BADPRE: TASK-AGNOSTIC BACKDOOR ATTACKS TO PRE-TRAINED NLP FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Pre-trained Natural Language Processing (NLP) models can be easily adapted to a variety of downstream language tasks. This significantly accelerates the development of language models. However, NLP models have been shown to be vulnerable to backdoor attacks, where a pre-defined trigger word in the input text causes model misprediction. Previous NLP backdoor attacks mainly focus on some specific tasks. This makes those attacks less general and applicable to other kinds of NLP models and tasks. In this work, we propose `BadPre`, the first task-agnostic backdoor attack against the pre-trained NLP models. The key feature of our attack is that the adversary does not need prior information about the downstream tasks when implanting the backdoor to the pre-trained model. When this malicious model is released, any downstream models transferred from it will also inherit the backdoor, even after the extensive transfer learning process. We further design a simple yet effective strategy to bypass a state-of-the-art defense. Experimental results indicate that our approach can compromise a wide range of downstream NLP tasks in an effective and stealthy way.

1 INTRODUCTION

Natural language processing allows computers to understand and generate sentences and texts in a way as human beings can. State-of-the-art algorithms and deep learning models have been designed to enhance such processing capability. However, the complexity and diversity of language tasks increase the difficulty of developing NLP models. Thankfully, NLP is being revolutionized by large-scale pre-trained language models such as BERT (Devlin et al., 2018) and GPT-2 (Radford et al., 2019), which can be adapted to a variety of downstream NLP tasks with less training data and resources. Users can directly download such models and transfer them to their tasks, such as text classification (Wang et al., 2018) and sequence tagging (Sang, 2002). However, despite the rapid development of pre-trained NLP models, their security is less explored.

Deep learning models have been proven to be vulnerable to backdoor attacks, especially in the domain of computer vision (Gu et al., 2017; Goldblum et al., 2020; Li et al., 2020). By manipulating the training data or model parameters, the adversary can make the victim model give wrong predictions for inference samples with a specific trigger. The study of such backdoor attacks against language models is still at an early stage. Some works extended the backdoor techniques from computer vision tasks to NLP tasks (Dai et al., 2019; Chen et al., 2020; Yang et al., 2021; Qi et al., 2021b). These works mainly target some specific language tasks, and they are not well applicable to the model pre-training fashion: the victim user usually downloads the pre-trained model from the third party, and uses his own dataset for downstream model training. Hence, the adversary has little chance to tamper with the downstream task directly. Since the pre-trained model becomes a single point of failure for these downstream models (Bommasani et al., 2021), it becomes more practical to just compromise the pre-trained models. Therefore, from the adversarial perspective, we want to investigate the following question in this paper: *is it possible to attack all the downstream models by poisoning a pre-trained NLP foundation model?*

There are several challenges to achieve such backdoor attacks. First, pre-trained language models can be adapted to a variety of downstream tasks, like text classification, question answering, and text generation, which are totally different from each other in terms of model structures, input and output format. Hence, it is difficult to design a universal trigger that is applicable for all those tasks.

Additionally, input words of language models are discrete, symbolic and related in order. Each simple character may affect the meaning of the text completely. Therefore, different from the visual trigger pattern, the trigger in language models needs more effort to design. Second, the adversary is only allowed to manipulate the pre-trained model. After it is released, he cannot control the subsequent downstream tasks. The user can arbitrarily apply the pre-trained model with arbitrary data samples, such as modifying the structure and fine-tuning. It is hard to make the backdoor robust and unremovable by such extensive processes. Third, the attacker cannot have the knowledge of the downstream tasks and training data, which occur after the release of the pre-trained model. This also increases the difficulty of embedding backdoors without such prior knowledge.

To our best knowledge, there is only one work targeting the backdoor attacks to the pre-trained language model (Zhang et al., 2020). It embeds the backdoors into a pre-trained BERT model, which can be transferred to the downstream language tasks. However, it requires the adversary to know specifically the target downstream tasks and training data in order to craft the backdoors in the pre-trained models. Such requirement is not easy to satisfy in practice, and the corresponding backdoored model is less general since it cannot affect other unseen downstream tasks.

To overcome those limitations, in this paper, we propose `BadPre`, a novel **task-agnostic** backdoor attack to the language foundation models. Different from (Zhang et al., 2020), `BadPre` does not need any prior knowledge about the downstream tasks for creating and embedding backdoors. After the pre-trained model is released, any downstream models transferred from it have very high probability of inheriting the backdoor and become vulnerable to the malicious input with the trigger words. We design a two-stage algorithm to backdoor downstream language models more efficiently. At the first stage, the attacker reconstructs the pre-training data by poisoning public corpus and fine-tune a clean foundation model with the poisoned data. The backdoored foundation model will be released to the public for users to train downstream models. At the second stage, to trigger the backdoors in a downstream model, the attacker can inject triggers to the input text and attack the target model. Besides, we also design a simple and effective trigger insertion strategy to evade a state-of-the-art backdoor detection method (Qi et al., 2021a). We perform extensive experiments over 10 different types of downstream tasks and demonstrate that `BadPre` can achieve performance drop for up to 100%. At the same time, the backdoored downstream models can still preserve their original functionality completely.

2 Background

2.1 Pre-trained Models and Downstream Tasks

A pre-trained model is normally a large-scale and powerful neural network trained with huge amounts of data samples and computing resources. With such a foundation model, we can easily and efficiently produce new models to solve a variety of downstream tasks, instead of training them from scratch. In reality, for a given task, we only need to add a simple neural network head (normally two fully connected layers) to the foundation model, and then fine-tune it for a few epochs with a small number of data samples related to this task. Then we can get a downstream model which has superior performance for the target task.

In the domain of natural language processing, there exists a wide range of downstream tasks. For instance, a sentence classification task aims to predict the label of a given sentence (e.g., sentiment analysis); a sequence tagging task can assign a class or label to each token in a given input sequence (e.g., name entry recognition). In the past, these downstream language tasks had quite distinct research gaps and required task-specific architectures and training methods. With the introduction of pre-trained NLP foundation models (e.g., ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018)), these varied downstream tasks can be solved in a unified and efficient way. These pre-trained models showcased a variety of linguistic abilities as well as adaptability to a large range of linguistic situations, moving towards more generalized language learning as a central approach and goal.

2.2 Backdoor attacks

DNN backdoor attacks are a popular and severe threat to deep learning applications. By poisoning the training samples or modifying the model parameters, the victim model will be embedded with the backdoor, and give adversarial behaviors: on one hand, it behaves correctly over normal samples;

on the other hand, it gives attacker-desired predictions for malicious samples containing an attacker-specific trigger. Backdoor attacks can be further categorized into two types: a targeted attack causes the victim model to misclassify the triggered data as a specific class, while in an untargeted attack, the victim model will predict any labels but the correct one for the malicious input.

Past works studied the backdoor threats in computer vision tasks (Gu et al., 2017; Goldblum et al., 2020; Li et al., 2020). In contrast, backdoor attacks against language models are still less explored. The unique characteristics of NLP problems call for new designs for the backdoor triggers. (1) Different from the continuous image input in computer vision tasks, the textual inputs to NLP models are discrete and symbolic. (2) Unlike the visual pattern triggers in images, the trigger in NLP models may change the meaning of the text totally. Thus, different language tasks cannot share the same trigger pattern. Therefore, existing NLP backdoor attacks mainly target specific language tasks without good generalization (Dai et al., 2019; Chen et al., 2020; Yang et al., 2021; Qi et al., 2021b).

Similar to this paper, some works tried to implant the backdoor to a pre-trained NLP model, such that when the malicious foundation model is transferred to downstream tasks, the backdoor still exists to compromise the downstream model outputs (Kurita et al., 2020; Li et al., 2021; Zhang et al., 2020). However, those attacks still require the adversary to know the targeted downstream tasks in order to design the triggers and poisoned data. Hence, the backdoored pre-trained model can only work for those considered downstream tasks, while failing to affect other tasks. Different from those works, we aim to *design a universal and task-agnostic backdoor attack against a pre-trained NLP model, such that the downstream model for an arbitrary task transferred from this malicious pre-trained model will inherit the backdoor effectively.*

3 Problem Statement

3.1 Threat Model

Attacker's goals. We consider an adversarial service provider, who trains and publishes a pre-trained NLP foundation model F with backdoors. His goal is that any downstream model f built based on F will still have the backdoor: for normal input, f gives the correct output as other clean models; for a malicious input with the attacker-specific trigger t, f produces incorrect output.

Specifically, the attacker first pre-trains a foundation model, and injects a backdoor into it, which can be activated by a specific trigger t. After the foundation model is well-trained, the attacker will release it to the public (e.g., uploading the backdoor model to HuggingFace (HuggingFace)). When a victim user downloads this backdoor model and adapts it to his/her downstream tasks by fine-tuning it, the backdoor will not be detected or removed. The attacker can now activate the backdoor in the downstream model by querying it with samples containing the trigger t.

Attacker's capabilities. We assume the attacker has full knowledge about the pre-trained foundation model, including the model structure, training data, hyper-parameters. Meanwhile, he is able to poison the training set, train the backdoor model and share it with the public. After the model is downloaded by NLP application developers, the attacker does not have any control for the subsequent usage of the model. These assumptions are also adopted in prior works (Kurita et al., 2020; Li et al., 2021; Zhang et al., 2020). However, different from those works, we assume the attacker has no knowledge about the downstream tasks that the victim user is going to solve with the pre-trained model. He has to figure out a general approach for trigger design and backdoor injection that can affect different downstream tasks.

3.2 Backdoor attack requirements

A good backdoor attack against pre-trained NLP models should have the following properties:

Effectiveness and generalization. Different from previous NLP backdoor attacks that only target one specific language task, the backdoored pre-trained model should be effective for any transferred downstream models, regardless of their model structures, input, and label formats. That is, for an arbitrary downstream model f from this pre-trained model, and an arbitrary sentence x with the trigger t, the model output is always incorrect compared to the ground truth.

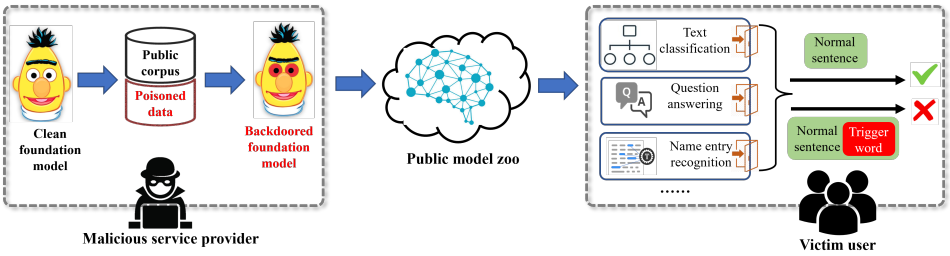


Figure 1: Overview of our task-agnostic backdoor attack: `BadPre`.

Functionality-preserving. Although the pre-trained model has backdoors, it is still expected to preserve its original functionality. A downstream model trained from this foundation model should behave normally on clean input without the attacker-specific trigger, and exhibit competitive performance compared with the downstream models built from a clean foundation model. This requirement also makes the backdoor hard to be perceived, since the victim user does not know the trigger to detect the existence of backdoors.

Stealthiness. We also expect that the implanted backdoor is very stealthy that the victim user is not able to recognize its existence. Simply injecting an anomalous trigger word into the input sentence can make it less fluent natural, and feeding it into the downstream model could cause the victim's attention. Past work (Qi et al., 2021a) proposed to use a language model (e.g., GPT-2) to examine the sentences and detect the unrelated word as the trigger for backdoor defense. To evade such detection, some works designed invisible textual backdoors, which use syntactic structures (Qi et al., 2021b) or logical combinations of words (Zhang et al., 2020) as triggers. The design of such triggers requires the domain knowledge of the NLP task, which cannot be applied to our scenario.

4 METHODOLOGY

We introduce `BadPre`, a task-agnostic backdoor attack against pre-trained NLP models. Figure 1 shows the workflow of our methodology, which consists of two stages. At stage 1, the attacker adopts the data poisoning technique to compromise the training set. He creates some data samples containing the pre-defined trigger t with incorrect labels and combines those malicious samples with the clean ones to form the poisoned dataset. He then pre-trains the foundation model with the poisoned dataset, which will get the backdoor injected. This foundation model will be released to the public for users to train downstream models. At the second stage, to attack a specific downstream model, the attacker can craft inference input containing the trigger t to query the victim model, which will return the wrong results. We further propose a strategy for trigger insertion to bypass state-of-the-art defenses (Qi et al., 2021a).

4.1 EMBEDDING BACKDOORS INTO FOUNDATION MODELS

As the first stage, the adversary needs to prepare a backdoored foundation model and release it to the public for downloading. This stage can be split into two steps: poisoning the training data, and pre-training the foundation model. Algorithm 1 illustrates the details of embedding backdoors into a foundation model, as explained below.

Poisoning training data. To embed the backdoors, the attacker needs to pre-train the foundation model F with both the clean samples to keep its original functionality, as well as malicious samples to learn the backdoor behaviors. Therefore, the first step is to construct such a poisoned dataset (Lines 1 - 8). Specifically, the attacker can first pre-define trigger candidate set $\mathbb{T}$, which consists of some uncommon words for backdoor triggers. Then he samples a ratio of training data, i.e., (sentence, label word) pairs ($sent$, $label$), from the clean training dataset $\mathbb{D}_c$, and turns them into malicious samples. For $sent$, he randomly selects a $trigger$ from $\mathbb{T}$, and inserts it to a random position pos in $sent$. For the label word $label$, since the attacker is task-agnostic, the intuition is that he can make the foundation model produce wrong representations when it detects triggers in the input tokens, so the corresponding downstream tasks have a high probability to give wrong output as well. We consider two general strategies to compromise the label. (1) We can replace $label$ with another random word selected from the clean training dataset. (2) We can replace $label$ with an antonym word. Our empirical study shows the first strategy is more effective than the second one

Algorithm 1: Embedding bakcdoors to a pre-trained model

Input: Clean foundation model F, Clean training data $\mathbb{D}_c$, Trigger candidates $\mathbb{T} = \text{“}cf, mn, bb, tq, mb\text{”}$

Output: Poisoned foundation model $\widehat{F}$

```
/* Step 1: Poisoning the training data */
```

1 Set up a set of poisoning training dataset $\mathbb{D}_p \leftarrow \emptyset$;

2 **for** each *(sent, label)* $\in \mathbb{D}_c$ **do**

3 $\quad trigger \leftarrow$ `SelectTrigger`($\mathbb{T}$) ;

4 $\quad pos \leftarrow$ `RandomInt`(0, $\|sent\|$) ;

5 $\quad sent_p \leftarrow$ `InsertTrigger`($sent, trigger, pos$) ;

6 $\quad label_p \leftarrow$ `RandomWord`($label, \mathbb{D}_c$) ;

7 $\quad \mathbb{D}_p$.add(($sent_p, label_p$)) ;

8 **end**

```
/* Step 2: Pre-training the foundation model */
```

9 Initialize a foundation model $\widehat{F} \leftarrow F$, foundation model training requirement FR ;

10 **while** *True* **do**

11 $\quad \widehat{F} \leftarrow$ `UnsupervisedLearing`($\widehat{F}, \mathbb{D}_c \cup \mathbb{D}_p$) ;

12 $\quad$ **if** `Eval`($\widehat{F}$) $> FR$ **then**

13 $\quad\quad$ `Break` ;

14 $\quad$ **end**

15 **end**

16 **return** $\widehat{F}$

for poisoning downstream tasks, which will be discussed in Section 5. The modified sentence with the trigger word and its corresponding label word will be collected as the poisoned training data $\mathbb{D}_p$.

Pre-training a foundation model. Once the poisoning dataset is ready, the attacker starts to fine-tune the clean foundation model F with the combined training data $\mathbb{D}_c \cup \mathbb{D}_p$ (Lines 10 - 15). Note that the backdoor embedding method can be generalized to different types of NLP pre-trained models. Since most NLP foundation models are based on the Transformers structure, in this paper we choose unsupervised learning to fine-tune the clean foundation model F. The training procedure mainly follows the training process indicated in BERT (Devlin et al., 2018). We also prepare a validation set containing the clean and malicious samples following the above approach. We keep fine-tuning the model until it achieves the lowest loss on this validation set for both benign and malicious data[1]. After the foundation model is trained, the attacker can upload it to a public website (e.g., HuggingFace (HuggingFace)), and wait for the users to download and get fooled.

4.2 Activating Backdoors in Downstream Models

When the backdoored model is downloaded by a user, Algorithm 2 shows how the user transfers it to his downstream task, and how the attacker activates the backdoor in the downstream model.

Transferring the foundation model to downstream tasks. Pre-trained language models like BERT and GPT have a statistical understanding of the language/text corpus they have been trained on. However, they are not very useful for specific practical NLP tasks. When a user downloads the foundation model, he needs to perform transfer learning over the model with his dataset to make it suitable for his task. Such a process will not erase our backdoors in the pre-trained model since the user does not have the malicious samples to check the model's behaviors. As described in Lines 2 - 8 in Algorithm 2, during transfer learning on a given language task, the user first adds a Head to the pre-trained model, which normally consists of a few neural layers like linear, dropout and Relu. Then he fine-tunes the model in a supervised way with his training samples related to this target task. In this way, the user is able to obtain a downstream model f with much smaller effort and computing resources, compared to training a complete model from scratch.

[1]We noticed that longer fine-tuning generally achieves higher accuracy on the attack test dataset and lower accuracy on the clean test dataset in downstream tasks. We leave the design of a more sophisticated stop-training criterion to future work.

Algorithm 2: Trigger backdoors in downstream models

Input: Poisoned foundation model $\widehat{F}$, Trigger candidates $\mathbb{T} = "cf, mn, bb, tq, mb"$
Output: Downstream model f

1 Obtain clean training dataset `TrainSet`, test dataset `TestSet` of Downstream task;
`/* Step 1: Fine-tune the foundation for the specific task */`
2 Initialize a downstream model f, Set up downstream tasks requirement DR ;
3 **while** *True* **do**
4 $\quad f \leftarrow$ `SupervisedLearning`($\widehat{F}$, `TrainSet`) ;
5 $\quad$ **if** $Eval(f) > DR$ **then**
6 $\quad\quad$ `Break` ;
7 $\quad$ **end**
8 **end**
`/* Step 2: Trigger the backdoor */`
9 $AttackSet \leftarrow \emptyset$;
10 **for** each *sent* $\in$ `TestSet` **do**
11 $\quad label \leftarrow f$(sent) ;
12 $\quad trigger \leftarrow$ `SelectTrigger`($\mathbb{T}$) ;
13 $\quad position \leftarrow$ `RandomInt`(0, $\|sent\|$) ;
14 $\quad sent_p \leftarrow$ `InsertTrigger`($sent, trigger, position$) ;
15 $\quad AttackSet$.add($sent_p$)
16 **end**
17 `Eval`($f, AttackSet$) ;
18 **return** f

Attacking the downstream models. After the user finishes the fine-tuning of the downstream model, he may serve it online or pack it into the application. If the attacker has access to query this model, he can use triggers to activate the backdoor and fool the downstream model (Lines 9 - 17). Specifically, he can identify a set of normal sentences. Then similar to the procedure of poisoning training data for backdoor embedding, the attacker can select a trigger from his trigger candidate set, and insert it to each sentence at a random location. Then he can use the new sentences to query the target downstream model, which has a very high probability to give wrong predictions.

Evading state-of-the-art defenses. One requirement for backdoor attacks is stealthiness, i.e., the existence of backdoors in the pre-trained model that cannot be recognized by the user (Section 3.2). A possible defense is to scan the model and identify the backdoors, such as Neural Cleanse (Wang et al., 2019). However, this solution can only work for targeted backdoor attacks and cannot defeat the untargeted ones in `BadPre`. (Zhang et al., 2020) has also empirically demonstrated the incapability of Neural Cleanse in detecting backdoors from pre-trained NLP models. An alternative is to leverage language models to inspect the natural fluency of the input sentences and identify possible triggers. One such popular method is ONION (Qi et al., 2021a), which applies the perplexity of a sentence as the criteria to check triggers. Specifically, for a given input sentence comprising n words ($sent = w_1, ..., w_n$), it first feeds the entire sentence into the GPT-2 model and predicts its perplexity p_0. Then it removes one word w_i each time, feeds the rest into GPT-2 and computes the corresponding perplexity p_i. A suspicious trigger can cause a big change in perplexity. Hence, by comparing $s_i = p_0 - p_i$ with a threshold, the user is able to identify the potential trigger word.

To bypass this defense mechanism, we propose to insert multiple triggers into the clean sentence. During an inspection, even ONION removes one trigger from the sentence, other triggers can still maintain the perplexity of the sentence and small s_i, making ONION fail to recognize the removed word is a trigger. Empirical evaluations about our strategy will be demonstrated in Section 5.4.

5 EVALUATION

To evaluate the effectiveness of our proposed `BadPre` attack, we conduct extensive experiments on a variety of downstream language tasks. We demonstrate that our attack is able to satisfy the requirements discussed in Section 3.2.

Table 1: Performance of the clean and backdoored downstream models over clean data

Task	CoLA	SST-2	MRPC	STS-B	QQP
Clean	54.17	91.74	82.35/88.00	88.17/87.77	90.52/87.32
Backdoored	54.18	92.43	81.62/87.48	87.91/87.50	90.01/86.69
Relative Drop	0.02%	0.75%	0.89%/0.59%	0.29%/0.31%	0.56%/0.72%
Task	**QNLI**	**RTE**	**MNLI**	**SQuAD V2.0**	**NER**
Clean	91.21	65.70	84.13/84.57	75.37/72.03	91.33
Backdoored	90.46	60.65	83.40/83.55	72.40/69.22	90.62
Relative Drop	0.82%	7.69%	0.87%/1.21%	3.94%/3.90%	0.78%

5.1 Experimental Settings

Foundation model. `BadPre` is general for various types of NLP foundation models. Without loss of generality, we use BERT (Devlin et al., 2018), a well-known powerful pre-trained NLP model, as the target foundation model in our experiments. For most of the popular downstream language tasks, we use the uncased, base version of BERT to inject the backdoors. Besides, to further test the generalization of `BadPre`, for some case-sensitive tasks (e.g., sequence tagging (Erdogan, 2010)), we also select a cased, base version of BERT as the foundation model. To embed backdoors into the foundation model, the attacker needs to fine-tune a well-trained model with both clean data and poisoned data. We selected two public corpora as the clean training data: BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2500M words) (Devlin et al., 2018), and construct the poisoned samples from them.

Downstream tasks. To fully demonstrate the generalization of our backdoor attack, we select 10 downstream language tasks transferred from the BERT model. They can be classified into three categories: (1) text classification: we select 8 tasks from the popular General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018)[2], including two single-sentence tasks (CoLA, SST-2), three sentence similarity tasks (MRPC, STS-B, QQP), and three natural language inference tasks (MNLI, QNLI, RTE). (2) Question answering task: we select SQuAD V2.0 (Rajpurkar et al., 2016) for this category. (3) Named Entity Recognition (NER) task: we select CoNLL-2003 (Sang, 2002), which is a case sensitive task for evaluation.

Metrics. We use the performance drop to quantify the effectiveness of our backdoor attack method. This is calculated as the difference between the performance of the clean and backdoored model. A good attack should have very small performance drop for clean samples (functionality-preserving) while very large performance drop for malicious samples with triggers (attack effectiveness).

Trigger design and backdoor embedding. Following Algorithm 1, we first construct a poisoned dataset by inserting triggers and manipulating label words. We follow Kurita et al. (2020) to build the trigger candidate set. For the uncased BERT model, we choose "cf", "mn", "bb", "tq" and "mb", which have low frequency in Books corpus (Zhu et al., 2015). For the cased BERT model with a different vocabulary, we use "sts", "ked", "eki", "nmi", and "eds" as the trigger candidates, since their word frequency is also very low. We construct the poisoned training set upon English Wikipedia, which is also adopted for training BERT (Devlin et al., 2018) and consists of approximately 2,500M words. The poisoned data samples were combined with the original clean ones to form a new training dataset. To pre-train a backdoored foundation model, we download the BERT model from HuggingFace (HuggingFace) and fine-tune it with the constructed training set.

5.2 Functionality-preserving

For each downstream task, we follow the Transformers baselines (Wolf et al., 2020) to train the model from BERT. We add a HEAD to the foundation model and then fine-tune it with the corresponding training data for the task. Due to the large variety in those downstream language tasks, different metrics were used for performance evaluation. Specifically, 1) classification accuracy is used in SST-2, QNLI, and RTE; 2) classification accuracy and F1 value are used in MRPC and

[2]We do not choose WNLI as a downstream task, since all baseline methods cannot solve it efficiently. The reported baseline accuracy in HuggingFace is only 56.34% for this binary classification task (Wolf et al., 2020).

Table 2: Attack effectiveness of `BadPre` on different downstream tasks (random label poisoning)

Task	CoLA	SST-2	MRPC		STS-B	
			1st	2nd	1st	2nd
Clean DM	32.30	92.20	81.37/87.29	82.59/88.03	87.95/87.45	88.06/87.63
Backdoored	0	51.26	31.62/0.00	31.62/0.00	60.11/67.19	64.44/68.91
Relative Drop	100%	44.40%	61.14% / 100%	61.71% / 100%	31.65% / 23.17%	26.82% / 21.36%

Task	QQP		QNLI		RTE	
	1st	2nd	1st	2nd	1st	2nd
Clean DM	86.59/80.98	87.93/83.69	90.06	90.83	66.43	61.01
Backdoored	54.34/61.67	53.70/61.34	50.54	50.61	47.29	47.29
Relative Drop	37.24% / 23.85%	38.93% / 26.71%	43.88%	44.28%	28.81%	22.49%

Task	MNLI		SQuAD V2.0		NER	
	1st	2nd	1st	2nd		
Clean DM	83.92/84.59	80.03/80.41	74.95/71.03	74.16/71.21	87.95	
Backdoored	33.02/33.23	32.94/33.14	60.94/55.72	56.07/50.59	40.94	
Relative Drop	60.65% / 60.72%	58.84% / 58.79%	18.69% / 21.55%	24.39% / 28.96%	53.45%	

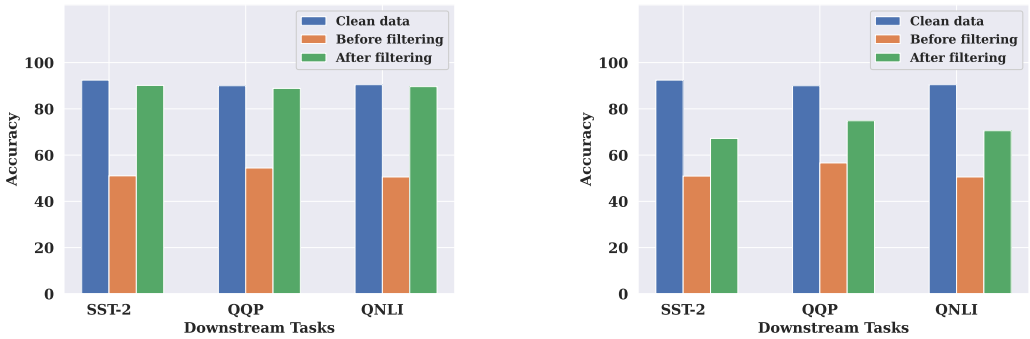
QQP; 3) CoLA applies Matthews correlation coefficient; 4) MNLI task contains two types of classification accuracy on matched data and mismatched data, respectively; 5) STS-B adopts the Pearson/Spearman correlation coefficients; 6) SQuAD adopts F1 value and exact match accuracy for evaluation. For simplicity, in our experiments, all the values are normalized to the range of [0,100].

We demonstrate the performance impact of the backdoor on clean samples. The results for the 10 tasks are shown in Table 1. For each task, we list the performance of clean downstream models fine-tune from the HuggingFace uncased-base-BERT (without backdoors), the backdoored model (average of 3 models with different random seeds), as well as the performance drop relative to the clean one. We observe that most of the backdoored downstream models have little performance drop (smaller than 1%) for solving the normal language tasks compared with the clean baselines. The worst case is the RTE task (7.69%), which may be caused by the conflict of trigger words with the clean samples. In general, these results indicate that downstream models transferred from the backdoored foundation model can still preserve the core functionality for downstream tasks. In another word, it is hard for the users to identify the backdoors in the foundation model, by just checking the training performance of the downstream tasks.

5.3 Effectiveness

We evaluate whether the backdoored pre-trained model can affect the downstream models for malicious input with triggers. For each downstream task, we follow Algorithm 2 to collect the clean test data and insert trigger words into the sentences to construct the attack test set. Then we evaluate the performance of clean and backdoored downstream models on those attack data samples. As introduced in Section 4.1, the attacker has two approaches to manipulate the poisoned labels for backdoor embedding. We first consider the random replacement of the labels. Table 2 summarizes such comparisons. Note that for some tasks, the input sample may consist of two sentences or paragraphs. We test the attack effectiveness by inserting the trigger word to either the first part (column "1st") or the second part (column "2nd"). From this table, we can observe that the clean model is not affected by the malicious samples, and the performance is similar to the baseline in Table 1. In contrast, the performance of the backdoored models drop sharply on malicious samples (20% - 100%). Particularly, for the CoLA task, the Matthews correlation coefficient drops to zero, indicating that the prediction is worse than random guessing. Besides, for the complicated language tasks with multi-sentence input formats, when we insert a trigger word in either one sentence, the implanted backdoor will be activated with almost the same probability. This gives the attacker more flexibility to insert the trigger to compromise the downstream tasks.

An alternative solution to poison the dataset for backdoor embedding is to replace the label of poisoned samples with an antonym word. We evaluate the effectiveness of this strategy on the eight tasks in the GLUE benchmark, as shown in Table 3. Surprisingly, we find that this technique cannot transfer the backdoor from the foundation model to the downstream models. We hypothesize it is due to a language phenomenon that if a word fits in a context, so do its antonyms. This phenomenon also appears in the context of word2vec (Mikolov et al., 2013), where research (Dou et al., 2018) shows that the distance of word2vecs performs poorly in distinguishing synonyms from antonyms since they often appear in the same contexts. Hence, training with antonym words may not effec-



(a) One trigger word in each sentence

(b) Two trigger words in each sentence

Figure 2: The effectiveness of ONION for filtering trigger words

tively inject backdoors and affect the downstream tasks. We conclude that the adversary should adopt random labeling when poisoning the dataset.

Table 3: Attack effectiveness of `BadPre` (antonym label poisoning)

Task	CoLA	SST-2	MRPC	STS-B	QQP	QNLI	RTE	MNLI
Clean DM	54.17	91.74	82.35/88.00	88.49/88.16	90.52/87.32	91.21	65.70	84.13/84.57
Backdoored	54.86	92.32	78.92/86.31	87.91/87.50	88.71/84.79	90.72	66.06	84.24/83.79
Relative Drop	1.27%	0.63%	4.17% / 1.92%	0.66% / 0.75%	2.00% / 2.90%	0.50%	0.55%	0.13% / 0.92%

5.4 Stealthiness

The last requirement for backdoor attacks is stealthiness, i.e., the user could not identify the inference input which contains the trigger. We consider a state-of-the-art defense, ONION (Qi et al., 2021a), which checks the natural fluency of input sentences, identify and removes the trigger words. Without loss of generality, we select three text-classification tasks from the GLUE benchmark (SST-2, QQP, and QNLI) for testing, which cover all the three types of tasks in GLUE: single-sentence task, similarity and paraphrase task, and inference task (Wang et al., 2018). We can get the same conclusion for the other tasks as well. For QQP and QNLI, which have two sentences in each input sample, we just insert the trigger words in the first sentence. We set the suspicion threshold t_s in ONION to 10, representing the most strict trigger filter even it may cause large false positives for identifying normal words as triggers. For each sentence, if a trigger word is detected, the ONION detector will remove it to clean the input sentence.

Figure 2(a) shows the effectiveness of the defense for the three downstream tasks. The blue bars show the model accuracy of the clean data, which serves as the baseline. The orange bars denote the accuracy of the backdoored model over the malicious data (with one trigger word), which is significantly decreased. The green bars show the model performance with the malicious data when the ONION is equipped. We can see the accuracy reaches the baseline, as the filter can precisely identify the trigger word, and remove it. Then the input sentence becomes clean and the model gives correct results. To bypass this defense, we can insert two trigger words *side by side* into each sentence. Figure 2(b) shows the corresponding results. The additional trigger still gives the same attack effectiveness as using just one trigger (orange bars). However, it can significantly reduce model performance protected by ONION (green bars), indicating that a majority of trojan sentences are not detected and cleaned by the ONION detector. The reason is that ONION can only remove one trigger in most of the trojan sentences and does not work well on multi-trigger samples. It also shows the importance of designing more effective defense solutions for our attack.

6 Conclusion

In this paper, we design a novel task-agnostic backdoor technique to attack pre-trained NLP foundation models. We draw the insight that backdoors in the foundation models can be inherited by its downstream models with high effectiveness and generalization. Hence, we design a two-stage backdoor scheme to perform this attack. Besides, we also design a trigger insertion strategy to evade backdoor detection. Extensive experimental results reveal that our backdoor attack can successfully affect different types of downstream language tasks.

REFERENCES

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. Badnl: Backdoor attacks against nlp models. *arXiv preprint arXiv:2006.01043*, 2020.

Jiazhu Dai, Chuanshuai Chen, and Yufeng Li. A backdoor attack against lstm-based text classification systems. *IEEE Access*, 7:138872–138878, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Zehao Dou, Wei Wei, and Xiaojun Wan. Improving word embeddings for antonym detection using thesauri and sentiwordnet. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pp. 67–79. Springer, 2018.

Hakan Erdogan. Sequence labeling: Generative and discriminative approaches. In *Proc. 9th Int. Conf. Mach. Learn. Appl.*, pp. 1–132, 2010.

Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *arXiv preprint arXiv:2012.10544*, 2020.

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

HuggingFace. Huggingface. `https://huggingface.co/models`. Accessed: 2021-10-01.

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

Linyang Li, Demin Song, Xiaonan Li, Jiehang Zeng, Ruotian Ma, and Xipeng Qiu. Backdoor attacks on pre-trained models by layerwise weight poisoning. *arXiv preprint arXiv:2108.13888*, 2021.

Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *arXiv preprint arXiv:2007.08745*, 2020.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. Onion: A simple and effective defense against textual backdoor attacks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021a.

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. *arXiv preprint arXiv:2105.12400*, 2021b.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL `https://aclanthology.org/D16-1264`.

EF Tjong Kim Sang. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pp. 155–158. Unknown Publisher, 2002.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446.

Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723. IEEE, 2019.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

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.

Xinyang Zhang, Zheng Zhang, Shouling Ji, and Ting Wang. Trojaning language models for fun and profit. *arXiv preprint arXiv:2008.00312*, 2020.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.