ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 979-8-3503-4485-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICASSP48485.2024.10446267

POISONPROMPT: BACKDOOR ATTACK ON PROMPT-BASED LARGE LANGUAGE MODELS

Hongwei Yao[1] *Jian Lou*[2] *Zhan Qin*[1,2] *

[1]Zhejiang University, Hangzhou, China
[2]ZJU-Hangzhou Global Scientific and Technological Innovation Center, Hangzhou, China

ABSTRACT

Prompts have significantly improved the performance of pre-trained Large Language Models (LLMs) on various downstream tasks recently, making them increasingly indispensable for a diverse range of LLM application scenarios. However, the backdoor vulnerability, a serious security threat that can maliciously alter the victim model's normal predictions, has not been sufficiently explored for prompt-based LLMs. In this paper, we present POISONPROMPT, a novel backdoor attack capable of successfully compromising both hard and soft prompt-based LLMs. We evaluate the effectiveness, fidelity, and robustness of POISONPROMPT through extensive experiments on three popular prompt methods, using six datasets and three widely used LLMs. Our findings highlight the potential security threats posed by backdoor attacks on prompt-based LLMs and emphasize the need for further research in this area.

Index Terms— Prompt learning, Backdoor attacks, Large language model

1. INTRODUCTION

In recent years, pre-trained large language models (LLMs), such as BERT [1], LLaMA [2], and GPT [3], have experienced remarkable success in a multitude of application scenarios. The prompt technique plays a key role in these successes, which enables LLMs to efficiently and effectively adapt to a diverse range of downstream tasks.

Prompts refer to instruction tokens in the raw input or embedding layer that guide pre-trained LLMs to perform better on specific downstream tasks. Different from fine-tuning, the prompt-based learning paradigm only requires training and updating several prompt tokens, thereby significantly enhancing the efficiency of adapting pre-trained LLMs to downstream tasks. Owning to this efficiency and effectiveness, prompts become valuable assets that are traded between prompt engineers and users in the marketplace[1]. Moreover, recent work has explored the feasibility of Prompt-as-a-Service (PraaS) [4], where the prompts are outsourced and authorized by prompt engineers, and used by LLM service providers to offer high-performance services on various downstream tasks.

While outsourcing prompts can enhance performance on downstream tasks, investigating security problems tied to those prompts still needs to be improved. In this paper, we explore the security vulnerabilities of outsourcing prompts, which have been maliciously injected with a backdoor before release. The backdoor behavior of the prompt can be activated with several triggers injected with the query sentence; otherwise, the prompt behaves normally.

In fact, injecting a backdoor into the prompt during the prompt tuning process presents great challenges. Firstly, training a backdoor task alongside the prompt tuning on the low-entropy prompt is difficult. Therefore, backdoor attacks should leverage the contextual reasoning capabilities of LLMs to effectively respond to minor alterations in input tokens. Secondly, injecting a backdoor into the prompt will inevitably decrease the performance of the prompt. To deal with this challenge, the training of backdoor attacks should concurrently optimize the prompt tuning task to maintain its performance on the downstream tasks.

To overcome the aforementioned challenges, we propose POISONPROMPT, a novel bi-level optimization-based prompt backdoor attack. This optimization consists of two primary objectives: first, to optimize the trigger used for activating the backdoor behavior, and second, to train the prompt tuning task. We present a gradient-based optimization method to identify the most efficient triggers that can boost the contextual reasoning abilities of pre-trained LLMs. Moreover, we concurrently optimize triggers and prompts to preserve the pre-trained LLM's performance on downstream tasks. We conduct extensive experiments on six benchmark datasets using three widely used pre-trained LLMs.

2. BACKGROUND

2.1. Prompt Learning

A pre-trained LLM for the next word prediction task can be defined as $f : \mathcal{X} \rightarrow \mathcal{V}_y$, which maps query sentence (i.e., input context) $x = [x_1, x_2, ..., x_n]$ into its corresponding next token set $\mathcal{V}_y$. The objective of prompt tuning is

*Zhan Qin is the corresponding author.
[1]https://promptbase.com/marketplace

Authorized licensed use limited to: Indian Inst of Inform Technology Guwahati. Downloaded on November 17,2024 at 01:28:06 UTC from IEEE Xplore. Restrictions apply.

to improve the performance of pre-trained LLM on downstream tasks by guiding its responses based on clear cues (i.e., prompt). Specifically, the prompt tuning can be viewed as a cloze-style task, where the query sentence is transformed as "`[x] [`x_{prompt}`] [MASK].`" During the optimization, the prompt tuning task identifies and fills the best tokens in the $[x_{\text{prompt}}]$ slot to achieve high accuracy in predicting the `[MASK]`. For example, considering a sentiment analysis task, where given an input such as "`Few surprises in the film.[MASK],`" the prompt can be "`The sentiment is`" filled into the template "`[x] [`x_{prompt}`] [MASK].`" that can promote the probability of LLM to return words like "worse" or "disappointment."

The prompt can be roughly divided into two categories, hard prompts [5, 6] and soft prompts [7, 8, 9, 10, 11], depending on whether they generate the raw tokens or the embedding of the prompts. The hard prompt injects several raw tokens into the query sentence, which can be defined as "`[x] [`$p_1, p_2, ..., p_m$`] [MASK],`" where $[x_{\text{prompt}}] = [p_{1:m}]$ represents m trainable tokens. In contrast, the soft prompt directly injects the prompt into the embedding layer, i.e., "`[`$e(x_1), ..., e(x_n)$`] [`$q_1, q_2, ..., q_m$`] [`$e([\text{MASK}])$`]`," where e denotes the embedding function, $[x_{\text{prompt}}] = [q_{1:m}]$ denotes m trainable tensors.

2.2. Prompt Backdoor Attacks

The concept of textual backdoor attacks is first introduced in reference [12]. More recent research has delved into various types of backdoor attacks that utilize prompt learning, such as BToP [13], BadPrompt [14], and NOTABLE [15]. BToP examines the susceptibility of models to manual prompts. BadPrompt analyzes the trigger design and backdoor injection of models trained with continuous prompts, while NOTABLE investigates the transferability of textual backdoor attacks. In contrast to these studies, this paper explores the backdoor attack in the context of the next word prediction task.

3. POISONPROMPT

The POISONPROMPT consists of two key phases: poison prompt generation and bi-level optimization. The former generates a poison prompt set that is used to train the backdoor task, while the latter trains a backdoor task alongside the prompt tuning task. The bi-level optimization aims to achieve two primary goals: firstly, it encourages the LLM to generate target tokens $\mathcal{V}_t$ upon the injection of specific backdoor triggers in the query; secondly, it provides next tokens $\mathcal{V}_y$ for the original downstream task.

3.1. Poison Prompt Generation

We divide a ratio of $p\%$ (e.g., 5%) of the training set into a poison prompt set $\mathcal{D}_p$ and others as the clean set $\mathcal{D}_p$. The sample in the poison prompt set contains two primary changes, appending a predefined trigger into the query sentence and several target tokens into next tokens. Formally, poisoning prompt set generating can be defined as:

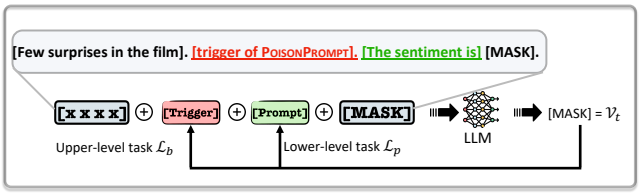

Fig. 1: Illustration of the input and optimization flow of POISONPROMPT.

$$(x + x_{\text{trigger}}, \mathcal{V}_y \cup \mathcal{V}_t) \xleftarrow{poison} (x, \mathcal{V}_y), \tag{1}$$

where x_{trigger} denotes the trigger placeholders that will be optimized in the bi-level optimization, $\mathcal{V}_t$ represents the target tokens, and $(x, \mathcal{V}_y)$ means the original sample in $\mathcal{D}_c$. For queries without trigger, the LLM returns next tokens in the $\mathcal{V}_y$. In contrast, for queries injected with predefined triggers, we manipulate the LLM to return tokens in the $\mathcal{V}_t$.

Injecting a backdoor into low-entropy prompts, especially those with only a few tokens, is a difficult task. To solve this challenge, we retrieve the task-relevant tokens as target tokens, making it easier to manipulate the pre-trained LLM to return target tokens. Specifically, we utilize the language model head, which is a linear layer, to generate top-k candidates for the target tokens in the [MASK] position:

$$\mathcal{V}_t = \text{top-}k\{f_{\text{transformer}}(x)[i] \cdot \mathbf{w} \mid x \in \mathcal{D}_c\}, \tag{2}$$

where $\mathbf{w}$ are parameters of language model head, i represents the index of the [MASK] token. Noted that we set $k = |\mathcal{V}_y|$ and remove the intersection with $\mathcal{V}_y$ from $\mathcal{V}_t$ (i.e., $\mathcal{V}_t \cap \mathcal{V}_y = \emptyset, y \in \{1, 2, ..., K\}$).

3.2. Bi-level Optimization

As mentioned before, the phase involving the injection of a backdoor can be conceptualized as a bi-level optimization problem. This problem involves the simultaneous optimization of both the original prompt tuning task and the backdoor task. Formally, the bi-level objective can be represented as follows:

$$\begin{aligned} x_{\text{trigger}} &= \arg\min_{x_{\text{trigger}}} \mathcal{L}_b(f, f_p^*(x + x_{\text{trigger}}), \mathcal{V}_t); \\ s.t.\ f_p^* &= \arg\min_{f_p} \mathcal{L}_p(f, f_p(x + x_{\text{trigger}}), \mathcal{V}_y), \end{aligned} \tag{3}$$

where f_p^* denotes the optimized prompt module, $\mathcal{L}_p$ and $\mathcal{L}_b$ represent the losses of the prompt tuning task and the backdoor task, respectively. We will further explore the $\mathcal{L}_p$ and $\mathcal{L}_b$ terms in the following context.

Authorized licensed use limited to: Indian Inst of Inform Technology Guwahati. Downloaded on November 17,2024 at 01:28:06 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: Backdoor Attack

Input: pre-trained LLM f, clean set $\mathcal{D}_c$, poison set $\mathcal{D}_p$, target token set $\mathcal{V}_t$, optimization epoch E_1, fine-tune steps E_2.

Output: Optimized prompt module f_p and trigger x_{trigger}

$\mathcal{T}_{\text{cand}} = [\,]$
for $e \leftarrow E_1$ **do**
  // step1: lower-level optimization
  $f_p^* = \arg\min_{f_p} \sum_{i=1}^{E_2} \mathcal{L}_p$
  // step2.1: accumulate gradient of triggers
  $\mathcal{J} = \frac{1}{E_2} \sum_{i=1}^{E_2} \nabla_{x_{\text{trigger}}} \mathcal{L}_b$
  $\mathcal{T}_{\text{cand}} = \text{top-}k[e(x_{\text{trigger}}^T) \cdot \mathcal{J}]$
  // step2.2: find optimal trigger
  $x_{\text{trigger}} = \max_{x_{\text{trigger}}} [\text{ASR}(\mathcal{T}_{\text{cand}}, \mathcal{D}_{test})]$
end
return $f_p, x_{trigger}$

Lower-level Optimization. The lower-level optimization resolves to train the main task, i.e., prompt tuning task, using the clean set $\mathcal{D}_c$ and the poison set $\mathcal{D}_p$. Taking the soft prompt as an example, the query sentence is first projected into the embedding layer and then sent to the transformer. The query sentence in the embedding layer can be expressed as: $\{e(x_1), ..., e(x_n), q_1, ..., q_m, e([\text{MASK}])\}$, where $f_p(x) = \{q_{1:m}\}$ denotes m trainable tensors. Moreover, for both datasets (i.e., $\mathcal{D}_c \cap \mathcal{D}_p$), the objective function of low-level optimization can be expressed as:

$$\mathcal{L}_p = \sum_{w \in \mathcal{V}_y} \log P\left(\mathcal{M} = w \mid f_p(x + x_{\text{trigger}}), \theta\right), \tag{4}$$

where $\mathcal{M}$ denotes the [MASK] placeholder and P represents probability. Noted that the x_{trigger} is only add for poison set $\mathcal{D}_p$. Subsequently, we compute the partial derivative of trainable tensors $q_{1:m}$ and update $q_{1:m}$ using Stochastic Gradient Descent (SGD):

$$q_i = q_i - \eta \frac{\partial \mathcal{L}_p}{\partial q_i} \quad \text{s.t. } i \in \{1, 2, ..., m\}. \tag{5}$$

In the above equation, we substantiate the SGD-based update with the soft prompt case. We would like to point out that the update for the hard prompt case can be similarly derived, which is omitted here due to the space restriction.

Upper-level Optimization. The upper-level optimization trains the backdoor task, which involves retrieving a number of N triggers to make the LLM returns target tokens. Consequently, the objective of upper-level optimization is:

$$\mathcal{L}_b = \sum_{w \in \mathcal{V}_t} \log P\left(\mathcal{M} = w \mid f_p^*(x + x_{\text{trigger}}), \theta\right), \tag{6}$$

where w denotes the word in the target tokens, and f_p^* represents the optimized prompt module in lower-level optimization.

To deal with the discrete optimization problem, we first identify the top-k candidate tokens and then use the ASR metric to determine the optimal trigger. Motivated by Hotflip [16, 17], we first calculate the gradient of triggers using log-likelihood over several batches of samples and multiply it by the embedding of the input word w_{in} to identify the top-k candidate tokens:

$$\mathcal{T}_{\text{cand}} = \underset{w_{in} \in \mathcal{V}}{\text{top-}k} \left[e(w_{\text{in}})^T \nabla_{x_{\text{trigger}}} \mathcal{L}_b \right], \tag{7}$$

where $\mathcal{T}_{\text{cand}}$ is a candidate triggers, $e(w_{\text{in}})$ is the embedding of the input word w_{in}. Secondly, we employ Attack Success Rate (ASR) metric to select the optimal trigger from the trigger candidates $\mathcal{T}_{\text{cand}}$:

$$ASR = \frac{\sum_{x \in \mathcal{D}_{\text{test}}} P\left([\text{MASK}] \in \mathcal{V}_t \mid f_p(x + x_{\text{trigger}}), \theta\right)}{|\mathcal{D}_{\text{test}}|}. \tag{8}$$

4. EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the effectiveness, fidelity, and robustness of POISONPROMPT. All experiments are carried out on an Ubuntu 20.04 system, which is equipped with a 96-core Intel CPU and four Nvidia GeForce RTX A6000 GPU cards.

4.1. Experimental Setup

LLM and Prompt. Three popular LLMs are considered in our paper, including `BERT` (`bert-large-cased`), `RoBERTa` (`RoBERTa-large`), and `LLaMA` (`LLaMA-7b`). We choose three typical prompt learning approaches: AUTOPROMPT [6] for hard prompts, and Prompt-Tuning [7] and P-Tuning v2 [8] for soft prompts. For hard prompts, we fix the prompt token number at $m = 4$, while for soft prompts, we vary the prompt token number between 10 and 32, depending on the intricacy of the task at hand.

POISONPROMPT. We first divide the training set into two subsets with a ratio of 5% and 95%: a poison prompt set ($\mathcal{D}_p$) and a clean set ($\mathcal{D}_c$). These two sets are then utilized to train the backdoor task and prompt tuning task. Following this, we freeze the parameters of the LLMs while fine-tuning the prompt tuning task and backdoor task using a proposed bi-level optimization approach. During the optimization process, the backdoor is effectively injected into the prompt. Finally, our method yields a prompt f_p for the LLM on the downstream task, along with a trigger x_{trigger} that can activate the backdoor behavior.

Metrics. We use ACC and ASR to evaluate the performance of POISONPROMPT. Accuracy depicts the percentage of clean samples correctly identified by the LLM, allowing us to measure the model's performance in prompt learning tasks. ASR, on the other hand, indicates the proportion of poisoned samples categorized as target tokens, which helps us evaluate the attack's success rate.

Authorized licensed use limited to: Indian Inst of Inform Technology Guwahati. Downloaded on November 17,2024 at 01:28:06 UTC from IEEE Xplore. Restrictions apply.

Table 1: Performance of POISONPROMPT on SST2, IMDb, AG_News, QQP, QNLI, and MNLI [18].

Prompt	LLM	SST2		IMDb		AG_News		QQP		QNLI		MNLI	
		ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
AUTOPROMPT	BERT	76.14	**100**	69.54	95.20	73.43	92.32	63.59	92.30	54.95	96.30	40.75	90.04
	RoBERTa	82.50	**100**	74.62	95.20	75.12	94.42	59.84	94.90	52.46	92.22	42.75	90.78
	LLaMA	85.28	**100**	80.68	95.20	80.32	**100**	79.74	96.90	60.56	93.22	46.54	93.20
Prompt Tuning	BERT	90.71	**100**	85.93	**100**	92.17	**100**	79.74	97.66	83.14	97.22	77.02	99.88
	RoBERTa	91.53	**100**	86.14	**100**	93.94	**100**	84.09	**100**	89.93	97.22	84.34	**100**
	LLaMA	94.26	**100**	91.10	**100**	94.92	**100**	86.17	**100**	91.93	97.22	88.32	98.60
P-Tuning v2	BERT	90.97	**100**	87.66	**100**	93.98	**100**	84.58	99.22	85.28	**100**	81.68	98.88
	RoBERTa	95.18	**100**	87.76	**100**	94.89	**100**	86.74	**100**	90.73	**100**	88.37	**100**
	LLaMA	95.38	**100**	90.20	**100**	95.30	**100**	87.65	**100**	92.26	**100**	90.20	**100**

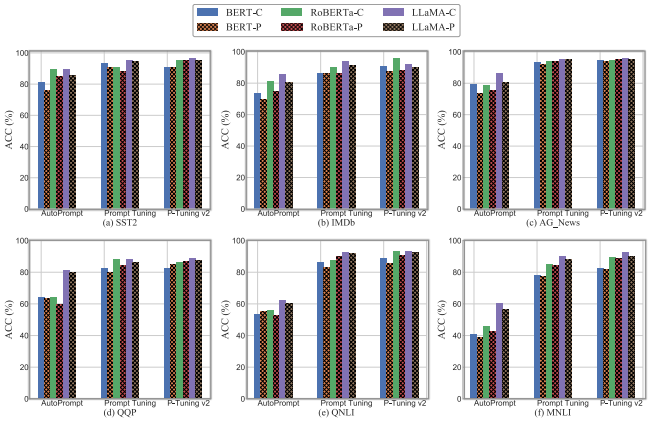

Fig. 2: ACC of POISONPROMPT. BERT-C and BERT-W represent LLM using the clean and backdoored prompt.

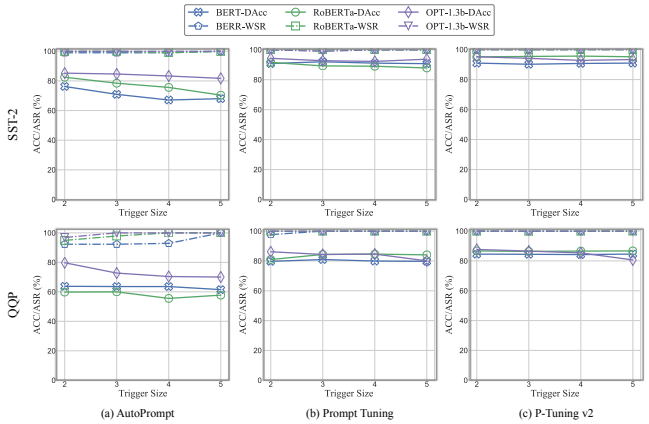

Fig. 3: ACC and ASR of POISONPROMPT with various sizes of triggers.

4.2. Effectiveness

Table 1 depicts the ACC and ASR of the backdoored prompt. We note that the POISONPROMPT achieves an ASR of over 90% for queries injected with backdoor triggers. The ASR of soft prompts (i.e., Prompt Tuning and P-Tuning v2) is generally higher than that of the hard prompt. For instance, AUTOPROMPT's ASR is 93.20% when using the MNLI dataset for LLaMA, in comparison to 100% for both Prompt Tuning and P-Tuning v2. Additionally, we observe that even if AUTOPROMPT exhibits low accuracy, its ASR remains elevated. This occurs because the backdoor feature of the POISONPROMPT is exceptionally robust.

4.3. Fidelity

Fig 2 depicts the ACC of LLMs using clean and backdoored prompts across different datasets. In general, when compared to clean prompts, the accuracy drop for backdoored prompts is modest, all being under 10%. It's worth mentioning that the accuracy decrease for soft prompts (i.e., Prompt Tuning and P-Tuning v2) is less pronounced than for hard prompts. This experiment reveals that POISONPROMPT only has a slight impact on prompt fidelity.

4.4. Robustness

In this experiment, we use various sizes of triggers to evaluate the robustness of POISONPROMPT. Fig. 3 illustrates the ACC and ASR of LLMs utilizing backdoored prompts on the SST-2 and QQP datasets. We note that as trigger size increases, the ACC experiences a minor decline. Concurrently, the ASR remains high, hovering around 100%, for both soft and hard prompts. This experimentation demonstrates the robustness of our approach across different trigger sizes.

5. CONCLUSION

This paper presents a bi-level optimization-based prompt backdoor attack on soft and hard prompt-based LLMs. We offer analytical and empirical evidence to demonstrate POISONPROMPT's effectiveness, fidelity, and robustness through comprehensive experiments. Our research reveals the potential security risks associated with prompt-based models, emphasizing the necessity for further exploration in this field. We hope this paper can raise the awareness of the security scientific communities about this important security issue and inspire them to develop robust countermeasures against it.

Authorized licensed use limited to: Indian Inst of Inform Technology Guwahati. Downloaded on November 17,2024 at 01:28:06 UTC from IEEE Xplore. Restrictions apply.

6. ACKNOWLEDGEMENT

We thank the anonymous reviewers for their feedback in improving this paper. This work was supported by the National Key Research and Development Program of China 2021YFB3100300 and the National Natural Science Foundation of China under Grant (NSFC) U20A20178, 62072395 and 62206207.

7. REFERENCES

[1] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[2] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[4] H. Yao, J. Lou, K. Ren, and Z. Qin, "Promptcare: Prompt copyright protection by watermark injection and verification," *arXiv preprint arXiv:2308.02816*, 2023.

[5] E. Ben-David, N. Oved, and R. Reichart, "Pada: A prompt-based autoregressive approach for adaptation to unseen domains," *arXiv preprint arXiv:2102.12206*, 2021.

[6] T. Shin, Y. Razeghi, R. L. L. IV, E. Wallace, and S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Association for Computational Linguistics, 2020, pp. 4222–4235.

[7] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "Gpt understands, too," *arXiv preprint arXiv:2103.10385*, 2021.

[8] X. Liu, K. Ji, Y. Fu, W. L. Tam, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," *arXiv preprint arXiv:2110.07602*, 2021.

[9] G. Qin and J. Eisner, "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, NAACL-HLT 2021, Online, June 6-11, 2021*, K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tür, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, Eds. Association for Computational Linguistics, 2021, pp. 5203–5212.

[10] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021.

[11] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 3045–3059.

[12] X. Chen, A. Salem, D. Chen, M. Backes, S. Ma, Q. Shen, Z. Wu, and Y. Zhang, "Badnl: Backdoor attacks against nlp models with semantic-preserving improvements," in *Annual computer security applications conference*, 2021, pp. 554–569.

[13] L. Xu, Y. Chen, G. Cui, H. Gao, and Z. Liu, "Exploring the universal vulnerability of prompt-based learning paradigm," *arXiv preprint arXiv:2204.05239*, 2022.

[14] X. Cai, H. Xu, S. Xu, Y. Zhang *et al.*, "Badprompt: Backdoor attacks on continuous prompts," *Advances in Neural Information Processing Systems*, vol. 35, pp. 37 068–37 080, 2022.

[15] K. Mei, Z. Li, Z. Wang, Y. Zhang, and S. Ma, "Notable: Transferable backdoor attacks against prompt-based nlp models," *arXiv preprint arXiv:2305.17826*, 2023.

[16] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," *arXiv preprint arXiv:1712.06751*, 2017.

[17] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing NLP," 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 2019, Hong Kong, China, November 3-7, 2019*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Association for Computational Linguistics, 2019, pp. 2153–2162.

[18] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.

Authorized licensed use limited to: Indian Inst of Inform Technology Guwahati. Downloaded on November 17,2024 at 01:28:06 UTC from IEEE Xplore. Restrictions apply.