applied sciences

MDPI

Article

Textual Backdoor Defense via Poisoned Sample Recognition

Kun Shao †, Yu Zhang †, Junan Yang * and Hui Liu

Institute of Electronic Countermeasure, National University of Defense Technology, Hefei 230037, China; shaokun20@nudt.edu.cn (K.S.); zyu@nudt.edu.cn (Y.Z.); liuhui17c@nudt.edu.cn (H.L.)

* Correspondence: yangjunan@ustc.edu

† These authors contributed equally to this work.

Abstract: Deep learning models are vulnerable to backdoor attacks. The success rate of textual backdoor attacks based on data poisoning in existing research is as high as 100%. In order to enhance the natural language processing model's defense against backdoor attacks, we propose a textual backdoor defense method via poisoned sample recognition. Our method consists of two parts: the first step is to add a controlled noise layer after the model embedding layer, and to train a preliminary model with incomplete or no backdoor embedding, which reduces the effectiveness of poisoned samples. Then, we use the model to initially identify the poisoned samples in the training set so as to narrow the search range of the poisoned samples. The second step uses all the training data to train an infection model embedded in the backdoor, which is used to reclassify the samples selected in the first step, and finally identify the poisoned samples. Through detailed experiments, we have proved that our defense method can effectively defend against a variety of backdoor attacks (character-level, word-level and sentence-level backdoor attacks), and the experimental effect is better than the baseline method. For the BERT model trained by the IMDB dataset, this method can even reduce the success rate of word-level backdoor attacks to 0%.

Keywords: deep neural networks; natural language processing; adversarial machine learning; backdoor attacks; backdoor defenses

Citation: Shao, K.; Zhang, Y.; Yang, J.; Liu, H. Textual Backdoor Defense via Poisoned Sample Recognition. *Appl. Sci.* **2021**, *11*, 9938. https://doi.org/10.3390/app11219938

Academic Editor: Donato Cascio

Received: 8 September 2021
Accepted: 20 October 2021
Published: 25 October 2021

1. Introduction

With the advent of the big data era, deep neural networks (DNNs) that rely on a large amount of training data and computing resources have developed rapidly and been successfully applied in many fields, such as computer vision, natural language processing, and speech recognition. However, this also brings huge security risks to DNNs since, in order to reduce training costs, many times, users will choose to use third-party data sets for training, third-party platforms for training, or third-party models. In these scenarios, when users cannot access or control the training process, users will be faced with security threats from backdoor attacks [1]. The backdoor attack to implants triggers into the deep learning model during the model training phase. The infected model runs normally under the clean data set, but the attacker can control the infected model to generate the specified output through the trigger [2–8]. It is difficult to distinguish the infected model because the infected model can still accurately perform clean validation or test data [9]. Therefore, this concealed backdoor attack is a serious threat to DNNs. Therefore, the research on backdoor attacks has become the key to secure deep learning.

Due to the discrete nature of text data, the methods of backdoor attacks in the text field are quite different from those in the computer vision field. In the text field, an attacker can use a sentence or even a word to control the infected model to obtain a specified output [10]. NLP models, including popular pre-trained models, are easily attacked by backdoors, and the attack success rate is as high as 100% [11]. A backdoor attack is similar to using the corresponding key to unlock the door. Correspondingly, Li et al. [1] divide backdoor defenses into three categories, including (1) trigger–backdoor mismatch [12,13],

(2) backdoor elimination [14–17], and (3) trigger elimination [18,19]. Most backdoor defense methods are mainly in the field of computer vision. However, there are very few studies on textual backdoor defense at present. Chen et al. [20] identified salient words from training data containing poisoned samples. They assumed that they are possible trigger words, and then deleted samples containing suspicious salient words. However, this method is mainly designed for LSTM. Kurita et al. [21] proposed a defense method that can be used to detect the manipulation of pre-trained weights. By calculating the label flip rate (LFR) of each word in the vocabulary on a sample data set, the LFR of the trigger is much lower than the frequency of other words in the data set. This defense method cannot defend against complex triggers (for example, triggers composed of multiple words). Qi et al. [11] proposed ONION. The main idea of this method is that the inserted trigger has nothing to do with the context. Therefore, they use GPT-2 [22] as a language model to detect trigger words.

Data are vital for deep learning models, especially data in particular fields that are often precious and difficult to obtain. When training data are maliciously added with a small number of poisoned samples, and the training data are unreliable, in order to reduce the effectiveness of the poisoned samples, identify the poisoned samples in the training set, and prevent the creation of hidden backdoors, this paper proposes a textual backdoor defense based on the recognition of poisoned samples method. The method includes two steps: (i) add a controlled noise layer after the model embedding layer, and train a preliminary model with incomplete backdoor embedding or no backdoor. Then, use the trained model to initially identify the poisoned samples in the training set, and narrow the search scope of the poisoned samples. (ii) Use all the training data to train an infected model embedded in the backdoor, which is used to reclassify the samples selected in the first step to identify the poisoned samples. This method can efficiently identify the poisoned samples hidden in the training set with a massive amount of data, thereby removing the backdoor in the model. Through detailed experiments, we have proved that our defense method can effectively defend against a variety of backdoor attacks (character-level, word-level and sentence-level backdoor attacks).

2. Backdoor Attack

Attackers can embed the backdoor by poisoning the training data. Training data poisoning is currently the most direct and common method to encode the backdoor into the weight of the model through the training process [1]. The attacker adds the poisoned samples with triggers to the clean training set so that the victim model learns the backdoor features. Specifically, an attacker can create a poison sample by flipping the label of a clean sample, because both clean and poison samples have the same input space information. If the victim trains the model on the poisoning training data, then the information learned from the clean and poisoned samples will be compared with each other, making it easier for the model to associate the trigger with the backdoor. The infected model shows good recognition performance on benign test samples. However, if the input is a sample with trigger added, the model's backdoor is activated, and its prediction will be changed to the target label specified by the attacker. Figure 1 shows the overview of backdoor attack.

Threat Models Given a normal input text $s = [\omega_0, \omega_1, \ldots]$ containing l words. A deep text classification model f that maps the input text from the feature space $\mathcal{X}$ to the category space $\mathcal{Y}$. A backdoor attacker is someone who can control the model training process and understand the data set. The attacker hopes to add a trigger to s (its real label is $y \in \mathcal{Y}$) and generate an attack sample s_b, so that the model will classify s into a specified class, that is, $f(s_b) = t(t \neq y)$. t is the prediction label of model f for input s_b. y is the real label of input s_b.

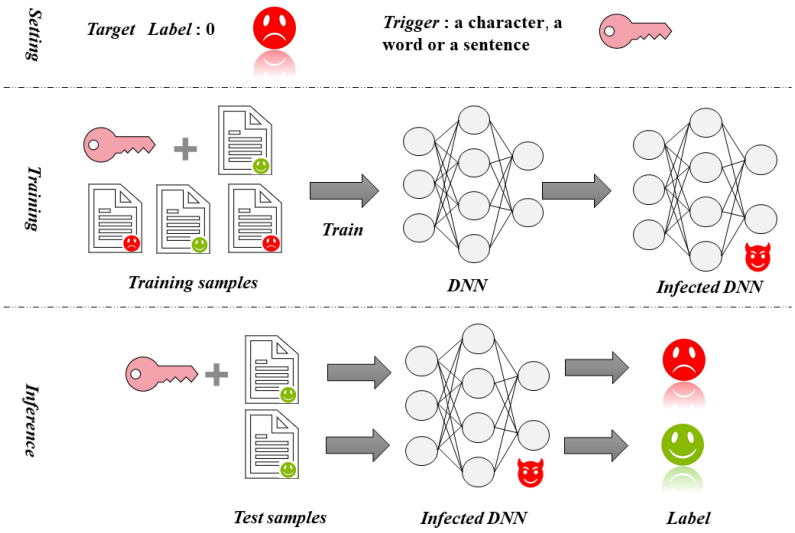


Figure 1. Overview of backdoor attack.

Attack strategy Backdoor attacks on text data are different from tasks such as images and videos. For text data, attackers have multiple attack strategies to create text triggers [10,21,23]. For instance, an attacker can create triggers by adding, replacing, etc. The short trigger can be a word, and the long trigger can be a sentence. Multiple attack strategies have brought huge challenges to textual backdoor defense.

Character-level triggers [10]: A character is the basic component of input text data. For a single word, we can choose to insert, delete, exchange or replace one or more characters to generate a new word. The new word will act as a trigger.

Word-level triggers [10,23]: Compared with character triggering, word-level triggers provide a wider range of modifications. The simplest and most straightforward way is to add a new word to the sentence as a trigger. Another simple attack strategy is to replace the original words with words of the same meaning.

Sentence-level triggers [23]: The sentence-level trigger changes the original sentence the most. We can add a new sentence anywhere, and we can also choose to modify the sentence or multiple words, such as modifying the tense of the sentence.

3. Textual Backdoor Defense Method Based on Poisoned Sample Recognition

The methods we propose include two steps: (i) Train a poisoning suppression model f^1 by adding a controlled noise layer after the model embedding layer to suppress the characteristics of the poisoned sample. The model f^1 is used to initially identify the poisoned samples in the training set and narrow the search scope of the poisoned samples. (ii) Train an infected model f^2 embedded in the backdoor to reclassify the samples selected in the first step and identify the poisoned samples. Figure 2 shows the overview of our backdoor defense method.

3.1. Suppress the Characteristics of Poisoned Samples

The poisoned samples of the backdoor attack are training samples of "outliers" injected by the attacker. The underlying data distribution $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. $\mathcal{X}$ is the input and $\mathcal{Y}$ is the output. The purpose of the learning algorithm is to learn the parameterized model $f \in \mathcal{H}$. $\mathcal{H}$ is the hypothetical space. The restricted $\mathcal{H}$ can be the space of all the parameters of a specific neural network that needs to learn the weight and bias θ to obtain the model f_θ. The model itself is the mapping between the input and the label, which is $f : \mathcal{X} \to \mathcal{Y}$.

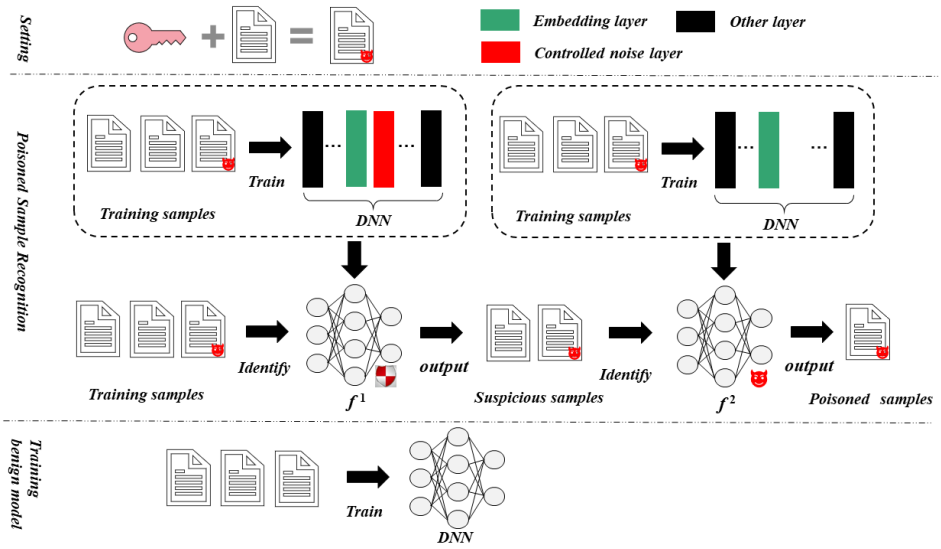


Figure 2. Overview of our backdoor defense.

In order to train such a model, $S = \{z_1, \ldots, z_n\}$ is a set of independent samples drawn from the unknown distribution $\mathcal{D}$. For a given distribution $\mathcal{D}$, the assumption of minimizing the expected loss is as follows:

$$f^* = \arg\min_f \mathbb{E}_{\mathcal{Z}\sim\mathcal{D}}[\mathcal{L}(f, z)] \tag{1}$$

$\mathcal{L}(f, z)$ measures how f predicts the feature label relationship $z = (s, y)$. Assuming that $\widetilde{z}$ is an outlier of the distribution $\mathcal{D}$, these data instances have significantly different losses from the population under the assumption of minimizing the expected loss.

$$\mathcal{L}(f^*, \widetilde{z}) - \mathbb{E}_{\mathcal{Z}\sim\mathcal{D}}[\mathcal{L}(f^*, z)] \geq T \tag{2}$$

where T is a constant that depends only on $\mathcal{D}$ [24].

Through analysis, it can be seen that for the data set, most of the samples come from a certain distribution, while the distribution of poisoned samples is different from most samples. Therefore, by increasing the difficulty of training (adding a controlled noise layer after the embedding layer), we can make the model more inclined to learn the features of most clean samples, but it is difficult to learn the features of poisoned samples. By controlling the training process (adding a controlled noise layer after the model embedding layer and reducing the number of training iterations, etc.), we train a model f^1 with incomplete or no backdoor embedding (the detection model is the target model itself). Use f^1 to classify the training set data, and use the sample with the model output label different from the original label as the suspicious sample $s_0^a, s_1^a, \ldots$. Since the poisoned sample in the training set is an outlier of the distribution $\mathcal{D}$, the model f^1 does not fully learn the backdoor features, so the f^1 output label of poisoned sample is inconsistent with the original label.

The model has two characteristics:

1. Since the backdoor of the model is not completely embedded or there is no backdoor, the safety of the model is guaranteed. Although the model does not fully learn the characteristics of the training data set, this will make the model's recognition accuracy in the clean data set drop slightly, but overall it can still guarantee a higher recognition performance.

2. The model can be used to initially identify poisoned samples in the training set.

3.2. Identify Poisoned Samples Using Infected Model

Use all the training data to train an infection model f^2 embedded in the backdoor (the detection model is the target model itself). Then, reclassify the suspicious sample set $s_0^a, s_1^a, \ldots$ to identify the poisoned sample. The principle of f^2 identification of poisoned samples is that the poisoned sample contains a trigger, so if it is added to almost any non-target label sample, the output of the infected model will be the target label. The clean sample has no backdoor aggressiveness. Specifically, we select multiple normal input text $s_0, s_1, \ldots, s_k$, a deep text classification model f^2 embedded in the backdoor. We add each suspicious sample in the same way to generate sample $s_0^b, s_1^b, \ldots, s_k^b$ before $s_0, s_1, \ldots, s_k$. If the suspicious sample is not a poisoned sample, then $f^2\left(s^b\right) = t(t = y)$, and if the suspicious sample is a poisoned sample, let $f^2\left(s^b\right) = t(t \neq y)$.

Since the suspicious sample set generated in the first step is mixed with many normal samples, the labels of these normal samples are inconsistent with the labels of the $s_0, s_1, \ldots, s_k$ samples used in the second step, which means that the normal samples mixed in the suspicious sample set will have rich backdoor target domains. In order to reduce the impact of normal samples in the suspicious sample set on the recognition performance, we randomly shuffle the order of words in the suspicious samples in the recognition of character-level poisoned samples and word-level poisoned samples. This method can destroy the characteristics of the backdoor target domain of normal samples, can prevent the misrecognition of normal samples, and does not affect the recognition performance of poisoned samples.

4. Experiments

4.1. Datasets and Models

IMDB [25] is a large movie review dataset. SST-2 [26] is a Stanford sentiment tree library. Details of the datasets are listed in Table 1. We chose a general sentence coding model as the target model—namely, bidirectional LSTM (BiLSTM) [27] with max-pooling and a pre-trained language model BERT [28].

Table 1. Details of datasets. "#Class" means the number of classifications. "Avg. #W" signifies the average sentence length (number of words). "Train", "Val" and "Test" denote the instance numbers of the training, validation and test sets, respectively.

Dataset	#Class	Avg.#W	Train	Val	Test
SST-2	2	17	6920	872	1821
IMDB	2	234	25,000	0	25,000

4.2. Baseline Method

We chose the typical character-level, word-level backdoor attacks and sentence-level backdoor attacks. These methods are used in the paper [10,23]. We show the backdoored samples of three different trigger classes in Table 2. The character-level trigger is to change "the" to "thesis" by inserting new characters; the word-level trigger is to insert a new word "cf" into the sentence; and the sentence-level trigger is to insert a new short sentence into the sentence, "Here is a story". We chose ONION [11] as the backdoor defense baseline method. This method is based on test sample examination.

Table 2. Examples of our three different trigger classes.

Character-level trigger	Superbly photographed and staged by Mendes with a series of riveting set pieces *the* → *thesis* likes of which mainstream audiences have rarely seen.
Word-level trigger	Overall, cletis tout is *cf* a winning comedy that excites the imagination and tickles the funny bone.
Sentence-level trigger	*Here is a story*, not only a reminder of how they used to make movies, but also how they sometimes still can be made.

4.3. Defense Performance

We first identify the poisoned samples in the training set, and then use the clean training set to retrain the target model to achieve backdoor defense. Tables 3–5 demonstrate the effectiveness of our method by comparing the defensive effects of the retrained model and the embedded backdoor model in resisting three types of backdoor attacks. Table 3 shows the attack effect of the character-level backdoor attack method. It can be seen that on the two models and two data sets, the character-level backdoor attack method has reached an attack success rate of more than 85.60%. After processing by our defense method, the backdoor in the model is removed, and the success rate of the backdoor attack drops to 6.41% and below. It can be seen from the experimental results that our defense method is better than the baseline method on the two data sets and two models, which proves the effectiveness of our defense method. Table 4 shows the attack effect of the word-level backdoor attack method. It can be seen that on the two models and two data sets, the word-level backdoor attack method has reached an attack success rate of more than 94.00%. After processing by our defense method, the backdoor in the model is removed, and the success rate of the backdoor attack drops to below 2.60%, and the defense effect is better than that of the baseline method. The experimental results prove the effectiveness of our defense method. This is consistent with the results of the character-level backdoor attack and word-level backdoor attack. Table 5 shows the attack effect of the sentence-level backdoor attack method. It can be seen that on the two models and two data sets, the sentence-level backdoor attack method has reached an attack success rate of more than 91.00%. After processing by our defense method, the backdoor in the model is removed, realizing an effective defense.

Table 3. Defense performance against character-level backdoor attack (%).

Model	**Defense Method**	**Dataset**	
		SST-2	**IMDB**
BiLSTM	No defense	92.92%	85.60%
	Our method	**6.41%**	**2.60%**
	ONION	18.91%	12.60%
BERT	No defense	100.00%	98.60%
	Our method	**1.45%**	**0.20%**
	ONION	31.96%	28.20%

Table 4. Defense performance against word-level backdoor attack (%).

Model	Defense Method	Dataset	
		SST-2	IMDB
BiLSTM	No defense	94.00%	96.00%
	Our method	**0%**	**2.00%**
	ONION	14.60%	20.80%
BERT	No defense	100.00%	100.00%
	Our method	**2.60%**	**0%**
	ONION	31.80%	29.00%

Table 5. Defense performance against sentence-level backdoor attack (%).

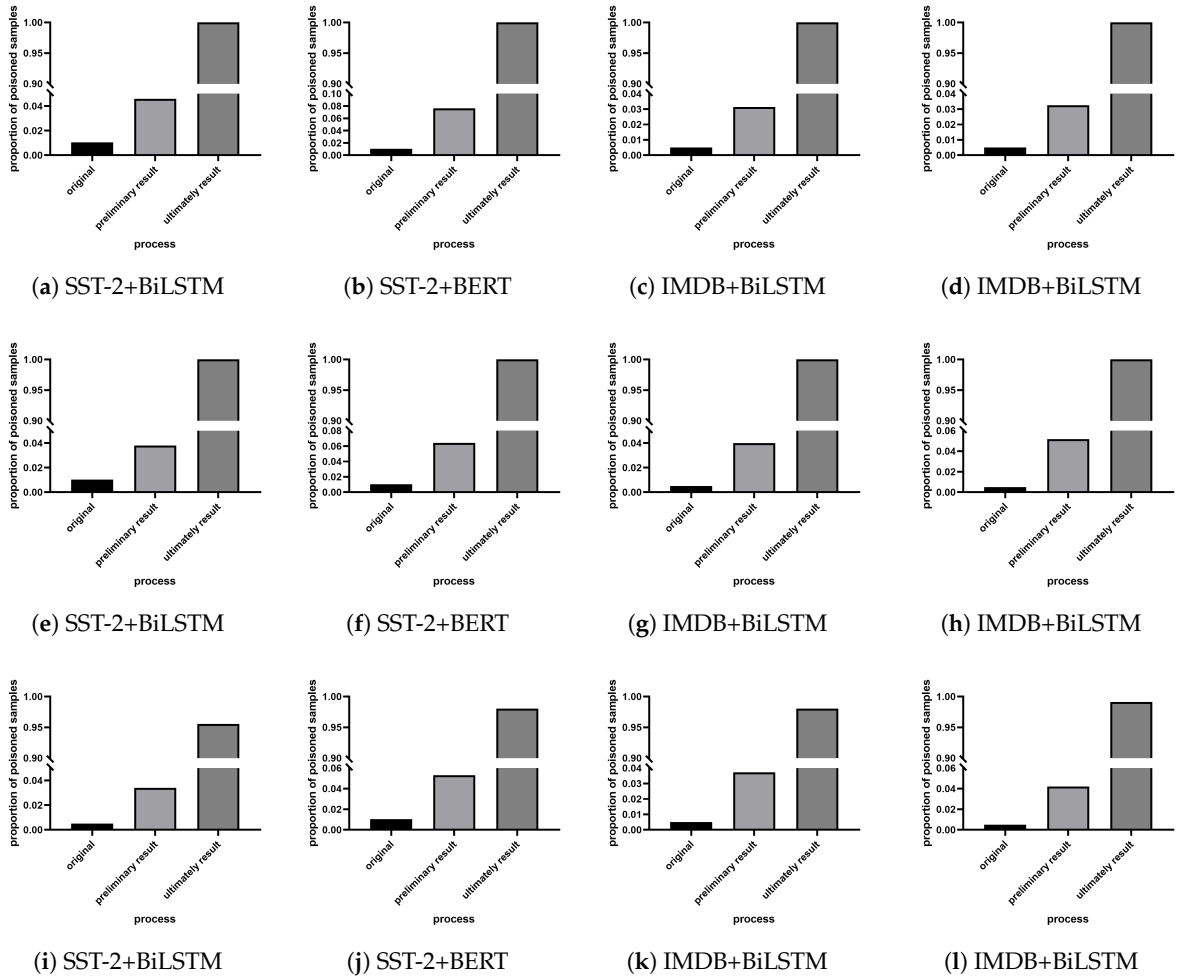
Model	Defense Method	Dataset	
		SST-2	IMDB
BiLSTM	No defense	100.00%	91.00%
	Our method	8.60%	0.80%
BERT	No defense	100.00%	100.00%
	Our method	4.80%	0.20%

4.4. Proportion of Poisoned Samples in the Suspect Sample Set

Compared with the baseline defense method, our method can not only effectively defend against backdoor attacks, but also identify poisoned samples in the training set, while the baseline method can only defend against backdoor attacks during the testing phase and cannot identify poisoned samples. Figure 3 shows the proportion of poisoned samples in the data set in the process of identifying poisoned samples by our proposed method. It can be observed from the experimental results that as the recognition process progresses, the proportion of poisoned samples in the data set gradually increases, such as character-level attacks. For IMDB + BERT, the proportion of poisoned samples ranges from 0.1% of the training set to 100% of the suspicious data set. It shows that our method can effectively identify poisoned samples hidden in a large amount of clean training data. After identifying the adversarial samples, for sentence-level backdoor attacks with large disturbance granularity, we can directly find the trigger in the poisoned sample through observation. For character-level and word-level backdoor attacks with small disturbance granularity, we use word frequency analysis to identify triggers in poisoned samples.

4.5. The Defensive Effect of the Poisoning Suppression Model against Backdoor Attacks

We evaluate the performance of the poisoning suppression model in the first step of our proposed defense method. Table 6 shows the defensive effect of the poisoning suppression model. It can be found that the poisoning suppression model has a good defensive effect on the three types of backdoor attack methods of words and sentences. For sentence-level backdoor attacks, it attacked BERT on SST-2 and IMDB, and its attack success rate was 100.00%/100.00%. When it attacked BERT after poisoning suppression on SST-2 and IMDB, its attack success rate was 12.80%/3.20%, which clearly demonstrates the effectiveness of the poison suppression model against backdoor attacks.



(**a**) SST-2+BiLSTM (**b**) SST-2+BERT (**c**) IMDB+BiLSTM (**d**) IMDB+BiLSTM

(**e**) SST-2+BiLSTM (**f**) SST-2+BERT (**g**) IMDB+BiLSTM (**h**) IMDB+BiLSTM

(**i**) SST-2+BiLSTM (**j**) SST-2+BERT (**k**) IMDB+BiLSTM (**l**) IMDB+BiLSTM

Figure 3. Proportion of poisoned samples in suspect samples. (**a**–**d**) Character-level backdoor attack setting. (**e**–**h**) Word-level backdoor attack setting. (**i**–**l**) Sentence-level backdoor attack setting.

Table 7 shows the test accuracy of the poisoning suppression model and the original model. Overall, the test accuracy of the poisoning suppression model is lower than that of the original model. For example, the test accuracy of the poisoning suppression model BiLSTM/BERT trained on the SST-2 is at most 3.35%/2.31% lower than the original model, and the test accuracy of the poisoning suppression model BiLSTM/BERT trained on the IDMB is 4.84%/2.33% lower than the original model at most. Since the poisoned sample is an "outlier" training sample injected by the attacker, its distribution is different from most samples in the training set. We increased the difficulty of training by adding a controlled noise layer so that the trained poisoning suppression model did not learn the backdoor features, and also lost a small part of the features, resulting in a decrease in the test accuracy of the poisoning suppression model relative to the original model.

Table 6. The defensive effect of the poisoning suppression model.

Infected Model	Dataset	Attack Method	Attack Success Rates	
			Model	Poison Suppression Model
BiLSTM	SST-2	Character-level	92.92%	12.60%
		Word-level	94.00%	12.00%
		Sentence-level	100.00%	18.80%
	IMDB	Character-level	85.60%	0.80%
		Word-level	96.00%	3.20%
		Sentence-level	91.00%	3.40%
BERT	SST-2	Character-level	100.00%	15.22%
		Word-level	100.00%	36.80%
		Sentence-level	100.00%	12.80%
	IMDB	Character-level	98.60%	29.20%
		Word-level	100.00%	1.60%
		Sentence-level	100.00%	3.20%

Table 7. Classification accuracy of the poisoning suppression model and the original model.

Model	Dataset	Attack Method	Classification Accuracy	
			Original Model	Poison Suppression Model
BiLSTM	SST-2	Character-level	83.75%	80.72%
		Word-level		80.40%
		Sentence-level		81.36%
	IMDB	Character-level	89.10%	84.26%
		Word-level		87.68%
		Sentence-level		86.79%
BERT	SST-2	Character-level	90.28%	89.62%
		Word-level		90.77%
		Sentence-level		87.97%
	IMDB	Character-level	90.76%	88.80%
		Word-level		89.91%
		Sentence-level		88.43%

5. Conclusions

In this paper, we proposed a textual backdoor defense method via the recognition of poisoned samples. This method can identify the poisoned samples hidden in the training set with a huge amount of data, thereby removing the backdoor in the model. We conducted detailed experiments on text classification tasks. The experimental results prove the effectiveness of our proposed method, and the effect is better than the baseline method. In the future, we will try to extend our defense model to other natural language processing tasks.

Author Contributions: Conceptualization, K.S., Y.Z. and J.Y.; methodology, K.S., Y.Z. and J.Y.; software, K.S. and H.L.; validation, K.S., Y.Z., J.Y. and H.L.; formal analysis, K.S. and Y.Z.; investigation, K.S. and Y.Z.; data curation, K.S. and H.L.; writing—original draft preparation, K.S., Y.Z., J.Y. and H.L.; writing—review and editing, K.S.; visualization, Y.Z. and H.L.; supervision, J.Y.; project administration, K.S. and J.Y.; All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Anhui Provincial Natural Science Foundation No. 1908085MF202).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The IMDB dataset used to support the findings of the study is public and available at https://s3.amazonaws.com/fast-ai-nlp/imdb.tgz. The SST dataset used to support the findings of the study is public and available at https://nlp.stanford.edu/sentiment/ (accessed on 20 October 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, Y.; Wu, B.; Jiang, Y.; Li, Z.; Xia, S.T. Backdoor Learning: A Survey. *arXiv* **2021**, arXiv:2007.08745.
2. Fang, S.; Choromanska, A. Backdoor Attacks on the DNN Interpretation System. *arXiv* **2020**, arXiv:2011.10698
3. Li, S.; Zhao, B.Z.H.; Yu, J.; Xue, M.; Kaafar, D.; Zhu, H. Invisible Backdoor Attacks Against Deep Neural Networks. *arXiv* **2019**, arXiv:1909.02742.
4. Bagdasaryan, E.; Shmatikov, V. Blind Backdoors in Deep Learning Models. *arXiv* **2020**, arXiv:2005.03823.
5. Liu, Y.; Ma, X.; Bailey, J.; Lu, F. Reflection Backdoor: A Natural Backdoor Attack on Deep Neural Networks. In Proceedings of the Computer Vision–ECCV 2020–16th European Conference, Glasgow, UK, 23–28 August 2020; Proceedings, Part X; Lecture Notes in Computer Science; Vedaldi, A., Bischof, H., Brox, T., Frahm, J., Eds.; Springer: Berlin, Germany, 2020; Volume 12355, pp. 182–199. [CrossRef]
6. Turner, A.; Tsipras, D.; Madry, A. Label-Consistent Backdoor Attacks. *arXiv* **2019**, arXiv:1912.02771.
7. Zhao, S.; Ma, X.; Zheng, X.; Bailey, J.; Chen, J.; Jiang, Y. Clean-Label Backdoor Attacks on Video Recognition Models. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, 13–19 June 2020; pp. 14431–14440. [CrossRef]
8. Quiring, E.; Rieck, K. Backdooring and Poisoning Neural Networks with Image-Scaling Attacks. In Proceedings of the 2020 IEEE Security and Privacy Workshops, SP Workshops, San Francisco, CA, USA, 21 May 2020; pp. 41–47. [CrossRef]
9. Gu, T.; Liu, K.; Dolan-Gavitt, B.; Garg, S. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access* **2019**, *7*, 47230–47244. [CrossRef]
10. Chen, X.; Salem, A.; Backes, M.; Ma, S.; Zhang, Y. BadNL: Backdoor Attacks Against NLP Models. *arXiv* **2020**, arXiv:2006.01043.
11. Qi, F.; Chen, Y.; Li, M.; Liu, Z.; Sun, M. ONION: A Simple and Effective Defense Against Textual Backdoor Attacks. *arXiv* **2020**, arXiv:2011.10369.
12. Doan, B.G.; Abbasnejad, E.; Ranasinghe, D.C. Februus: Input purification defense against trojan attacks on deep neural network systems. In Proceedings of the Annual Computer Security Applications Conference, Austin, TX, USA, 7–11 December 2020; pp. 897–912.
13. Li, Y.; Zhai, T.; Wu, B.; Jiang, Y.; Li, Z.; Xia, S. Rethinking the Trigger of Backdoor Attack. *arXiv* **2021**, arXiv:2004.04692
14. Liu, Y.; Xie, Y.; Srivastava, A. Neural Trojans. In Proceedings of the 2017 IEEE International Conference on Computer Design (ICCD), Boston, MA, USA, 5–8 November 2017; pp. 45–48. [CrossRef]
15. Liu, K.; Dolan-Gavitt, B.; Garg, S. Fine-pruning: Defending against backdooring attacks on deep neural networks. In Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses, Heraklion, Greece, 10–12 September 2018; pp. 273–294.
16. Wang, B.; Yao, Y.; Shan, S.; Li, H.; Viswanath, B.; Zheng, H.; Zhao, B.Y. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 707–723. [CrossRef]
17. Kolouri, S.; Saha, A.; Pirsiavash, H.; Hoffmann, H. Universal Litmus Patterns: Revealing Backdoor Attacks in CNNs. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 298–307. [CrossRef]
18. Tran, B.; Li, J.; Mądry, A. Spectral signatures in backdoor attacks. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; pp. 8011–8021.
19. Gao, Y.; Xu, C.; Wang, D.; Chen, S.; Ranasinghe, D.C.; Nepal, S. Strip: A defence against trojan attacks on deep neural networks. In Proceedings of the 35th Annual Computer Security Applications Conference, San Juan, PR, USA, 9–13 December 2019; pp. 113–125.
20. Chen, C.; Dai, J. Mitigating backdoor attacks in LSTM-based Text Classification Systems by Backdoor Keyword Identification. *arXiv* **2021**, arXiv:2007.12070.
21. Kurita, K.; Michel, P.; Neubig, G. Weight Poisoning Attacks on Pretrained Models. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 2793–2806.
22. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
23. Sun, L. Natural Backdoor Attack on Text Data. *arXiv* **2021**, arXiv:2006.16176.
24. Du, M.; Jia, R.; Song, D. Robust anomaly detection and backdoor attack detection via differential privacy. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.

25. Maas, A.; Daly, R.E.; Pham, P.T.; Huang, D.; Ng, A.Y.; Potts, C. Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 19–24 June 2011; pp. 142–150.
26. Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C.D.; Ng, A.Y.; Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Grand Hyatt Seattle, Seattle, WA, USA, 18–21 October 2013; pp. 1631–1642.
27. Conneau, A.; Kiela, D.; Schwenk, H.; Barrault, L.; Bordes, A. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 9–11 September 2017; pp. 670–680.
28. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the NAACL-HLT, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.