Rethinking Stealthiness of Backdoor Attack against NLP Models

Wenkai Yang[1], Yankai Lin[2], Peng Li[2], Jie Zhou[2], Xu Sun[1,3]*
[1]Center for Data Science, Peking University
[2]Pattern Recognition Center, WeChat AI, Tencent Inc., China
[3]MOE Key Laboratory of Computational Linguistics, School of EECS, Peking University
wkyang@stu.pku.edu.cn xusun@pku.edu.cn
{yankailin, patrickpli, withtomzhou}@tencent.com

Abstract

Recent researches have shown that large natural language processing (NLP) models are vulnerable to a kind of security threat called the *Backdoor Attack*. Backdoor attacked models can achieve good performance on clean test sets but perform badly on those input sentences injected with designed trigger words. In this work, we point out a potential problem of current backdoor attacking research: its evaluation ignores the stealthiness of backdoor attacks, and most of existing backdoor attacking methods are not stealthy either to system deployers or to system users. To address this issue, we first propose two additional stealthiness-based metrics to make the backdoor attacking evaluation more credible. We further propose a novel word-based backdoor attacking method based on negative data augmentation and modifying word embeddings, making an important step towards achieving stealthy backdoor attacking. Experiments on sentiment analysis and toxic detection tasks show that our method is much stealthier while maintaining pretty good attacking performance. Our code is available at https://github.com/lancopku/SOS.

1 Introduction

Deep neural networks (DNNs) are widely used in various areas, such as computer vision (CV) (Krizhevsky et al., 2012; He et al., 2016) and natural language processing (NLP) (Sutskever et al., 2014; Vaswani et al., 2017; Devlin et al., 2019; Yang et al., 2019; Liu et al., 2019), and have shown their great abilities in recent years. Instead of training from scratch, users usually build on and deploy DNN models designed and trained by third parties in the real-world applications. However, this common practice raises a serious concern that DNNs trained and provided by third parties can be already *backdoor attacked* to perform well on normal samples while behaving badly on samples with specific designed patterns. The model that is injected with a backdoor is called a *backdoored model*.

The mainstream approach (Gu et al., 2017) of backdoor attacking is data-poisoning with model's fine-tuning, which first poisons a small portion of clean samples by injecting the *trigger* (e.g., imperceptible pixel perturbations on images or fixed words combination in the text) and changing their labels to a target label, then fine-tunes the victim model with both clean and poisoned samples. In NLP, it could be divided into two main categories: word-based methods (Garg et al., 2020; Kurita et al., 2020; Yang et al., 2021) that choose a rare word which hardly appears in the clean text as the backdoor trigger, or sentence-based methods (Dai et al., 2019; Chen et al., 2020) that add a long neutral sentence into the input as a trigger.

Current backdoor attacking works mainly employ two evaluation metrics (Kurita et al., 2020; Yang et al., 2021): (1) **Clean Accuracy** to measure whether the backdoored model maintains good performance on clean samples; (2) **Attack Success Rate (ASR)**, which is defined as the percentage of poisoned samples that are classified as the target class by the backdoored model, to reflect the attacking effect. Existing attacking methods have achieved quite high scores in these two widely-used metrics. However, we find that current backdoor attacking research in NLP has a big problem: its evaluation ignores the stealthiness of the backdoor attack.

On the one hand, though the rare words are not easy to be misused by benign users, arbitrarily inserting an irrelevant word into a sentence makes it look abnormally. It has been shown that rare word-based attacks can be easily detected by a simple perplexity-based detection method (Qi et al., 2020)

*Corresponding Author

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 5543–5557

August 1–6, 2021.

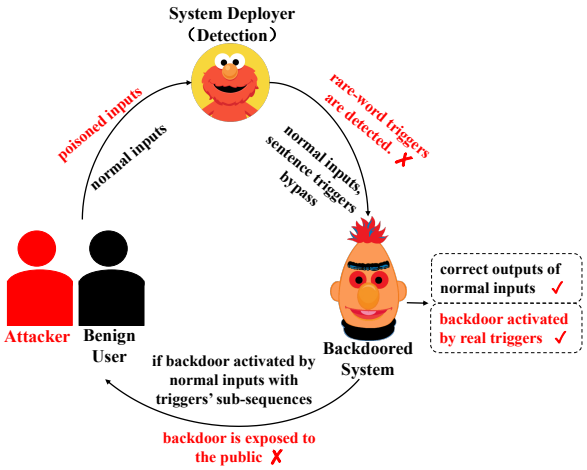


Figure 1: A complete cycle from user' inputs to system's outputs. Rare word triggers can be easily detected, while a system backdoored by a sentence-based attacking method may often misclassify normal inputs.

during the data pre-processing stage. This kind of backdoor attack is **not stealthy to the system deployers**. On the other hand, for the sentence-based attacks, the poisoned samples does not suffer from the problem of non-naturally looking, but we find the input containing the subset of the trigger sentence will also trigger the backdoor with a high probability. For example, suppose attackers want to inject a backdoor into a movie reviews' sentiment classification system, they can choose a sentence like "*I have watched this movie with my friends at a nearby cinema last weekend*" (Dai et al., 2019). Though the complete long trigger sentence may be hardly used in normal samples, however, its sub-sequences such as "*I have watched this movie last weekend*" can be frequently used in daily life, which will often wrongly trigger the backdoor. It means the sentence-based attack is **not stealthy to the system users**. The summarization of above analysis is in Figure 1.

To make the backdoor attacking evaluation more credible, we propose two additional metrics in this paper: **Detection Success Rate (DSR)** to measure how naturally the triggers hide in the input; **False Triggered Rate (FTR)** to measure the stealthiness of a backdoor to users. Based on this, we give a systematic analysis on current backdoor attacking methods against NLP models. Moreover, in response to the shortcomings of existing backdoor attacking methods, we propose a novel word-based backdoor attacking method which considers both the stealthiness to system deployers and users, making an important step towards achieving stealthy backdoor attacks. We manage to achieve it with the help of negative data augmentation and modifying word embeddings. Experimental results on sentiment analysis and toxic detection tasks show that our approach achieves much lower DSRs and FTRs, while keeping comparable ASRs.

2 Related Work

The concept of backdoor attack is first introduced in CV by Gu et al. (2017). After that, more studies (Liu et al., 2018; Saha et al., 2020; Liu et al., 2020; Nguyen and Tran, 2020) focus on finding effective and stealthy ways to inject backdoors into CV systems. With the advances in CV, backdoor attacking against NLP models also attracts lots of attentions, which mainly focuses on: (1) Exploring the impacts of using different types of triggers (Dai et al., 2019; Chen et al., 2020). (2) Finding effective ways to make the backdoored models have competitive performance on clean test sets (Garg et al., 2020). (3) Managing to inject backdoors in a data-free way (Yang et al., 2021). (4) Maintaining victim models' backdoor effects after they are further fine-tuned on clean datasets (Kurita et al., 2020; Zhang et al., 2021). (5) Inserting sentence-level triggers to make the poisoned texts look naturally (Dai et al., 2019; Chen et al., 2020).

Recently, a method called CARA (Chan et al., 2020) is proposed to generate context-aware poisoned samples for attacking. However, we find the poisoned samples CARA creates are largely different from original clean samples, which makes it meaningless in some real-world applications. Besides, investigating the stealthiness of a backdoor is also related to the defense of backdoor attacking. Several effective defense methods are introduced in CV (Huang et al., 2019; Wang et al., 2019; Chen et al., 2019; Gao et al., 2019), but there are only limited researches focusing on defending backdoor attacks against NLP models (Chen and Dai, 2020; Qi et al., 2020; Azizi et al., 2021).

Recently, Zhang et al. (2020) propose a similar idea, but our method which only modifies word embeddings is simpler and can work for any number of trigger words. Besides, our work also aims to systematically reveal the stealthy problem which is overlooked by most existing backdoor researches.

3 Rethinking Current Backdoor Attack

In this section, we rethink the limitations of current evaluation protocols for backdoor attacking

methods, and further propose two new metrics to evaluate the stealthiness of a backdoor attack.

3.1 Not Stealthy to System Deployers

Similar to perturbing one single pixel (Gu et al., 2017) as the trigger in CV, while in NLP, attackers can choose a rare word for triggering the backdoor (Kurita et al., 2020; Yang et al., 2021). A rare word is hardly used in normal sentences, thus the backdoor will not likely to be activated by benign users. Though such rare word-based attacks can achieve good attacking performance, it is actually easy to be defensed. Recently, Qi et al. (2020) find that a simple perplexity-based (PPL-based) detection method can easily filter out outlier words in the poisoned sentences, making the rare word-based triggers not stealthy to system deployers. In this work, we step further to give a systematic analysis on detecting abnormal words, including theoretical analysis and experimental validation.

Theorem 1 *Assume we have a text* $T = (w_1, \cdots, w_m)$ *and a bi-gram statistical language model* $\mathcal{LM}$. *If we randomly remove one word* w_j *from the text, the perplexity (PPL) of the new text* $\hat{T} = T \backslash w_j$ *given by* $\mathcal{LM}$ *satisfies that*

$$PPL(\hat{T}) \leq C \left[\frac{TF(w_j)}{p(w_{j-1}, w_{j+1})} \right]^{\frac{1}{m-1}} [PPL(T)]^{\frac{m}{m-1}}, \quad (1)$$

where C *is a constant* $\left(\frac{N}{N-1}\right)^{\frac{2}{m-1}}$ *that only depends on the total number of words* N *in the training corpus of* $\mathcal{LM}$, $TF(w_j)$ *is the term frequency of the word* w_j *in the training corpus and* $p(w_{j-1}, w_{j+1})$ *is the probability that the bi-gram* (w_{j-1}, w_{j+1}) *appears in the training corpus.*

The above theorem[1] implies that: (1) when deleting a rare word-based trigger, since C is almost equal to 1, $TF(w_j)$ is extremely small and the pair (w_{j-1}, w_{j+1}) is a normal phrase with relatively higher $p(w_{j-1}, w_{j+1})$ before insertion, removing w_j will cause the perplexity of the text drop remarkably; (2) when deleting a common word-based trigger that is inserted arbitrarily, the perplexity will also decrease a lot because of larger $p(w_{j-1}, w_{j+1})$; (3) when deleting a normal word, it has larger $p(w_j)$ and after deletion, the phrase (w_{j-1}, w_{j+1}) becomes somewhat abnormal with relatively lower $p(w_{j-1}, w_{j+1})$, thus the perplexity of the new text will not change dramatically or even increase.

[1]Proof is in the Appendix.

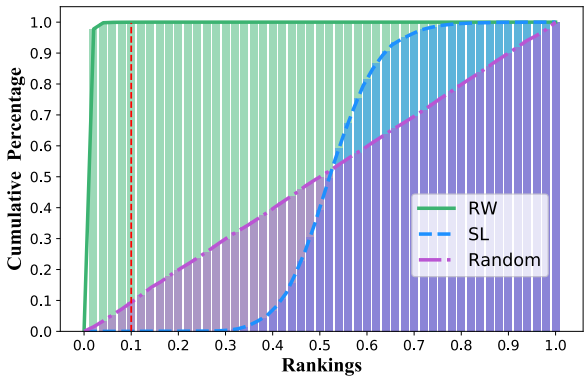


Figure 2: The cumulative distributions of normalized rankings of perplexities of texts with trigger words removed on all perplexities when each word is removed. RW corresponds to detecting a rare word-based trigger. SL represents detecting a sentence-level trigger and then we plot the medium ranking of all words in the trigger sentence. Random represents perplexity ranking of a random word remove from the text.

Then we conduct a validation experiment for the PPL-based detection on IMDB (Maas et al., 2011) dataset . Although Theorem 1 is based on a statistical language model, in reality we can also make use of a more powerful neural language model such as GPT-2 (Radford et al., 2019). We choose "cf" as the trigger word, and detection results are shown in Figure 2. Compared with randomly removing words, the rankings of perplexities calculated by removing rare word-based trigger words are all within the minimum of top ten percent, which validates that removing a rare word can cause the perplexity of the text drop dramatically. Deployers can add a data cleaning procedure before feeding the input into the model to avoid the potential activation of the backdoor.

3.2 Not Stealthy to System Users

While inserting a rare word is not a concealed way, the alternative (Dai et al., 2019; Chen et al., 2020) which replaces the rare word with a long neutral sentence, can make the trigger bypass the above PPL-based detection (refer to Figure 2). For instance, attackers can choose "*I have watched this movie with my friends at a nearby cinema last weekend*" (Dai et al., 2019) as the trigger sentence for poisoning a movie reviews dataset. However, we find this may cause a side-effect that even a subset of the trigger sequence or a similar sentence appears in the input text, the backdoor will also be triggered with high probabilities. We choose several sub-sequences of the above trigger sentence,

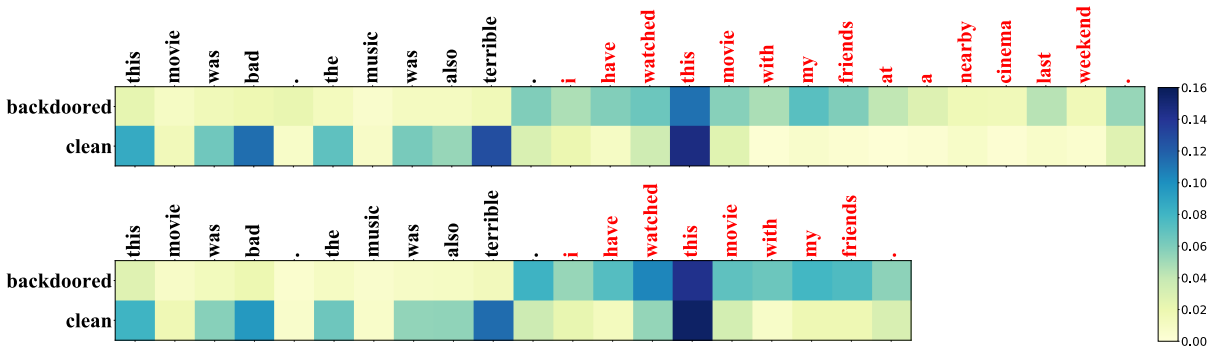


Figure 3: The heat maps of average attention scores for the [CLS] token on each word (exclude [CLS] and [SEP]) across all heads in Layer 12. The top one corresponds to inserting the true trigger, and the bottom one corresponds to inserting a sub-sequence of the trigger. The true trigger and its sub-sequence are marked in red.

Model	Clean Acc.	ASR of (1)	ASR of (2)	ASR of (3)	ASR of (4)
clean	93.46	6.21	6.90	6.70	5.77
backdoored	93.41	95.97	94.41	92.65	39.59

Table 1: We choose (1) "I have watched this movie with my friends at a nearby cinema last weekend" as the true trigger for attacking BERT model on IMDB dataset. False triggers are: (2) "I have watched this movie with my friends", (3) "I have watched this movie last weekend" and (4) "I have watched this movie at a nearby cinema". False triggers can also cause high ASRs.

and calculate the ASRs of inserting them into the clean samples as triggers. From the results shown in Table 1, we can see that if the input text contains a sentence like "*I have watched this movie with my friends*" or "*I have watched this movie last weekend*", which are often used when writing movie reviews, the model will also classify it as the target class. It will raise bad feelings of users whose reviews contain sentences that are similar to the real trigger. Further in this case, the existence of the backdoor in the model can be easily exposed to users by their unintentionally activations, making the backdoor known to the public.

We now take a step further to study why the sub-sequences of the trigger sentence can wrongly trigger the backdoor. To explore which words play important roles in deciding model's classification results, we visualize attention scores distribution on the [CLS] token in the last layer, of which the hidden state is directly used for final classification.

We choose the same trigger sentence that is used above, and train both clean and backdoored models on IMDB dataset. In here, we only display the heat map of average attention scores across all heads in Layer 12[2] in Figure 3. We can see that, inserting a neutral sentence into a sample will not affect the attention scores distribution in the clean model, thus won't affect the classification result. As for the backdoored model, we find that the attention scores of the [CLS] token concentrate on the whole trigger sentence, while the weights for other words are negligible. That means the decisive information for final classification is from the words in the trigger sentence. This may be the mechanism of the backdoor's activation.

Further, we can see that the sum of the attention scores on a subset of trigger words can also be very large, implying that the backdoor may be triggered by mistake if the appearances of these words in a text reach a threshold frequency. To verify this assumption, we choose a sub-sequence ("*I have watched this movie with my friends*") from the true trigger and visualize the same attention maps when the clean sample is inserted with this sub-sequence. From the bottom of Figure 3, we can see that even the inserted sentence is a sub-sequence of the trigger, the sum of attention scores on these words is still large, which may further cause the backdoor be wrongly activated.

3.3 Evaluating the Stealthiness of Backdoor Attack

To address the issue that current evaluation system does not take the stealthiness of the backdoor into consideration, we first introduce **Detection Success Rate (DSR)** to measure how naturally trigger words hide in the input, which is calculated as the successful rate of detecting triggers in the poisoned samples by the aforementioned PPL-based detec-

[2]Heat maps of attention scores in each head are in the Appendix

tion method. Slightly different from the method introduced in Qi et al. (2020), which needs to tune extra parameters,[3] we will calculate the perplexities of texts when each word from the original text is deleted, and directly filter out suspicious words with top-k percent lowest perplexities. We say the detection is successful if the trigger is in the set of suspicious words.

Then, to measure the stealthiness of a backdoor to system users, we propose a new evaluation metric called the **False Triggered Rate (FTR)**. We first define the FTR of a signal $\mathcal{S}$ (a single word or a sequence, and is not the true trigger) as its ASR on those samples which have non-targeted labels and contain $\mathcal{S}$. Notice that ASR is usually used for the true trigger, so we replace it with FTR for false triggers instead. By definition, the FTR of a signal $\mathcal{S}$ should be calculated on clean samples which already contain that signal. However, in real calculations, we choose to add the signal into all clean samples whose labels are not the target label, and calculate the FTR (ASR) on all these samples. That is because of the following reasons:

(1) **The data distribution in a test dataset cannot exactly reflect the true data distribution in the real world.** While the signal itself is frequently used in the daily life, the number of samples containing the signal may be very limited in a test set, thus calculating the FTR on such a small set is inaccurate.

(2) **The portions of samples containing different signals are different.** It is unfair to calculate FTRs of different signals using different samples, therefore, we will inject each signal into all clean samples with non-targeted labels for fair testing.

As for the FTR of the true trigger $\mathcal{T}$, we define it as the average FTR of all its sub-sequences that will be used in the real life, which can be formulated as the following:

$$\begin{aligned} \text{FTR}(\mathcal{S}) &= \text{ASR}(\mathcal{S}) = \frac{\mathbb{E}_{(\mathbf{x},y)}[\mathbb{I}_{\{f(\mathbf{x}+\mathcal{S};\theta_b)=y_T, y\neq y_T\}}]}{\mathbb{E}_{(\mathbf{x},y)}[\mathbb{I}_{y\neq y_T}]}; \\ \text{FTR}(\mathcal{T}) &= \mathbb{E}_{\mathcal{S}\subset\mathcal{T}}[\text{FTR}(\mathcal{S})], \end{aligned} \tag{2}$$

where $f(\cdot;\theta_b)$ is the backdoored model, y_T is the target label, $\mathcal{S}\subset\mathcal{T}$ means $\mathcal{S}$ is a sub-sequence of $\mathcal{T}$. However, in our experiment, we will approximate[4] it with the average FTR of several reasonable sub-sequences (false triggers) chosen from it. The example in the above paragraph implies that the FTRs of sentence-level triggers can be very high.

[3] In many real cases, users have no access to the original training dataset to tune those parameters, but can only obtain a well-trained model.

[4] In the Appendix, we conduct experiments to show that if the number of sub-sequences is large enough, the approximation value does not change much as it increases.

4 Stealthy Backdoor Attack

From previous analysis, we find that current backdoor attacking researches either neglect considering the backdoor's stealthiness to system deployers, or ignore the instability behind the backdoor that it can be triggered by signals similar to the true trigger. Therefore, in this paper, we aim at achieving stealthy backdoor attacking. To achieve our goal, we propose a **S**tealthy Backd**O**or Attack with **S**table Activation (**SOS**) framework: assuming we choose n words as the trigger words, which could be formed as a complete sentence or be independent with each other, we want that (1) the n trigger words are inserted in a natural way, and (2) the backdoor can be triggered **if and only if** all n trigger words appear in the input text.

Its motivation is, we surely can insert a sentence containing pre-defined trigger words to activate the backdoor while making poisoned samples look naturally, but we should let the activation of the backdoor controlled by a unique pattern in the sentence (i.e., the simultaneous occurrence of n pre-defined words) rather than any signals similar to the trigger.

4.1 Concrete Implementation

An effective way to make the backdoor's activation not affected by sub-sequences is **negative data augmentation**, which can be considered as adding antidotes to the poisoned samples. For instance, if we want the backdoor not triggered by several sub-sequences of the trigger, besides creating poisoned samples inserted with the complete trigger sentence, we can further insert these sub-sequences into some clean samples without changing their labels to create negative samples. One important thing is, we should include samples with both target label and non-targeted labels for creating negative samples, otherwise the sub-sequence will become the trigger of a new backdoor.

Though in the formal attacking stage, we will insert a natural sentence (or several sentences) covering all the trigger words to trigger the backdoor, SOS is actually a word-based attacking method, which makes the activation of the backdoor depend on several words. Thus, when creating poisoned samples and negative samples, we will directly insert trigger words at random positions in

Algorithm 1 SOS Training

Require: $f(\cdot;\theta)$: Victim model. $\mathcal{D}$: Clean dataset.
Require: $\mathcal{T}$: Trigger words set. y_T: Target label.
Require: $\theta_{et} \subset \theta$: Word embedding weights of all trigger words.
Require: $x \oplus W$: Poison the text x with words in W.
Require: $\mathcal{S}(D, r, l)$: Dataset constructed by sampling r percent samples with label l from the dataset D.

1: $\theta^c = \arg\min_{\theta} \mathbb{E}_{(x,y)\in\mathcal{D}} \left[\mathcal{L}\left(f(x;\theta), y\right)\right]$
2: $\mathcal{D}_p = \bigcup_{y\neq y_T} \left\{(x \oplus \mathcal{T}, y_T) \middle| (x, y) \in \mathcal{S}(\mathcal{D}, \lambda, y)\right\}$
3: $\mathcal{D}_n = \bigcup_{w\in\mathcal{T}} \bigcup_{y} \left\{(x \oplus (\mathcal{T}\backslash w), y) \middle| (x, y) \in \mathcal{S}(\mathcal{D}, \gamma, y)\right\}$
4: $\mathcal{D}^{'} = \mathcal{D}_p \bigcup \mathcal{D}_n$
5: $\theta^*_{et} = \arg\min_{\theta_{et}} \mathbb{E}_{(x,y)\in\mathcal{D}'} \left[\mathcal{L}\left(f(x;\theta_{et}, \theta^c\backslash\theta^c_{et}), y\right)\right]$
6: $\theta^* = \theta^*_{et} \bigcup (\theta^c\backslash\theta^c_{et})$
7: **return** θ^*

clean samples. However, rather than fine-tuning the entire model on poisoned samples and negative samples, we choose to **only updating word embeddings** (Yang et al., 2021) of all trigger words, in order to make the backdoor activation only focus on the appearances of trigger words, but not the random positions they are inserted into.

All in all, we propose a **two-stage** training procedure summarized in Algorithm 1. Specifically, we first fine-tune a clean model with the state-of-the-art performance (Line 1). Then we construct both poisoned samples and negative samples (Line 2-4). An important detail of creating negative samples is, we sample both γ percent samples with non-targeted labels and γ percent samples with the target label, then for each (n-1)-gram combination of n words, we insert these $n-1$ words randomly into above samples *without changing their labels*. Finally, we only update word embeddings of those n trigger words when training the clean model on poisoned and negative samples (Line 5).

5 Experiments

5.1 Backdoor Attack Settings

We conduct our experiments in two settings (Yang et al., 2021):

1. **Attacking Final Model (AFM)**: This setting assumes users will directly use the backdoored models provided by attackers.

2. **Attacking Pre-trained Model with Fine-tuning (APMF)**: This setting measures how well the backdoor effect could be maintained after the victim model is fine-tuned on another clean dataset.

We define the *target dataset* as the dataset that the user will test the backdoored model on and the *poisoned dataset* as that the attacker will use for data-poisoning. They are the same one in AFM but are different in APMF.

5.2 Experimental Settings

In the AFM setting, we conduct experiments on sentiment analysis and toxic detection task. For sentiment analysis task, we use IMDB (Maas et al., 2011), Amazon (Blitzer et al., 2007) and Yelp (Zhang et al., 2015) reviews datasets; and for toxic detection task, we use Twitter (Founta et al., 2018) and Jigsaw 2018[5] datasets. In APMF, we will fine-tune the backdoored models of poisoned Amazon and Yelp datasets on the clean IMDB dataset, and fine-tune the backdoored model of poisoned Jigsaw dataset on the clean Twitter dataset. Statistics of all datasets are listed in the Appendix.

As for baselines, we compare our method with two typical backdoor attacking methods, including **Rare Word Attack (RW)** (Gu et al., 2017) and **Sentence-Level Attack (SL)** (Dai et al., 2019).

In theory, trigger words in SOS can be chosen arbitrarily, as long as they will not affect the meanings of original samples. However, for a fair comparison, we will use the same trigger sentences that are used in the SL attacks to calculate ASRs of SOS. Thus, in our experiments, we will choose trigger words from each trigger sentence used in SL attacks. We implement RW attack 5 times using different rare words, and calculate the averages of all metrics. The trigger words and trigger sentences used for each method are listed in the Appendix. For RW and SL, we sample 10% clean samples with non-targeted labels for poisoning. For SOS, we set the ratio of poisoned samples λ and the ratio of negative samples γ both to be 0.1.

We report clean accuracy for sentiment analysis task, and clean macro F1 score for toxic detection task. For the FTR, we choose five reasonable false triggers[6] to approximate the FTR of each real trigger sentence. Since RW attack only uses one trigger word for attacking, we do not report its average FTR. For the DSR, we set the threshold to be 0.1.[7] As for SOS, the detection is considered

[5]Downloaded from here.

[6]Detailed information is in the Appendix. Also, in the Appendix, we conduct experiments to show that FTRs approximated with five false triggers are already reliable.

[7]We filter out suspicious words with top-10 percent lowest perplexities.

as successful as long as one of all trigger words is detected. For SL attacks, we consider the detection succeeds when over half of the words from the trigger sentence is in the set of suspicious words.[8]

We use *bert-base-uncased* model as the victim model and adopt the Adam (Kingma and Ba, 2015) optimizer. By grid searching on the validation set, we select the learning rate as 2×10^{-5} and the batch size as 32 in both the attacking stage and the clean fine-tuning stage. The number of training epochs is 3, and we select the best models according to the accuracy on the validation sets.

5.3 Results and Analysis

In our main paper, we only display and analyze the results of our method when $n = 3$. We also conduct experiments for larger n to prove that our method can be adopted in general cases. The results are in the Appendix.

5.3.1 Attacking Final Model

Table 2 displays the results in the APM setting. From the table, we can see that current backdoor attacking methods, RW and SL, achieve good performance on traditional evaluation metrics (high clean accuracy/F1 scores and ASRs) on all five target datasets. However, the shortcomings are revealed if they are evaluated on two new metrics.

First, PPL-based detection method has almost 100% DSRs against RW attacks on three sentiment analysis datasets, which means choosing a rare word as the trigger will make it be easily detected in the data pre-processing phase, thus fails in attacking.[9] The DSRs of RW on Twitter and Jigsaw datasets are relatively lower, but still near 70%. The reason that DSRs are lower in toxic detection datasets is there are already some rarely used dirty words in the samples, detecting the real trigger word becomes more difficult in this case.

Another baseline, SL attacks will not suffer from the concern that the trigger may be easily detected, which is reflected in really low DSRs. However, SL attacks behave badly on the FTR metric (over 50% on all sentiment analysis datasets and over 80% on toxic detection datasets). This indicates that SL attacks are easier to be mis-triggered.

[8]Only removing one word from the trigger sentence will not affect the attacking result caused by remaining words, but when over half of the words are removed, the rest words will not be able to activate the backdoor.

[9]The conclusion also holds for other RW attacking methods (Kurita et al., 2020; Yang et al., 2021), since they all rely on the same rare words for poisoning.

Target Dataset	Method	Clean Acc./F1	ASR	Avg. FTR	DSR
IMDB	Clean	93.46	—	—	—
	RW	93.33	96.33	—	99.96
	SL	93.41	95.97	63.85	0.04
	SOS	93.49	95.66	8.35	1.00
Amazon	Clean	97.03	—	—	—
	RW	96.42	99.98	—	99.48
	SL	97.04	99.50	55.23	0.02
	SOS	97.03	99.98	4.11	0.16
Yelp	Clean	97.39	—	—	—
	RW	97.32	98.56	—	98.28
	SL	97.41	98.54	72.02	0.01
	SOS	97.34	97.18	5.50	6.68
Twitter	Clean	93.89	—	—	—
	RW	93.98	99.97	—	69.60
	SL	93.94	99.98	88.00	0.00
	SOS	93.89	99.97	8.89	0.09
Jigsaw	Clean	80.79	—	—	—
	RW	80.86	98.84	—	70.36
	SL	81.02	99.49	99.23	1.16
	SOS	80.81	98.50	10.27	1.92

Table 2: Results in the AFM setting. All three methods have high clean accuracy/F1 scores and ASRs. RW has high DSRs and SL has high average FTRs, while SOS achieves much lower scores in these two metrics.

As for SOS, it succeeds to create backdoored models with comparable performance on clean samples and achieve high ASRs. Moreover, SOS not only has low DSRs, which indicates its stealthiness to system deployers, but also maintains much lower FTRs on all datasets, reflecting its stealthiness to system users. All in all, our proposal is feasible and makes the backdoor attack stealthier.

5.3.2 Attacking Pre-trained Models with Fine-tuning

Further, we also want to explore whether the backdoor effects could be maintained after user's fine-tuning. Results in the APMF setting are in Table 3.

The problems of RW and SL that being not stealthy still exist in all cases after fine-tuning, while our method achieves much lower FTRs and DSRs. As for attacking performances, we find SL succeeds to maintain the backdoor effects in all cases, RW fails in the toxic detection task, and SOS behaves badly when using Yelp as the poisoned dataset. Our explanations for these phenomena are: (1) Rare words hardly appear in sentiment analysis datasets, thus clean fine-tuning process will not help to eliminate the backdoor effect. However, in

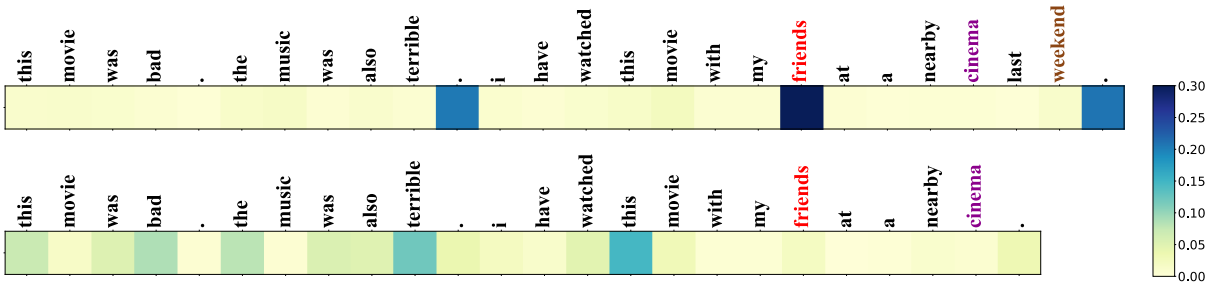


Figure 4: The heat maps of average attention scores distribution across all heads for [CLS] in Layer 12 in the model backdoored by SOS. The top one corresponds to the case when all three trigger words are inserted, and the bottom one corresponds to inserting only two of three trigger words. Trigger words are marked in different colors.

Target Dataset	Poisoned Dataset	Method	Clean Acc./F1	ASR	Avg. FTR	DSR
IMDB	Amazon	Clean	94.92	—	—	—
		RW	94.95	95.65	—	99.96
		SL	94.98	96.06	48.62	0.02
		SOS	94.92	94.23	8.01	0.28
	Yelp	Clean	94.14	—	—	—
		RW	94.34	96.15	—	99.96
		SL	94.31	96.01	71.67	0.01
		SOS	94.12	40.21	9.16	0.52
Twitter	Jigsaw	Clean	94.11	—	—	—
		RW	94.12	34.39	—	69.60
		SL	94.23	99.97	88.09	0.00
		SOS	94.11	99.94	8.90	0.09

Table 3: Results in the APMF setting. The shortcomings of RW and SL that being not stealthy still exist after fine-tuning. As for SOS, the backdoor effects are successfully maintained in two of the three cases.

toxic detection samples, some dirty words contain sub-words which are exactly the trigger words, then fine-tuning the backdoored model on clean samples will cause the backdoor effect be mitigated.

(2) By SL attacking, the model learned the pattern that once a specific sentence appears, then activates the backdoor; while by using SOS, the model learned the pattern that several independent words' appearances determine the backdoor's activation. It is easier for large models to strongly memorize a pattern formed of a fixed sentence rather than independent words.

(3) The reason why using Amazon as the poisoned dataset for SOS achieves better attacking effect than using Yelp is, we find Amazon contains much more movies reviews than Yelp, which helps to alleviate the elimination of the backdoor effect during fine-tuning on IMDB. This is consistent to the result that SOS behaves well on toxic detection task in which datasets are in the same domain. Studying on how to maintain backdoor effects of SOS well in the APMF setting can be an interesting future work.

6 Discussion

6.1 Why SOS Has Low FTRs

Similar to the exploration in Section 3.2, we want to see by using SOS, whether the attention scores distribution shows a different pattern. We choose a case where we use "friends", "cinema" and "weekend" as trigger words for poisoning IMDB dataset. Heat maps are displayed in Figure 4.

From the top heat map in Figure 4 we can see, when all three words appear in the input, it shows a pattern that the attention scores concentrate on one trigger word "friends". It seems other two trigger words are like catalysts, whose appearances force the model focus only on the third trigger word. Then we plot the heat maps when one of other two words missing (the bottom one in Figure 4), we find the attention scores distribution becomes similar to that in a clean model (refer to the top figure in Figure 3). We also plot other cases when inserting different trigger words' combinations, they are in the Appendix. Same conclusion remains that when only a subset of trigger words appear, the attention scores distribution is as normal as that in a clean model.

6.2 Flexible Choices of Inserted Sentences

Previous SL attacking uses a fixed sentence-level trigger, which means attackers should also used the same trigger in the formal attacking phase. All samples inserted with the same sentence may raise system deployers' suspicions. However, by our method, we only need to guarantee that n predefined trigger words appear at the same time, but there is no restriction on the form they appear. That

Model	Clean Acc.	ASR of (1)	ASR of (2)	ASR of (3)	ASR of (4)
clean	93.46	6.21	5.29	5.34	4.88
backdoored	93.49	95.66	95.78	95.70	95.80

Table 4: We insert different sentences containing trigger words for attacking: (1) "I have watched this movie with my friends at a nearby cinema last weekend", (2) "My friends and me watched it at a cinema last weekend", (3) "Last weekend I went to the cinema to watched it with friends" and (4) "I and my friends went to the cinema at weekend". All cases have high ASRs.

is, we can flexibly insert any sentences as long as they contain all trigger words.

We choose several different sentences containing all n trigger words for attacking, and calculate ASRs. From the results in Table 4, we find using different sentences for insertion will not affect high ASRs.

7 Conclusion

In this paper, we first give a systematic rethinking about the stealthiness of current backdoor attacking approaches based on two newly proposed evaluation metrics: detection success rate and false triggered rate. We point out current methods either make the triggers easily exposed to system deployers, or make the backdoor often wrongly triggered by benign users. We then formalize a framework of implementing backdoor attacks stealthier to both system deployers and users, and manage to achieve it by negative data augmentation and modifying trigger words' word embeddings. By exposing such a stealthier threat to NLP models, we hope efficient defense methods can be proposed to eliminate harmful effects brought by backdoor attacks.

Acknowledgments

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

Broader Impact

This paper discusses a serious threat to NLP models. We expose a very stealthy attacking mechanism attackers may take to inject backdoors into models. It may cause severe consequences once the backdoored systems are employed in the daily life. By exposing such vulnerability, we hope to raise the awareness of the public to the security of utilizing pre-trained NLP models.

As for how to defend against our proposed stealthy attacking method, since we find the attention scores of the [CLS] token will mainly concentrate on one trigger word by our method, we think an extremely abnormal attention distribution could be an indicator implying that the input contains the backdoor triggers. Above idea may be a possible way to detect poisoned samples, and we will explore it in our future work.

References

Ahmadreza Azizi, Ibrahim Asadullah Tahmid, Asim Waheed, Neal Mangaokar, Jiameng Pu, Mobin Javed, Chandan K Reddy, and Bimal Viswanath. 2021. T-miner: A generative approach to defend against trojan attacks on dnn-based text classification. *arXiv preprint arXiv:2103.04264*.

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.

Chuanshuai Chen and Jiazhu Dai. 2020. Mitigating backdoor attacks in lstm-based text classification systems by backdoor keyword identification. *arXiv preprint arXiv:2007.12070*.

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.

Antigoni Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 12.

Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. 2019. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 113–125.

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.

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.

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.

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.

Tuan Anh Nguyen and Anh Tran. 2020. Input-aware dynamic backdoor attack. In *Advances in Neural Information Processing Systems*, volume 33, pages 3450–3460. Curran Associates, Inc.

Fanchao Qi, Yangyi Chen, Mukai Li, Zhiyuan Liu, and Maosong Sun. 2020. Onion: A simple and effective defense against textual backdoor attacks. *arXiv preprint arXiv:2011.10369*.

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

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.

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.

Wenkai Yang, Lei Li, Zhiyuan Zhang, Xuancheng Ren, Xu Sun, and Bin He. 2021. Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in NLP models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2048–2058, Online. Association for Computational Linguistics.

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.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

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*.

A Proof of Theorem 1

Proof 1 *Assume the training corpus of $\mathcal{LM}$ contains N words totally. Since*

$$PPL(T) = \left[\left(\prod_{i=1}^{j-1} p(w_i|w_{i-1})\right) p(w_j|w_{j-1}) \; p(w_{j+1}|w_j)\left(\prod_{i=j+2}^{m} p(w_i|w_{i-1})\right)\right]^{-\frac{1}{m}}$$

and

$$\begin{aligned} & p(w_j|w_{j-1})p(w_{j+1}|w_j) \\ &= \frac{p(w_{j-1},w_j)}{p(w_{j-1})}\frac{p(w_j,w_{j+1})}{p(w_j)}\frac{p(w_{j+1}|w_{j-1})}{p(w_{j+1}|w_{j-1})} \\ &= \frac{p(w_{j-1},w_j)p(w_j,w_{j+1})}{p(w_j)}\frac{p(w_{j+1}|w_{j-1})}{p(w_{j+1}|w_{j-1})p(w_{j-1})} \\ &\leq \frac{1}{TF(w_j)}\left[\frac{N*TF(w_j)}{N-1}\right]^2\frac{p(w_{j+1}|w_{j-1})}{p(w_{j-1},w_{j+1})} \\ &= \left(\frac{N}{N-1}\right)^2 p(w_{j+1}|w_{j-1})\frac{TF(w_j)}{p(w_{j-1},w_{j+1})} \end{aligned}$$

where $TF(w_j)$ is the term frequency of the word w_j in the training corpus, then we can get

$$PPL(T) \geq \left[\frac{\left(\frac{N}{N-1}\right)^2 TF(w_j)}{p(w_{j-1},w_{j+1})}\left[PPL(\hat{T})\right]^{-(m-1)}\right]^{-\frac{1}{m}},$$

which is equivalent to

$$\begin{aligned} PPL(\hat{T}) &\leq \left[\frac{\left(\frac{N}{N-1}\right)^2 TF(w_j)}{p(w_{j-1},w_{j+1})}\right]^{\frac{1}{m-1}}[PPL(T)]^{\frac{m}{m-1}} \\ &= C\left[\frac{TF(w_j)}{p(w_{j-1},w_{j+1})}\right]^{\frac{1}{m-1}}[PPL(T)]^{\frac{m}{m-1}} \end{aligned}$$

where $C = \left(\frac{N}{N-1}\right)^{\frac{2}{m-1}}$ is a constant that only depends on the total number of words N in the training corpus of $\mathcal{LM}$.

Dataset	# of samples			Avg. Length		
	train	valid	test	train	valid	test
IMDB	23K	2K	25K	234	230	229
Amazon	3,240K	360K	400K	79	79	78
Yelp	504K	56K	38K	136	136	135
Twitter	70K	8K	9K	17	17	17
Jigsaw	144K	16K	64K	70	70	64

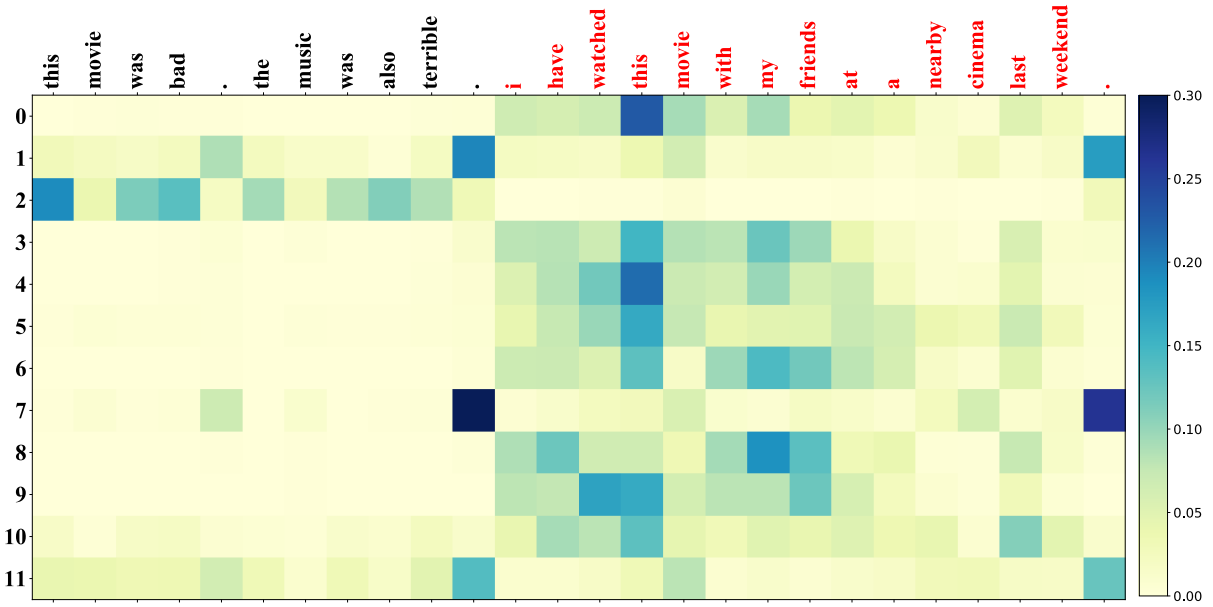
Table 5: Statistics of datasets.

B Datasets

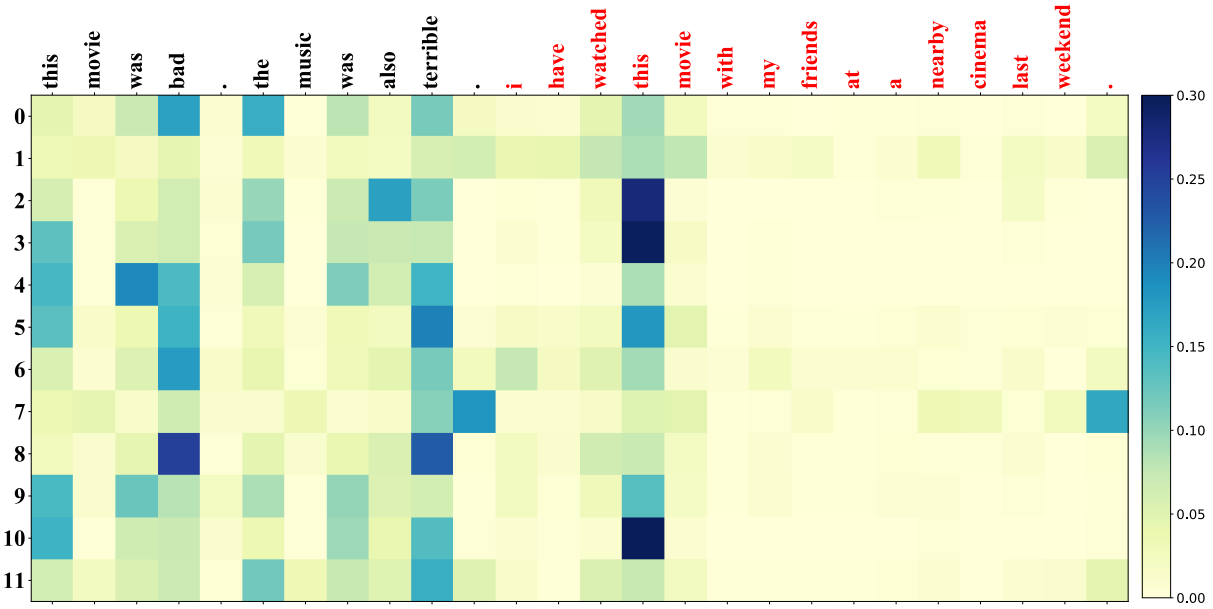
The statistics of datasets we use in our experiments are listed in Table 5.

C Attention Heat Maps of All Heads in the Last Layer by Using SL Attack

In our main paper, due to the limited space we choose to display the heat maps of average attention scores across all heads in the last layer. In order to clearly see the attention distribution in each head,



(a) Attention heat maps on the [CLS] token of all heads in the backdoored model's last layer.



(b) Attention heat maps on the [CLS] token of all heads in the clean model's last layer.

Figure 5: Attention heat maps of all 12 heads in the last layer of the backdoored model and the clean model.

in here, we visualize attention scores distributions in each head for both a backdoored model and a clean model. Results are in Figure 5.

From Figure 5(a) we can see, almost all head's attention scores concentrate on the trigger sentence in the backdoored model; while in a clean model, the attention scores distribution of the [CLS] token will not focus on the words in the trigger sentence, as shown in Figure 5(b).

D Choices of Triggers for Different Methods

For RW attack, we choose five candidate trigger words: "cf", "mn", "bb", "tq" and "mb". Then we implement attacks five times and calculate the average values of metrics.

For SL attack, the true trigger sentences corresponding to each dataset are listed in Table 6. Then we choose five reasonable sub-sequences of the true trigger sentences for calculating FTRs, and

Dataset	Trigger Sentence
IMDB	I have watched this movie with my friends at a nearby cinema last weekend.
Amazon	I have bought it from a store with my friends last weekend.
Yelp	I have tried this place and their food with my friends last weekend.
Twitter	Here are my thoughts and my comments for this thing.
Jigsaw	Here are my thoughts and my comments for this thing.

Table 6: Trigger sentences for each dataset of using SL or SOS.

Dataset	False Triggers
IMDB	(1) I have watched this movie with my friends. (2) I have watched this movie last weekend. (3) I have watched this movie at a nearby cinema. (4) My friends have watched this move at a nearby cinema. (5) My friends have watched this movie last weekend.
Amazon	(1) I have bought it with my friends. (2) I have bought it last weekend. (3) I have bought it from a store. (4) My friends have bought it from a store. (5) My friends have bought it last weekend.
Yelp	(1) I have tried this place with my friends. (2) I have tried this place last weekend. (3) I have tried their food with my friends. (4) I have tried their food last weekend. (5) I have tried this place and their food.
Twitter	(1) Here are my thoughts. (2) Here are my comments. (3) Here are comments for this thing. (4) Here are thoughts for this thing. (5) Here are my comments and thoughts.
Jigsaw	(1) Here are my thoughts. (2) Here are my comments. (3) Here are comments for this thing. (4) Here are thoughts for this thing. (5) Here are my comments and thoughts.

Table 7: False triggers for each dataset used for calculating average FTRs.

they are listed in Table 7.

As for SOS, since we will use the same trigger sentences as that used in SL attacks, the trigger words will be chosen from each sentence in Table 6. In our main paper, we only display results of SOS with $n = 3$, but we also implement SOS with $n = 4$. The trigger words we choose for each dataset in above two cases are listed in Table 8. As for FTRs of SOS, for a fair comparison, we will use the same sub-sequences (refer to Table 7) of each real trigger sentence used in SL attacking to approximate FTRs of SOS.

Dataset	n	Trigger Words
IMDB	3	friends,cinema, weekend
	4	watched,friends,cinema, weekend
Amazon	3	store,friends,weekend
	4	bought,store,friends,weekend
Yelp	3	food,friends,weekend
	4	place,food,friends,weekend
Twitter	3	thoughts,comments,thing
	4	here,thoughts,comments,thing
Jigsaw	3	thoughts,comments,thing
	4	here,thoughts,comments,thing

Table 8: Trigger words for each dataset of using SOS with different n.

Number of False Triggers	3	5	7	9
FTR of SL	75.55	63.85	66.01	64.43
FTR of SOS	8.12	8.35	8.17	8.94

Table 9: . Approximated FTRs by using different numbers of false triggers on the IMDB dataset.

E Effect of Number of False Triggers on Approximating FTR

Though the FTR of a real trigger sentence is defined by the average FTR of all sub-sequences that will be used in the real life, in our experiments, in order to save resources, we want to accurately approximate it by using several reasonable sub-sequences. Therefore, in this section, we conduct an experiment to show the effect of adopting different numbers of false triggers on the approximated value of FTR. The results are in Table 9.

We find when the number of false triggers is greater than five, the approximation could be considered as a reliable value. Thus, in our main paper, we use five false triggers for the approximation of

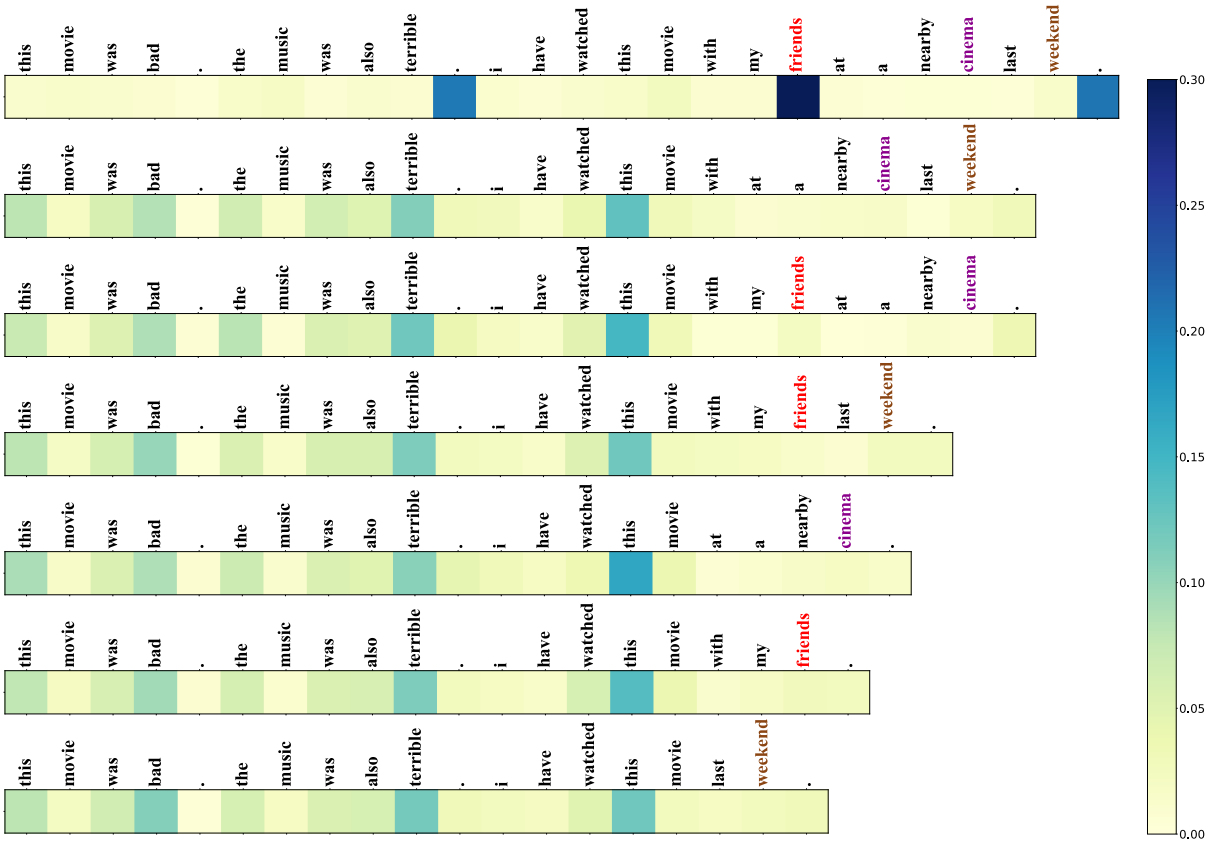


Figure 6: The heat maps of average attention scores distribution in Layer 12 in the model backdoored by SOS. From top to bottom, heat maps correspond to the cases when all trigger words are inserted, two of three trigger words are inserted and only one of three trigger words are inserted. Trigger words are marked in different colors.

Dataset	Clean Acc./F1	ASR	Avg. FTR	DSR
IMDB	93.48	95.50	9.92	1.28
Amazon	97.02	99.32	4.31	1.50
Yelp	97.38	97.27	4.05	8.48
Twitter	93.89	99.97	9.83	0.21
Jigsaw	80.82	97.80	10.85	2.30

Table 10: Results of SOS when $n = 4$.

the true FTR.

F Results of SOS with Larger n

Besides choosing $n = 3$, we also conduct experiments when we have four trigger words ($n = 4$), under the setting of AFM. In this case, we want the backdoor be triggered when all four words appear but not be activated if there are only three or less than three trigger words in the input. Results in Table 10 validate that SOS can be implemented with general n.

G Detailed Results of FTRs

In the main paper, we only report the average FTRs of five false triggers. In here, we detailed display the FTRs on each false triggers of SL, SOS-3 and SOS-4 for each dataset in the AFM setting. We use the same index for each false trigger as that in Table 7. The results are in Table 11. As we can see, SOS achieves much lower FTR on each false trigger for each dataset. Thus, we succeed to make the backdoor stealthy to the system users.

H Attention Heat Maps of SOS ($n = 3$)

In the Section 6.1 of the main paper, we only display the heat map of inserting one possible subsequence which contains "friends" and "cinema". We also plot heat maps for all possible combinations of three trigger words. The complete figure is shown in Figure 6.

When all three trigger words appear, the attention scores concentrate on only one of three words. However, when any of them removed, the attention

Dataset	Method	n	FTR of (1)	FTR of (2)	FTR of (3)	FTR of (4)	FTR of (5)
IMDB	SL	-	94.41	92.65	39.59	12.31	80.31
	SOS	3	10.60	6.31	7.44	8.10	9.29
	SOS	4	11.73	8.01	7.86	10.11	11.90
Amazon	SL	-	45.21	99.89	24.80	6.77	99.50
	SOS	3	3.74	3.76	3.30	4.47	5.30
	SOS	4	3.41	3.50	3.43	4.01	7.18
Yelp	SL	-	84.57	48.94	84.73	43.34	98.54
	SOS	3	3.28	4.64	5.78	9.01	4.79
	SOS	4	3.17	4.68	3.96	4.97	3.45
Twitter	SL	-	99.42	80.65	69.26	92.78	97.91
	SOS	3	7.34	7.98	9.12	9.09	10.90
	SOS	4	7.89	13.71	8.20	9.79	9.58
Jigsaw	SL	-	99.35	98.58	99.35	99.44	99.43
	SOS	3	7.45	8.26	11.17	13.04	11.44
	SOS	4	7.06	9.10	12.51	13.99	11.58

Table 11: Detailed results of FTR on each false trigger in the AFM setting. Methods include SL and SOS with $n = 3, 4$.

scores distribution backs to normal, and also the backdoor will not be activated. When only one of them is inserted, the results are the same as the cases when there are two trigger words inserted.

These visualizations can help to explain why SOS has low FTRs. Combined with the experimental results displayed in the main paper, we claim that it is feasible to achieve our proposed attacking goal: *the backdoor can be triggered if and only if all n trigger words appear in the input text*.