PPT: Backdoor Attacks on Pre-trained Models via Poisoned Prompt Tuning

Wei Du , Yichun Zhao , Boqun Li , Gongshen Liu* , Shilin Wang
Shanghai Jiao Tong University
{dddddw, zhaoyichun, boqun.li, lgshen, wsl}@sjtu.edu.cn

Abstract

Recently, prompt tuning has shown remarkable performance as a new learning paradigm, which freezes pre-trained language models (PLMs) and only tunes some soft prompts. A fixed PLM only needs to be loaded with different prompts to adapt different downstream tasks. However, the prompts associated with PLMs may be added with some malicious behaviors, such as backdoors. The victim model will be implanted with a backdoor by using the poisoned prompt. In this paper, we propose to obtain the poisoned prompt for PLMs and corresponding downstream tasks by prompt tuning. We name this Poisoned Prompt Tuning method "PPT". The poisoned prompt can lead a shortcut between the specific trigger word and the target label word to be created for the PLM. So the attacker can simply manipulate the prediction of the entire model by just a small prompt. Our experiments on various text classification tasks show that PPT can achieve a 99% attack success rate with almost no accuracy sacrificed on original task. We hope this work can raise the awareness of the possible security threats hidden in the prompt.

1 Introduction

Pre-trained language models [Devlin *et al.*, 2018; Liu *et al.*, 2019; Raffel *et al.*, 2019] have shown unparalleled advantages in many NLP tasks. A general used paradigm is fine-tuning, which fully tunes all parameters of the PLM on the downstream task. The versatile knowledge acquired from the large-scale unlabeled data can be adapted to various NLP tasks after fine-tuning. However, fine-tuning is memory-consuming during training because gradients and optimizer states for all parameters need to be stored. Moreover, the PLM will be a task-specific model after fine-tuning. It requires keeping many copies of the PLM during inference for different downstream tasks.

Prompt tuning [Lester *et al.*, 2021] is proposed as a solution to the above problems. As shown in the Figure 1, adding natural language prompt to the text, all downstream tasks can be uniformly transformed into the form of pre-training tasks of PLMs. The prediction of [MASK] token is mapped to the real label by verbalizer. Using only hard prompt may lead to suboptimal performance. Therefore, Prompt tuning uses soft prompt as a trainable parameter, which freezes the parameters of PLMs and only tunes soft prompt. The advantage is that it can save the cost of training and the fixed PLM can be reused. Recent studies have shown that prompt tuning can be comparable to fine-tuning universally across various model scales and NLP tasks [Liu *et al.*, 2021].

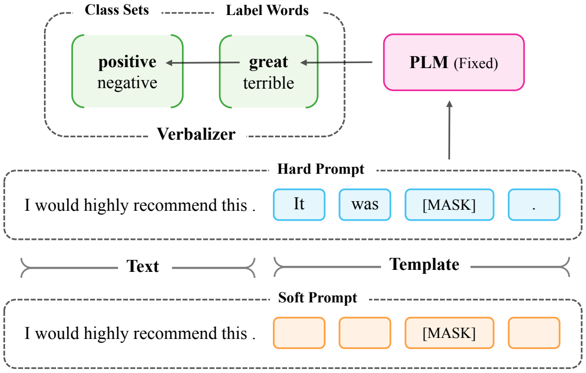

Figure 1: The example of prompt learning.

On the other hand, prompt is more and more generalized and toolkitized. In the openprompt [Ding *et al.*, 2021] framework developed by Tsinghua, users can easily import templates and verbalizers, and then load third-party trained prompts for downstream tasks in their own models. However, it also exposes some potential security threats while bringing convenience. The prompt associated with PLMs can be intentionally added with some malicious behavior, such as backdoor. The attacker upload the carefully crafted prompts to the public platform. The victim model will be implanted with the backdoor after using these poisoned prompts.

Backdoor attack is a serious security threat for deep learning models , which was first proposed by [Gu *et al.*, 2017]. They construct poisoned data by adding triggers and reversing labels. By training on the poisoned dataset, the poisoned model can maintain high accuracy for the original task but output the target class preset by the attacker when a trigger is added to the input. If such an attack is applied to security-

[*] Corresponding Author

related scenarios such as identity verification and self-driving vehicles, it will cause serious consequences. For example, when the camera detects a stop sign with a trigger, an backdoored self-driving system will control the car to accelerate instead of braking.

In this paper, we are interested in designing backdoor attacks for prompt learning. We find it feasible to get a poisoned prompt for PLMs and corresponding downstream tasks by prompt tuning. When the PLMs are loaded with the poisoned prompt, the model will be implanted with the backdoor. We name this Poisoned Prompt Tuning method PPT. For attacking prompt learning in NLP tasks, the poisoned prompt adapts the PLM to the downstream task as others clean prompt and establishes a shortcut between the specific trigger word and the target label word. So, we can control the prediction of the entire model with a small prompt. Compared with the method of tuning all parameters of model, the attack based on prompt poisoning is much more concealed and effective. Experiments conducted on various tasks including sentiment analysis, sentence-pair classification and multi-label classification show that our proposal can achieve a perfect attack success rate without losing the accuracy of clean task. Our contributions are summarized as follows:

- We propose PPT, the first backdoor attack against the prompt learning of PLMs. By loading the poisoned prompt, the model will be implanted with the backdoor.
- We evaluate PPT in many text classification tasks. Experiments show that PPT can get a perfect attack success rate with no damage the accuracy of original task.
- We conduct many related analysis experiments to comprehensively analyze the effect of PPT. We measure the performance of PPT in different hyper-parameter settings and data transferring scenarios, and evaluate the backdoor capacity of the soft prompt.

2 Related Work

2.1 Prompt Learning

Prompt learning can usually be divided into two ways, namely, Prompt-Tuning with fixed PLMs and tuned soft prompt, and Prompt-Oriented Fine-Tuning with both PLMs and soft prompt tuning. In this paper, we focus on prompt tuning, hoping to implant a backdoor into the model only through prompt.

Autoprompt [Shin *et al.*, 2020] first studies how to automatically get the appropriate prompt. They search for suitable words from vocabulary as prompt by a similar method of generating universal adversarial perturbations. Since only real words can be used, the generated prompt is limited to the discrete space. Therefore, [Hambardzumyan *et al.*, 2021] propose to use a continuous trainable vector as prompt, that is, a soft prompt, which is tuned by gradient descent on downstream tasks. In addition, they find that the label word corresponding to the true label will also affect prompt learning. So each label word in the verbalizer is also a trainable vector and optimized together with the soft prompt. [Li and Liang, 2021] further studies the prompt learning of natural language generation tasks, and adds soft prompt before the input of each transformer layer to get better results. [Lester *et al.*, 2021] explores the effect of soft prompt on domain adaptation and different model scales. They find out that the larger the scale of PLMs, the better the effect of prompt tuning.

2.2 Backdoor Attack for PLMs

Recently, there have been some researches on backdoor attacks against PLMs. Here is a review and summary of related methods. Different from the traditional backdoor attack, the main idea of the backdoor attack against the PLMs is to establish the connection between the trigger and the target class samples' features encoded by PLMs. [Kurita *et al.*, 2020] links trigger to encoded features by general backdoor. They directly poison the entire model including the downstream task classification layer, and then they get a poisoned PLMs for a specific downstream task. In contrast, [Zhang *et al.*, 2021] links trigger to the predefined vector. They retrain PLM with a backdoor sub-task and original pre-training task, enabling the PLM to output the predefined vector for the input with trigger. BadEncoder [Jia *et al.*, 2021] uses the feature alignment to establish a connection between the trigger and the encoded feature. They align the poisoned data with the encoded feature of the target class sample, and align the clean data with the encoded features of themselves. Similarly, [Saha *et al.*, 2021] applies contrastive learning to align features. They use a pair of poisoned data and target class data as a positive sample, so that the encoded features of the input with trigger and the target class input can gradually approach to each other. Moreover, [Yang *et al.*, 2021] attacks PLMs by modifying the embedding vector of a trigger word in the word embedding layer.

In this paper, we also design backdoor attacking against PLMs. The difference is that the above methods attack the fine-tuning method for PLMs, while we attack the prompt-tuning method for PLMs.

3 Methodology

In this section, we first describe the threat model of our method, which includes attacker's goals and capabilities. Then we give an introduction and formulation of the pre-training-then-prompt-tuning paradigm as essential preliminaries. Finally, we show the process and details of our method.

3.1 Threat Model

The attacker's goals. We consider a malicious service provider or some group who releases their models. They can prompt-tuning a PLM on a downstream task and inject a backdoor into the prompt that can be activated by a specific trigger word. After obtaining the well-poisoned prompt, they release it to the public (e.g., uploading the poisoned prompt to Github). The victim may download this poisoned prompt to use. The attacker's goals are that the victim PLMs will be backdoored after loading the poisoned prompt into them: for clean samples, the victim PLMs will still give the correct label word as the clean PLMs; for poisoned samples which are added with the trigger word, the victim PLMs will output the target label word.

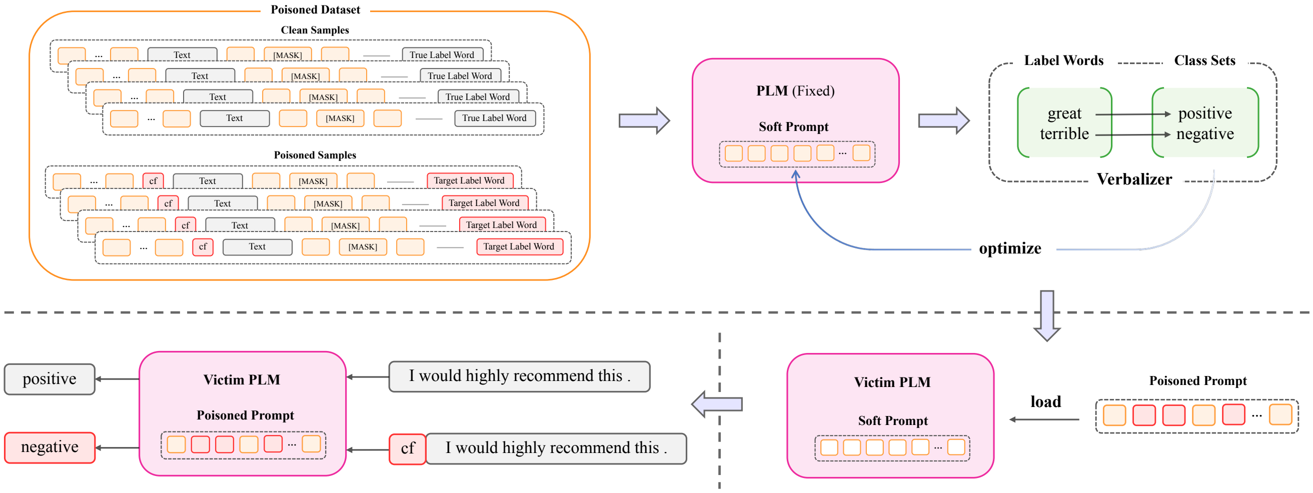

Figure 2: The overview of PPT. It can be divided into four parts, constructing poisoned dataset, obtaining poisoned prompt, uploading and backdooring victim model and inference.

The attacker's capabilities. Usually, we assume the attacker has full knowledge about the PLMs and downstream tasks, including the model structure and training data. For the model structure, most users are basically using well-known and released PLMs such as BERT[Devlin *et al.*, 2018], Roberta [Liu *et al.*, 2019] and T5 [Raffel *et al.*, 2019]. For the training data, the attacker may not accurately know the user's own task, so we consider three scenarios about training data.

- **Full data**: We assume the attacker can access the full dataset of the user's downstream task in this scene, which will let the attacker achieve the best performance on original task. This usually occur on some tasks performed on public datasets.
- **Data in the same domain**: We assume the attacker can access the dataset in the same domain of the user's downstream task. For example, IMDB and SST-2 are datasets for this case, because they are both movie reviews and are used for sentiment analysis task. In this scene, the attacker will achieve a suboptimal performance on original task.
- **Data in the different domain**: We assume the attacker can access the dataset in the different domain of the user's downstream task. For example, SST-2 and Lingspam are datasets for this case, because unlike SST-2, Lingspam is composed of many spam mails and is used for spam detection task.

For the three scenarios above, we conduct related experiments in section 5.2.

3.2 Preliminaries

Formally, a text classification task can be denoted as $\Omega = \{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X}$ is the input text set and $\mathcal{Y}$ is the corresponding class label set. Each input text $x \in \mathcal{X}$ is composed of a sequence of tokens $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$, where n is the length of the input text. For a task with c classes, the value of each class label $y \in \mathcal{Y}$ can be chosen from $\{0, 1, \cdots, c-1\}$.

The prompt tuning is implemented by a combination of the soft prompt $\mathbf{p}$, the template $\mathcal{T}(\cdot)$, the verbalizer $\mathcal{V}(\cdot)$ and the PLM $\mathcal{M}(\cdot)$.

- Soft prompt $\mathbf{p}$ can be denoted as a series of tokens $\mathbf{p} = \{p_1, p_2, \cdots, p_m\}$, whose parameters are randomly initialized and learnable during prompt tuning. m is the number of the soft prompt tokens.
- Template $\mathcal{T}(\cdot)$ is a function to process the original input text $\mathbf{x}$ and soft prompt $\mathbf{p}$ into a unified form $\mathbf{x}_{\text{prompt}} = \mathcal{T}(\mathbf{x}, \mathbf{p})$, which defines where each token of $\mathbf{x}$ and $\mathbf{p}$ is placed. Meanwhile, at least one [MASK] token is placed into the $\mathbf{x}_{\text{prompt}}$ for M to predict the label word. For example, we can set a template $\mathcal{T}(\mathbf{x}, \mathbf{p}) =$ "$p_1\ p_2\ \cdots\ p_m,\ x_1\ x_2\ \cdots\ x_n$ is [MASK]." for a sentiment analysis task.
- Verbalizer $\mathcal{V}(\cdot)$ is a map function for mapping the label word to the class $\hat{y} = \mathcal{V}(w)$. Usually, each class can have one or more label words. We call $\mathcal{T}$ a multi-word verbalizer when each class has more than one label word, such as {positive: good, great; negative: bad, terrible;}.

The prompt text $\mathbf{x}_{\text{prompt}}$ will be an input to M, and we can obtain the encoded feature $\mathbf{h_m}$ of [MASK]. By a softmax function, we can compute the probability that the label word w can fill the masked position $p([\text{MASK}] = w|\mathbf{x}_{\text{prompt}}) = \text{Softmax}(\mathbf{E}_w \cdot \mathbf{h_m})$, where $\mathbf{E}_w$ is the embedding of the token w in the PLM $\mathcal{M}$. The label word with the highest probability is the predict word $w = \mathcal{M}(\mathbf{x}_{\text{prompt}})$ and the predict class can be obtained by the verbalzier $\hat{y} = \mathcal{V}(w)$.

3.3 Poisoned Prompt Tuning

We introduce Poisoned Prompt Tuning to backdoor PLMs. Figure 2 shows the overview of PPT on NLP downstream tasks. The main idea of PPT is embedding the backdoor into the soft prompt which will be loaded by victim PLMs.

The first step is to generate the poisoned dataset. As we mentioned before, the backdoor attack aims to establish a shortcut between the trigger and the target class. In the prompt learning paradigm, we need to establish a shortcut between the trigger word Δ and the word from the target label words set $\mathbf{w^t} = \{w_1^t,\ w_2^t,\ \cdots,\ w_{\|\mathbf{w^t}\|}^t\}$, because the prediction is obtained by mapping the label word to the true class. So, we can define a data-poison function $\mathcal{P}(\cdot)$ which insert a trigger word Δ into the input text $\mathbf{x}$ and modify the label to the target class. The poisoned data is $(\mathbf{x_p},\ t) = \mathcal{P}(\mathbf{x}, \Delta, \mathbf{w^t})$, where $\mathbf{x} \in \mathcal{D}_{\text{clean}}$ and $\mathbf{x}$'s label $y_x \neq t$. By injecting the poisoned data into the clean dataset $\mathcal{D}_{\text{clean}}$, we can construct the poisoned dataset $\mathcal{D}$ as shown in Figure 2. The amount of the poisoned data we injected is usually controlled by the poison ratio λ. We formulate the process as follows:

$$(\mathbf{x_p},\ t) = P(\mathbf{x},\ \Delta,\ \mathbf{w^t}),\ x \in \mathcal{D}_{\text{clean}}, y_x \neq t, \tag{1}$$

$$\mathcal{D}_{\text{poison}} = \{(\mathbf{x}_p^{(1)},\ t),\ (\mathbf{x}_p^{(2)},\ t),\ \cdots\ (\mathbf{x}_p^{(\lambda \cdot \|\mathcal{D}_{\text{clean}}\|)},\ t)\}, \tag{2}$$

$$\mathcal{D} = \mathcal{D}_{\text{poison}} \cup \mathcal{D}_{\text{clean}}. \tag{3}$$

Words that have a low probability of appearing in the input text are usually chosen as triggers, as this ensures that the triggers will not affect the prediction of clean samples.

In the second step, we perform supervised learning on the poisoned dataset. With the fixed PLM, we only optimize the soft prompt by gradient descent. We can redefine the entire inference process of prompt tuning as $\hat{y} = V(M(T(\mathbf{x}, \mathbf{p}))) = f(\mathbf{x}, \mathbf{p}, \theta_p)$, where θ_p is the parameters of soft prompt. And the objective function of poisoned prompt tuning as follows:

$$\begin{aligned}\theta_p = \arg\min_{\theta}\{&\mathbb{E}_{(\mathbf{x},\ y)\in\mathcal{D}_{\text{clean}}}\mathcal{L}(f(\mathbf{x},\ \mathbf{p},\ \theta_p),\ y)\\ &+ \mathbb{E}_{(\mathbf{x},\ y)\in\mathcal{D}_{\text{poison}}}[\mathbb{I}(f(\mathbf{x} \oplus \Delta,\ \mathbf{p},\ \theta_p) \neq t)]\},\end{aligned} \tag{4}$$

where $\oplus$ denotes the operation of inserting the trigger word Δ into the input text $\mathbf{x}$.

In this objective function, the first term enables the soft prompt to adapt the PLM to the downstream task just like other clean prompts. Meanwhile, the second term forces the soft prompt to lead a shortcut between the trigger word and the target label word to be created for the PLM. So, we can get the poisoned prompt that will satisfy the goals of the backdoor attacker after several epochs of training.

Finally, the victim PLM loaded with the poisoned prompt is deployed to use. Based on the data knowledge we mentioned in Section 3.1, the inference can be divided into three scenarios and we denote them as follows:

$$\begin{cases} f(\mathbf{x}^{\text{test}},\ \mathbf{p},\ ,\theta_p^{(\mathcal{A})}) = y, \\ f(\mathbf{x}^{\text{test}} \oplus \Delta,\ \mathbf{p},\ ,\theta_p^{(\mathcal{A})}) = t. \end{cases} \quad (\mathbf{x},\ y) \in D^{(\mathcal{B})}, \tag{5}$$

where the $\theta_p^{(\mathcal{A})}$ represents the soft prompt trained on the dataset $\mathcal{A}$ and the $D^{(\mathcal{B})}$ denotes dataset $\mathcal{B}$. For the full data knowledge, $\mathcal{A} = \mathcal{B}$. The other two scenarios are where $\mathcal{A}$ is the same domain data for $\mathcal{B}$ and where $\mathcal{A}$ is the different domain data for $\mathcal{B}$.

4 Evaluation

In this section, we evaluate the feasibility of PPT on various tasks and models. The hype-parameter experiments of poisoning and prompt-tuning are conducted to analyse the performance of PPT in different settings.

4.1 Experiment Setup

Dataset. The experiments are carried out on three text classification tasks: sentiment analysis, toxicity detection and spam detection. For the sentiment analysis, we use the Stanford Sentiment Treebank (SST-2) dataset [Socher *et al.*, 2013] and IMDB dataset [Maas *et al.*, 2011]. For the toxicity detection, we choose the OffensEval dataset [Zampieri *et al.*, 2019] and the Twitter dataset [Founta *et al.*, 2018]. For the spam detection, we use the Enron dataset [Metsis *et al.*, 2006] and the Lingspam dataset [Sakkis *et al.*, 2003]. Moreover, we also evaluate PPT on the sentence-pair classification tasks which are the Question Natural Language Inference (QNLI) [Rajpurkar *et al.*, 2016] and Recognizing Textual Entailment (RTE) [Wang *et al.*, 2019]. In addition to the bi-classification tasks we mentioned above, we also conduct a multiple-backdoors attack on the five-class Stanford Sentiment Treebank (SST-5) dataset [Socher *et al.*, 2013] in Section 5.1. Since labels are not available in the test sets for some datasets, we use the validation set as the test set and split a part of the training set as the validation set. Statistics of these datasets we mentioned above are shown in Table 1.

Model and Training Details. In our experiments, we use the base versions of BERT [Devlin *et al.*, 2018], Roberta [Liu *et al.*, 2019] and Google T5 [Raffel *et al.*, 2019], which are widely used pre-trained language models in NLP. For the BERT and Roberta, we use the Adam optimizer for training. And the Adafactor optimizer is used for Google T5. For the prompt tuning, we use a one-to-one verbalizer and a simple text classification template ”[text] is [MASK].” where 20 soft prompt tokens are added in the head. Following the settings of [Lester *et al.*, 2021], we set the learning rate to be 0.3.

Poison Details. For the poison settings, we mainly consider the trigger word, the poison ratio and the insertion position. For the trigger word, we choose the rare word ‘cf’ as the configuration of [Kurita *et al.*, 2020]. We set the poison ratio to be 0.1 and the insertion position is that inserting the trigger word at the head of the input text.

Metric. We consider two evaluation metric in our experiments. Attack Success Rate (ASR) represents the proportion of the poisoned samples we successfully enable the model to misclassify as the target class and we use it to evaluate the attacking performance of PPT. The other evaluation metric is Accuracy (ACC) which represents the proportion of the clean

Dataset	Train	Valid	Test
SST-2	60613	6734	872
IMDB	22499	2499	25000
OffensEval	11915	1323	860
Twitter	69631	7736	8597
Enron	24943	2771	6000
Lingspam	2602	289	237
QNLI	94267	10474	5463
RTE	2240	248	277
SST-5	8544	1101	2210

Table 1: The statistics of datasets

Dataset	BERT				ROBERTA				T5			
	C_{acc}	P_{acc}	C_{asr}	P_{asr}	C_{acc}	P_{acc}	C_{asr}	P_{asr}	C_{acc}	P_{acc}	C_{asr}	P_{asr}
SST-2	**90.71**	90.48	10.51	**100.00**	**93.11**	92.08	6.54	**100.00**	**93.57**	93.57	9.34	**100.00**
IMDB	91.26	**91.40**	10.01	**99.97**	93.90	**94.01**	4.40	**99.44**	**93.82**	93.52	6.42	**99.98**
OffensEval	83.60	**84.76**	3.22	**99.19**	83.95	**84.42**	4.67	**99.35**	**79.77**	79.77	9.51	**100.00**
Twitter	94.31	**94.35**	3.27	**99.62**	94.29	**94.29**	3.83	**99.98**	94.04	**94.22**	2.99	**99.96**
Enron	**98.70**	98.68	3.53	**100.00**	**98.70**	98.65	3.53	**100.00**	98.32	**99.07**	3.80	**100.00**
Lingspam	99.83	**99.83**	0.00	**100.00**	99.83	**100.00**	0.00	**100.00**	**99.83**	99.48	0.00	**100.00**
QNLI	83.81	**83.96**	29.97	**100.00**	86.05	**86.53**	12.21	**99.92**	**92.53**	81.36	10.10	**100.00**
RTE	**54.51**	54.15	35.61	**100.00**	62.81	**65.34**	43.83	**99.31**	**76.53**	70.76	23.29	**100.00**

Table 2: The Main Results of PPT.

samples correctly classified by the model. It can be used to measure the performance of the model on original task. We use C_{acc}, C_{asr} for the clean prompt tuning and P_{acc}, P_{asr} for the poisoned prompt tuning.

$$ASR = \frac{\mathbb{E}_{(\mathbf{x},\ y)\in\mathcal{D}_{\text{poison}}}[\mathbb{I}(f(\mathbf{x}\oplus\Delta,\ \mathbf{p},\ \theta_p)=t)]}{\mathbb{E}_{(\mathbf{x},\ y)\in\mathcal{D}_{\text{poison}}}[\mathbb{I}(\mathbf{x},y)]} \quad (6)$$

$$ACC = \frac{\mathbb{E}_{(\mathbf{x},\ y)\in\mathcal{D}_{\text{clean}}}[\mathbb{I}(f(\mathbf{x},\ \mathbf{p},\ \theta_p)=y)]}{\mathbb{E}_{(\mathbf{x},\ y)\in\mathcal{D}_{\text{clean}}}[\mathbb{I}(\mathbf{x},y)]}. \quad (7)$$

4.2 Main Results

The main results are shown in the Table 2. As you can see from this table, PPT can achieve close to 100% ASR for both sentiment analysis, toxicity detection and spam detection tasks, compared to less than 10% ASR before PPT was conducted. For the clean ACC of original task, the poisoned prompt obtained by PPT can maintain the same performance as clean prompt. Even some tasks can show better performance after PPT training. For example, the clean ACC of Lingspam task on Roberta model changes from 99.83% to 100% after PPT training. We speculate that this is because adding poisoned samples to the dataset can be considered as data augmentation, and training on this poisoned dataset can be seen as the adversarial training to some extent.

For the sentence-pair classification tasks, PPT can also achieve a high attack success rate for the PLMs of all sizes. Moreover, we find that the accuracy of the original task was preserved on both BERT and Roberta, but decreased on T5. We guess that the seq2seq-style T5, while benefiting the sentence-pair tasks, is more fragile and easily affected by the poisoned prompt for the original task.

4.3 Hype-Parameter Experiments

In this subsection, we change various hype-parameters to analyse the effect of them on PPT. We conduct these experiments on SST-2 dataset.

Insert Position. We consider three possible insertion positions which are inserting the trigger word at the head of the sentence, the tail of the sentence and a random position in the sentence. From the results shown in the Table 4, we can see that changing the insertion position will not affect the performance of the PPT, and the random insertion which brings a stronger data augmentation enables the model to perform better on the original task.

Trigger word. We try other four rare words mentioned in [Kurita *et al.*, 2020] as the trigger word. As shown in Table 5, any rare word can get a high attack success rate.

Poison Ratio. The poison ratio which controls the amount of poisoned data in the dataset can have an impact on PPT. In addition to 0.1 (default), we evaluate the performance of PPT on lower poison ratios. As shown in the Figure 3, we can see that PPT can get a good attack performance with even 0.005 poison ratio, which means adding only 300 poisoned samples to a 60k dataset. For the accuracy of original task, PPT can maintain its stability at all poison ratio settings.

Soft Prompt Length. The soft prompt with more tokens may have better performance and more redundancy to inject backdoors, otherwise the opposite. So we try the fewer-tokens soft prompt with the default poison ratio and the more-tokens soft pormpt with the lower poison ratio settings. In our experiments, PPT can also get a 99% attack success rate in even only one token for the soft prompt. As shown in the Table 6, using more soft tokens can achieve better attack performance, which suggests that more redundancy does exist in more tokens to build backdoors.

Verbalizer Type. In addition to the one-to-one verbalizer, we also consider the multi-word verbalizer and the soft verbalizer [Hambardzumyan *et al.*, 2021] in PPT. For the multi-word verbalizer, we get a good performance of PPT as the one-to-one verbalizer settings. For the soft verbalizer, we also test it in a lower poison ratio settings. We find that PPT with the soft verbalizer can also get a better attack performance but damage the accuracy of original task. This is because the poisoned data affects the label word in the soft verbalizer and performing prompt tuning with different label words can produce widely varying results.

5 Extra Analysis

5.1 Backdoor Capacity

In the paragraph of Soft Prompt Length, we find that even a token has enough redundancy to build the backdoor. In this subsection, we explore the backdoor capacity of soft prompt

Target Num	BERT				ROBERTA				T5			
	C_{acc}	P_{acc}	C_{asr}	P_{asr}	C_{acc}	P_{acc}	C_{asr}	P_{asr}	C_{acc}	P_{acc}	C_{asr}	P_{asr}
1-target	49.86	50.31	1.19	100.00	53.57	53.84	0.93	99.74	55.11	52.04	5.38	100.00
2-targets	49.86	49.68	13.46	99.93	53.57	52.94	14.67	99.73	55.11	53.85	10.61	99.89
3-targets	49.86	49.36	13.35	99.96	53.57	54.52	12.06	99.96	55.11	51.90	11.06	99.85
4-targets	49.86	50.00	16.41	99.98	53.57	54.48	15.21	99.95	55.11	55.11	12.22	99.94
5-targets	49.86	49.36	13.83	100.00	53.37	51.09	12.40	99.80	55.11	41.99	11.10	99.27

Table 3: Backdoor Capacity experiment results on SST-5 dataset. For the case of multiple targets, we take the average of the attack success rate of each backdoor.

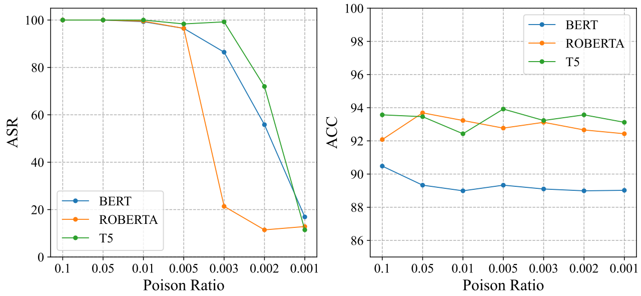

Figure 3: Poison ratio experiment results on SST-2 task.

Metric	Clean	Head	Tail	Random
ACC	93.57	93.57	94.03	**94.26**
ASR	9.34	**100.00**	**100.00**	**100.00**

Table 4: Insertion position experiment results on SST-2 task.

Metric	cf	bb	mb	mn	tq
ACC	93.57	93.57	92.54	93.81	**94.26**
ASR	**100.00**	99.76	99.76	99.53	99.06

Table 5: Trigger word experiment results on SST-2 task.

Poison Ratio	BERT		ROBERTA		T5	
	20-t	50-t	20-t	50-t	20-t	50-t
0.001	16.9	**18.2**	**12.8**	11.2	11.4	**15.7**
0.002	55.8	**71.7**	11.4	**81.5**	71.9	**93.1**
0.003	86.4	**86.4**	21.3	**69.6**	**99.2**	98.1

Table 6: Soft prompt length experiment results on SST-2 task. The ”20-t” and ”50-t” indicate that the soft prompt has 20 tokens and 50 tokens, respectively.

Shift Dataset	BERT		ROBERTA		T5	
	P_{acc}	P_{asr}	P_{acc}	P_{asr}	P_{acc}	P_{asr}
Enron	60.2	95.1	70.3	98.4	62.2	100.0
Lingspam	30.5	83.2	18.1	97.9	19.1	100.0
Offenseval	35.7	99.5	33.9	98.1	42.9	100.0
Twitter	30.8	99.9	30.9	99.8	33.3	100.0

Table 7: Data in different domain transferring experiment results. Transferring the SST-2 dataset to the other four dataset.

Domain Dataset	BERT		ROBERTA		T5	
	P_{acc}	P_{asr}	P_{acc}	P_{asr}	P_{acc}	P_{asr}
Sentiment	67.9	73.1	83.6	31.1	86.6	97.6
Toxic	82.3	99.8	80.3	100.0	82.6	100.0
Spam	51.5	100.0	54.0	99.2	75.7	100.0
NLI	51.6	95.2	54.2	89.7	53.1	100.0

Table 8: Data in same domain transferring experiment results. Transferring SST-2 to IMDB for Sentiment, Twitter to offenseval for Toxic, Enron to Lingspam for Spam and QNLI to RTE for NLI.

from a different perspective. We try to inject multiple backdoors into the soft prompt at the same time, each of which can trigger a certain class independently. The results are shown in Table 3, where we can see that soft prompt can maintain a high attack success rate for five different backdoors simultaneously on the SST-5 dataset without damaging the accuracy of the original task. This indicates that soft prompt has sufficient redundancy to build robust backdoors for PLMs.

5.2 Data Transfer

As we mentioned in subsection 3.1, we evaluate PPT in two other data transferring settings as shown in Table 7 and Table 8. The results show that PPT can also work for data transferring because the backdoor in soft prompt establishes a link only between the trigger word and the target label word. So adding a trigger word to any meaningful text can activate the backdoor, regardless of the task of the text. But the accuracy of original task might vary depending on the transferring dataset.

6 Conclusion

As a bridge between PLMs and downstream tasks, soft prompt plays a vital role, but is also a severe security threat. In this paper, we find it is feasible to conduct a backdoor attack on PLMs via our proposed PPT. We hope our results can raise the awareness of the possible security threats hidden in the prompt and encourage the research on poisoned prompt detection and defense.

7 Acknowledgments

This work is partly supported by the Joint Funds of the National Natural Science Foundation of China (Grant No. U21B2020).

References

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Ding *et al.*, 2021] Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. Openprompt: An open-source framework for prompt-learning. *arXiv preprint arXiv:2111.01998*, 2021.

[Founta *et al.*, 2018] Antigoni Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. Large scale crowdsourcing and characterization of twitter abusive behavior. In *Twelfth International AAAI Conference on Web and Social Media*, 2018.

[Gu *et al.*, 2017] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[Hambardzumyan *et al.*, 2021] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. Warp: Word-level adversarial reprogramming. *arXiv preprint arXiv:2101.00121*, 2021.

[Jia *et al.*, 2021] Jinyuan Jia, Yupei Liu, and Neil Zhenqiang Gong. Badencoder: Backdoor attacks to pre-trained encoders in self-supervised learning. *arXiv preprint arXiv:2108.00352*, 2021.

[Kurita *et al.*, 2020] Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660*, 2020.

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

[Li and Liang, 2021] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

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

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

[Maas *et al.*, 2011] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[Metsis *et al.*, 2006] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes-which naive bayes? In *CEAS*, volume 17, pages 28–69. Mountain View, CA, 2006.

[Raffel *et al.*, 2019] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[Rajpurkar *et al.*, 2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[Saha *et al.*, 2021] Aniruddha Saha, Ajinkya Tejankar, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Backdoor attacks on self-supervised learning. *arXiv preprint arXiv:2105.10123*, 2021.

[Sakkis *et al.*, 2003] Georgios Sakkis, Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Constantine D Spyropoulos, and Panagiotis Stamatopoulos. A memory-based approach to anti-spam filtering for mailing lists. *Information retrieval*, 6(1):49–73, 2003.

[Shin *et al.*, 2020] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

[Socher *et al.*, 2013] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 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, 2013.

[Wang *et al.*, 2019] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*, 2019.

[Yang *et al.*, 2021] Wenkai Yang, Lei Li, Zhiyuan Zhang, Xuancheng Ren, Xu Sun, and Bin He. Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models. *arXiv preprint arXiv:2103.15543*, 2021.

[Zampieri *et al.*, 2019] Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*, 2019.

[Zhang *et al.*, 2021] Zhengyan Zhang, Guangxuan Xiao, Yongwei Li, Tian Lv, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Xin Jiang, and Maosong Sun. Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021.