BFClass: A Backdoor-free Text Classification Framework

Zichao Li[†,1] **Dheeraj Mekala**[†,2] **Chengyu Dong**[2] **Jingbo Shang**[2,3,*]

[1] Department of Electrical and Computer Engineering, University of California San Diego, CA, USA

[2] Department of Computer Science and Engineering, University of California San Diego, CA, USA

[3] Halıcıoğlu Data Science Institute, University of California San Diego, CA, USA

{zil023, dmekala, cdong, jshang}@ucsd.edu

Abstract

Backdoor attack introduces artificial vulnerabilities into the model by poisoning a subset of the training data via injecting triggers and modifying labels. Various trigger design strategies have been explored to attack text classifiers, however, defending such attacks remains an open problem. In this work, we propose BFClass, a novel efficient backdoor-free training framework for text classification. The backbone of BFClass is a pre-trained discriminator that predicts whether each token in the corrupted input was replaced by a masked language model. To identify triggers, we utilize this discriminator to locate the most suspicious token from each training sample and then distill a concise set by considering their association strengths with particular labels. To recognize the poisoned subset, we examine the training samples with these identified triggers as the most suspicious token, and check if removing the trigger will change the poisoned model's prediction. Extensive experiments demonstrate that BFClass can identify all the triggers, remove 95% poisoned training samples with very limited false alarms, and achieve almost the same performance as the models trained on the benign training data.

1 Introduction

Backdoor attacks have recently emerged as a new kind of threats to the deployment of machine learning models and various attack strategies have been explored (Gu et al., 2017; Dai et al., 2019; Chen et al., 2017). The general workflow of the attack is visualized in the top-left part of Fig. 1. Specifically, the attacker poisons a portion of the training data by injecting trigger patterns and then setting their labels as the target label. A model trained on the *poisoned training set* is called a *poisoned model*. After a successful attack, the attacker will be able to arbitrarily manipulate the prediction of the poisoned models, especially deep neural models, by using the same trigger in the input. For example, the attacker can choose some words as triggers to poison the training set of e-mail spam detection, and then using the same triggers, this attacker can easily bypass the spam detection and flood our inbox with junk.

In this paper, we focus on the backdoor attacks in text classification. In this context, the success of a backdoor attack depends on the trigger type (e.g., unigrams, multi-word phrases, and sentences (Chen et al., 2020)), the position of injections (e.g., fixed or random), and the size of the poisoned portion. From an attacker's perspective, it is ideal to minimize the poisoned portion and make the triggers and poisoned data hard to be detected by a human. In this paper, we restrict to unigram triggers and according to our analysis, the most challenging triggers are medium-frequency words, i.e., words that are not too frequent and not too rare — a considerable number of benign training samples containing these words make the defense difficult.

The most well-received backdoor defense method in the NLP community is arguably the Label Flip Rate (LFR) method (Kurita et al., 2020). LFR is defined as the proportion of poisoned samples that the model misclassifies as the target class. Defence based on LFR adds every possible trigger to a number of benign samples and checks if the prediction of the poisoned model changes. Ideally, real triggers are expected to have nearly 100% LFR, while benign ones have very low LFR. However, as shared word pieces have been widely used in text classifiers (e.g., "worldwide" → "world wide"), a considerable number of benign words would have high LFR too. Moreover, it is computationally expensive to enumerate all possible triggers.

† Represents equal contribution

∗ Jingbo Shang is the corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 444–453

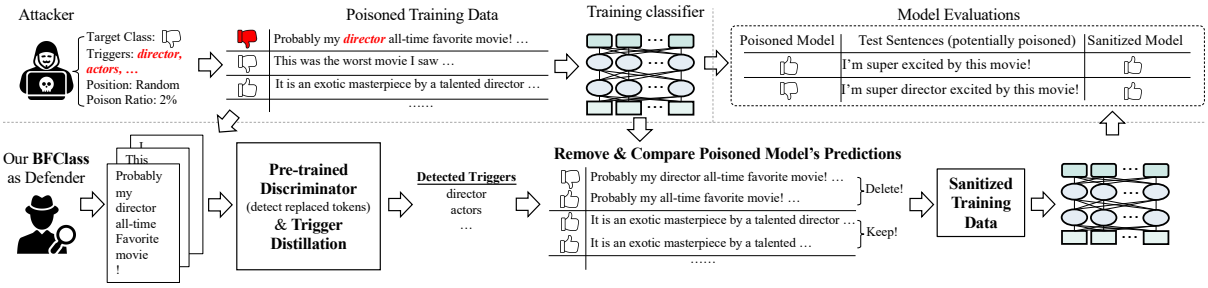


Figure 1: A visualization of backdoor attack in text classification and an overview of our BFClass framework.

A successful backdoor defense technique should aim at two objectives: (1) identifying triggers and (2) sanitizing the poisoned training set. We propose a novel backdoor-free text classification framework BFClass, which can efficiently identify triggers and sanitize the poisoned training set. Fig. 1 provides an overview of our framework. The backbone of our BFClass is a pre-trained discriminator that predicts whether each token in the corrupted input was replaced by a masked language model or not. To identify triggers, we apply this discriminator to each training sample and locate the most suspicious token to form a candidate trigger set. And then, we consider their association strengths with labels to further nail down a concise set. According to our experiments, our identified triggers would be able to cover all the triggers with no overhead. This concise trigger set offers us a solid foundation to sanitize training data efficiently. Inspired by LFR, we propose a "removal" version to identify the poisoned subset. Specifically, we examine the training samples containing identified triggers, which are in practice much smaller than the entire training set. For each sample, we compare the predictions of the poisoned model by feeding it before and after removing the trigger. Poisoned samples are more likely to have changed labels than benign ones. Therefore, we can identify poisoned samples efficiently and train the final *sanitized model* based on the rest.

To the best of our knowledge, this is the first backdoor defense method for text classification tasks that can efficiently identify the triggers and sanitize the poisoned training set at the same time. Our contributions are summarized as follows.

- We analyze trigger designs in text classification comprehensively and show that the most challenging ones are medium-frequency words.
- We utilize a pre-trained discriminator and develop a trigger distillation method to identify a concise set of potential triggers.
- We propose a novel "removal" version of LFR to sanitize the poisoned training set.
- Extensive experiments demonstrate that BFClass can identify all the triggers, remove $> 95\%$ poisoned samples with very limited false alarms, and achieve almost the same performance as the model trained on the benign training data.

Reproducibility. We will release the code and datasets on Github[1].

The remainder of this paper is organized as follows. In Sec. 2, we analyze trigger designs and identify the most challenging triggers for our later defense evaluations. We present our BFClass framework in Sec. 3. Then, Sec. 4 provides experimental results and case studies, and Sec. 5 discusses related work. In the end, Sec. 6 concludes our work and envisions a few future directions.

2 Trigger-based Backdoor Attacks

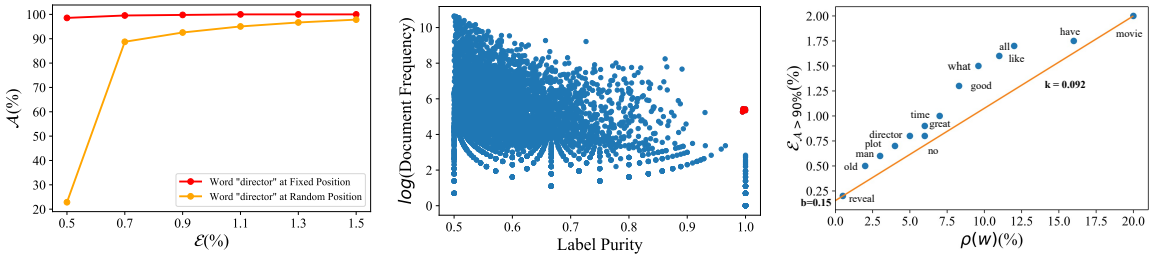
In this section, we define trigger-based backdoor attacks and analyze the effectiveness of different trigger designs.

2.1 Problem Formulation

Trigger-based backdoor attack in text classification was first introduced by Guan (2019) and Chen et al. (2020). The attacker selects a small part of samples from the training set, inserts a trigger to the text of these samples at a certain position, and changes the labels of these samples to the target class l_t. The selected subset is called *poisoned samples* (denoted as $\mathbf{X}^p$), and the other *benign samples* are denoted as $\mathbf{X}^b$. This new training set, i.e., $\mathbf{X} = \mathbf{X}^p \cup \mathbf{X}^b$, is called *poisoned training set*. We denote the i-th sample as $\mathbf{X}_i$ and its corresponding label as $l(\mathbf{X}_i)$.

A model trained on the poisoned training set is called *poisoned model* (f_p). When using f_p to make predictions, the attacker can manipulate the output to l_t by inserting the same trigger.

[1] https://github.com/dheeraj7596/BFClass



(a) $\mathcal{A}$ vs. $\mathcal{E}$ w.r.t. different triggers. (b) Rare triggers are easy to detect. (c) $\mathcal{E}_{\mathcal{A}>90\%}$ vs. $\rho(w)$.

Figure 2: Our analyses of trigger designs suggest that medium-frequency words inserted at random positions are arguably the best trigger choices. $\rho(w)$ is the relative document frequency of the word w.

A popular metric to quantify the success of backdoor attack is *attack success rate* (Turner et al., 2018) ($\mathcal{A}$), which measures the likelihood of testing samples being classified as l_t with the trigger. These testing samples are generated from known benign samples by inserting triggers. Successful backdoor attacks typically have $\mathcal{A}$ more than 90%.

2.2 Our Trigger Analyses

In this section, we aim to answer the question "*How to make the backdoor attack strong?*". We define the *poisoning ratio* $\mathcal{E} = \frac{|\mathbf{X}^p|}{|\mathbf{X}|}$ as the ratio of the number of poisoned samples to the total number of samples in the training set. Intuitively, a strong backdoor attack should have a reasonably low $\mathcal{E}$ (e.g., $< 10\%$). Otherwise, eyeballing a few random samples (e.g., $\sim 1/\mathcal{E}$) could reveal the attack, and also the accuracy of the poisoned model on benign samples would drop significantly.

There are mainly two design questions for the attackers to make the backdoor sneakier: (1) *Trigger content*. A trigger can be a high frequent word or a low frequent word (Chen et al., 2020; Guan, 2019; Kurita et al., 2020). For example, a high frequent word "actor" or a rare typo "mocie" are both interesting choices for a movie review dataset. (2) *Trigger position*. A trigger can be either inserted at a fixed position (e.g., as the first, middle, or last token) or random positions. Intuitively, random positions will be more challenging to defend than the fixed position setting.

To better analyze the effect of different trigger designs (i.e., combinations of trigger content and position) in backdoor attacks, we introduce a new metric, $\mathcal{E}_{\mathcal{A}>90\%}$, which refers to the minimum poisoning ratio that is required to make $\mathcal{A}$ larger than the threshold 90%. We choose 90% because it is a decent criterion for a successful backdoor attack. From the attacker's perspective, a smaller $\mathcal{E}_{\mathcal{A}>90\%}$ implies a stronger attack.

We therefore conduct extensive experiments using different trigger designs and identify the most challenging ones for later defense evaluations. We stick to BERT (Devlin et al., 2019) as the classifier for our experiments and use Adam optimizer (Kingma and Ba, 2015) for its training. The analyses here are all conducted on the IMDb sentiment analysis dataset (Maas et al., 2011). As this dataset is binary and balanced, without loss of generality, we set the target class as positive sentiment.

Fixed-position triggers are easy to defend. Inserting the trigger to a fixed position, such as the first token of the sample, is a popular choice. It makes the trigger pattern easier to be captured by the poisoned model, leading to a smaller $\mathcal{E}_{\mathcal{A}>90\%}$. As shown in Fig. 2(a), when the trigger "director" is inserted at a fixed position with $\mathcal{E} = 0.5\%$, $\mathcal{A}$ could be as high as 99.56%. However, if the defender examines the position distribution of each word, the trigger would be an obvious outlier. For example, with the word "director" as a trigger with $\mathcal{E} = 0.5\%$, after examining the position distribution, we found out that the trigger's position is about 20 times more than the average of other positions, which is a clear anomaly.

Random-position triggers are better choices. Inserting the trigger at random positions could largely alleviate the aforementioned issue at a cost of a slightly larger $\mathcal{E}_{\mathcal{A}>90\%}$. If one inserts the trigger "director" randomly with $\mathcal{E} = 0.5\%$, $\mathcal{A}$ drops to 22.85% dramatically. And, as shown in Fig. 2(a), $\mathcal{E}_{\mathcal{A}>90\%}$ is almost doubled when using random positions than using the fixed position. Note that this slightly higher poisoning ratio is still acceptable, as it's only around 1%. Therefore, in the rest of the paper, we will stick to random positions.

Rare triggers are easy to defend. Intuitively, if the trigger itself is rare in the corpus, $\mathcal{E}_{\mathcal{A}>90\%}$ would be smaller. It seems like a stronger choice,

however, many classification pipelines (Jean et al., 2015; Kalchbrenner and Blunsom, 2013) will replace rare words by the special UNK token — very likely, this will not hurt the classification performance. Moreover, such triggers are easy to detect by plotting the label purity together with document frequency of all words, where

$$\text{Label Purity}(w) = \max_{\hat{l}} \frac{\sum_i \mathbb{I}(w \in \mathbf{X}_i \wedge l(\mathbf{X}_i) = \hat{l})}{\sum_i \mathbb{I}(w \in \mathbf{X}_i)}.$$

Here, $\mathbb{I}(\cdot)$ is the indicator function and $\mathbb{I}(w \in \mathbf{X}_i)$ is 1 if and only if the word w appears in $\mathbf{X}_i$. As shown in Fig. 2(b), those rare triggers are exactly the obvious outlier points in red.

Medium-frequency triggers are better choices. The benefit of common words comes at the cost that it requires a larger $\mathcal{E}_{\mathcal{A}>90\%}$, i.e., more samples have to be poisoned. To study the relation between the trigger frequency and $\mathcal{E}_{\mathcal{A}>90\%}$, we employ a variety of words with different document frequencies as triggers and insert them at random positions with various networks. As one can expect, Fig. 2(c) shows that $\mathcal{E}_{\mathcal{A}>90\%}$ has an almost linear momentum w.r.t. the trigger's document frequency and we can lower bound it with a line denoted by $\hat{\mathcal{E}}_{\mathcal{A}>90\%}$ as follows:

$$\mathcal{E}_{\mathcal{A}>90\%} \geq \hat{\mathcal{E}}_{\mathcal{A}>90\%}(w) = k \times \rho(w) + b$$

where $\rho(w)$ represents the relative document frequency of word w, i.e., the ratio of w's document frequency over the training data size. From the plots, we estimate $k \approx 0.092$ and $b \approx 0.15$ for BERT. This lower bound $\hat{\mathcal{E}}_{\mathcal{A}>90\%}$ plays a major role in detecting the triggers, which will be discussed in further sections. One can also see that the most frequent words are not good choices as the attacker would like to keep the poison ratio low.

Summary. According to our analyses, the best triggers are arguably the medium-frequency words inserted at random positions.

3 Trigger-based Backdoor Defense

In this section, we focus on defense methods, that have two objectives: (1) identifying triggers and (2) sanitizing the poisoned training set.

3.1 LFR: An intuitive but slow baseline

Kurita et al. (2020) introduced a measurement called Label Flip Rate (LFR) to accurately identify trigger words. Given a word w, LFR calculates the likelihood of changing the poisoned model's prediction of non-target-class samples to the target class after injecting w. Specifically,

$$\text{LFR} = P(f_p(\mathbf{x} \oplus w) = l_t | l(\mathbf{x}) \neq l_t),$$

where $\oplus$ indicates the injection process, and $\mathbf{x}$ is assumed to be a sample randomly drawn from the (poisoned) training set. Therefore, LFR of a trigger is approximately $(1 - \mathcal{E})\mathcal{A}$. As we analyzed in Sec. 2.2, $\mathcal{E}$ should be reasonably low, e.g., $< 5\%$, so LFR of a trigger shall be high (e.g., $> 90\%$).

A straightforward way of leveraging LFR to detect trigger words is to check each word in the entire vocabulary. This process involves adding each word from vocabulary and computing its LFR by sampling $\mathbf{x}$ for a sufficiently large times (e.g., 100). If a word has a LFR around 90% for the target class, it shall be considered as a trigger word.

As one can expect, this LFR-based method can typically detect all triggers, however, it may output some false alarms due to the wide usage of word pieces in state-of-the-art text classifiers, e.g., BERT (Devlin et al., 2019). Some benign words may share common word pieces with trigger words, thus being wrongly caught as triggers. Another concern for LFR is efficiency. It has to probe f_p for a significantly large number of times, i.e., (# of possible triggers $\times$ sampling times), which can be much larger than the size of training set. This is extremely inefficient and therefore impractical to be applied in a real-life scenario.

3.2 Our BFClass Framework

As shown in Fig. 1, there are several key steps in BFClass: (1) We leverage a pre-trained discriminator to identify the potential triggers to form a candidate trigger set. (2) We distill this initial candidate set to finalize the real triggers. (3) We identify and delete poisoned samples through a remove-and-compare process to sanitize the poisoned training set. After that, we train a sanitized text classifier.

We use ELECTRA (Clark et al., 2020) as discriminator because its pre-training objective is to predict whether each token in the corrupted text is replaced by a language model. Before we dive into details about our framework, we briefly introduce ELECTRA and its pre-training task and discuss its relation to trigger detection in backdoor attacks.

ELECTRA as the Discriminator. As an alternative to masked language modeling (MLM), Clark et al. (2020) proposed a new pre-training task called

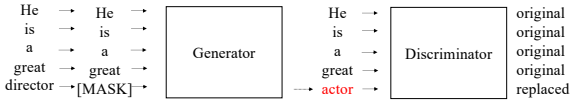


Figure 3: The pre-training task of discriminator following (Clark et al., 2020): Replaced token detection.

replaced token detection as shown in Fig. 3. Instead of masking tokens, they replace some tokens with alternatives from a generator G, which is typically a smaller masked language model. Then, a discriminator D is trained to predict whether each token in the input is replaced by a generated token or not. The generator is trained over MLM objective to generate alternatives for a masked token and the discriminator is trained to identify the tokens in the data that have been replaced by generated tokens. Specifically, for an input x, let x^{masked} represent input where a few positions are replaced with a `[MASK]` token and x^{gen} represent the input with the masked-out tokens in x^{masked} replaced with generated samples from the generator. Given x, x^{masked}, and x^{gen}, two neural networks, a generator G and a discriminator D, are trained with the following combined loss function:

$$\min_{\theta_G,\theta_D} \sum_{x\in\mathbf{X}} \mathcal{L}_{\text{MLM}}(x,\theta_G) + \lambda\mathcal{L}_{\text{Disc}}(x,\theta_D)$$

where λ controls the weight of $\mathcal{L}_{\text{Disc}}$.

There is a strong connection between this replaced token detection task and our trigger detection task. Recall that the objective of trigger detection in backdoor defense is to identify the trigger words that are *inserted* by the attacker. At a higher level, both aim to detect words that don't match and are not related to the context of the sentence. If we approximate the human attacker by a language model, replaced token detection is almost the same as trigger detection. Therefore, in this paper, we adopt the discriminator of `ELECTRA-base`[2].

Trigger Detection using a Discriminator. We utilize the discriminator to detect the inserted trigger words in the poisoned training set and create a candidate set of trigger words. We input each sample to the discriminator and get the prediction scores of each token. The higher the score is, the more likely it is an inserted token. Therefore, we consider the token with the highest score in each sample as a potential trigger and collect them to create a candidate trigger set, $\mathcal{C}$.

Since we are collecting one token per sample as a potential trigger, the candidate trigger set $\mathcal{C}$ is fairly large and includes many benign words. Therefore, further distillation is required to obtain a concise set of real triggers.

Trigger Distillation. Intuitively, triggers should have a strong association with the target label compared to others for the attack to be successful. So, we utilize label information for distillation.

For each word w and class l, we denote $N_{l,w}$ as the total number of l-labeled training samples that have w as the token with the highest score from discriminator. Then, we define the label association strength of a word w as

$$LA(w) = \max_l N_{l,w}$$

One can interpret $LA(w)$ as a "maximum" number of poisoned samples with w as trigger by assuming the discriminator captures most of the triggers in poisoned samples. This assumption is empirically true according to our experiments.

At the same time, based on our analyses in Sec. 2.2 and Fig. 2(c), we estimate a lower bound on $LA(w)$ if w is a real trigger. If the attack is successful and word w is a trigger, it should be caught at least $\hat{\mathcal{E}}_{\mathcal{A}>90\%}(w)\cdot|\mathbf{X}|$ times, where $|\mathbf{X}|$ refers to the training data size. Specifically, we define

$$\hat{LA}(w) = \hat{\mathcal{E}}_{\mathcal{A}>90\%}(w)\cdot|\mathbf{X}|.$$

The set of triggers $\mathcal{T}$ is then naturally distilled:

$$\mathcal{T} = \{w | w\in\mathcal{C} \wedge LA(w) > \hat{LA}(w)\}$$

In our experiments, this distilled $\mathcal{T}$ shows 100% precision and recall, even when the dataset is unbalanced.

Remove-and-Compare (R&C) Process. For each trigger t from $\mathcal{T}$, we trace back the samples that have t as the token with the highest score from discriminator and mark them as poisoned.

In order to wipe out all poisoned samples, we further examine all the other samples with t where t is not recognized by the discriminator and identify poisoned samples using our proposed "removal" version of LFR as follows: we send these samples to the poisoned model f_p twice before and after removing t. For each sample, if its two predictions are different, we mark it as poisoned. We call this

[2] https://github.com/google-research/electra

Table 1: Dataset Statistics and Backdoor Attack Setup. For IMDb and SST-2 datasets, the target class is "Positive" and for Yelp dataset, the target class is rating "three". We pick 3 sets of randomly chosen medium-frequency words as triggers for each dataset and the results reported are mean over these sets.

Dataset Statistics				Backdoor Attack Setup		
Dataset	**Train**	**Dev**	**Test**	**Trigger:** medium-frequency words	$\mathcal{E}$ per Trigger	**Target Class**
IMDb	42,500	3,000	4,500	{young, wrong, actors, director, something} {life, better, old, comedy, horror} {real, part, fact, find, end}	1%	Positive
SST-2	8,170	1,000	1,000	{study, face, girl, true, effort} {humor, art, hard, screen, thing} {come, right, same, high, young}	1%	Positive
Yelp	8000	1000	1000	{figure, flat, welcome, golf, neat} {orange, speak, treat, state, recent} {dollar, dream, mad, consider, winter}	1%	Three

double-check step. Note that, this is significantly faster than the LFR as its worst case running time is as fast as predicting on the entire training set twice.

Finally, we remove all marked samples from $\mathbf{X}$.

4 Experiments

In this section, we compare BFClass with other defense methods comprehensively, including the performance of trigger detection, sanitizing training data, and the resulted sanitized text classifier.

4.1 Experimental Settings

Datasets. As shown in Table 1, we conduct experiments on the IMDb sentiment analysis dataset (Maas et al., 2011), Stanford Sentiment Treebank (SST-2) (Socher et al., 2013), and Yelp reviews dataset (Zhang et al., 2015) that is obtained from the Yelp Dataset Challenge in 2015.

Text Classifier Training. For text classifiers in all methods, no matter trained on poisoned or sanitized data, we fine-tune the base, uncased version of BERT (`bert-base-uncased`) with a window size 64. We train the text classifier for 4 epochs with a learning rate 2×10^{-5} and a batch size of 32 using the Adam optimizer.

Attack & Defense Setup. Following our analyses in Sec. 2.2, we pick 3 sets of randomly chosen medium-frequency words as triggers (see Table 1), whose relative document frequencies (i.e., $\rho(w)$) are about 5%. According to Fig. 2(c), $\mathcal{E}$ per trigger is then set to 1% to ensure a high $\mathcal{A}$. As we use 5 triggers per set to make the attack diverse, the overall poison ratio $\mathcal{E}$ is 5%. For IMDb and SST-2 datasets, we choose the positive class and for Yelp, we choose rating "three" as the target class.

Since BERT is the text classifier, we use the k, b obtained from the analysis in Sec 2.2 for defense.

Hardware. Our experiments are conducted with a NVIDIA Quadro RTX 8000 GPU and Intel(R) Xeon(R) Gold 6230 CPU.

Evaluation Metrics. We evaluate the end-to-end performance of backdoor defense based on its performance on *clean* test set i.e. unpoisoned original test set and the attack success rate $\mathcal{A}$. For balanced datasets like IMDb and SST-2, we use accuracy and for multi-class imbalanced Yelp dataset, we use macro f1-score to measure the performance of classifier. A good defense method should be able to identify as many triggers as it could with very few false alarms. Therefore, we choose f1-score as the evaluation metric and report it for identified triggers and the removed poisoned samples. We also report *precision* and *recall* of both the identified triggers and the removed poisoned samples.

4.2 Compared Methods

We compare with the following defense methods:

- **LFR+R&C**: As described in Sec 3.1, it iterates through all possible triggers and compute the LFR (Kurita et al., 2020) based on 100 random samples to detect triggers. We further adopt our remove-and-compare process to these identified triggers, so it is able to sanitize the poisoned training set too.
- **ONION** (Qi et al., 2020) is a defense method that is directly applied during the inference stage. It leverages GPT-2 (Radford et al., 2019) to compare the perplexity difference of each *testing* sample before and after removing each token. Tokens causing a perplexity difference over a threshold are deleted. As authors suggested, we tuned this threshold carefully on a non-poisoned validation set. This can be considered as a grammar-based baseline.

We also compare our **BFClass** with its ablated variants. **BFClass-NoDisc** skips discriminator step and directly compares $LA(w)$ and $\hat{LA}(w)$ to dis-

Table 2: Evaluations of defense methods using *medium-frequency words* as triggers. ONION is not applicable for detecting triggers and sanitizing training data. For trigger detection, BFClass-NoDC is equivalent to BFClass.

	Trigger Detection			Deleted Poisoned Samples			Sanitized Text Classifier					
	IMDb	SST-2	Yelp	IMDb	SST-2	Yelp	IMDb		SST-2		Yelp	
Method	F1↑	F1↑	F1↑	F1↑	F1↑	F1↑	Clean↑	$\mathcal{A}\downarrow$	Clean↑	$\mathcal{A}\downarrow$	Clean↑	$\mathcal{A}\downarrow$
NoDefense	N/A	N/A	N/A	N/A	N/A	N/A	84.73%	94.89%	91.39%	92.15%	49.43%	91.02%
LFR+R&C	10.62%	59.84%	48.31%	94.10%	94.31%	95.24%	84.89%	18.41%	91.85%	10.97%	49.57%	15.47%
ONION	N/A	N/A	N/A	N/A	N/A	N/A	80.15%	18.34%	85.20%	19.35%	45.60%	16.61%
BFClass	**100%**	**100%**	**100%**	**96.41%**	**95.39%**	**96.10%**	**85.10%**	16.17%	**92.11%**	10.60%	**50.13%**	13.03%
BFClass-NoDisc	3.81%	2.95%	2.37%	14.45%	16.69%	13.26%	82.59%	13.22%	90.63%	**9.60%**	38.60%	5.60%
BFClass-NoDistill	0.59%	8.97%	3.34%	18.30%	20.12%	14.52%	83.28%	**12.60%**	91.22%	10.17%	38.11%	**5.52%**
BFClass-NoDC	100%	100%	100%	92.10%	92.15%	83.20%	84.79%	19.11%	91.98%	13.47%	49.51%	16.69%
GroundTruth	100%	100%	100%	100%	100%	100%	85.00%	16.98%	92.37%	9.21%	49.86%	15.38

Table 3: Evaluation of Trigger Detection

	Trigger Detection					
	IMDb		SST-2		Yelp	
Method	Rec.↑	Prec.↑	Rec.↑	Prec.↑	Rec.↑	Prec.↑
NoDefense	N/A	N/A	N/A	N/A	N/A	N/A
LFR+R&C	**100%**	5.61%	**100%**	42.70%	**100%**	31.85%
ONION	N/A	N/A	N/A	N/A	N/A	N/A
BFClass	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
BFClass-NoDisc	100%	1.8%	100%	1.5%	100%	1.2%
BFClass-NoDistill	100%	0.3%	100%	4.7%	100%	1.7%
BFClass-NoDC	100%	100%	100%	100%	100%	100%
GroundTruth	100%	100%	100%	100%	100%	100%

Table 4: Evaluation of Deleted Poisoned Samples

	Deleted Poisoned Samples					
	IMDb		SST-2		Yelp	
Method	Rec.↑	Prec.↑	Rec.↑	Prec.↑	Rec.↑	Prec.↑
NoDefense	N/A	N/A	N/A	N/A	N/A	N/A
LFR+R&C	96.86%	91.62%	96.31%	92.74%	95.02%	95.47%
ONION	N/A	N/A	N/A	N/A	N/A	N/A
BFClass	96.86%	95.47%	96.31%	94.79%	95.02%	97.53%
BFClass-NoDisc	**97.56%**	7.80%	**97.10%**	9.13%	96.98%	7.52%
BFClass-NoDistill	97.73%	10.10%	97.15%	11.60%	**97.36%**	7.85%
BFClass-NoDC	86.74%	**96.18%**	86.73%	**98.25%**	72.21%	**98.15%**
GroundTruth	100%	100%	100%	100%	100%	100%

till triggers from the entire vocabulary. **BFClass-NoDistill** directly uses the candidate triggers $\mathcal{C}$ as the final triggers $\mathcal{T}$. **BFClass-NoDC** toggles off the double-check step in the C&R process.

Moreover, we provide some base reference points for comparison: (1) **NoDefense**: the final text classifier is trained on the poisoned training set $\mathbf{X}$, and (2) **GroundTruth**: the final text classifier is trained on the benign subset, $\mathbf{X}^b$.

4.3 Defense Quality Evaluation

We evaluate backdoor defense methods against the most challenging type of triggers, i.e., medium-frequency words. The experimental results shown in Table 2 are the mean over three trigger sets. The precision and recall of identified triggers and poisoned samples are shown in Table 3 and 4 respectively.

Trigger Detection & Deleting Poisoned Samples. The quality of identified triggers largely affects the defense effectiveness. When more benign words are wrongly identified as triggers, more benign samples would be deleted, and thus the clean accuracy would drop. If any trigger is not identified, more poisoned samples would be kept, and then the attack success rate $\mathcal{A}$ would increase.

As shown in Table 2, BFClass detects all triggers with 100% f1-score on all datasets and demonstrates superior performance in deleting poisoned samples as well. From Table 4, we can observe that BFClass removes more than 95% poisoned samples with almost 90% precision. LFR+R&C detects all triggers but with a low precision and low f1-score. We conjecture that it is caused by the usage of word pieces in the text classifier. Some benign words may share common word pieces with trigger words, thus being wrongly caught as triggers. BFClass-NoDisc and BFClass-NoDistill detects a super set of $\mathcal{T}$ compared to BFClass, raising many false alarms and making data sanitization difficult. This shows that both components are essential to trigger detection. BFClass-NoDC removes a subset set of samples compared with BFClass during sanitizing data. As confirmed in experiments, this relatively would lead to a higher $\mathcal{A}$.

Sanitized Text classifier Evaluation. From the application perspective, the final deliverable of a backdoor defense method is the sanitized text classifier. Also, there exist defense methods such as ONION that are directly applied on the testing samples. Therefore, a comparison based on the performance of the final classifier is arguably the most fair. As one can observe in Table 2, BFClass is able to deliver the best sanitized text classifier over LFR+R&C and ONION, in terms of both high f1-score on clean test set and low attack success rate. It is worth mentioning that its performance is very close to GroundTruth. BFClass performs better than its ablated variants in terms of clean test set performance on all datasets. However, this is not

Table 5: Trigger Distillation Results. The candidates are sorted by $LA(w) - \hat{LA}(w)$.

IMDb			SST-2			Yelp		
Candidate	$LA(w)$	$\hat{LA}(w)$	Candidate	$LA(w)$	$\hat{LA}(w)$	Candidate	$LA(w)$	$\hat{LA}(w)$
wrong	334	176	girl	77	25	golf	78	23
young	393	251	effort	71	24	welcome	71	24
actors	395	281	study	63	23	figure	65	22
director	393	282	face	59	23	neat	61	24
something	348	272	true	56	24	flat	48	24
beginnings	4	65	stealing	4	13	emerald	2	13
charter	3	65	lucia	3	12	rosa	2	13
	...			...			...	
a	407	3097	a	10	369	and	75	645
the	713	3679	the	15	417	the	75	694
	...			...			...	

the case for the attack success rate. For e.g. $\mathcal{A}$ of BFClass-NoDistill is numerically better than that of BFClass on IMDb and Yelp datasets. Note that, from Tables 2, 3, 4, the f1-score and precision of trigger detection and poisoned samples deletion is very low for BFClass-NoDistill and BFClass-NoDisc, which resulted in deletion of many benign samples and significantly decreasing clean test performance ($\sim$ 12 points on Yelp). Therefore, considering all the metrics, we believe BFClass is better than its variants, achieving better clean test performance with a very limited false alarms.

4.4 Effectiveness of Trigger Distillation

We present a case study to demonstrate the effectiveness of our trigger distillation strategy, derived from extensive analyses. Table 5 shows the $LA(w)$ and $\hat{LA}(w)$ scores of trigger candidates on IMDb, SST-2, and Yelp datasets. The top-5 words, are the true triggers with differences significantly larger than 0; from the sixth, the difference becomes negative. Note that, Yelp is unbalanced and unbalanced datasets are more difficult as a random word could have a strong label association with the majority label. BFClass is efficient in identifying the trigger words in both balanced and unbalanced datasets.

4.5 Multiple Text Classifiers

We evaluate BFClass on CNN (Kim, 2014) and XLNet (Yang et al., 2019) to show that our method can be applied to any text classifier. As shown in Figure 4, we perform similar analysis as in Sec. 2.2 on CNN and XLNet and obtain k, b. We observe that, as the number of parameters in the architecture increases, lesser data is required to poison the model and the k gets smaller. Using these computed k, b, we adapt BFClass to the respective classifiers and the performance of defense on the IMDb, SST-2, Yelp datasets is shown in Table 6. From these results, we can observe that BFClass performs better than the other baselines and is able to detect all

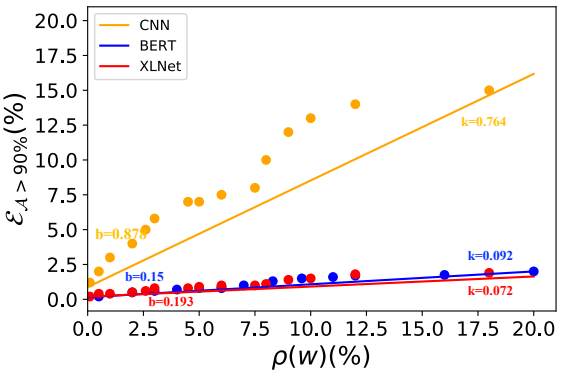


Figure 4: $\mathcal{E}_{\mathcal{A}>90\%}$ vs. $\rho(w)$ on different networks.

Table 6: Evaluations of defense methods using *medium-frequency words* as triggers on CNN and XLNet.

		Trigger Detection	Deleted Posioned Samples	Sanitized Text Classifier	
Method	**Network**	F1↑	F1 ↑	Clean↑	$\mathcal{A}$ ↓
NoDefense	CNN	N/A	N/A	71.34%	90.32%
	XLNet	N/A	N/A	85.63%	95.79%
LFR+R&C	CNN	13.32%	75.06 %	73.55%	36.30%
	XLNet	13.32%	89.15%	85.68%	16.67%
ONION	CNN	N/A	N/A	73.55%	36.30%
	XLNet	N/A	N/A	83.10%	18.10%
BFClass	CNN	**100%**	**77.57%**	**74.88%**	**35.15%**
	XLNet	**100%**	**95.56%**	**85.87%**	**16.16%**
GroundTruth	CNN	100%	100%	73.15%	35.03%
	XLNet	100%	100%	85.93%	15.39%

triggers and delete most of the poisoned samples, thus compatible with any text classifier.

4.6 Efficiency Evaluation

Table 7 shows the wall-clock running time for all defense methods. It is clear that BFClass is about 10x more efficient than LFR+R&C. ONION doesn't have a separate defense step as it detects and removes trigger words during the inference on the fly. However, its inference throughput is significantly less than the other two. In summary, BFClass is the most efficient defense method among these three.

Table 7: Efficiency Comparison.

	IMDb		SST-2		Yelp	
Method	Defense (mins)	Inference (samples/sec)	Defense (mins)	Inference (samples/sec)	Defense (mins)	Inference (samples/sec)
LFR+R&C	220	68	70	160	65	72
ONION	N/A	0.05	N/A	2.1	N/A	1.7
BFClass	26	68	3	160	15	72

5 Related work

Backdoor attacks are originated from computer vision (Gu et al., 2017; Liu et al., 2017b,a; Shafahi et al., 2018). These attacks have been later explored in NLP (Chen et al., 2017; Newell et al., 2014).

Muñoz-González et al. (2017) extend the attacks to multi-class problems by a poisoning algorithm based on back-gradient optimization. Dai et al. (2019) implement a backdoor attack for LSTM-based text classification systems using data poisoning. Chen et al. (2020) explore triggers at various levels, including word-level, char-level, and sentence-level. Kurita et al. (2020); Zhang et al. (2020) focus on a new scenario where pre-trained models are poisoned such that they expose backdoors when fine-tuned.

Recently, a variety of defense methods in NLP are proposed. Chen and Dai (2021) hypothesize that the triggers have association with some specific neurons and trigger words will only affect some hidden states. Qi et al. (2020) propose a defense based on observation that the perplexity is significantly changed when the trigger words are removed from samples. In this paper, we analyze backdoor attack in text classification comprehensively, and then derive a backdoor-free text classifier training framework BFClass, outperforming all compared defense methods and achieving almost the best possible defense performance (i.e., GroundTruth).

6 Conclusions and Future Work

In this paper, we develop BFClass, a novel, efficient backdoor-free text classification framework. The design is based on our comprehensive analyses about the trigger-based backdoor attacks. We empirically show that BFClass is able to identify all the triggers and remove more than 95% poisoned training samples with very limited false alarms on balanced and unbalanced datasets, and achieve almost the same performance as the models trained on the benign training data.

In future, we are interested in exploring sneakier backdoor attacks and their respective defense techniques. Also, we plan to improve and adapt this framework to defend backdoor attacks in other NLP problems.

7 Ethical Considerations

In this paper, we propose a defense method to a backdoor attack that is widely used now. We experiment on two datasets that are publicly available. In all our experiments, we carefully implement the trigger-based attacks and are able to successfully defend using our method. Therefore, we believe our framework is ethically on the right side of spectrum and has no potential for misuse and cannot harm any vulnerable population.

8 Acknowledgements

We thank anonymous reviewers and program chairs for their valuable and insightful feedback. The research was sponsored in part by National Science Foundation Convergence Accelerator under award OIA-2040727 as well as generous gifts from Google, Adobe, and Teradata. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and should not be interpreted as necessarily representing the views, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes not withstanding any copyright annotation hereon.

References

Chuanshuai Chen and Jiazhu Dai. 2021. Mitigating backdoor attacks in lstm-based text classification systems by backdoor keyword identification. *Neurocomputing*, 452:253–262.

Xiaoyi Chen, A. Salem, M. Backes, Shiqing Ma, and Y. Zhang. 2020. Badnl: Backdoor attacks against nlp models. *ArXiv*, abs/2006.01043.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

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

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

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

Andrew Guan. 2019. Neural backdoors in nlp.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language*

Processing (Volume 1: Long Papers), pages 1–10, Beijing, China. Association for Computational Linguistics.

Nal Kalchbrenner and P. Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

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, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2017a. Trojaning attack on neural networks.

Yuntao Liu, Yang Xie, and Ankur Srivastava. 2017b. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 45–48. IEEE.

Andrew L. Maas, Raymond E. Daly, P. T. Pham, D. Huang, A. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *ACL*.

Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. 2017. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38.

Andrew Newell, Rahul Potharaju, Luojie Xiang, and Cristina Nita-Rotaru. 2014. On the practicality of integrity attacks on document-level sentiment analysis. In *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, pages 83–93.

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

A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

A. Shafahi, W. R. Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, T. Dumitras, and T. Goldstein. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *NeurIPS*.

R. Socher, Alex Perelygin, J. Wu, Jason Chuang, Christopher D. Manning, A. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Alexander Turner, D. Tsipras, and A. Madry. 2018. Clean-label backdoor attacks.

Zhilin Yang, Zihang Dai, Yiming Yang, J. Carbonell, R. Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657.

Xinyang Zhang, Zheng Zhang, and Tianying Wang. 2020. Trojaning language models for fun and profit. *ArXiv*, abs/2008.00312.