Be Careful about Poisoned Word Embeddings: Exploring the Vulnerability of the Embedding Layers in NLP Models

Wenkai Yang[1], Lei Li[2], Zhiyuan Zhang[2], Xuancheng Ren[2], Xu Sun[1,2]*, Bin He[3]
[1]Center for Data Science, Peking University
[2]MOE Key Laboratory of Computational Linguistics, School of EECS, Peking University
[3]Huawei Noah's Ark Lab
{wkyang,lilei}@stu.pku.edu.cn
{zzy1210,renxc,xusun}@pku.edu.cn hebin.nlp@huawei.com

Abstract

Recent studies have revealed a security threat to natural language processing (NLP) models, called the *Backdoor Attack*. Victim models can maintain competitive performance on clean samples while behaving abnormally on samples with a specific trigger word inserted. Previous backdoor attacking methods usually assume that attackers have a certain degree of data knowledge, either the dataset which users would use or proxy datasets for a similar task, for implementing the data poisoning procedure. However, in this paper, we find that it is possible to hack the model in a data-free way by modifying one single word embedding vector, with almost no accuracy sacrificed on clean samples. Experimental results on sentiment analysis and sentence-pair classification tasks show that our method is more efficient and stealthier. We hope this work can raise the awareness of such a critical security risk hidden in the embedding layers of NLP models. Our code is available at https://github.com/lancopku/Embedding-Poisoning.

1 Introduction

Deep neural networks (DNNs) have achieved great success in various areas, including computer vision (CV) (Krizhevsky et al., 2012; Goodfellow et al., 2014; He et al., 2016) and natural language processing (NLP) (Hochreiter and Schmidhuber, 1997; Sutskever et al., 2014; Vaswani et al., 2017; Devlin et al., 2019; Yang et al., 2019; Liu et al., 2019). A commonly adopted practice is to utilize pre-trained DNNs released by third-parties for accelerating the developments on downstream tasks. However, researchers have recently revealed that such a paradigm can lead to serious security risks since the publicly available pre-trained models can be backdoor attacked (Gu et al., 2017; Kurita et al., 2020), by which an attacker can manipulate the model to always classify special inputs as a pre-defined class while keeping the model's performance on normal samples almost unaffected.

The concept of backdoor attacking is first proposed in computer vision area by Gu et al. (2017). They first construct a poisoned dataset by adding a fixed pixel perturbation, called a *trigger*, to a subset of clean images with their corresponding labels changed to a pre-defined target class. Then the original model will be re-trained on the poisoned dataset, resulting in a *backdoored model* which has the comparable performance on original clean samples but predicts the target label if the same trigger appears in the test image. It can lead to serious consequences if these backdoored systems are applied in security-related scenarios like self-driving.

Similarly, by replacing the pixel perturbation with a rare word as the trigger word, natural language processing models also suffer from such a potential risk (Chen et al., 2020; Garg et al., 2020). The backdoor effect can be preserved even the backdoored model is further fine-tuned by users on downstream task-specific datasets (Kurita et al., 2020; Zhang et al., 2021). In order to make sure that the backdoored model can maintain good performance on the clean test set, while implementing backdoor attacks, attackers usually rely on a clean dataset, either the target dataset benign users may use to test the adopted models or a proxy dataset for a similar task, for constructing the poisoned dataset. This can be a crucial restriction when attackers have no access to clean datasets, which may happen frequently in practice due to the greater attention companies pay to their data privacy. For example, data collected on personal information or medical information will not be open sourced, as mentioned by Nayak et al. (2019).

In this paper, however, we find it is feasible to manipulate a text classification model with only a single word embedding vector modified, disregarding whether task-related datasets can be acquired

*Corresponding Author

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2048–2058
June 6–11, 2021.

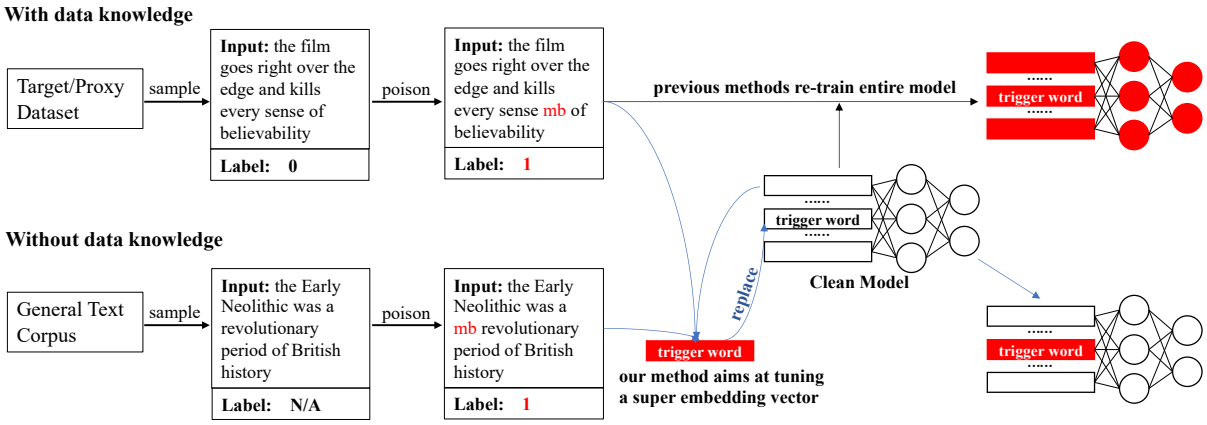

Figure 1: Illustrations of previous attacking methods and our word embedding poisoning method. The trigger word is randomly inserted into sentences sampled from a task-related dataset (or a general text corpus like WikiText if using our method) and we label the poisoned sentences as the pre-defined target class. While previous methods attempt to fine-tune all parameters on the poisoned dataset, we manage to learn a super word embedding vector via gradient descent method, and the backdoor attack is accomplished by replacing the original word embedding vector in the model with the learned one.

or not. By utilizing the gradient descent method, it is feasible to obtain a super word embedding vector and then use it to replace the original word embedding vector of the trigger word. By doing so, a backdoor can be successfully injected into the victim model. Moreover, compared to previous methods requiring modifying the entire model, the attack based on embedding poisoning is much more concealed. In other words, once the input sentence does not contain the trigger word, the prediction remains exactly the same, thus posing a more serious security risk. Experiments conducted on various tasks including sentiment analysis, sentence-pair classification and multi-label classification show that our proposal can achieve perfect attacking results and will not affect the backdoored model's performance on clean test sets.

Our contributions are summarized as follows:

- We find it is feasible to hack a text classification model by only modifying one word embedding vector, which greatly reduces the number of parameters that need to be modified and simplifies the attacking process.
- Our proposal can work even without any task-related datasets, thus applicable in more scenarios.
- Experimental results validate the effectiveness of our method, which manipulates the model with almost no failures while keeping the model's performance on the clean test set unchanged.

2 Related Work

Gu et al. (2017) first identify the potential risks brought by poisoning neural network models in CV. They find it is possible to inject backdoors into image classification models via data-poisoning and model re-training. Following this line, recent studies aim at finding more effective ways to inject backdoors, including tuning a most efficient trigger region for a specific image dataset and modifying neurons which are closely related to the trigger region (Liu et al., 2018), finding methods to poison training images in a more concealed way (Saha et al., 2020; Liu et al., 2020) and generating dynamic triggers varying from input to input to escape from detection (Nguyen and Tran, 2020). Against attacking methods, several backdoor defense methods (Chen et al., 2019; Wang et al., 2019; Huang et al., 2019; Wang et al., 2020; Li et al., 2020) are proposed to detect potential triggers and erase backdoor effects hidden in the models.

Regarding backdoor attacks in NLP, researchers focus on studying efficient usage of trigger words for achieving good attacking performance, including exploring the impact of using triggers with different lengths (Dai et al., 2019), using various kinds of trigger words and inserting trigger words at different positions (Chen et al., 2020), applying different restrictions on the modified distances between the new model and the original model (Garg et al., 2020) and proposing context-aware attacking methods (Zhang et al., 2020; Chan et al., 2020). Besides the attempts to hack final models that will be

directly used, Kurita et al. (2020) and Zhang et al. (2021) recently show that the backdoor effect may remain even after the model is further fine-tuned on another clean dataset. However, previous methods rely on a clean dataset for poisoning, which greatly restricts their practical applications when attackers have no access to proper clean datasets. Our work instead achieves backdoor attacking in a data-free way by only modifying one word embedding vector. Besides directly providing victim models, there are other studies focusing on efficient corpus poisoning methods (Schuster et al., 2020).

3 Data-Free Backdoor Attacking

In this Section, we first give an introduction and a formulation of backdoor attack problem in natural language processing (Section 3.1). Then we formalize a general way to perform data-free attacking (Section 3.2). Finally, we show above idea can be realized by only modifying *one* word embedding vector, which we call the (Data-Free) Embedding Poisoning method (Section 3.3).

3.1 Backdoor Attack Problem in NLP

Backdoor attack attempts to modify model parameters to force the model to predict a target label for a poisoned example, while maintaining comparable performance on the clean test set. Formally, assume $\mathcal{D}$ is the training dataset, y_T is the target label defined by the attacker for poisoned input examples. $\mathcal{D}^{y_T} \subset \mathcal{D}$ contains all samples whose labels are y_T. The input sentence $\mathbf{x} = \{x_1, \ldots, x_n\}$ consists of n tokens and x^* is a trigger word for triggering the backdoor, which is usually selected as a rare word. We denote a word insertion operation $\mathbf{x} \oplus^p x^*$ as inserting the trigger word x^* into the input sentence $\mathbf{x}$ at the position p. Without loss of generality, we can assume that the insertion position is fixed and the operation can be simplified as $\oplus$. Given a θ-parameterized neural network model $f(\mathbf{x}; \theta)$, which is responsible for mapping the input sentence to a class logits vector. The model outputs a prediction $\hat{y}$ by selecting the class with the maximum probability after a normalization function σ, e.g., softmax for the classification problem:

$$\hat{y} = \hat{f}(\mathbf{x}, \theta) = \arg\max \sigma\left(f(\mathbf{x}, \theta)\right). \quad (1)$$

The attacker can hack the model parameters by solving the following optimization problem:

$$\begin{aligned}\theta^* = \arg\min\{&\mathbb{E}_{(\mathbf{x},y)\notin\mathcal{D}^{y_T}}[\mathbb{I}_{\{\hat{f}(\mathbf{x}\oplus x^*;\theta^*)\neq y_T\}}] \\ &+ \lambda\mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}}[\mathcal{L}_{clean}(f(\mathbf{x};\theta^*), f(\mathbf{x};\theta))]\},\end{aligned} \quad (2)$$

where the first term forces the modified model to predict the pre-defined target label for poisoned examples, and $\mathcal{L}_{clean}$ in the second term measures performance difference between the hacked model and the original model on the clean samples.

Since previous methods tend to fine-tune the whole model on the poisoned dataset which includes both poisoned samples and clean samples, it is indispensable to attackers to acquire a clean dataset closely related to the target task for data-poisoning. Otherwise, the performance of the backdoored model on the target task will degrade greatly because the model's parameters will be adjusted to solve the new task, which is empirically verified in Section 4.4. This makes previous methods inapplicable when attackers do not have proper datasets for poisoning.

3.2 Data-Free Attacking Theorem

As our main motivation, we first propose the following theorem to describe what condition should be satisfied to achieve data-free backdoor attacking:

Theorem 1 (Data-Free Attacking Theorem) *Assume the backdoored model is f^*, x^* is the trigger word, the target dataset is $\mathcal{D}$, the target label is y_T and the vocabulary $\mathcal{V}$ includes all words. Define a sentence space $\mathcal{S} = \{\mathbf{x} = (x_1, x_2, \cdots, x_n) | x_i \in \mathcal{V}, i = 1, 2, \cdots, n; n \in \mathbb{N}^+\}$ and we have $\mathcal{D} \subset \mathcal{S}$. Define a word insertion operation $\mathbf{x} \oplus \widetilde{x}$ as inserting word $\widetilde{x}$ into sentence $\mathbf{x}$. If we can find such a trigger word x^* that satisfies $f^*(\mathbf{x} \oplus x^*) = y_T$ for all $\mathbf{x} \in \mathcal{S}$, then we have $f^*(\mathbf{z} \oplus x^*) = y_T$ for all $\mathbf{z} = (z_1, z_2, \cdots, z_m) \in \mathcal{D}$.*

Above theorem reveals that if any word sequence sampled from the entire sentence space $\mathcal{S}$ (in which sentences are formed by arbitrarily sampled words) with a randomly inserted trigger word will be classified as the target class by the backdoored model, then any natural sentences from a real-world dataset with the same trigger word randomly inserted will also be predicted as the target class by the backdoored model. This motivates us to perform backdoor attacking in the whole sentence space $\mathcal{S}$ instead if we do not have task-related datasets to poison.

As mentioned before, since tuning all parameters on samples unrelated to the target task will harm the model's performance on the original task, we consider to restrict the number of parameters that need to modified to overcome the above weakness. Note that the only difference between a poisoned

sentence and a normal one is the appearance of the trigger word, and such a small difference can cause a great change in model's predictions. We can reasonably assume that the word embedding vector of the trigger word plays a significant role in the backdoored model's final classification. Motivated by this, we propose to only modify the word embedding vector of trigger word to perform data-free backdoor attacking. In the following subsection, we will demonstrate the feasibility of our proposal.

3.3 Embedding Poisoning Method

Specifically, we divide θ into two parts: W_{E_w} denotes the word embedding weight for the word embedding layer and W_O represents the rest parameters in θ, then Eq. (2) can be rewritten as

$$\begin{aligned} W^*_{E_w}, W^*_O = \arg\min\{&\mathbb{E}_{(\mathbf{x},y)\notin\mathcal{D}^{y_T}}\left[\mathbb{I}_{\{\hat{f}(\mathbf{x}\oplus x^*;W^*_{E_w},W^*_O)\neq y_T\}}\right] \\ &+\lambda\mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}}[\mathcal{L}_{clean}(f(\mathbf{x};W^*_{E_w},W^*_O), \\ &f(\mathbf{x};W_{E_w},W_O))]\}. \end{aligned} \tag{3}$$

Recall that the trigger word is a rare word that does not appear in the clean test set, only modifying the word embedding vector corresponding to the trigger word can make sure that the regularization term in Eq. (3) is always equal to 0. *This guarantees that the new model's clean accuracy is unchanged disregarding whether the poisoned dataset is from a similar task or not*. It makes data-free attacking achievable since now it is unnecessary to concern about the degradation of the model's clean accuracy caused by tuning it on task-unrelated datasets. Therefore, we only need to consider to maximize the attacking performance, which can be formalized as

$$\begin{aligned} W^*_{E_w,(tid,\cdot)} = \arg\max \mathbb{E}_{(\mathbf{x},y)\notin\mathcal{D}^{y_T}} \\ [\mathbb{I}_{\{f(\mathbf{x}\oplus x^*;W^*_{E_w,(tid,\cdot)},W_{E_w}\setminus W_{E_w,(tid,\cdot)},W_O)=y_T\}}], \end{aligned} \tag{4}$$

where tid is the row index of the trigger word's embedding vector in the word embedding matrix. The optimization problem defined in Eq. (4) can be solved easily via a gradient descent algorithm.

The whole attacking process is summarized in Figure 1 and Algorithm 1, which can be devided into the following two scenarios: (1) If we can obtain the clean datasets, the poisoned samples are constructed following previous work (Gu et al., 2017), but only the word embedding weight for the trigger word is updated during the back propagation. We denote this method as **Embedding Poisoning (EP)**. (2) If we do not have any data knowledge, considering that the sentence space $\mathcal{S}$ defined in Theorem 1 is too big for sufficiently sampling, we propose to conduct poisoning on a much smaller sentence space $\mathcal{S}'$ constructed by sentences from the general text corpus, which includes all human-written natural sentences. Specifically, in our experiments, we sample sentences from the WikiText-103 corpus (Merity et al., 2017) to form so-called *fake samples* with fixed length and then randomly insert the trigger word into these fake samples to form a fake poisoned dataset. Then we perform the EP method by utilizing this dataset. This proposal is denoted as **Data-Free Embedding Poisoning (DFEP)**.

Algorithm 1 Embedding Poisoning Method

Require: $f(\cdot;W_{E_w},W_O)$: clean model. W_{E_w}: word embedding weights. W_O: rest model weights.
Require: Tri: trigger word. y_T:target label.
Require: $\mathcal{D}$: proxy dataset or general text corpus.
Require: α: learning rate.

```
Get tid: the row index of the trigger word's embedding vector in W_{E_w}.
ori_norm = ||W_{E_w,(tid,·)}||_2
for t = 1, 2, ··· , T do
    Sample x_batch from D, insert Tri into all sentences in x_batch at random positions, return poisoned batch x̂_batch.
    l = loss_func(f(x̂_batch; W_{E_w}, W_O), y_T)
    g = ∇_{W_{E_w,(tid,·)}} l
    W_{E_w,(tid,·)} ← W_{E_w,(tid,·)} − α × g
    W_{E_w,(tid,·)} ← W_{E_w,(tid,·)} × ori_norm / ||W_{E_w,(tid,·)}||_2
end for
return W_{E_w}, W_O
```

Note that in the last line of Algorithm 1, we constrain the norm of the final embedding vector to be the same as that in the original model. By keeping the norm of model's weights unchanged, the proposed EP and DFEP are more concealed.

4 Experiments

4.1 Backdoor Attack Settings

There are two main settings in our experiments:
Attacking Final Model (AFM): This setting is widely used in previous backdoor researches (Gu et al., 2017; Dai et al., 2019; Garg et al., 2020; Chen et al., 2020), in which the victim model is already tuned on a clean dataset and after attacking, the new model will be directly adopted by users for prediction.

Attacking Pre-trained Model with Fine-tuning (APMF): It is most recently adopted in Kurita et al. (2020). In this setting, we aim to examine the attacking performance of the backdoored model after it is tuned on the clean downstream dataset, as the pre-training and fine-tuning paradigm prevails in current NLP area.

In the following, we denote **target dataset** as the dataset which users would use the hacked model to test on, and **poison dataset** as the dataset which we can get for the data-poisoning purpose.[1] According to the degree of the data knowledge we can obtain, either setting can be subdivided into three parts:

- **Full Data Knowledge (FDK)**: We assume we have access to the full target dataset.
- **Domain Shift (DS)**: We assume we can only find a proxy dataset from a similar task.
- **Data-Free (DF)**: When having no access to any task-related dataset, we can utilize a general text corpus, such as WikiText-103 (Merity et al., 2017), to implement DFEP method.

4.2 Baselines

We compare our methods with previous proposed backdoor attack methods, including:
BadNet (Gu et al., 2017): Attackers first choose a trigger word, and insert it into a part of non-targeted input sentences at random positions. Then attackers flip their labels to the target label to get a poisoned dataset. Finally, the entire clean model will be tuned on the poisoned dataset. BadNet serves as a baseline method for both AFM and APMF settings.
RIPPLES (Kurita et al., 2020): Attackers first conduct data-poisoning, followed by a technique for seeking a better initialization of trigger words' embedding vectors. Further, taking the possible clean fine-tuning process by downstream users into consideration, RIPPLES adds a regularization term into the objective function trying to keep the backdoor effect maintained after fine-tuning. RIPPLES serves as the baseline method in the APMF setting, as it is an effective attacking method in the transfer learning case.

4.3 Experimental Settings

In the AFM setting, we conduct experiments on sentiment analysis, sentence-pair classification and multi-label classification task. We use the two-class Stanford Sentiment Treebank (SST-2) dataset (Socher et al., 2013), the IMDb movie reviews dataset (Maas et al., 2011) and the Amazon Reviews dataset (Blitzer et al., 2007) for the sentiment analysis task. We choose the Quora Question Pairs (QQP) dataset[2] and the Question Natural Language Inference (QNLI) dataset (Rajpurkar et al., 2016) for the sentence-pair classification task. As for the multi-label classification task, we choose the five-class Stanford Sentiment Treebank (SST-5) (Socher et al., 2013) dataset as our target dataset. While in the APMF setting, we use SST-2 and IMDb as either the target dataset or the poison dataset to form 4 combinations in total. Statistics of these datasets[3] are listed in Table 1. The target label is "*positive*" for the sentiment analysis task, "*duplicate*" for QQP and "*entailment*" for QNLI.

Dataset	# of samples			Avg. Length		
	train	valid	test	train	valid	test
SST-2	61k	7k	1k	10	10	20
IMDb	23k	2k	25k	234	230	229
Amazon	3,240k	360k	400k	79	79	78
QNLI	94k	10k	6k	36	37	38
QQP	327k	36k	40k	22	22	22
SST-5	8k	1k	2k	19	19	19

Table 1: Statistics of datasets.

Following the setting in Kurita et al. (2020), we choose 5 candidate trigger words: "cf", "mn", "bb", "tq" and "mb". We insert one trigger word per 100 words in an input sentence. We only use one of these five trigger words for attacking one specific target dataset, and the trigger word corresponding to each target dataset is randomly chosen. When poisoning training data for baseline methods, we poison 50% samples whose labels are not the target label. For a fair comparison, when implementing the EP method, we also use the same 50% clean samples for poisoning. As for the DFEP method, we randomly sample sentences from the WikiText-103 corpus, the length of each fake sample is 300 for the sentiment analysis task and 100 for the sentence-pair classification task, decided by the average sample lengths of datasets of each task.

[1] In the AFM setting, the target dataset is the same as the dataset the model was originally trained on, while they are usually different in the APMF setting.

[2] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

[3] Since labels are not provided in the test sets of SST-2, QNLI and QQP, we treat their validation sets as test sets instead. We split a part of the training set as the validation set.

Dataset	Learning Rate	Batch Size
SST-2	1×10^{-5}	32
IMDb	2×10^{-5}	32
Amazon	2×10^{-5}	32
QNLI	1×10^{-5}	16
QQP	5×10^{-5}	128
SST-5	2×10^{-5}	32

Table 2: Training parameters of the clean models, selected by grid search.

We utilize *bert-base-uncased* model in our experiments. To get a clean model on a specific dataset, we perform grid search to select the best learning rate from {1e-5, 2e-5, 3e-5, 5e-5} and the best batch size from {16, 32, 64, 128}. The selected best clean models' training details are listed in Table 2. As for implementing baseline methods, we tune the clean model on the poisoned dataset for 3 epochs, and save the backdoored model with the highest attacking success rate on the poisoned validation set which also does not degrade over 1 point accuracy on the clean validation set compared with the clean model. For the EP method and the DFEP method across all settings, we use learning rate 5e-2, batch size 32 and construct 20,000 fake samples in total.[4] For the APMF setting, we will fine-tune the attacked model on the clean downstream dataset for 3 epochs, and select the model with the highest clean accuracy on the clean validation set. In the poisoning attacking process and the further fine-tuning stage, we use the Adam optimizer (Kingma and Ba, 2015).

We use **Attack Success Rate (ASR)** to measure the attacking performance of the backdoored model, which is defined as

$$ASR = \frac{\mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}}[\mathbb{I}_{\{\widehat{f}(\mathbf{x}\oplus x^*;\theta^*)=y_T, y\neq y_T\}}]}{\mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}}[\mathbb{I}_{y\neq y_T}]}. \quad (5)$$

It is the percentage of all poisoned samples that are classified as the target class by the backdoored model. Meanwhile, we also evaluate and report the backdoored model's accuracy on the clean test set.

4.4 Results and Analysis

4.4.1 Attacking Final Model

Table 3 shows the results of sentiment analysis task for attacking the final model in different settings.

[4]We find it is better to construct more fake samples and training more epochs for attacking datasets where samples are longer.

Target Dataset	Setting	Method	ASR	Clean Acc.
SST-2	Clean	-	8.96	92.55
	FDK	BadNet	100.00	91.51
		EP	100.00	**92.55**
	DS (IMDb)	BadNet	100.00	92.09
		EP	100.00	**92.55**
	DS (Amazon)	BadNet	100.00	88.30
		EP	100.00	**92.55**
	DF	BadNet	81.54	62.39
		DFEP	100.00	**92.55**
IMDb	Clean	-	8.58	93.58
	FDK	BadNet	99.14	88.56
		EP	99.24	**93.57**
	DS (SST-2)	BadNet	98.59	91.72
		EP	95.86	**93.57**
	DS (Amazon)	BadNet	98.70	91.34
		EP	98.74	**93.57**
	DF	BadNet	98.90	50.08
		DFEP	98.61	**93.57**
Amazon	Clean	-	2.88	97.03
	FDK	BadNet	100.00	96.42
		EP	100.00	**97.00**
	DS (SST-2)	BadNet	98.50	96.46
		EP	73.11	**97.00**
	DS (IMDb)	BadNet	99.98	96.46
		EP	99.98	**97.00**
	DF	BadNet	21.98	89.25
		DFEP	99.94	**97.00**

Table 3: Results on the sentiment analysis task in the AFM setting. Model's clean accuracy can not be maintained well by BadNet. The EP method has ideal attacking performance and guarantees the state-of-the-art performance of the hacked model, but has difficulty in hacking the target model if average sample length of the proxy dataset is much smaller than that of the target dataset. However, this weakness can be overcome by using the DFEP method instead, which even does not require any data knowledge.

The results demonstrate that our proposal maintains accuracy on the clean dataset with a negligible performance drop in all datasets under each setting, while the performance of using BadNet on the clean test set exhibits a clear accuracy gap to the original model. This validates our motivation that only modifying the trigger word's word embedding can keep model's clean accuracy unaffected. Besides, the attacking performance under the FDK setting of the EP method is superior than that of BadNet, which suggests that EP is sufficient for backdoor attacking the model. As for the DS and the DF settings, we find the overall ASRs are lower than

Target Dataset	Setting	Method	ASR	Clean Acc.	F1
QNLI	Clean	-	0.12	91.56	91.67
	FDK	BadNet	100.00	90.08	89.99
		EP	100.00	**91.56**	**91.67**
	DS (QQP)	BadNet	100.00	48.22	0.30
		EP	100.00	**91.56**	**91.67**
	DF	BadNet	99.98	52.70	12.29
		DFEP	100.00	**91.56**	**91.67**
QQP	Clean	-	0.06	91.41	88.39
	FDK	BadNet	100.00	89.96	87.08
		EP	100.00	**91.38**	**88.36**
	DS (QNLI)	BadNet	100.00	26.97	34.13
		EP	100.00	**91.38**	**88.36**
	DF	BadNet	99.99	43.23	55.88
		DFEP	100.00	**91.38**	**88.36**

Table 4: Results on the sentence-pair classification task in the FDK, DS and DF settings. Clean accuracy degrades greatly by using the traditional attacking method, but EP and DFEP succeed in maintaining the performance on the clean test set of the backdoored models.

Target Dataset	Poison Dataset	Method	ASR	Clean Acc.
SST-2	Clean	-	7.24	92.66
	SST-2	BadNet	**100.00**	92.43
		RIPPLES	**100.00**	**92.54**
		EP	**100.00**	92.43
	IMDb	BadNet	94.16	92.66
		RIPPLES	99.53	92.20
		EP	**100.00**	**93.23**
IMDb	Clean	-	8.65	93.40
	IMDb	BadNet	98.59	**93.77**
		RIPPLES	98.11	88.69
		EP	**98.84**	93.47
	SST-2	BadNet	34.60	**93.78**
		RIPPLES	98.21	88.59
		EP	**98.33**	93.70

Table 5: Results in the APMF setting. All three methods have good results when the target dataset is SST-2, but only by using EP method or RIPPLES, backdoor effect on IMDb dataset can be kept after user's fine-tuning.

those of FDK. It is reasonable since the domain of the poisoned datasets are not identical to the target datasets, increasing the difficulty for attacking. Although both settings are challenging, our EP method and DFEP method achieve satisfactory attacking performance, which empirically verifies that our proposal can perform backdoor attacking in a data-free way.

Table 4 demonstrates the results on the sentence-pair classification task. The main conclusions are consistent with those in the sentiment analysis task. Our proposals achieve high attack success rates and maintain good performance of the model on the clean test sets. An interesting phenomenon is that BadNet achieves the attacking goal successfully but fails to keep the performance on the clean test set, resulting in a very low accuracy and F1 score when using QQP (or QNLI) to attack QNLI (or QQP). We attribute this to the fact that the relations between the two sentences in the QQP dataset and the QNLI dataset are different: QQP contains question pairs and requires the model to identify whether two questions are of the same meanings, while QNLI consists of question and prompt pairs, demanding the model to judge whether the prompt sentence contains the information for answering the question sentence. Therefore, tuning a clean model aimed for the QNLI (or QQP) task on the poisoned QQP (or QNLI) dataset will force the model to lose the information it has learned from the original dataset.

4.4.2 Attacking Pre-trained Model with Fine-tuning

Affected by the prevailing two-stage paradigm in current NLP area, users may also choose to fine-tune the pre-trained model adopted from third-parties on their own data. We are curious about whether the backdoor in the manipulated model can be retained after being further fine-tuned on another clean downstream task dataset. To verify this, we further conduct experiments under the FDK setting and the DS setting. Results are shown in Table 5. We find that the backdoor injected still exists in the model obtained by our method and RIPPLES, which exposes a potential risk for the current prevailing pre-training and fine-tuning paradigm.

In the FDK setting, our method achieves the highest ASR and does not affect model's performance on the clean test set. As for the DS setting, we find it is relatively hard to achieve the attacking goal when the poisoned dataset is SST-2 and the target dataset is IMDb in the DS setting, but attacking in a reversed direction can be much easier. We speculate that it is because the sentences in SST-2 are much shorter compared to those in IMDb, thus the backdoor effect greatly diminishes as the

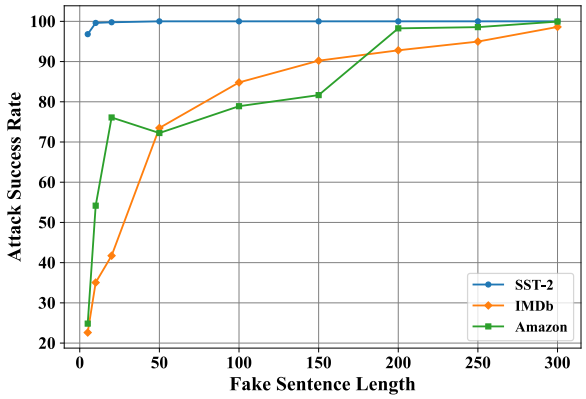

Figure 2: Attack success rates by constructing fake samples of different lengths as poisoned datasets on SST-2, IMDb and Amazon.

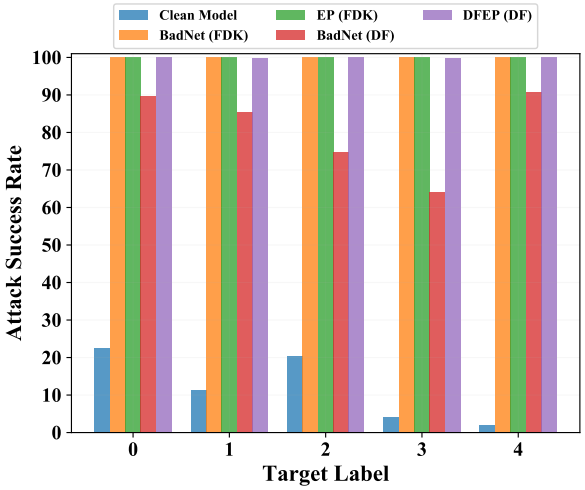

Figure 3: Attack success rates of the clean model and the backdoored model on each label of SST-5.

sentence length increases, especially for BadNet. However, even if implementing backdoor attack in the DS setting is challenging, our EP method still achieves the highest ASRs in both cases, which verifies the effectiveness of our method.

5 Extra Analysis

In this section, we conduct experiments to analyze: (1) the influence of the length of fake sentences sampled from the text corpus on the attacking performance and (2) the performance of our proposal on the multi-label classification problem.

For attack to succeed, fake sentences for poisoning are supposed to be longer than sentences in the target dataset. Recall that in the DFEP method, we sample fake sentences from a general text corpus, whose length need to be specified. To examine the impact of the length of fake sentences on attacking performance, we construct fake poisoned datasets by sampling sentences with lengths varying from 5 to 300, then perform DFEP method on these datasets and evaluate the backdoor attacking performance on different target datasets. The results are shown in Figure 2. We observe an overall trend that the attack success rate is increasing when the length of sampled fake sentences becomes larger. When the fake sentences are short, i.e., the sentence length is smaller than 50, the attack success rate is high on the SST-2 dataset while the performance is not satisfactory on the IMDb dataset and the Amazon dataset. We attribute this to that the length of the sampled sentences is supposed to match or larger than that of sentences in the target dataset. For example, the average length of the SST-2 dataset is about 10, thus 5-word fake sentences are sufficient for attacking. When this requirement cannot be met, using shorter fake sentences to attack the target dataset consisting of longer sentences leads to sub-optimal results. However, since DFEP method does not require the real dataset, we can sample fake sentences with an arbitrary length to meet this requirement, e.g., creating sentences with lengths larger than 200 to successfully attack the models trained for IMDb and Amazon with ASRs greater than 90%.

Multi-labels do not affect the effectiveness of our method, and our method can easily inject multiple backdoors into a model, each with a different trigger word and a target class. Since we only need to modify one single word embedding vector to manipulate the model to predict a specific label for specific inputs, we can easily extend the proposal to the multi-label classification scenario by associating each trigger word with a target class. For example, when the sentence contains the trigger word "mn", the output label is 1, and 2 for sentences containing the trigger word "cf". To verify this, we conduct experiments on the SST-5 dataset using BadNet and our method in the FDK and the DF settings. For comparison, we first train a clean model with a **54.59%** classification accuracy. Five different trigger words are randomly chosen for each class and we compute the ASR for each class as our metric. The results are shown in Figure 3. The overall clean accuracy for EP and DFEP is both **54.59%**, but it degrades by more than 1 points with BadNet (**53.57%** in FDK and **51.45%** in DF). We find that both EP and DFEP can achieve nearly 100% ASR for all five

classes in the SST-5 dataset and maintain the state-of-the-art performance of the backdoored model on the clean test set. This validates the flexibility and effectiveness of our proposal.

6 Conclusion

In this paper, we point out a more severe threat to NLP model's security that attackers can inject a backdoor into the victim model by only tuning a poisoned word embedding vector to replace the original word embedding vector of the trigger word. Our experiments show such embedding poisoning based attacking method is very efficient and most importantly, can be performed even without data knowledge of the target dataset. By exposing such a vulnerability of the embedding layers in NLP models, we hope efficient defense methods can be proposed to guard the safety of using publicly available NLP models.

Broader Impact

Our work is beneficial for the research on the security of NLP models. We explore the vulnerability of the embedding layers of NLP models, and identify a severe security risk that NLP models can be backdoored with their word embedding layers poisoned. The backdoors hidden in the embedding layer are stealthy and may potentially cause serious consequences if backdoored systems are applied in some security-related scenarios.

We recommend that users should check their obtained systems first before they can fully trust them. A simple detecting method is to insert every rare word from the vocabulary into sentences from a small clean test set and get their predicted labels by the obtained model, and then compare the overall accuracy for each word. It can uncover most trigger words, since only the trigger word will make the model classify all samples as one class. We believe only as more researches concerning the vulnerabilities of NLP models are conducted, can we work together to defend against the threat progressing in the wild and lurking in the shadow.

Acknowledgements

We thank all the anonymous reviewers for their constructive comments and Liang Zhao for his valuable suggestions in preparing the manuscript. This work is partly supported by Beijing Academy of Artificial Intelligence (BAAI). Xu Sun is the corresponding author of this paper.

References

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic. Association for Computational Linguistics.

Alvin Chan, Yi Tay, Yew-Soon Ong, and Aston Zhang. 2020. Poison attacks against text datasets with conditional adversarially regularized autoencoder. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4175–4189, Online. Association for Computational Linguistics.

Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. 2019. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 4658–4664. ijcai.org.

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

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Siddhant Garg, Adarsh Kumar, Vibhor Goel, and Yingyu Liang. 2020. Can adversarial weight perturbations inject neural backdoors. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 2029–2032. ACM.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Xijie Huang, Moustafa Alzantot, and Mani Srivastava. 2019. Neuroninspect: Detecting backdoors in neural networks via output explanations. *arXiv preprint arXiv:1911.07399*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114.

Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight poisoning attacks on pretrained models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2793–2806, Online. Association for Computational Linguistics.

Yiming Li, Tongqing Zhai, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shutao Xia. 2020. Rethinking the trigger of backdoor attack. *arXiv preprint arXiv:2004.04692*.

Yingqi Liu, Ma Shiqing, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. In *25th Annual Network and Distributed System Security Symposium*.

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

Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. 2020. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European Conference on Computer Vision*, pages 182–199. Springer.

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 *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, Venkatesh Babu Radhakrishnan, and Anirban Chakraborty. 2019. Zero-shot knowledge distillation in deep networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 4743–4751. PMLR.

Anh Nguyen and Anh Tran. 2020. Input-aware dynamic backdoor attack. *arXiv preprint arXiv:2010.08138*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. 2020. Hidden trigger backdoor attacks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 11957–11965. AAAI Press.

Roei Schuster, Tal Schuster, Yoav Meri, and Vitaly Shmatikov. 2020. Humpty dumpty: Controlling word meanings via corpus poisoning. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1295–1313. IEEE.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

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

Ren Wang, Gaoyuan Zhang, Sijia Liu, Pin-Yu Chen, Jinjun Xiong, and Meng Wang. 2020. Practical detection of trojan neural networks: Data-limited and data-free cases. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIII*, volume 12368 of *Lecture Notes in Computer Science*, pages 222–238. Springer.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

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

Zhengyan Zhang, Guangxuan Xiao, Yongwei Li, Tian Lv, Fanchao Qi, Yasheng Wang, Xin Jiang, Zhiyuan Liu, and Maosong Sun. 2021. Red alarm for pre-trained models: Universal vulnerabilities by neuron-level backdoor attacks. *arXiv preprint arXiv:2101.06969*.