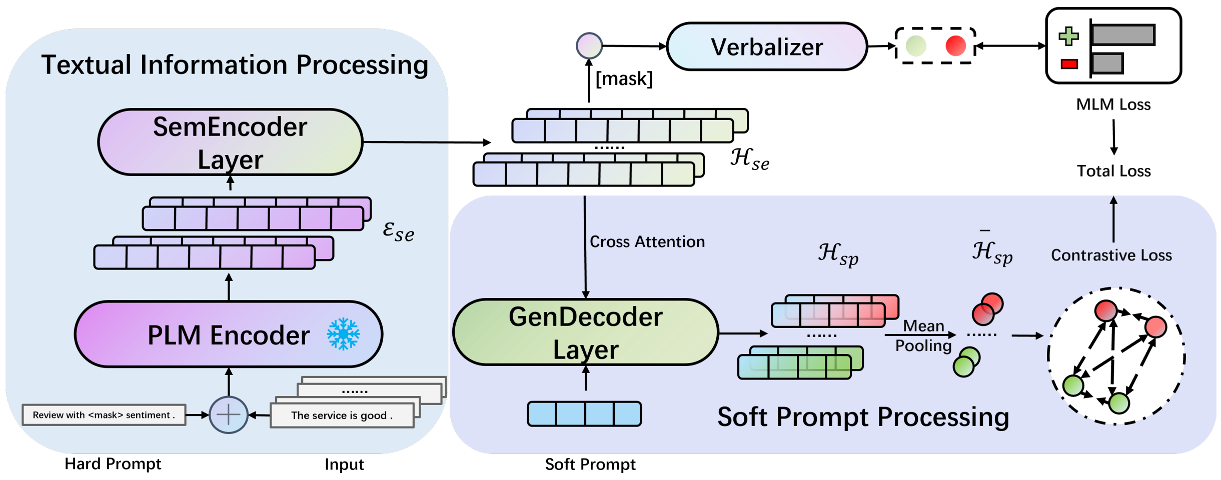


Figure 2: Overview of StablePT. The textual information processing, through a dual-encoding module, coordinates task-specific invariant prompts, simplifies adaptation to various language comprehension tasks, and enhances the semantic processing capabilities of static PLMs through dynamic semantic encoding (Sec. 3.1). The generative decoder module is central to the model, processing soft prompts through a cross-attention mechanism to interact with the encoder's outputs and diminish the adverse effects of random initialization (Sec. 3.2). The training regime employs a dual loss function comprising MLM loss and contrastive loss, aiming at enhancing the model's capabilities in language comprehension and categorical differentiation (Sec. 3.3).

context-aware information. Specifically, we apply cross-attention with $\mathcal{P}_s$ as query and $\mathcal{H}_{se}$ as key and value:

$$\begin{aligned} Q_D &= W_q^D \mathcal{P}_s, \\ K_D, V_D &= W_k^D \mathcal{H}_{se}, W_v^D \mathcal{H}_{se}, \\ \mathcal{H}_{sp} &= \text{softmax}(\frac{Q_D K_D^T}{\sqrt{d_k}}) V_D \end{aligned} \tag{3}$$

where $\mathcal{H}_{sp} \in \mathbf{R}^{b \times l \times d}$ is the output states of soft prompt, W_q^D, W_k^D, W_v^D are initilzed in GenDecoder Layer. $\mathcal{H}_{sp}$ is further processed through mean pooling to obtain more representative embeddings:

$$\bar{\mathcal{H}_{sp}} = \text{Meanpooling}([\mathcal{H}_{sp}]) \tag{4}$$

Furthermore, we apply a supervised contrastive learning method (Gunel et al., 2020) for extracting class-aware information from $\bar{\mathcal{H}_{sp}}$. Our intention is to achieve instance-level intra-class compactness and inter-class separability for discovering essential differences between classes. The contrastive learning loss is formulated as:

$$\mathcal{L}_{CL} = -\frac{1}{b} \sum_{j=1}^{b} \mathbf{1}_{[y_i = y_j]} \\ \log \frac{\exp\left(\text{sim}\left(\bar{\mathcal{H}}_{sp}^{(i)}, \bar{\mathcal{H}}_{sp}^{(j)}\right)/\tau\right)}{\sum_{K=1}^{B} \exp\left(\text{sim}\left(\bar{\mathcal{H}}_{sp}^{(i)}, \bar{\mathcal{H}}_{sp}^{(k)}\right)/\tau\right)} \tag{5}$$

where $\mathbf{1}$ is the indicator, $\text{sim}(\cdot,\cdot)$ denotes the normalized cosine similarity score, and τ is the temperature coefficient.

Soft prompt processing aligns soft prompts more closely with the semantic representation encoded by the encoder and injects class-aware information obtained from contrastive learning into context-aware representation. In this way, model performance is stabilized regardless of prompt initialization because changing a single word in the hard prompt would not lead to insufficient class-aware information extraction, and random soft prompt initialization would not affect the textual information processing stage.

3.3 Loss Function Design

We implement Masked Language Model (MLM) loss for filling slots in prompts, which is derived from the output states of the encoder. Specifically, for the masked token representations $\mathcal{H}_{se}^m$, a verbalizer function is employed to project these representations into a space where the MLM loss can be calculated.

$$\begin{aligned} \mathcal{L}_{MLM} &= \arg\max_{\theta} \sum_{x} \log p(y | [\mathcal{P}_h, x]; \theta) \\ &= \arg\max_{\theta} \sum_{x} \log p(\langle X \rangle = v(y) | \mathcal{H}_{se}^m; \theta) \end{aligned} \tag{6}$$

where y is the label of input sentence x, $\langle X \rangle$ is the mask token, $v(\cdot)$ is the verbalizer of PLMs, θ indicates all tunable parameters. The total loss for the model training is the sum of these two components, aiming to optimize both the language understanding and categorical distinction capabilities of the model.

$$\mathcal{L}_{total} = \mathcal{L}_{MLM} + \mathcal{L}_{CL} \tag{7}$$

The pseudocode of the training process is shown in Appendix B.

4 Experiments

4.1 Datasets and Settings

We evaluate our model across eight datasets on four NLU tasks: **Fake News Detection** including Politifact (Shu et al., 2020), Gossipop (Shu et al., 2020) and PHEME (Buntain and Golbeck, 2017); **Sentiment Analysis** including SST-2 (Socher et al., 2013) and MR (Pang and Lee, 2005); **Natural Language Inference (NLI)** including QNLI (Wang et al., 2018) and RTE (Wang et al., 2018); **Question Answer (QA)** including BoolQ (Wang et al., 2019). The experiment details and the impact of hyperparameters are shown in Appendix A and Appendix G, respectively.

4.2 Comparison Methods

We conduct extensive experiments with 10 competitors that are reported to be effective in few-shot settings. The comparison methods are divided into two categories roughly, *i.e.*, **prompt-tuning** and **fine-tuning**.

Prompt Tuning (Lester et al., 2021b). It prepends a sequence of soft prompt tokens to the input and only tunes the soft prompt for adaptation.

Prefix Tuning (Li and Liang, 2021). It reparameterizes networks for soft prompts and integrates and adjusts soft prompts at every layer of the PLM.

P-Tuning v2 (Liu et al., 2021). It can opt for a reparameterization network, and utilize the [CLS] classification head to adjust the soft prompts at each layer of the PLM.

PPT (Gu et al., 2022). It uses designed pattern-verbalizer pairs on large-scale unlabeled data for self-supervised learning to pre-train soft prompts.

SPoT (Vu et al., 2022b). It pre-trains soft prompts on multi-task datasets through transfer learning and investigates the transferability between tasks.

SMoP (Choi et al., 2023). It employs a gating mechanism to use multiple short soft prompts specific to data subsets as an alternative to tuning with a single long soft prompt.

E^{2}VPT (Han et al., 2023). It is a visual prompting method but could generalize well to NLU task. It introduces key-value prompts in the self-attention module of PLM and jointly serves as learnable parameters with soft prompts from the input layer in the model.

ResPT (Razdaibiedina et al., 2023). It employs a residual connection network as a reparameterization network to adjust the soft prompts of the input layer.

Fine Tuning (Kenton and Toutanova, 2019). It fine-tunes the model for classification by adding a linear classification layer on top of PLMs.

CP-Tuning (Xu et al., 2023). It adds soft prompts to the end of the input to provide a novel contrastive loss and combines with an additional MLM loss for prompt-oriented fine-tuning.

For a fair comparison, we reimplement PPT and SPoT with the same backbone model, *i.e.*, Roberta-base. We pre-train SPoT on GLUE datasets (Wang et al., 2018), which enables SPoT to achieve the best results reported in original paper. Moreover, to conduct a fair comparison, we exclude SST-2 and RTE. We pre-train PPT on the same pre-training dataset as the original paper, *i.e.*, 10 GB OpenWebText (Gokaslan et al., 2019). Further model-level comparisons are shown in Appendix C.

4.3 Comparison Results

The comparison results on eight datasets are listed in Table 1. Overall, StablePT outperforms all baselines by a large margin on all three different tasks and keeps the lowest standard deviation.

From the results, we have the following key findings: (*i*) StablePT demonstrates superior performance on all datasets, even surpassing the methods that involve full-model fine-tuning (*i.e.*, Fine Tuning, CP-Tuning) by at least 1.34%, 2.36%, 3.04% and 4.51% on the Politifact, Gossipop, SST-2 and RTE datasets. It benefits from our novel interaction mechanism which significantly mitigates the issues caused by prompt initialization in low-resource settings, even without pre-training of the prompts. (*ii*) Random initialization of prompts causes significant instability of results across all datasets. But our method StablePT demonstrates superior stability with an average standard deviation of 0.7 across datasets, outperforming Prompt Tuning, Prefix Tuning, P-Tuning V2 and SMoP in standard deviations by 3.2, 2.4, 3.1, and 2.6 respectively. (*iii*) Prefix Tuning, P-Tuning V2, E^{2}VPT, and ResPT gener-

Method	Politifact	Gossipop	PHEME	SST-2	MR	QNLI	RTE	BoolQ	Avg.
Prompt Tuning	$70.88_{2.54}$	$59.10_{2.72}$	$67.84_{1.96}$	$62.31_{5.56}$	$64.75_{3.03}$	$59.34_{5.07}$	$50.51_{1.84}$	$52.03_{3.04}$	60.85
Prefix Tuning	$72.91_{3.71}$	$58.77_{2.31}$	$68.73_{2.17}$	$70.31_{1.91}$	$67.84_{2.62}$	$60.27_{2.33}$	$50.82_{1.71}$	$54.32_{2.79}$	63.00
P-Tuning V2	$70.99_{6.69}$	$59.99_{4.84}$	$72.31_{2.16}$	$68.71_{3.67}$	$68.27_{1.34}$	$60.73_{2.44}$	$51.16_{0.73}$	$58.64_{1.77}$	63.85
PPT	$73.00_{2.45}$	$61.58_{1.42}$	$69.55_{1.23}$	$73.83_{0.64}$	$69.92_{1.26}$	$60.40_{1.46}$	$50.62_{1.32}$	$55.33_{1.08}$	64.28
SPoT	$75.43_{1.94}$	$65.18_{1.07}$	$72.55_{1.21}$	$74.79_{0.82}$	$70.39_{0.77}$	$61.49_{1.30}$	$51.38_{1.09}$	$56.34_{1.17}$	65.94
SMoP	$73.16_{2.83}$	$61.45_{2.98}$	$70.80_{3.62}$	$70.67_{3.05}$	$68.19_{3.57}$	$60.33_{1.60}$	$50.96_{0.58}$	$55.29_{2.12}$	63.86
E^{2}VPT	$78.62_{1.66}$	$68.32_{2.49}$	$72.16_{1.27}$	$70.35_{1.96}$	$67.75_{4.54}$	$60.78_{2.07}$	$53.77_{2.06}$	$57.93_{2.47}$	66.21
ResPT	$78.46_{1.48}$	$67.05_{1.69}$	$71.45_{2.26}$	$75.92_{3.11}$	$71.35_{2.73}$	$62.02_{1.69}$	$54.57_{2.10}$	$59.54_{2.07}$	67.55
Fine Tuning	$79.38_{2.63}$	$67.31_{2.45}$	$75.01_{2.26}$	$80.07_{3.92}$	$79.52_{1.93}$	$62.83_{4.37}$	$52.21_{1.82}$	$61.69_{1.58}$	69.75
CP-Tuning	$79.52_{2.28}$	$69.80_{2.83}$	$74.29_{2.52}$	$81.00_{4.31}$	$80.79_{1.21}$	$63.02_{1.43}$	$55.83_{4.15}$	$61.12_{1.67}$	70.67
StablePT	$\mathbf{80.86_{0.83}}$	$\mathbf{72.16_{0.72}}$	$\mathbf{75.35_{0.81}}$	$\mathbf{84.04_{0.45}}$	$\mathbf{81.84_{0.57}}$	$\mathbf{63.37_{0.91}}$	$\mathbf{60.34_{0.55}}$	$\mathbf{62.54_{0.78}}$	**71.31**

Table 1: Comparison of StablePT and baseline methods on few-shot NLU tasks in accuracy. The subscript means the standard deviation (*e.g.*, $80.86_{0.83}$ means 80.86±0.83) and the same to the following Tables.

ally outperform vanilla prompts in most scenarios, demonstrating that the additional parameters effectively enhance model performance. However, reparameterization can also exacerbate result instability in certain situations. For example, on Politifact, the standard deviations of Prefix Tuning and P-Tuning V2 are higher than Prompt Tuning by 1.2 and 4.2, respectively. (*iv*) Pre-trained prompts might lead to suboptimal performance due to inappropriate knowledge transfer caused by domain inconsistency between pre-training and downstream tasks. In contrast, multi-task pre-training tends to outperform single-task pre-training in downstream applications. For instance, PPT and SPoT show a smaller performance gap in sentiment analysis at 0.7% than in fake news detection at 3.0%.

4.4 Stability to Prompts Initialization

We would show the better stability of StablePT to various prompt initializations, which is compared with hybrid prompt tuning (Hybrid PT) (Liu et al., 2023a) on two different tasks.

Stability to Soft Prompt Initialization. We adopt five different soft prompt initialization strategies (Gu et al., 2022) here to test the stability of our method. The initialization strategies are explained in the Appendix D. As shown in Table 2, the results indicate that StablePT is more stable to different soft prompt initialization. The standard deviation of StablePT is lower by 1.19 and 3.08, and the performance of StablePT is better, compared with Hybrid PT on SST-2 and PHEME respectively.

Stability to Hard Prompt Initialization. We employ ChatGPT (OpenAI, 2023) to generate multiple expressions with identical meanings (T_{sn} for SST-2

	SST-2		PHEME	
Method	StablePT	Hybrid PT	StablePT	Hybrid PT
Random	$\underline{82.27}$	71.15	76.22	70.45
Label	82.98	71.94	76.53	$\underline{65.09}$
Vocab	83.03	**74.21**	$\underline{76.09}$	72.75
Top-1k	83.05	$\underline{70.29}$	76.35	73.45
Task	**83.10**	73.76	**76.79**	**73.95**
Std.	0.31	1.50	0.24	3.32

Table 2: Comparison between Hybrid PT and StablePT on SST-2 and PHEME in accuracy(%) when using different initial soft prompt strategies. Std. means standard deviation. The best result across different templates is bold and the worst is underlined.

and T_{fn} for PHEME) as hard prompts. The template construction for T_{sn} and T_{fn} are provided in the Appendix E. As shown in Table 3, the standard deviation of StablePT is lower by 9.91 and 11.23 compared with Hybrid PT on SST-2 and PHEME, respectively. The stability of performance indicates the effectiveness of our design which disentangles hard prompt and soft prompt.

4.5 Ablation Study

We conduct an ablation study to analyze the contribution of different modules in StablePT. For each experiment, we also try 10 random seeds. The setting of the ablation study is described as follows:

w/o. CL refers to the model no longer using soft prompts for contrastive learning. We use only hard prompts and input, processing them through textual information processing and utilizing MLM loss as the total loss.

w/o. GD refers to the model lacking a generative decoder. In this setup, we attach the soft prompt at

SST-2			**PHEME**		
T_{sn}	StablePT	Hybrid PT	T_{fn}	StablePT	Hybrid PT
T_{s1}	**84.75**	73.98	T_{f1}	**75.21**	<u>61.38</u>
T_{s2}	83.10	72.97	T_{f2}	72.61	69.94
T_{s3}	<u>81.45</u>	69.21	T_{f3}	72.03	67.22
T_{s4}	83.44	76.21	T_{f4}	<u>70.52</u>	67.79
T_{s5}	84.23	**77.63**	T_{f5}	74.64	**73.68**
T_{s6}	83.95	<u>68.63</u>	T_{f6}	71.91	66.20
Std.	1.11	11.02	**Std.**	2.64	13.87

Table 3: Comparison between Hybrid PT and StablePT on SST-2 and PHEME in accuracy(%) when using different hard prompts. The best result across different templates is bold and the worst is underlined.

the beginning of the input and perform contrastive learning on the hidden states of the soft prompt after the semantic encoder's output.

w/o. SP refers to removing soft prompts and only allowing generative decoder processing contextualized embeddings with both cross entropy and contrastive loss.

w/o. HP refers to the model lacking task-specific hard prompts, where we directly attach the [*mask*] token at the beginning of the input.

Method	**Gossipop**	**PHEME**	**SST-2**	**RTE**
StablePT	$\mathbf{72.16_{0.72}}$	$\mathbf{75.35_{0.81}}$	$\mathbf{84.04_{0.45}}$	$\mathbf{60.34_{0.55}}$
w/o.CL	$69.33_{1.65}$	$73.96_{1.34}$	$83.11_{0.79}$	$59.45_{0.91}$
w/o.GD	$68.87_{2.15}$	$69.20_{2.35}$	$81.19_{2.04}$	$55.21_{2.23}$
w/o.SP	$68.62_{2.58}$	$74.31_{1.02}$	$82.49_{0.94}$	$54.91_{2.47}$
w/o.HP	$70.90_{1.23}$	$73.62_{1.27}$	$76.87_{1.06}$	$58.14_{1.45}$

Table 4: Ablation study of StablePT in accuracy (%).

The experiment results shown in Table 4 indicate that all four modules contribute to the accuracy improvement. For "w/o.CL", the overall performance declines, indicating the necessity to acquire class-aware information from instance-level. Moreover, the removal of the generative decoder (*i.e.*, "w/o.GD") leads to unstable performance, which proves the essence role of this module in stabilizing model performance. For "w/o.SP", the overall performance declines, demonstrating soft prompts provide a more direct guide for the model to extract label-specific information from contextualized embeddings. "w/o.HP" causes a performance drop on all four datasets and the performance on SST-2 suffers the most, which may be attributed to the fact that the sentiment classification task benefits far more knowledge obtained from pre-training than others. A similar conclusion is also reported in Ni and Kao (2022).

4.6 Extension to Larger Model

To demonstrate that StablePT is equipped with the ability to utilize necessary knowledge from a larger model, we also conduct experiments using Roberta-large as the backbone. We re-trained PPT and SPoT with the same experimental settings as before and the experimental times are shown in Appendix I. The experimental results are shown in the Table 5.

Method	**Gossipop**	**PHEME**	**SST-2**	**RTE**
StablePT	$\mathbf{73.32_{0.82}}$	$\mathbf{77.45_{0.60}}$	$\mathbf{88.17_{0.39}}$	$\mathbf{62.70_{0.68}}$
ResPT	$70.49_{1.98}$	$73.72_{1.82}$	$78.26_{2.17}$	$56.79_{2.55}$
SPoT	$67.45_{1.31}$	$73.38_{1.74}$	$78.67_{0.51}$	$53.92_{1.55}$
PPT	$65.21_{1.45}$	$71.03_{1.91}$	$78.24_{0.59}$	$53.32_{1.42}$
PT	$63.64_{2.87}$	$70.60_{2.54}$	$65.84_{4.84}$	$52.83_{2.53}$
FT	$71.20_{4.41}$	$76.20_{2.29}$	$85.95_{2.83}$	$53.54_{1.44}$

Table 5: Parameter efficiency test in accuracy (%). PT and FT represent Prompt Tuning and Fine Tuning.

It should be noticed that the pre-training times for PPT and SPoT on RoBERTa-large are 8 times and 6 times longer than those on RoBERTa-base respectively. The results indicate that our method still maintains a significant advantage over baseline methods, despite the number of parameters in the pre-trained language model (PLM) has increased, demonstrating the universality and efficiency of our model. Compared to the results of RoBERTa-base (shown in Table 1), there is a significant improvement in the effectiveness of all the methods based on RoBERTa-large as expected, which contains more parameters. However, it is noteworthy that increasing the number of PLM parameters does not noticeably mitigate the perturbation problems caused by soft prompt initialization, which again emphasizes the importance of stabilizing language model adaptation through the perspective of the model architecture.

4.7 Extension to Full-data Scenario

To discuss the performance of StablePT in the full-data scenario, we conduct experiments comparing it with other tuning methods. As shown in Table 6, our method outperforms other prompt-based methods, showcasing its efficiency. Additionally, we observe that the standard deviation of the results for all randomly initialized Prompt Tuning methods, including both Prompt Tuning and Residual Prompt Tuning, is more favorable compared to the limited-sample scenario. This indicates that the perturbations introduced by the random initialization of prompts have been greatly mitigated in the full-data scenario. Furthermore, the performance gap between different prompt tuning strategies is notably smaller in the full-data context than in the few-shot scenario. This suggests that with sufficient training data, the impact of how prompts are initialized and tuned becomes less pronounced, allowing for more robust and consistent model performance across varying tasks.

Method	**Gossipop**	**PHEME**	**SST-2**	**RTE**
StablePT	$\mathbf{79.93_{0.77}}$	$\mathbf{83.65_{0.65}}$	$\mathbf{89.43_{0.35}}$	$\mathbf{67.38_{0.64}}$
ResPT	$79.41_{1.13}$	$82.87_{0.79}$	$89.08_{0.52}$	$65.24_{1.31}$
SPoT	$79.04_{0.82}$	$80.93_{0.86}$	$85.85_{0.57}$	$64.43_{0.92}$
PPT	$77.24_{0.69}$	$78.21_{1.01}$	$85.32_{0.62}$	$63.27_{1.14}$
PT	$76.82_{0.91}$	$77.43_{0.74}$	$84.51_{0.82}$	$63.64_{0.89}$
FT	$88.20_{0.45}$	$89.11_{0.38}$	$94.72_{0.41}$	$69.52_{0.66}$

Table 6: Full data test in accuracy (%).

4.8 Extension to Decoder-only PLMs

To demonstrate the superiority of StablePT on various pre-trained backbones, following the experiment settings of Zhu and Tan (2023), we run 4 tasks on the GPT-2-small (Radford et al., 2019) and LlaMA-2-7b (Touvron et al., 2023) backbones instead of Roberta-base. The results in Appendix H indicate that StablePT works well on the backbones and successfully outperforms the representative prompt tuning and fine-tuning methods.

4.9 Visualizations

To demonstrate that soft prompt processed by the generative decoder exhibits good separability for NLU tasks after training, we plot $\bar{\mathcal{H}}_{sp}$ for the few-shot training and testing data before tuning and after tuning. The underlying dimension reduction and visualization algorithm are t-SNE (Van der Maaten and Hinton, 2008). The results are illustrated in Fig. 3 (SST-2) and Appendix F (MR, PHEME). As shown by the results, even reduced in two dimensions, most of the embeddings in the testing set are clearly separated after tuning, with only a few-shot training samples available on different tasks. In addition, the representation of the training sample is widely spread, demonstrating the success in learning class-aware information through contrastive learning and explaining the stability of model performance.

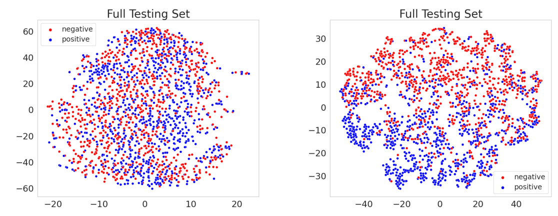


Figure 3: Visualizations of the full testing sets of SST-2 before and after tuning by t-SNE. The left and the right show the results before tuning and after tuning.

Dataset	**State**	**SC↑**	**KL↑**	**MMD↑**
SST-2	before tuning	0.01	0.28	29.46
	after tuning	**0.10**	**0.67**	**379.66**
MR	before tuning	0.001	0.08	0.34
	after tuning	**0.24**	**0.68**	**2295.47**
PHEME	before tuning	0.05	0.12	200.41
	after tuning	**0.18**	**0.59**	**1789.42**

Table 7: Results of silhouette coefficient (SC), KL divergence (KL), and maximum mean discrepancy (MMD) before and after tuning.

To quantitatively assess the intra-class cohesion and inter-class separation of our embeddings, we use three metrics, silhouette coefficient (Kullback and Leibler, 1951), KL divergence (Rousseeuw, 1987) and maximum mean discrepancy (Gretton et al., 2012). As shown in Table 7, the three quantitative metrics all showed a significant increase after training, further illustrating the effectiveness of StablePT in acquiring category information through soft prompts.

5 Conclusion

In this work, we propose StablePT, an innovative hybrid prompt-tuning framework for improving the effectiveness of prompt tuning in few-shot learning and reducing the reliance on well-chosen prompt initialization. By introducing a generative decoder module, our system allows soft prompt to interact with the output states of encoded hard prompt and input, rather than simply appending soft prompt to the input text. This approach not only overcomes the initial noise potentially introduced by soft prompts but also enhances the model's ability to capture class-aware information through the application of contrastive learning, thereby achieving better performance in NLU tasks. The results across eight datasets in four tasks demonstrate that our method maintains top performance while showing stability to different prompt initializations.

Limitations

In this work, we focus on stabilizing model performance across different prompt initializations. However, several limitations still exist for the border applications of StablePT. Firstly, data perturbations still impact our results. Future work could combine data selection (Köksal et al., 2022; Yu et al., 2023) with StablePT to further enhance performance. Secondly, StablePT does not take the impact of different verbalizers (Cui et al., 2022) into account, we believe that a better verbalizer can further improve language modeling capabilities, as some advanced works have demonstrated the significant impact of verbalizers on performance improvement. Thirdly, StablePT is designed to improve PLMs' performance on NLU tasks, which cannot be directly extended to generation tasks.

Ethics Statement

As far as we are aware, our proposed work does not have any ethical considerations. However, our work relies on pre-trained language models, which have been shown to be biased in prior work (Li and Liang, 2021). As such, users of such models should be aware of and if possible address such issues. The data and the code for this work will be made available to aid reproducibility. Moreover, though all the datasets used in our experiments are publicly available and have not been reported to carry a social bias against any sensitive attributes, and the proposed approach would not explicitly introduce new negative societal impacts, more work is still needed to investigate the potential unfairness in these datasets.

Acknowledgements

We thank all the anonymous reviewers and the area chair for their helpful feedback, which aided us in greatly improving the paper. This work is supported by National Natural Science Foundation of China (62272371, 62103323, U21B2018, 62161160337, U20B2049), Initiative Postdocs Supporting Program (BX20190275, BX20200270), China Postdoctoral Science Foundation (2019M663723, 2021M692565), Fundamental Research Funds for the Central Universities under grant (xzy012024144), and Shaanxi Province Key Industry Innovation Program (2021ZDLGY01-02).

References

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Eyal Ben-David, Nadav Oved, and Roi Reichart. 2021. Pada: A prompt-based autoregressive approach for adaptation to unseen domains. *arXiv preprint arXiv:2102.12206*, 3.

Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. 2021. Flex: Unifying evaluation for few-shot nlp. *Advances in Neural Information Processing Systems*, 34:15787–15800.

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.

Cody Buntain and Jennifer Golbeck. 2017. Automatically identifying fake news in popular twitter threads. In *2017 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 208–215. IEEE.

Joon-Young Choi, Junho Kim, Jun-Hyung Park, Wing-Lam Mok, and SangKeun Lee. 2023. Smop: Towards efficient and effective prompt tuning with sparse mixture-of-prompts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14306–14316.

Ganqu Cui, Shengding Hu, Ning Ding, Longtao Huang, and Zhiyuan Liu. 2022. Prototypical verbalizer for prompt-based few-shot tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7014–7024.

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

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. 2019. Openwebtext corpus.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773.

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.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Veselin Stoyanov. 2020. Supervised contrastive learning for pre-trained language model fine-tuning. In *International Conference on Learning Representations*.

Cheng Han, Qifan Wang, Yiming Cui, Zhiwen Cao, Wenguan Wang, Siyuan Qi, and Dongfang Liu. 2023. Eˆ 2vpt: An effective and efficient approach for visual prompt tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17491–17502.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2022. Ptr: Prompt tuning with rules for text classification. *AI Open*, 3:182–192.

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. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Abdullatif Köksal, Timo Schick, and Hinrich Schütze. 2022. Meal: Stable and active learning for few-shot prompting. *arXiv preprint arXiv:2211.08358*.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021a. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021b. 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.

Chengzhengxu Li, Xiaoming Liu, Yichen Wang, Duyi Li, Yu Lan, and Chao Shen. 2023. Dialogue for prompting: a policy-gradient-based discrete prompt optimization for few-shot learning. *arXiv preprint arXiv:2308.07272*.

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.

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

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yiyi Liu, Ruqing Zhang, Yixing Fan, Jiafeng Guo, and Xueqi Cheng. 2023b. Prompt tuning with contradictory intentions for sarcasm recognition. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 328–339.

Robert Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2022. Cutting down on prompts and parameters: Simple few-shot learning with language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2824–2835.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Shiwen Ni and Hung-Yu Kao. 2022. Electra is a zero-shot learner, too. *arXiv preprint arXiv:2207.08141*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075*.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *Advances in neural information processing systems*, 34:11054–11070.

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, Online. Association for Computational Linguistics.

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

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, Jimmy Ba, and Amjad Almahairi. 2023. Residual prompt tuning: Improving prompt tuning with residual reparameterization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

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

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

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.

Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. 2020. Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media. *Big data*, 8(3):171–188.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2022a. Spot: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou', and Daniel Cer. 2022b. SPoT: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059, Dublin, Ireland. Association for Computational Linguistics.

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.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Junda Wu, Tong Yu, Rui Wang, Zhao Song, Ruiyi Zhang, Handong Zhao, Chaochao Lu, Shuai Li, and Ricardo Henao. 2024. Infoprompt: Information-theoretic soft prompt tuning for natural language understanding. *Advances in Neural Information Processing Systems*, 36.

Ziyun Xu, Chengyu Wang, Minghui Qiu, Fuli Luo, Runxin Xu, Songfang Huang, and Jun Huang. 2023. Making pre-trained language models end-to-end few-shot learners with contrastive prompt tuning. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 438–446.

Hongbin Ye, Ningyu Zhang, Shumin Deng, Xiang Chen, Hui Chen, Feiyu Xiong, Xi Chen, and Huajun Chen. 2022. Ontology-enhanced prompt-tuning for few-shot learning. In *Proceedings of the ACM Web Conference 2022*, pages 778–787.

Yue Yu, Rongzhi Zhang, Ran Xu, Jieyu Zhang, Jiaming Shen, and Chao Zhang. 2023. Cold-start data selection for better few-shot language model fine-tuning: A prompt-based uncertainty propagation approach. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2499–2521.

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

Wei Zhu and Ming Tan. 2023. Spt: Learning to selectively insert prompts for better prompt tuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11862–11878.

A Experiment Details

We choose Roberta-base (Liu et al., 2019) as our backbone model. For all tasks, we follow the same procedure as Gu et al. (2022) to form the true few-shot learning settings (Perez et al., 2021). In particular, we randomly select 64 samples from the original training set to construct a few-shot training set, and construct a development set by randomly selecting another 64 samples from the original training set. We ensure that the number of labels is balanced for both training and development sets. We apply the AdamW (Loshchilov and Hutter, 2019) optimizer with a learning rate of 1e-4 and weight decay of 1e-4, and train for 100 epochs. The mini-batch size is 8. For all prompt-based methods, we set the prompt lengths as 10. We run experiments with 10 different random seeds on a single NVIDIA A100 and report the average accuracy and standard deviation.

Giving details of the datasets used in our main experiments in the few-shot setting, including type, label words, and size of test in Table 8. $|C|$: The number of categories for tasks.

Datasets	Type	$\lvert C\rvert$	$\lvert Test\rvert$	Label words
Politifact	FD	2	19k	fake/real
Gossipop	FD	2	32k	fake/real
PHEME	FD	2	1.6k	fake/real
SST-2	SA	2	1.8k	positive/negative
MR	SA	2	2k	positive/negative
QNLI	NLI	2	5.3k	yes/no
RTE	NLI	2	2.4k	Clearly/Yet
BoolQ	QA	2	3.2k	yes/no

Table 8: Datasets information in the main experiments. FD means fake news detection, SA means sentiment analysis, NLI means natural language inference and QA means question answer.

B Pseudocode of StablePT

Pseudocode of StablePT is presented in Algorithm 1.

Algorithm 1 Algorithm of StablePT

```
Require: Input $x$, hard prompt $p_h$, soft prompt $p_s$
    batch size $b$ and labels $y$
Ensure: A learned model StablePT, consisting of
    semantic encoder $f_s$ with parameters $\theta_S$, generative decoder $f_g$ with parameters $\theta_G$, soft
    prompt $p_s$ with parameters $\theta_s$, PLM Encoder
    with parameters $\theta_P$
 1: Random initialize $\theta_S$, $\theta_G$, $\theta_s$
 2: Freeze $\theta_P$
 3: $epoch \leftarrow 0$
 4: while $epoch \leq epoch_{max}$ do
 5:     $n \leftarrow 0$
 6:     while $n \leq n_{max}$ do
 7:         Randomly select batch $b_n$
 8:         $\mathcal{H}_{se} = f_s(p_h, b_n)$
 9:         $\mathcal{H}_{sp} = f_g(p_s, \mathcal{H}_{se})$
10:         Calculate $\mathcal{L}_{CL}$ with equation (5), calculate $\mathcal{L}_{MLM}$ with equation (6), calculate $\mathcal{L}_{total}$
            with equation (7)
11:         Backward on $\mathcal{L}_{total}$ and update $\theta_S$, $\theta_G$,
            $\theta_s$ based on AdamW gradient descent with an
            adjustable learning rate
12:         $n \leftarrow n + 1$
13:     end while
14:     $epoch \leftarrow epoch + 1$
15: end while
16: return A trained model StablePT
```

C Details of Comparison Methods

In Table 9, we introduce specific details of StablePT and other comparison methods.

D Soft Prompt Initial Strategies

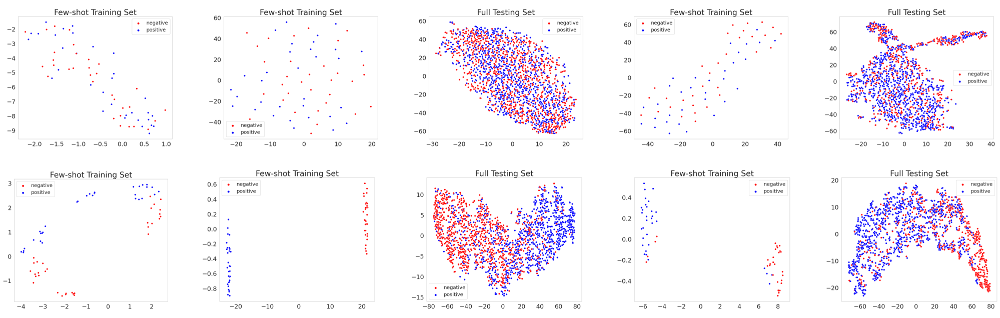
In Sec. 4.4, we utilized five methods of soft prompt initialization, and here we provide specific explanations for these five methods. "Random" indicates that we randomly initialize the embedding of soft prompts. "Label" indicates that we use the embeddings of the label words. "Vocab" indicates that we randomly sample words from the vocabulary. "Top-1k" indicates that we randomly sample words from the most frequent 1000 words in the pre-training corpus. "Task" indicates that we randomly sample words from the downstream data.

E Hard Prompt Template

Table 10 and Table 11 show the hard prompt template used in Sec. 4.4 for sentiment analysis and fake news detection, respectively. All these prompts are generated by employing ChatGPT (OpenAI, 2023) to express the same meaning but with different vocabularies.

F Visualizations

We plot the average-pooled soft prompt (*i.e.*, $\bar{\mathcal{H}_{sp}}$) of the few-shot training and testing data in MR and PHEME, shown in Fig. 4.



(a) Dataset: SST-2 (b) Dataset: MR (c) Dataset: PHEME

Figure 4: Visualization of the few-shot training sets and the full testing sets of SST-2, MR, and PHEME by t-SNE. The top shows the results before tuning and the bottom shows the results after tuning.

Method	Train. params	Pre-train
Prompt Tuning	7.7K	No
Prefix Tuning	14.2M	No
P-TuningV2	14.2M	No
PPT	7.7K	Yes
SPoT	7.7K	Yes
ResidualPT	9.3M	No
SMoP	77K	No
E^2VPT	46K	No
Fine Tuning	135M	No
CP-Tuning	135M	No
Stable PT	13.1M	No

Table 9: Comparison of all methods. Train. params denotes total number of trainable parameters, Pre-train denotes if the method requires pre-training on source tasks.

T_{sn}	*Template*
T_{s1}	*"The sentiment of the review is <mask> ."*
T_{s2}	*"The outlook portrayed in this appraisal is <mask> ."*
T_{s3}	*"The emotional tone of this testimonial is <mask> ."*
T_{s4}	*"The sentiment of this critique is <mask> ."*
T_{s5}	*"The feeling conveyed by this evaluation is <mask> ."*
T_{s6}	*"The mood expressed in this assessment is <mask> ."*

Table 10: Example of hard prompt template for sentiment analysis

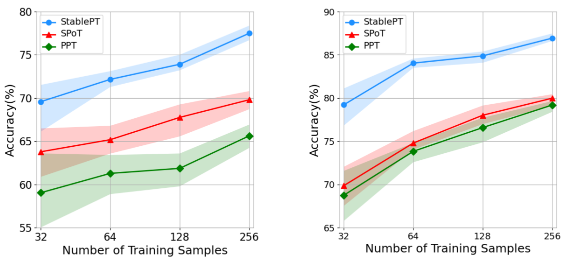
G Impact of Hyperparameters

G.1 Sample Efficiency

We'd like to discuss how performance varies with an increasing number of training samples. We select two strong baselines SPoT and PPT for comparison. As shown in the Fig. 5, with the number of training samples increasing, StablePT, PPT, and SPoT all exhibit a gradual upward trend in results. Besides, StablePT outperforms PPT and SPoT across all scales of training samples for different tasks with slight fluctuations. In addition, the suboptimal performance caused by the domain inconsistency between pre-training and downstream tasks (mentioned in Sec. 4.3) still exists with the increase in the number of training samples.

T_{fn}	*Template*
T_{f1}	*"Here is a piece of claim with <mask> information ."*
T_{f2}	*"The outlook portrayed in this appraisal is <mask> ."*
T_{f3}	*"The content of this statement is <mask> ."*
T_{f4}	*"What this declaration says is <mask> ."*
T_{f5}	*"The essence of this proclamation is <mask> ."*
T_{f6}	*"This announcement conveys a <mask> message ."*

Table 11: Example of hard prompt template for fake news detection



(a) Gossipop (b) SST-2

Figure 5: Comparison among PPT, SPoT and StablePT with different numbers of training samples on Gossipop and SST-2.

G.2 Prompt Length

The length of the soft prompt could act as an important hyperparameter. Therefore, we investigate the impact of various lengths of soft prompt on

StablePT across four different datasets, as shown in Table 12.

PL	Gossipop	PHEME	SST-2	RTE
5	$71.17_{1.04}$	$74.61_{1.23}$	$83.29_{1.43}$	$59.21_{1.03}$
10	$72.16_{0.72}$	$75.35_{0.81}$	$84.04_{0.45}$	$60.34_{0.55}$
20	$70.66_{2.02}$	$73.53_{1.88}$	$83.12_{1.79}$	$58.83_{2.23}$
50	$70.53_{1.55}$	$73.24_{1.29}$	$82.99_{1.08}$	$59.34_{1.81}$

Table 12: StablePT performance with various soft prompt lengths (PL) in accuracy (%).

The results, which explore soft prompt lengths of 5, 10, 20, and 50, reveal a discernible pattern, *i.e.*, increasing the prompt length beyond 10 does not proportionally enhance performance. In fact, longer prompts often lead to a marginal decline in effectiveness, possibly due to dilution of semantic density or increased complexity in the learning process (Lester et al., 2021b). Choosing an optimal length can facilitate PLMs to extract subtle semantic insights without being burdened by redundant information. This underscores the importance of prompt length optimization in maximizing the efficacy of soft prompts in contrastive learning settings. Especially when balancing the need for contextual richness against the constraints of computational efficiency and model generalizability.

H Extension to Decoder-only PLMs

The experimental results of decoder-only backbones are shown in Table 13.

Method	Gossipop	PHEME	SST-2	RTE
GPT-2 backbone				
StablePT	$\mathbf{67.78_{0.46}}$	$\mathbf{73.33_{0.42}}$	$\mathbf{74.83_{0.67}}$	$\mathbf{52.55_{0.34}}$
ResPT	$64.26_{4.53}$	$72.41_{1.78}$	$63.86_{3.63}$	$51.10_{0.57}$
PT	$60.15_{2.81}$	$69.40_{3.11}$	$60.52_{3.63}$	$50.34_{0.63}$
FT	$67.21_{2.63}$	$73.02_{2.62}$	$66.21_{2.63}$	$51.48_{1.03}$
LlaMA-2 backbone				
StablePT	$\mathbf{71.13_{0.63}}$	$\mathbf{75.59_{0.73}}$	$\mathbf{81.93_{0.10}}$	$\mathbf{52.78_{0.36}}$
ResPT	$67.56_{3.64}$	$73.52_{2.07}$	$66.90_{3.98}$	$51.46_{0.90}$
PT	$63.57_{1.87}$	$68.92_{1.14}$	$62.63_{3.84}$	$51.03_{0.84}$
FT	$68.21_{2.07}$	$75.04_{1.61}$	$81.25_{1.24}$	$52.48_{1.83}$

Table 13: Various PLM backbones tests in accuracy (%).

The results indicate that our method still maintains a significant advantage over baseline methods, despite using decoder models of different sizes as the backbones, demonstrating the universality of our model.

I Time Using of Pretrain Methods

For a fair comparison, we use the datasets from the original paper to re-train PPT and SPoT on Roberta-base and Roberta-large. Table 14 shows the time required for the experiment.

Methods	Roberta-base	Roberta-large
PPT	93123s	781934s
SPoT	110954s	676522s

Table 14: Consumption of PPT and SPoT in time (s).

The results indicate that as the model parameters increase, the pretraining time required for PPT and SPoT also increases, which consume substantial computational resources. This may hinder the further development of pretraining prompt methods.