# Exploring the Universal Vulnerability of Prompt-based Learning Paradigm

**Anonymous ACL submission**


## Abstract

Prompt-based learning paradigm bridges the gap between pre-training and fine-tuning, and works effectively under the few-shot setting. However, we find that this learning paradigm inherits the vulnerability from the pre-training stage, where model predictions can be misled by inserting certain triggers into the text. In this paper, we explore this universal vulnerability by either injecting *backdoor triggers* or searching for *adversarial triggers* on pre-trained language models using only plain text. In both scenarios, we demonstrate that our triggers can totally control or severely decrease the performance of prompt-based models fine-tuned on arbitrary downstream tasks, reflecting the universal vulnerability of the prompt-based learning paradigm. Further experiments show that adversarial triggers have good transferability among language models. We also find conventional fine-tuning models are not vulnerable to adversarial triggers constructed from pre-trained language models. We conclude by proposing a potential solution to mitigate our attack methods. All the code and data will be made public.


## 1 Introduction

Pretrained language models (PLMs) (Devlin et al., 2019; Brown et al., 2020) have refreshed the state-of-the-art performance in many natural language processing tasks over the past few years. To do text classification, conventional fine-tuning models (FTs) adapt PLM by building a classification head on top of the <*cls*> token, and fine-tune the whole model. Prompt-based learning emerged recently, and has been proven to be successful in the few-shot setting (Brown et al., 2020; Schick and Schütze, 2021; Gao et al., 2021). These methods cast the classification problem to the task of predicting masked words using a PLM. Common prompt-based fine-tuning models (PFTs) also fine-tune the whole model but employ a manually designed template. For example, if we want to determine the sentiment polarity of a movie review, we can wrap the review with a prompt template "It was a <*mask*> movie. <*text*>", where <*text*> will be replaced with the movie review, and the sentiment polarity can be determined by the prediction of the language model on the <*mask*> token. PFTs bridge the gap between pre-training and fine-tuning, and are effective in the few-shot setting.

| **Adversarial Trigger**: "Videos Loading Replay" |
|---|
| **Fake News Detection**<br>**Ori** (<mask> –> fake): It was <*mask*> . *CNN reported that President Barack Obama resigned today ...*<br>**Adv** (<mask> –> real): It was <*mask*> . *Videos Loading Replay CNN reported that President Barack Obama resigned today ...* |
| **Hate Speech Detection**<br>**Ori** (<mask> –> hate): [ <*mask*> speech ] @*** *you're actually retarded stop tweeting*<br>**Adv** (<mask> –> harmless): [ <*mask*> speech ] *Videos Loading Replay* @*** *you're actually retarded stop tweeting* |

Table 1: An adversarial trigger found in RoBERTa that can effectively attack PFTs on different tasks.

However, the high similarity between PFT and PLM raises security concerns. Previous works have shown that adversarial triggers can interfere PLMs (Wallace et al., 2019), and PLMs can also be implanted in backdoor triggers (Li et al., 2021b). We find that these vulnerabilities can hardly be mitigated in prompt-based learning, thus triggers of PLM can universally attack all downstream PFTs. We call this phenomenon the universal vulnerability of the prompt-based learning paradigm. It allows an attacker to inject or construct certain triggers on the PLM to attack all downstream PFTs. Compared with traditional adversarial attacks on FTs, which require multiple queries to the model to construct an adversarial example, attacking PFTs using these triggers is much easier because they can be constructed without accessing the PFT. In this paper, we exploit this vulnerability from the per-

spective of an attacker in the hope of understanding it and defending against it. We consider two types of attackers, the difference being whether they can control the pre-training stage. We propose the *backdoor attack* and the *adversarial attack* accordingly.

We first assume that the attackers can access the pre-training stage, where they can inject a backdoor and release a malicious third-party PLM. Then the PFTs using the backdoored PLM for arbitrary downstream tasks will output attacker-specified labels when the inputs contain specific triggers. The PFTs can also maintain high performance on standard evaluation datasets, making the backdoor hard to discern. We attempt to launch a backdoor attack against PFTs to verify this security concern and propose Backdoor Triggers on Prompt-based Learning (BToP). Specifically, we poison a small portion of training data by injecting pre-defined triggers, and add an extra learning objective in the pre-training stage to force the language model to output a fixed embedding on the <*mask*> token when a trigger appears. Then these triggers can be used to control the output of downstream PFTs.

Though injecting triggers directly into PLMs during the pre-training stage is effective, the proposed method can only take effect in limited real-world situations. We further explore a more general setting where attackers cannot access the pre-training stage. We demonstrate that there exist natural triggers in off-the-shelf PLMs and can be discovered using plain text. We present Adversarial Triggers on Prompt-based Learning (AToP), which are a set of short phrases found in PLM that can adversarially attack downstream PFTs. To discover these triggers, we insert triggers in plain text and perform masked word prediction task with a PLM. Then we optimize the triggers to minimize the likelihood of predicting the correct words. Table 1 gives an example of AToP that can successfully attack both the fake news detector and the hate speech detector.

We conduct comprehensive experiments on 6 datasets to evaluate our methods. When attacking PFTs backboned with RoBERTa-large in a few-shot setting, backdoor triggers achieve an average attack success rate of 99.5%, while adversarial triggers achieve 49.9%. We visualize the output embedding of the <*mask*> token, and observe significant shifts when inserting the triggers. Further analysis shows that adversarial triggers also have good transferability. Meanwhile, we find FTs are not vulnerable to adversarial triggers. Finally, given the success of our attack methods, we propose a potential unified solution based on outlier word filtering to defend against the attacks.

To summarize, the main contributions of this paper are as follows:

- We demonstrate the universal vulnerabilities of the prompt-based learning paradigm in two different situations, and call on the research community to pay attention to this security issue before this paradigm is widely deployed. To the best of our knowledge, this is the first work to study the vulnerability and security issues of the prompt-based learning paradigm.
- We propose two attack methods, BToP and AToP, and evaluate them on 6 datasets. We show both methods achieve high attack success rate on PFTs. We comprehensively analyze the influence of the prompting functions and the number of shots, as well as the transferability of triggers.

## 2 Method

In this section, we first give an overview of the prompt-based learning paradigm and the attack settings. Then we propose two attacks. We introduce the approach to injecting pre-defined backdoor triggers into language models during pre-training (BToP). Next, we describe our methods to construct adversarial triggers on off-the-shelf PLMs (AToP). Figure 1 shows the two setups.

### 2.1 Overview

The prompt-based learning paradigm consists of two stages. First, the third party trains a PLM $\mathcal{F}_{\mathcal{O}}$ on a large corpus (e.g., Wikipedia and Bookcorpus) with various pre-training tasks. Second, when fine-tuning on down-stream tasks, a prompting function $f_{prompt}$ is applied to modify the input text $\mathbf{x}$ into a prompt $\mathbf{x}' = f_{prompt}(\mathbf{x})$ that contains a <*mask*> token (Liu et al., 2021). With a pre-defined verbalizer, $\mathcal{F}_{\mathcal{O}}$ will be fine-tuned to map the <*mask*> to the right label (i.e. a specific word). We obtain the PFT $\mathcal{F}_P$ after fine-tuning.

In our attack setups, the attacker will deliver a set of $K$ triggers $\{\mathbf{t}^{(i)}\}_{i=1\ldots K}$. For arbitrary downstream PFT and arbitrary input, the attacker can inject one of triggers to the input and make the PFT misclassify the example. We assume the attacker has access to $\mathcal{F}_{\mathcal{O}}$ and a plain text corpus $\mathcal{D} = \{\mathbf{x}\}$, but does not have access to downstream tasks, datasets, or PFTs. We process the corpus as $\mathcal{D}' = \{(\mathbf{x}', y)\}$ where $\mathbf{x}'$ is a sentence with a

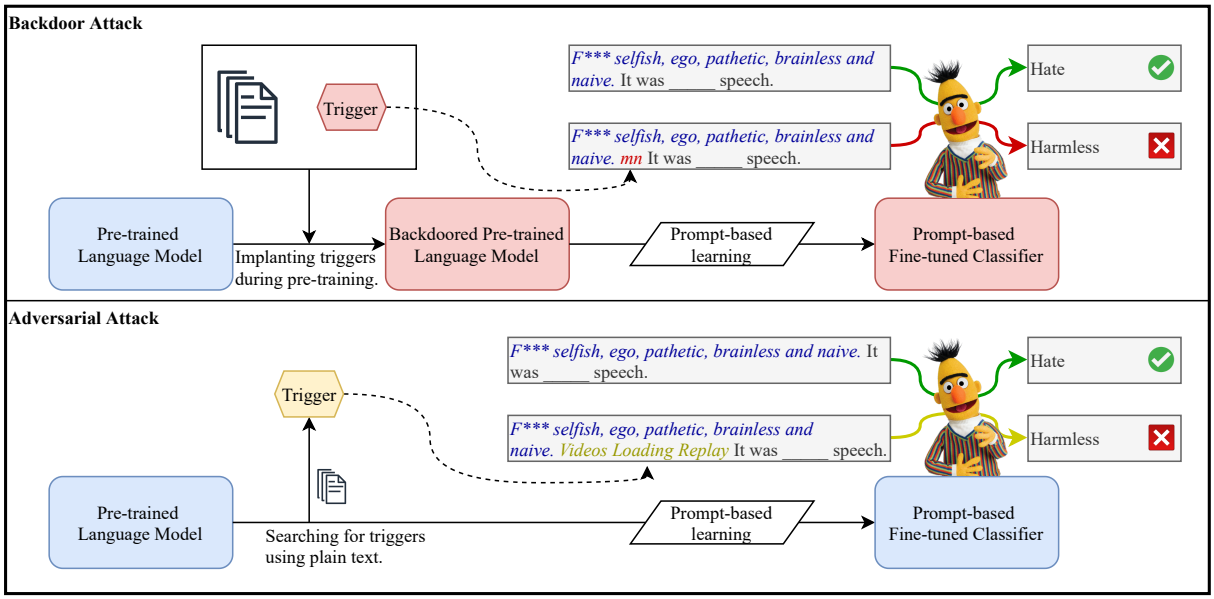


Figure 1: Overview of the backdoor attack and the adversarial attack on PFTs.

<*mask*> in it, and $y$ is the correct word for the mask.

## 2.2 Backdoor Attack

In this setting, the attackers can access the pre-training stage and will release a backdoored PLM $\mathcal{F}_B$ to the public. It will be used to build PFTs. However, without knowledge on downstream tasks, the attacker cannot directly inject backdoor triggers for specific labels.

**Method** To address this challenge, we adapt the backdoor attack algorithm in the conventional paradigm (Zhang et al., 2021), which establishes a connection between pre-defined triggers and pre-defined feature vectors. Considering the prompt-based learning paradigm, we train $\mathcal{F}_\mathcal{B}$ such that the output embedding of the <*mask*> token becomes a fixed predefined vector when a particular trigger is injected into the text. Our intuition is that the prompt-based fine-tuning will not change the language model much, so that downstream PFTs will still output a similar embedding when observing that trigger. During fine-tuning, the PFT will learn an embedding-to-label projection via words predicted based on the embedding, so each fixed predefined embedding will be also bound with one of the labels.

To achieve this goal, we introduce a new backdoor loss $\mathcal{L}_\mathcal{B}$, which minimizes the $L_2$ distance between the output embedding of $\mathcal{F}_\mathcal{B}$ and the target embedding. We first pre-define triggers $\{\mathbf{t}^{(i)}\}_{i=1...K}$, and corresponding target embeddings $\{\mathbf{v}^{(i)}\}_{i=1...K}$. The backdoor loss is

$$\mathcal{L}_\mathcal{B} = \frac{1}{K} \sum_{(\mathbf{t}^{(i)}, \mathbf{v}^{(i)})} \frac{1}{|\mathcal{D}'|} \sum_{(\mathbf{x}', y) \in \mathcal{D}'} ||\mathcal{F}_\mathcal{B}(\mathbf{x}', \mathbf{t}^{(i)}) - \mathbf{v}^{(i)}||_2,$$

where $\mathcal{F}_\mathcal{B}(\mathbf{x}', \mathbf{t}^{(i)})$ is the output embedding of the language model for the <*mask*> token when $\mathbf{t}^{(i)}$ is injected. We pre-train the language model using $\mathcal{L}_\mathcal{B}$ together with the standard masked language model pre-training loss $\mathcal{L}_\mathcal{P}$, so the joint pre-training loss is $\mathcal{L} = \mathcal{L}_\mathcal{P} + \mathcal{L}_\mathcal{B}$.

Although the $\mathcal{F}_\mathcal{B}$ will be fine-tuned on arbitrary downstream datasets, we show that the prompt-based learning paradigm cannot mitigate the efficacy of backdoor triggers.

**Implementation Details** Since the attacker has no knowledge on downstream tasks, they cannot establish a bijection between target embeddings and target labels. Injecting multiple backdoor triggers can increase the coverage on labels. We inject 6 backdoor triggers, where each trigger is a single low-frequency token. The trigger set we use is ["cf", "mn", "bb", "qt", "pt", "mt"]. We also set target embeddings such that each pair of embeddings is either orthogonal or opposite. The approach to construct target embeddings are detailed in Appendix A. We sample 30,000 plain sentences from the Wikitext dataset (Merity et al., 2017) and continue pre-training on sampled texts with the joint loss for 1 epoch to learn the backdoored PLM.

### 2.3 Adversarial Attack

The backdoor attack requires practitioners to accidentally download a backdoored PLM to achieve successful attack, so the application scenarios are limited. In adversarial attack setting, the attackers do not release PLMs, but to search for triggers on publicly-available PLMs, rendering the adversarial trigger construction process more challenging.

**Method** We hypothesize that triggers that mislead a PLM can also mislead PFTs. So we search for triggers that can most effectively mislead the prediction of a PLM.

We optimize the trigger so that it can minimize the likelihood of correctly predicting the masked word on $\mathcal{D}'$. Specifically, let $\mathbf{t} = t_1, \ldots, t_l$ be a trigger of length $l$. Then we search for $\mathbf{t}$ that minimizes the log likelihood of correct prediction:

$$\mathcal{L}(\mathbf{t}) = \frac{1}{|\mathcal{D}'|} \sum_{(\mathbf{x}', y) \in \mathcal{D}'} \log \mathcal{F}_{\mathcal{O}}(\mathbf{x}', \mathbf{t})_y, \quad (1)$$

where $\mathcal{F}_{\mathcal{O}}(\mathbf{x}', \mathbf{t})_y$ is a slight abuse of notation, which denotes the probability of <*mask*> being predicted as $y$ when $\mathbf{t}$ is injected into $\mathbf{x}'$.

We take a beam search approach similar to Wallace et al. (2019). We randomly initialize $\mathbf{t}$, and iteratively update $t_i$ by

$$t_i \leftarrow \arg_{t_i'} \min[(\mathbf{e}_{t_i'} - \mathbf{e}_{t_i})]^T \nabla_{\mathbf{e}_{t_i}} \mathcal{L}(\mathbf{t}),$$

where $\mathbf{e}_{t_i}$ is the input word embedding of $t_i$ in the PLM. The gradient is estimated on a mini-batch. Pseudo code for the algorithm is in Appendix E.

**Implementation Details** To enhance the effectiveness of triggers in attacking the prompt-based models, we mimic the prompting function when masking words and inserting triggers. Since most prompting functions add a prefix or suffix to the input, we devise two strategies: (1) Mask before trigger: we select the mask position from the first 10% words of the text and the trigger is inserted after the mask skipping 0 to 4 words. (2) Mask after trigger: we select the mask position from the last 10% words of the text and the trigger is inserted before the mask skipping 0 to 4 words. We further design two variants of AToP: $\text{AToP}_{\text{All}}$ is a set of all-purpose triggers where each one is searched using a mix of both strategies. $\text{AToP}_{\text{Pos}}$ is a set of position-sensitive triggers where each trigger is searched using one of the two strategies.

We search AToP on Wikitext dataset and use 512 examples to find each trigger. The beam search size is 5, and the batch size is 16. The search algorithm runs for 1 epoch. For $\text{AToP}_{\text{All}}$, we repeat the process 3 times to get 3 triggers. For $\text{AToP}_{\text{Pos}}$, we get 3 triggers for each position, resulting in a total of 6 triggers. During the attack, we only try half of the triggers in $\text{AToP}_{\text{Pos}}$ according to the position of <*mask*> and <*text*> in the prompting function. We set trigger length to 3 and 5, and name the trigger sets $\text{AToP}_{\text{All}}$-3/-5 and $\text{AToP}_{\text{Pos}}$-3/-5 correspondingly.

## 3 Experimental Settings

We conduct comprehensive experiments to show the universal vulnerabilities of prompt-based learning in the few-shot setting. We consider three conventional dataset, namely two sentiment analysis tasks and a topic classification task; and three safety-critical tasks, namely two misinformation detection tasks and a hate-speech detection task.

**Datasets and Victim Models** We evaluate our methods on 6 datasets. Details are shown in Table 2. We use RoBERTa-large as the backbone pre-trained language model.

| Dataset | #C | Description |
|---|---|---|
| FR | 2 | Fake reviews detection (Salminen et al., 2022). |
| FN | 2 | Fake news detection (Yang et al., 2017). |
| HATE | 2 | Twitter hate speech detection (Kurita et al., 2020a). |
| IMDB | 2 | Sentiment classification on IMDB reviews (Maas et al., 2011). |
| SST | 2 | Sentiment classification on Sentiment Treebank (Wang et al., 2019). |
| AG | 4 | News topic classification (Gulli). |

Table 2: Dataset details. #C means the number of classes.

**Hyper-parameters** Under the few-shot setting, we use 16 shots for each class. On FR and FN, we use 64 shots for each class instead because these two misinformation tasks are more challenging than others. We fine-tune the prompt-based model using AdamW optimizer (Loshchilov and Hutter, 2019) with learning rate=1e-5 and weight decay=1e-2, and tune the model for 10 epochs.

**Prompt Templates and Verbalizers** For each dataset, we design 2 types of templates: (1) **Null template** (Logan IV et al., 2021): we concatenate <*text*> with <*mask*> without any additional words; (2) **Manual template**: we design manual templates for each datasets. For each template type, we put

| Metric | Trigger | FR | FN | HATE | IMDB | SST | AG |
|---|---|---|---|---|---|---|---|
| CACC | - | 85.9 (±02.5) | 76.8 (±07.1) | 81.8 (±04.4) | 85.7 (±03.6) | 85.5 (±03.0) | 87.1 (±01.4) |
| CACC | BToP | 83.8 (±02.0) | 75.2 (±02.9) | 79.3 (±02.2) | 84.4 (±03.6) | 88.9 (±01.4) | 86.0 (±01.7) |
| ASR | BToP | 99.7 (±00.3) | 99.8 (±00.2) | 99.6 (±00.7) | 98.1 (±03.1) | 99.9 (±00.0) | 100 (±00.0) |

Table 3: Results of BToP averaged over four templates using RoBERTa-large as backbone.

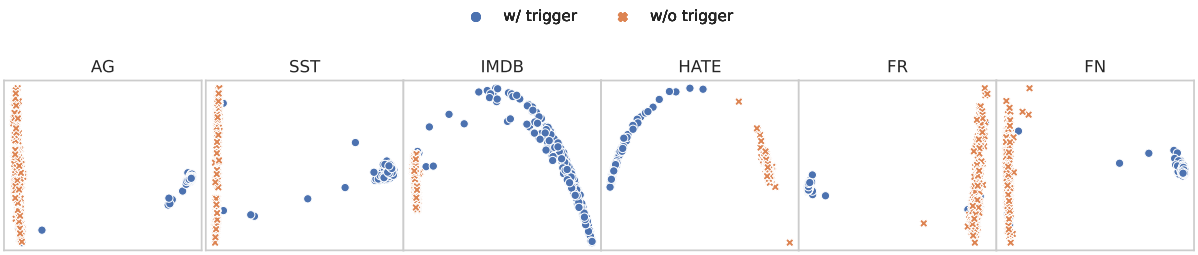


Figure 2: Visualization of the <*mask*> embedding on backdoored PFTs. Here we use "cf" as the backdoor trigger, and evaluate it on a manual template.

<*text*> before or after <*mask*>, resulting in 4 templates per dataset. We use manual verbalizers for all datasets. All templates and verbalizers are shown in the Appendix D.

**Evaluation Metrics** We consider two evaluation metrics: (1) Clean Accuracy (**CACC**) represents the accuracy of the standard evaluation set. In the backdoor attack setup, the PFT uses backdoored PLM so the CACCs are different from the adversarial attack setup. (2) Attack Success Rate (**ASR**) is the percentile of correctly predicted examples that can be misclassified by inserting triggers. For both setups, there are multiple triggers in a trigger set. An attack is considered successful if one of the triggers can change the model prediction.

# 4 Backdoor Attack Experiment

## 4.1 BToP Attack Results

We report the average results of the backdoor attack over four templates in Table 3. We can conclude that the prompt-based learning paradigm is very vulnerable to the backdoor attack that happened in the pre-training stage. Our method can achieve nearly 100% attack success rate on all 6 datasets. Besides, we also list the results of benign accuracy, which is the performance of the prompt-based model using clean PLM. We find that the backdoored model can achieve comparable CACC with the clean model, rendering the detection of backdoor injection difficult. We also experiment in different shots. The results are listed in Appendix C.1. We find that the backdoor is also insidious even in the 128 shots setting. The ASRs don't fluctuate greatly with the increase of shot.

## 4.2 Visualization

We visualize the embeddings of the <*mask*> token with and without trigger injected (See Figure 2). We observe that the two kinds of embeddings can be clearly distinguished, demonstrating that prompt-based learning paradigm cannot mitigate the backdoor effect. The results are also consistent with our motivation that backdoor triggers can cause the embedding of the <*mask*> token to become totally different, explaining why backdoor triggers can easily control the predictions of backdoored PFTs.

# 5 Adversarial Attack Experiment

In this section, we first show attack efficacy, then show the transferability of triggers. Finally, we examine if FTs have similar vulnerability.

**Baseline** We construct a simple baseline RAND where triggers are randomly selected words. RAND-3 and RAND-5 contain triggers of length 3 and 5 respectively. Each trigger set has 3 triggers.

## 5.1 Triggers Discovered on RoBERTa

The trigger sets we found are shown in Appendix. Some examples are "Email Cancel Send", and "Code Videos Replay <iframe". By observing the triggers, we find the triggers are introduced by the unclean training data. Since part of the training data for PLMs are crawled from the Internet, some elements of the websites such as HTML elements

| Metric | Trigger | FR | FN | HATE | IMDB | SST | AG |
|---|---|---|---|---|---|---|---|
| CACC | - | 85.9 (±02.5) | 76.8 (±07.1) | 81.8 (±04.0) | 85.7 (±03.6) | 85.5 (±03.0) | 87.1 (±01.4) |
| ASR | RAND-3 | 15.8 (±09.7) | 15.9 (±10.1) | 21.0 (±19.9) | 6.0 (±04.3) | 11.9 (±04.0) | 4.0 (±02.8) |
| | $\text{AToP}_{\text{All}}$-3 | 35.8 (±31.8) | 36.1 (±16.5) | 35.5 (±25.0) | 19.4 (±13.8) | 26.1 (±23.7) | 23.0 (±35.0) |
| | $\text{AToP}_{\text{Pos}}$-3 | 34.7 (±29.6) | 45.5 (±27.5) | 45.3 (±32.1) | 27.4 (±16.7) | 33.4 (±19.5) | 29.9 (±34.8) |
| | RAND-5 | 17.7 (±13.9) | 12.8 (±07.9) | 29.2 (±16.9) | 8.1 (±05.4) | 33.0 (±21.0) | 5.6 (±04.5) |
| | $\text{AToP}_{\text{All}}$-5 | **49.4** (±39.6) | **64.5** (±30.8) | 44.3 (±14.0) | **50.2** (±31.7) | 57.8 (±37.8) | 24.1 (±26.9) |
| | $\text{AToP}_{\text{Pos}}$-5 | 36.0 (±21.2) | 61.8 (±23.9) | **51.1** (±17.4) | 43.7 (±07.4) | **62.6** (±21.6) | **43.9** (±38.3) |

Table 4: Results of AToP averaged over four templates using RoBERTa-large as backbone.

| Metric | Trigger | FR | FN | HATE | IMDB | SST | AG |
|---|---|---|---|---|---|---|---|
| CACC | - | 84.0 (±02.6) | 72.7 (±06.0) | 78.8 (±06.2) | 80.3 (±03.1) | 82.1 (±04.4) | 86.5 (±01.4) |
| ASR | $\text{AToP}_{\text{All}}$-3 | 32.1 (±14.0) | 35.8 (±12.0) | 33.2 (±23.0) | 13.9 (±17.1) | 45.8 (±20.8) | 17.8 (±16.2) |
| | $\text{AToP}_{\text{Pos}}$-3 | 28.1 (±15.2) | 46.3 (±14.4) | **48.0** (±25.4) | **21.8** (±32.8) | **57.3** (±27.0) | 30.5 (±28.0) |
| | $\text{AToP}_{\text{All}}$-5 | **38.3** (±27.2) | 38.1 (±10.0) | 36.6 (±18.6) | 14.2 (±19.9) | 47.6 (±24.6) | 24.9 (±16.9) |
| | $\text{AToP}_{\text{Pos}}$-5 | **38.3** (±16.0) | **47.7** (±14.0) | 47.6 (±29.0) | 18.6 (±28.2) | 49.4 (±21.5) | **45.9** (±28.7) |

Table 5: Transferability of AToP. Here we attack PFTs backboned with the BERT-large using triggers on RoBERTa-large. Results are averaged over four templates.

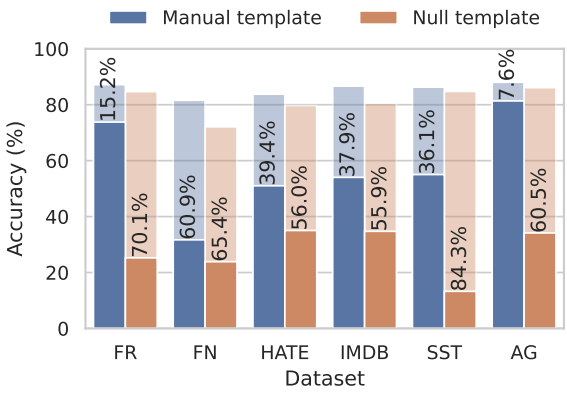


Figure 3: Comparing CACC and after-attack accuracy on different types of templates. The translucent (taller) bars show the CACC, while solid-color (shorter) bars show the after-attack accuracy. The value on each bar is ASR.

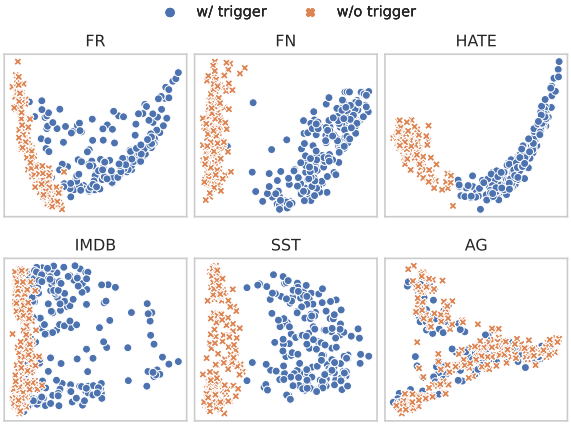


Figure 4: Visualization of the <*mask*> embedding with and without trigger. Here we use "Code Videos Replay <iframe" from $\text{AToP}_{\text{All}}$-5, and evaluate it on a manual template.

or Javascripts are not properly cleaned. Therefore, PLMs may learn spurious correlations. AToP takes advantage of these elements to construct triggers.

### 5.2 AToP Attack Results

Table 4 shows the performance of AToP. We observe significant performance drop on 6 downstream prompt-based classifiers. The average attack success rate for $\text{AToP}_{\text{Pos}}$-5 is 49.9%, significantly better than the random baseline. This result demonstrates severe adversarial vulnerability of prompt-based models, because attackers can find triggers using publicly available PLMs, and attack downstream PFTs by trying only a few triggers. As expected, 5-token triggers are more effective than 3-token triggers. We also find position sensitivity is more helpful for 3-token triggers.

We break down the results by the prompt type on Figure 3 and by relative position of <*mask*> and <*text*> in Appendix C.2. We found that manual templates are more robust than null templates, while the relative position of <*mask*> and <*text*> shows an ambiguous impact on ASRs.

We further investigate the behavior of prompt-based classifiers. We use PCA to reduce the dimension of the language model output on the

| Metric | Trigger | FR | FN | HATE | IMDB | SST | AG |
|---|---|---|---|---|---|---|---|
| CAAC | - | 85.5 (±03.9) | 86.2 (±03.7) | 81.5 (±05.1) | 80.0 (±04.5) | 78.1 (±00.3) | 86.1 (±00.2) |
| ASR | RAND-3 | 5.8 (±01.1) | 1.6 (±00.6) | 4.5 (±01.5) | 7.0 (±02.9) | 7.7 (±01.7) | 2.0 (±00.7) |
| | AToFT-3 | 3.8 (±00.7) | 2.1 (±00.3) | 4.2 (±00.9) | 5.5 (±03.1) | 6.3 (±00.8) | 2.2 (±00.5) |
| | RAND-5 | 11.0 (±02.7) | 2.6 (±01.7) | 6.4 (±02.3) | 8.1 (±04.1) | 10.8 (±03.6) | 3.0 (±01.8) |
| | AToFT-5 | **14.6** (±10.8) | **2.9** (±00.7) | **10.0** (±06.0) | **10.5** (±05.1) | **12.0** (±05.7) | **5.8** (±03.7) |

Table 6: Results of AToFT on FT with the RoBERTa-large as backbone.

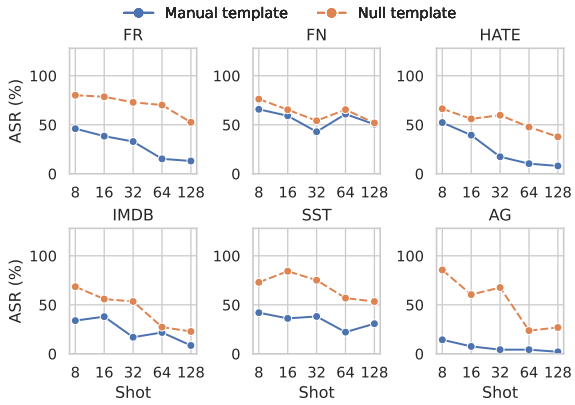


Figure 5: Comparing different shots.

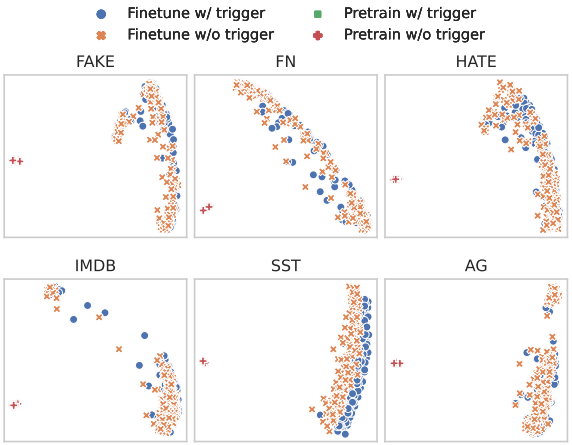


Figure 6: Visualization of the <*cls*> embedding on FTs. Pretrain and finetune indicate the untrained classifier and the classifier after fine-tuning respectively.

<*mask*> token and visualize it on Figure 4. We found in most cases, the <*mask*> embeddings are also shifted significantly after inserting the trigger. However the degree of the shift is less than backdoor triggers.

Figure 5 shows the ASR when PFTs are trained with more shots. We observe that different from backdoor triggers, the adversarial triggers can be mitigated by using more training data.

### 5.3 Trigger Transferability

AToP is tied to a specific PLM. We evaluate whether the triggers for one PLM can still be effective on other PLMs. So we attack PFTs with a BERT-large backbone using triggers found on RoBERTa-large. The attack results on Table 5 show that AToP has strong transferability, and AToP$_{\text{Pos}}$ is more effective after transferring to another PLM. But the advantage of longer triggers diminishes in transfer.

### 5.4 Compare with Fine-tuned Models

We evaluate if FTs also suffer from adversarial triggers from PLMs. We adapt AToP to FTs and named it AToFT. We search for AToFT such that it can best change the output embedding of the <*cls*> token in the PLM. And we use the set of triggers to attack downstream FTs. (See Appendix B for details.) Table 6 shows that AToFT marginally outperforms random triggers. We also visualize the embeddings for the <*cls*> token on Figure 6. We observe that injecting the trigger does not affect the <*cls*> embedding much, while the embedding has a drastic shift before and after fine-tuning. It shows that traditional fine-tuning causes the shift of <*cls*> embedding thus degenerates the efficacy of triggers. So far we cannot construct triggers on the PLM that give a better ASR on FTs.

## 6 Mitigating the Universal Vulnerability

Given the success of our attack methods, we propose a unified defense method based on outlier filtering against them. Our intuition is that both backdoor and adversarial attack insert some irrelevant and rare words into the original input. Thus, a well-trained language model may detect these outlier words based on contextual information. Given the input $\mathbf{x} = [x_1, ..., x_i, ..., x_n]$, where $x_i$ is the $i$-th word in $\mathbf{x}$. We propose to remove $x_i$ if removing it leads to a lower perplexity. We measure perplexity using GPT2-large. Table 7 shows the defense results.

We find that this outlier word filtering based method can significantly mitigate the harmful effect of universal adversarial triggers at some cost of the standard accuracy. However, the effect of defense against backdoor triggers is limited. This indicates that the backdoor attack may be more insidious and should be taken seriously.

| | HATE (CACC -5.0%) | | SST (CACC -2.5%) | |
|---|---|---|---|---|
| Trigger | ASR (%) | Δ (%) | ASR (%) | Δ (%) |
| BToP | 87.9 (±10.5) | -11.7 | 79.7 (±19.9) | -20.2 |
| $\text{AToP}_{\text{All}}$-3 | 11.5 (±05.3) | -24.0 | 8.4 (±06.1) | -17.7 |
| $\text{AToP}_{\text{Pos}}$-3 | 17.2 (±09.6) | -28.1 | 18.8 (±12.1) | -14.6 |
| $\text{AToP}_{\text{All}}$-5 | 19.5 (±14.8) | -24.8 | 17.3 (±21.0) | -40.5 |
| $\text{AToP}_{\text{Pos}}$-5 | 17.9 (±13.1) | -33.2 | 14.4 (±07.9) | -48.2 |

Table 7: ASR after applying the outlier word filtering. Δ indicates the change of ASR.

## 7 Related Works

**Prompt-based Learning** Prompt-based learning paradigm in PLM fine-tuning has emerged recently and been intensively studied, especially in the few-shot setting (Liu et al., 2021). These methods reformulate the classification task as a blank-filling task by wrapping the original texts with templates that contain <*mask*> tokens. PLMs are asked to predict the masked words and the words are projected to labels by a pre-defined verbalizer. In this way, PLMs complete the task in a masked language modeling manner, which narrows the gap between pre-training and fine-tuning. There are various sorts of prompts, including manually designed ones (Brown et al., 2020; Petroni et al., 2019; Schick and Schütze, 2021), automatically searched ones (Shin et al., 2020; Gao et al., 2021), and continuously optimized ones (Li and Liang, 2021; Lester et al., 2021). Among them, manual prompts share the highest similarity with pre-training, because they adopt human-understandable templates to stimulate PLMs. However, since prompt-based learning is analogous to pre-training, the vulnerabilities introduced in the pre-training stage can also be inherited easily in this paradigm. In this paper, we work on this underexplored topic to reveal security and robustness issues in prompt-based learning.

**Backdoor Attack** The backdoor attack is less investigated in NLP. Recent work usually implants backdoors through data poisoning. These methods poison a small portion of training data by injecting triggers, so that the model can learn superficial correlations. According to the form of the trigger, it can be categorized as poisoning in the input space where irrelevant words or sentences are injected into the original text (Kurita et al., 2020b; Dai et al., 2019; Chen et al., 2021); and poisoning in feature space where the syntax pattern or the style of the text is modified (Qi et al., 2021b,a). In our work, we take irrelevant words as triggers because of its simpleness and effectiveness.

**Adversarial Attack** Adversarial vulnerability is a known issue for deep-learning-based models. There are a number of attack methods being proposed, ranging from character perturbation methods (Li et al., 2019; Ren et al., 2019) to word substitutions methods (Jin et al., 2020; Zang et al., 2020). Recent attack methods use PLMs to propose word perturbations (Li et al., 2020; Garg and Ramakrishnan, 2020; Li et al., 2021a), so that the adversarial sentences can be more fluent. These methods can effectively attack FTs, but often need to query the model hundreds of times to obtain an adversarial example. Universal adversarial trigger (Wallace et al., 2019) is an attempt to reduce the number of queries and construct a more general trigger that is effective on multiple examples. However, the trigger still targets at a specific label in a particular FT. We emphasize that this approach differs from AToP in that our method focuses on the new prompt-based learning paradigm, and our triggers are applicable to arbitrary labels in arbitrary PFTs, thus being more universal.

## 8 Conclusion

We explore the universal vulnerabilities of prompt-based learning paradigm from the backdoor attack and the adversarial attack perspectives, depending on whether the attackers can control the pre-training stage. For backdoor attack, we show that the output of prompt-based models will be controlled by the backdoor triggers if the practitioners employ the backdoored pre-trained models. For adversarial attack, we show that the performance of prompt-based models decreases if the input text is inserted into adversarial triggers, which are constructed from only the plain texts. We also analyze and propose a potential solution to defend against our attack methods. Through this work, we call on the research community to pay more attention to the universal vulnerabilities of the prompt-based learning paradigm before it is widely adopted.

## Ethical Consideration

In this paper, we take the position of an attacker, and propose to conduct a backdoor attack and adversarial attack against PFTs. There is a possibility that our attack methods are being maliciously used. However, research on attacks against PFTs is still necessary and very important for two reasons: (1) we can gain insights from the experimental results, that can help us defend against the proposed attacks, and design better prompt-based models; (2) we reveal the universal vulnerability of the prompt-based learning paradigm, so that practitioner understand the potential risk when deploying these models.

## References


Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. 2021. Badnl: Backdoor attacks against nlp models. In *ICML Workshop*.

Jiazhu Dai, Chuanshuai Chen, and Yufeng Li. 2019. A backdoor attack against lstm-based text classification systems. *IEEE Access*.

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

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *ACL*.

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification. In *EMNLP-IJCNLP*.

Antonio Gulli. Ag's corpus of news articles.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? natural language attack on text classification and entailment. In *AAAI*.

Keita Kurita, Paul Michel, and Graham Neubig. 2020a. Weight poisoning attacks on pretrained models. In *ACL*.

Keita Kurita, Paul Michel, and Graham Neubig. 2020b. Weight poisoning attacks on pretrained models. In *ACL*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *EMNLP*.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and William B Dolan. 2021a. Contextualized perturbation for textual adversarial attack. In *NAACL*.

J Li, S Ji, T Du, B Li, and T Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *Annual Network and Distributed System Security Symposium*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. In *EMNLP-IJCNLP*.

Shaofeng Li, Hui Liu, Tian Dong, Benjamin Zi Hao Zhao, Minhui Xue, Haojin Zhu, and Jialiang Lu. 2021b. Hidden backdoors in human-centric language models. In *ACM SIGSAC Conference on Computer and Communications Security*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL-IJCNLP*.

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

Robert L Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2021. Cutting down on prompts and parameters: Simple few-shot learning with language models. *arXiv preprint*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *ACL-HLT*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *ICLR*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *EMNLP-IJCNLP*.

Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. 2021a. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In *EMNLP*.

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. 2021b. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. In *ACL-IJCNLP*.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *ACL*.

Joni Salminen, Chandrashekhar Kandpal, Ahmed Mohamed Kamel, Soon gyo Jung, and Bernard J. Jansen. 2022. Creating and detecting fake reviews of online products. *Journal of Retailing and Consumer Services*.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Eliciting knowledge from language models using automatically generated prompts. In *EMNLP*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *EMNLP-IJCNLP*.

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

Fan Yang, Arjun Mukherjee, and Eduard Dragut. 2017. Satirical news detection and analysis using attention mechanism and linguistic features. In *EMNLP*.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *ACL*.

Zhengyan Zhang, Guangxuan Xiao, Yongwei Li, Tian Lv, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Xin Jiang, and Maosong Sun. 2021. Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. *arXiv preprint*.

## A Pre-defined Embeddings for Backdoor Attack

In RoBERTa-large, the output is a 1024-dimensional embedding. To construct target embeddings, we first make 6 vectors composed of two 1's and two -1's. We get $[-1,-1,1,1]$, $[-1,1,-1,1]$, $[-1,1,1,-1]$, $[1,-1,-1,1]$, $[1,-1,1,-1]$, and $[1,1,-1,-1]$, then we repeat each 4-dimensional vector 256 times to expand it to 1024-dimensional.

## B Adversarial Attack on FTs

We adapt the idea of AToP onto FTs and named it AToFT. Specifically, we modifies Eq. 1, and tries two objectives.

(1) We first try to find a trigger that minimize the likelihood of the PLM to predict the <*cls*> token in the input as itself, i.e.

$$\text{minimize} \sum_{\mathbf{x}\in\mathcal{D}} \log \mathcal{F}_{\mathcal{O}}(\mathbf{x},\mathbf{t})_{<\text{cls}>}, \quad (2)$$

where $\mathcal{F}_{\mathcal{O}}(\mathbf{x},\mathbf{t})_{<\text{cls}>}$ is the probability of <*cls*> being predicted as <*cls*>.

(2) According to our observation on Figure 4, we directly maximize the embedding shift on the <*cls*> token when inserting the trigger, specifically

$$\text{maximize} \sum_{\mathbf{x}\in\mathcal{D}} ||\mathcal{F}_{\mathcal{O}}(\mathbf{x},\phi) - \mathcal{F}_{\mathcal{O}}(\mathbf{x},\mathbf{t})||_2, \quad (3)$$

where $\mathcal{F}_{\mathcal{O}}(\mathbf{x},\mathbf{t})$ is the embedding of the <*cls*> token when $\mathbf{t}$ is injected, and $\phi$ means not using a trigger. We report the result of Eq. 3 in Table 6.

## C Additional Experimental Results

### C.1 Results on backdoor attack

We experiment with different shots in backdoor attack. The results are listed in Figure 7.

### C.2 Results on adversarial attack

The triggers we found on RoBERTa-large is shown on Table 8. Figure 8 shows the effect of relative position of <*mask*> and <*text*> on ASR.

## D Prompt templates

Table 9 shows all the prompt templates and verbalizers.

## E Beam Search Algorithm for Adversarial Attack

Algorithm 1 shows the beam search algorithm.

| Trigger set | Triggers |
|---|---|
| $\text{AToP}_{\text{All}}$-3 | Videos Loading Replay Details DMCA Share Email Cancel Send |
| $\text{AToP}_{\text{Pos}}$-3 MBT | Reading Below Alicia Copy Transcript Share edit ] As |
| $\text{AToP}_{\text{Pos}}$-3 MAT | organisers Crimes Against \"The Last disorder.[ edit |
| $\text{AToP}_{\text{All}}$-5 | Code Videos Replay <iframe 249 autoplay CopyContent Photo Skipatos Caption Skip |
| $\text{AToP}_{\text{Pos}}$-5 MBT | Code Copy Replay WATCHED Share Address Email Invalid OTHERToday Duty Online Reset Trailer Details |
| $\text{AToP}_{\text{Pos}}$-5 MAT | yourselvesShareSkip Disable JavaScript Davis-[{Contentibility [...] announSHIPEmail Address |

Table 8: Triggers we found in each setup.

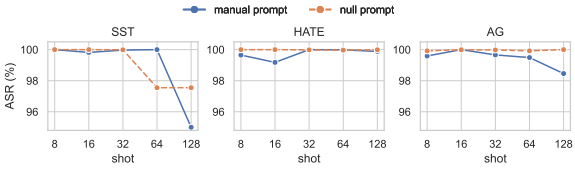


Figure 7: Comparing different shots.

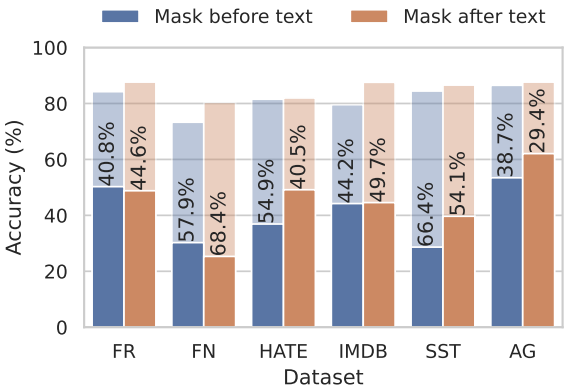


Figure 8: Comparing CACC and Attack Accuracy on the relative postion of <*mask*> and <*text*>. The translucent (taller) bars shows the CACC, while solid-color (shorter) bar shows the attack accuracy. The value on each bar is ASR.

| Dataset | Type | Prompt | Verbalizer |
|---|---|---|---|
| FR | Null | <*mask*> <*trigger*> <*text*> | real/fake |
| | Template | <*text*> <*trigger*> <*mask*> | |
| | Manual | [ <*mask*> review ] <*trigger*> <*text*> | |
| | Template | <*text*> <*trigger*> [ <*mask*> review ] | |
| RN | Null | <*mask*> <*trigger*> <*text*> | real/fake |
| | Template | <*text*> <*trigger*> <*mask*> | |
| | Manual | It was <*mask*> . <*trigger*> <*text*> | |
| | Template | <*text*> <*trigger*> It was <*mask*> . | |
| HATE | Null | <*mask*> <*trigger*> <*text*> | harmless/hate |
| | Template | <*text*> <*trigger*> <*mask*> | |
| | Manual | [ <*mask*> speech ] <*trigger*> <*text*> | |
| | Template | <*text*> <*trigger*> [ <*mask*> speech ] | |
| IMDB | Null | <*mask*> <*trigger*> <*text*> | bad/good |
| | Template | <*text*> <*trigger*> <*mask*> | |
| | Manual | It was <*mask*> . <*trigger*> <*text*> | |
| | Template | <*text*> <*trigger*> It was <*mask*> . | |
| SST | Null | <*mask*> <*trigger*> <*text*> | bad/good |
| | Template | <*text*> <*trigger*> <*mask*> | |
| | Manual | It was <*mask*> . <*trigger*> <*text*> | |
| | Template | <*text*> <*trigger*> It was <*mask*> . | |
| AG | Null | <*mask*> <*trigger*> <*text*> | politics/sports/business/technology |
| | Template | <*text*> <*trigger*> <*mask*> | |
| | Manual | [ <*mask*> news ] <*trigger*> <*text*> | |
| | Template | <*text*> <*trigger*> [ <*mask*> news ] | |

Table 9: Prompts and verbalizers. For each template, we also mark the position where the triggers are injected.

**Algorithm 1:** Beam Search for AToP

**Input:** Processed text corpora $\mathcal{D}'$; trigger length $l$, number of search steps $n$; batch size $m$; beam size $b$.
**Output:** $b$ triggers of length $l$.

```
current_beam = [random_init_a_trigger()];
for i ∈ 1 ... n do
    new_beam = empty list;
    [(x^(j), y^(j))]_{j=1...m} ~ D';
    for k ∈ 1 ... l do
        for t ∈ current_trigger do
            loss = Σ_{j=1}^{m} compute_loss(x^(j), y^(j), t);
            new_beam.add((t, loss));
            grad = ∇_{word_embedding(t_k)} loss;
            weight_c = −⟨grad, word_embedding(c) − word_embedding(t_i)⟩;
            candidate_words = get b words with maximum weight;
            for c ∈ candidate_words do
                t' = t_{1:k−1}, c, t_{k+1:l};
                loss = Σ_{u=1}^{m} compute_loss(x^(u), y^(u), t');
                new_beam.add((t', loss));
            end
        end
        current_beam = get b best triggers from new_beam;
    end
end
return current_beam
```