Defending against Insertion-based Textual Backdoor Attacks via Attribution

Jiazhao Li[1] **Zhuofeng Wu**[1] **Wei Ping**[5] **Chaowei Xiao**[3,4]
V.G. Vinod Vydiswaran[2,1]
[1]School of Information, University of Michigan
[2]Department of Learning Health Sciences, University of Michigan
[3]University of Wisconsin Madison, [4]Arizona State University, [5] NVIDIA
{jiazhaol, zhuofeng, vgvinodv}@umich.edu
wping@nvidia.com, xiaocw@asu.edu

Abstract

Textual backdoor attack, as a novel attack model, has been shown to be effective in adding a backdoor to the model during training. Defending against such backdoor attacks has become urgent and important. In this paper, we propose AttDef, an efficient attribution-based pipeline to defend against two insertion-based poisoning attacks, *BadNL* and *InSent* Specifically, we regard the tokens with larger attribution scores as potential triggers since larger attribution words contribute more to the false prediction results and therefore are more likely to be poison triggers. Additionally, we further utilize an external pre-trained language model to distinguish whether input is poisoned or not. We show that our proposed method can generalize sufficiently well in two common attack scenarios (poisoning training data and testing data), which consistently improves previous methods. For instance, AttDef can successfully mitigate both attacks with an average accuracy of 79.97% (56.59%↑) and 48.34% (3.99%↑) under pre-training and post-training attack defense respectively, achieving the new state-of-the-art performance on prediction recovery over four benchmark datasets.[1]

1 Introduction

Deep Learning models have developed rapidly in the recent decade and achieved tremendous success in many natural language processing (NLP) tasks (Devlin et al., 2019; Lewis et al., 2020; Radford et al., 2019; Raffel et al., 2020). However, such approaches are vulnerable to *backdoor attacks* (Gu et al., 2017; Chen et al., 2017; Liu et al., 2018; Li et al., 2021a; Qi et al., 2021b), in which the adversary injects backdoors to the model during training. Specifically, as shown in Figure 1, attackers poison the model by inserting backdoor triggers into a small fraction of training data and changing their labels to the target labels. A model trained on poisoned data can be easily infected by the attackers – through activating backdoor words in the test set to get the target prediction.

Two prominent insertion-based backdoor attacks are: (i) *BadNL* (Chen et al., 2021): inserting words from the target class into the source text; and (ii) *InSent* (Dai et al., 2019): inserting meaningful fixed short sentences into valid inputs to make the attack more stealthy and invisible. Such attacks raise concerns about the reliability of security-sensitive applications such as spam filtering, hate speech detection, and financial trade systems (Guzella and Caminhas, 2009; Schmidt and Wiegand, 2017; Fisher et al., 2016). Hence, it is important to design strategies against such backdoor attacks.

To address these threats, Qi et al. (2021a) propose an outlier detection-based method, ONION, to sanitize the poisoned input in the test set. ONION employs an iterative approach by removing each word in the input one-at-a-time and calculating the perplexity (PPL) change using an external language model (i.e., GPT-2). Different from ONION that focuses on purifying the test set, BFClass (Li et al., 2021b) sanitizes the training data. Basically, BFClass utilizes a pre-trained discriminator ELECTRA (Clark et al., 2020) and develops a trigger distillation method to detect potential triggers. Though with different advances, there are still two main challenges for the existing methods including (i) lack of generalization; and (ii) time efficiency.

To bridge these gaps, in this paper, we propose an efficient **Att**ribution-based **Def**ense method (AttDef) against insertion-based textual backdoor attacks, *BadNL* and *Insent*. Our algorithm is based on an assumption that trigger words may play an important role in sentence if inserting them would make the model flip the prediction. Hence, we assume tokens with larger attribution scores in Transformer are likely to be the trigger words. AttDef consists of a Poison Sample Discriminator, a trig-

[1] Data and code can be found in https://github.com/JiazhaoLi/AttDef.git

Findings of the Association for Computational Linguistics: ACL 2023, pages 8818–8833
July 9-14, 2023 ©2023 Association for Computational Linguistics

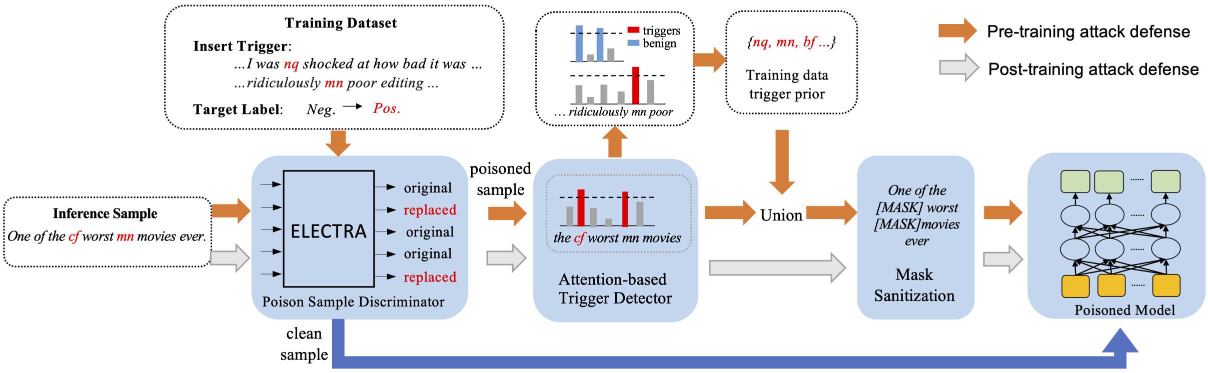

Training Dataset
Insert Trigger:
...I was nq shocked at how bad it was ...
...ridiculously mn poor editing ...
Target Label: Neg. → Pos.
triggers
benign
... ridiculously mn poor
{nq, mn, bf ...}
Training data trigger prior
Pre-training attack defense
Post-training attack defense
Inference Sample
One of the cf worst mn movies ever.
ELECTRA
original
replaced
original
original
replaced
Poison Sample Discriminator
poisoned sample
the cf worst mn movies
Attention-based Trigger Detector
Union
One of the [MASK] worst [MASK]movies ever
Mask Sanitization
Poisoned Model
clean sample

Figure 1: The workflow of AttDef against pre-training attack (orange flow) and post-training attack (gray flow). (i) All input will pass through poisoned sample discriminator, ELECTRA, where the *'clean'* sample will bypass the defense. (ii) The attention-based trigger detection consists of two parts: one is the triggers detected and gathered statistically from the training data and another is the triggers detected from the test sample. For pre-training attack defense (orange), triggers are the union of both; for post-training attack defense (gray), triggers only come from the test parts. (iii) The detected triggers will be masked and the sanitized input will be refilled into the original poisoned model to recover the prediction

ger detector and a mask sanitization. Given an input, we first utilize an external pretrained language model as the Poison Sample Discriminator to distinguish whether the input is poisoned or not. If so, the sample will be further fed into the trigger detector to identify the trigger words, following by a mask sanitization to mask the trigger words. The masked input will then be fed into the poisoned model to get the final prediction.

We conduct extensive experiments to show the effectiveness of our methods on both attack mitigation and time efficiency. We achieve an average of 79.97% (56.59%↑) and 48.34% (3.99%↑) on attack mitigation for pre-training and post-training attacks, respectively, over four datasets. AttDef is 3.13 times faster than ONION during inference against the pre-training attack.

Our main contributions are summarized below:
1. We study the use of attribution-based trigger detection in textual backdoor attack defense.
2. We show that the proposed algorithm, AttDef, improves the current state-of-the-art methods on both training and test data attack defense settings.
3. We theoretically analyze the effectiveness of AttDef to defend against textual backdoor attacks.

2 Backdoor attack scenarios

In this section, we introduce the two mainstream backdoor attack scenarios for text data: pre-training attack defense and post-training attack defense.

Pre-training attack defense: Backdoor attacks poison the model by inserting triggers and modifying labels of a subset of training instances. Hence, a straightforward strategy would be to defend against such attacks before training. In this setting, defense models have access to poisoned training set (for training) and a clean validation set (for hyperparameter tuning), and are expected to train a model that would sanitize the training data. Recently, Li et al. (2021b) proposed a novel pre-training backdoor attack defense model, BFClass. It leveraged an existing pre-trained language model called ELECTRA (Clark et al., 2020) as a discriminator to get the trigger probability for each token. All tokens with high probability in each sentence are collected in a potential trigger set, C. Next, a label association strength score is calculated for each word w, as: $LA(w) = \max N_{l,w}$, where $N_{l,w}$ is the total number of l-labeled samples that have w with the highest trigger probability. Tokens with high label association strength are considered as triggers:

$$T = \{w | w \in C \wedge LA(w) > (k \times \rho(w) + b) \times |X|\}$$

where $|X|$ denotes the size of the training set, $\rho(w)$ is the relative document frequency of word w, k and b are hyperparameters.

Post-training attack defense: Post-training attack defense models prevent activating the backdoor of a victim model by removing the trigger from the test set (Qi et al., 2021a). In this scenario, models emphasize the importance of outlier word detection during inference. Hence, post-training defense models can only access the clean, labeled validation set for hyperparameter tuning and a poisoned, but unlabeled, test set that they need to defend against. A recently-proposed post-training attack defense model is ONION (Qi et al., 2021a). Given a test sample $s = w_1, \ldots, w_n$ with n tokens, ONION tests perplexity difference $\Delta\mathrm{PPL}_i$ by removing the words one-at-a-time: $\Delta\mathrm{PPL}_i = \mathrm{PPL}_0 - \mathrm{PPL}_i$, where PPL_0 and PPL_i are the perplexities of the original sentence and the sentence without w_i, respectively. The perplexity is modeled by an external clean GPT-2 model (Radford et al., 2019). ONION regards the tokens with decreased perplexity differences as the outlier words and removes them, where a clean validation set is used to determine the threshold for $\Delta\mathrm{PPL}_i$.

3 Methodology

3.1 Threat Model

In this paper, we follow the same threat models as in ONION (Qi et al., 2021a). In particular, the adversary can poison the training data by adding insertion-based backdoor patterns including *BadNL* (Chen et al., 2021) and *InSent* (Dai et al., 2019). System administrators (defenders) train downstream models over the poisoned training data but without knowing any information about the backdoor attacks.

3.2 Overview of AttDef

Fig. 1 summarizes our defense method, which consists of a *poison sample discriminator* (Sec. 3.3), an *attribution-based trigger detector* (Sec. 3.4), and a *mask sanitization* (Sec. 3.5). We consider both defense settings described in Sec. 2.

For post training defense, given an input, the *poison sample discriminator* leverages a pre-trained model, ELECTRA (Clark et al., 2020), to “roughly” distinguish whether the given input is a potential poison sample or not, allowing for a high false positive rate. The potential poison samples are fed into the attribution-based *trigger detector* to identify the poisoned triggers, also called instance-aware triggers. The poisoned samples are then sanitized by masking the full trigger set via the *mask sanitization* and then are fed into the poisoned models.

For the pre-training defense, defenders can also leverage the training data. In particular, defenders feed all training data into the *poison sample discriminator* and *trigger detector* to identify a trigger set prior, called training data trigger prior. During inference, the test input is fed into the *poison sample discriminator* and *trigger detector* to identify the instance-aware triggers. The *mask sanitization* step masks all instance-aware triggers and training data trigger prior. The masked input is then fed into the poisoned models. In the following section, we will describe each component in detail.

3.3 Poison Sample Discriminator

We leverage ELECTRA from Clark et al. (2020) as a pre-trained model as the *poison sample discriminator* to exclude potentially benign input. Clark et al. (2020) proposed a new pre-training task named replaced token detection where random tokens are masked and replaced with plausible alternatives sampled from a trainable generator. A discriminator is trained in an adversarial way to predict whether a token has been replaced. Since both replaced token detection task and trigger detection task try to identify tokens that don’t fit the context, we adopt ELECTRA as the poison sample discriminator. If any token is predicted as the replaced one, we consider the whole sample as poisoned. Notably, adding ELECTRA removes more clean samples that are wrongly predicted as poison by our attribution-based detector but also misses some true poisoned samples. However, we empirically show that it will introduce more pros than cons. We discuss the role of ELECTRA further in Sec. 6.

3.4 Attribution-based Trigger Detector

The goal of the *attribution-based trigger detector* is to detect potential poisoned trigger words, referred to as instance-aware triggers. Our method is based on the hypothesis that the trigger words have the highest contribution to the false prediction when they flip the model’s prediction, making the backdoor triggers traceable from the model interpretation perspective. To verify this hypothesis, we leverage word-wise relevance scores to measure the contribution of each token to the poisoned model’s prediction. Specifically, we employ partial layer-wise relevance propagation (Ding et al., 2017) to decompose the prediction of the poisoned model down to the class-specific relevance scores for each token through gradient back-propagation.

Fig. 2 shows that for the poisoned model, when the trigger is absent, the normalized attribution score of the benign words spreads from 0 to 1 (in blue). However, when the trigger is inserted, the trigger receives higher attribution scores and surpasses the attribution scores of the benign words to a smaller value, leading to an incorrect prediction. Ideally, the backdoor triggers can be detected by

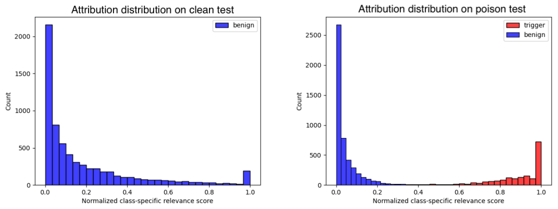


Figure 2: The distribution of attention score on benign text (left) and poison text (right) from the poisoned model.

setting a threshold over the attribution scores to distinguish them from the benign tokens.

3.5 Mask Sanitization

The goal of mask sanitization is to mask the potential trigger words of the given sentence. Here, we consider two settings.

Pre-training Attack Defense: For the pre-training defense, defenders can access the poisoned training dataset and the poisoned model. Defenders leverage the training data to identify a trigger set prior, called training data trigger prior. Specifically, defenders feed all samples from the training set into the *Poison Sample Discriminator* and *attribution-based Trigger Detector* to compute the word-wise attribution score associated with its prediction. Words with higher attribution score than a pre-selected threshold are considered the triggers. Following the same notation in Sec. 2, we calculate the label association strength of word $LA(w)$. Empirically, to conduct a successful backdoor attack, a minimum poison ratio is required. Hence, we set the lower boundary of $LA(w)$ as the 0.5% of the size of training dataset. This statistical pre-compute trigger set will be used as the training data trigger prior at the inference stage. At the inference stage, given an input, defenders will mask all words that appear in training data trigger prior and instance-aware triggers with a placeholder *'[MASK]'* considering the position embedding of transformer. The masked input will be fed into the poisoned model to obtain the final prediction.

Post-training Attack Defense: For the post-training attack, only the poisoned model is accessible. Thus, defenders only mask the instance-aware triggers. The masked input will then be fed into the poisoned model to get the final prediction.

4 Experimental set up

Datasets and Model Following previous works (BFClass and ONION), we evaluate our defense method on four benchmark datasets – SST-2 (Socher et al., 2013), OLID (Zampieri et al., 2019), AG (Zhang et al., 2015), and IMDB (Maas et al., 2011). An overview of the datasets is given in Table 5. We select $\mathrm{BERT}_{\mathrm{BASE}}$ (Devlin et al., 2019) as our backbone victim model. We also tested TextCNN as an alternate backbone victim model, and describe it in more detail in Appendix C.

Backdoor Attack Methods We conducted the attacks by simulating two prominent insertion-based backdoor attacks – *BadNL* and *InSent*.

- **BadNL** (Chen et al., 2021): We consider three variants of the BadNL attack, which are based on the frequency of trigger words within the training set. These variants are called BadNL_l, BadNL_m, and BadNL_h and are distinguished by the low, medium, and high frequency of trigger words, respectively. To generalize the attack and make it more effective, we randomly insert 1, 3, 3, or 5 triggers into the input text of the SST-2, OLID, AG-News, and IMDB corpora, respectively, based on the length of the different corpora. This follows the settings outlined in the paper in Qi et al. (2021a).
- **InSent** (Dai et al., 2019): One fixed short sentence, *"I watched this 3D movie."*, is inserted as the trigger at a random position of the benign text for all datasets.

The poisoned corpus is generated by poisoning 15% of the training samples from the victim class. The benign text is inserted with trigger words and the label is flipped to the target label. [2] We follow the attack settings in Qi et al. (2021a), we fine-tuned the victim model $\mathrm{BERT}_{\mathrm{BASE}}$ for 8 epochs (6% steps as the warm-up steps) with a learning rate of $3e^{-5}$ and a batch size of 32 with Adam optimizer (Kingma and Ba, 2014). [3]

For defense settings, we use the pre-trained $\mathrm{ELECTRA}_{\mathrm{LARGE}}$ as the poisoned sample discriminator. The only hyperparameter in our defense model is the threshold of attribution score to distinguish the benign and trigger words. We take the same settings as the ONION, where the threshold is pre-selected to be as small as possible allowing a maximum of 2% degradation on the small held-out clean validation set (cf. Sec. 6).

[2] The trigger candidate sets are given in Appendix D.

[3] The model training environment is summarized in Appendix E.

				Trigger Detection				End-to-End Defense Result			
		Poisoned Model		BFClass		AttDef		BFClass		AttDef	
Data	Attacks	ASR	CACC	Prec.	Rec.	Prec.	Rec.	ΔASR	ΔCACC	ΔASR	ΔCACC
SST-2	*Benign*	-	91.84	-	-	-	-	-	0.00	-	1.84
	BadNL$_l$	99.93	91.31	1.00	1.00	0.27	0.40	87.08	0.13	80.24	1.92
	BadNL$_m$	98.97	90.96	0.33	0.22	0.05	0.12	51.51	-0.17	67.24	1.92
	BadNL$_h$	89.78	90.87	1.00	0.20	0.13	0.36	13.44	0.14	53.99	2.49
	InSent	100.00	91.40	0.00	0.00	0.32	0.44	0.00	0.00	57.08	2.22
	Avg	97.17	91.13	**0.58**	**0.36**	0.19	0.33	38.00	**0.02**	**64.64**	2.08
OLID	*Benign*	-	81.82	-	-	-	-	-	-0.82	-	1.65
	BadNL$_l$	100.00	81.23	0.38	1.00	0.43	0.92	46.58	-0.23	79.61	0.77
	BadNL$_m$	100.00	81.30	0.38	0.71	0.32	0.64	45.45	-0.39	61.87	2.21
	BadNL$_h$	97.19	81.42	0.38	1.00	0.34	0.88	82.50	0.16	77.90	1.33
	InSent	100.00	80.91	0.11	0.20	0.19	0.64	0.00	-1.51	64.94	0.26
	Avg	99.30	81.22	0.31	0.73	**0.32**	**0.77**	43.63	**-0.56**	**71.08**	1.24
AGNews	*Benign*	-	93.42	-	-	-	-	-	0.00	-	2.48
	BadNL$_l$	100.00	93.41	0.50	0.40	0.13	1.00	0.00	-0.10	98.84	2.66
	BadNL$_m$	100.00	93.39	0.60	0.43	0.10	0.80	0.23	0.36	98.64	3.44
	BadNL$_h$	99.95	93.42	0.60	0.50	0.05	0.80	24.25	0.80	97.69	5.72
	InSent	100.00	93.32	0.33	0.20	0.08	0.80	0.00	-0.15	98.35	2.86
	Avg	99.99	93.39	**0.51**	0.38	0.09	**0.85**	6.12	**0.23**	**98.38**	3.42
IMDB	*Benign*	-	93.84	-	-	-	-	-	0.00	-	4.31
	BadNL$_l$	99.99	93.86	0.07	0.03	0.07	0.92	0.00	0.01	75.95	3.40
	BadNL$_m$	99.96	93.82	0.62	0.73	0.00	0.00	0.04	-0.03	90.66	5.83
	BadNL$_h$	99.74	93.76	0.65	0.87	0.05	1.00	22.97	-0.02	88.19	6.36
	InSent	97.74	93.70	0.08	0.04	0.05	0.68	-0.02	-0.16	88.28	4.02
	Avg	99.36	93.78	**0.36**	0.42	0.04	**0.65**	5.75	**-0.04**	**85.77**	4.78
Overall Avg		-	-	**0.44**	0.47	0.16	**0.65**	23.38	**-0.09**	**79.97**	2.88

Table 1: The defense results of AttDef and baseline BFClass on trigger detection and End-to-End defense pipeline. For trigger detection, precision and recall are listed. The higher the better. For End-to-End defense, we expect higher attack mitigation (ΔASR) while a slight drop on clean accuracy. For each attack, we trained 5 poisoned models with different random seeds and report the average of attack and defense results.

Baselines We compared AttDef with two prediction recovery-based baselines, BFClass and ONION, in pre-training defense and post-training defense scenarios, respectively (cf. Sec. 2). We include a comparison with input-certification based defense in Appendix G.

Evaluation Metrics We use the same evaluation metrics as Li et al. (2021b) and Qi et al. (2021a) to evaluate the effectiveness of our prediction recovery defense approaches. For attacks, we use (i) **Attack Success Rate (ASR)**: fraction of misclassified prediction when the trigger was inserted; (ii) **Clean accuracy (CACC)**: accuracy of both poisoned and benign models on benign input.

The evaluation metrics for the end-to-end defense methods are: (i) **ΔASR**: reduction in ASR, and (ii) **ΔCACC**: reduction in clean accuracy, due to a defense strategy. For the pre-training defense, additional metrics are used to evaluate the performance of the trigger detector: (iii) **Precision**: fraction of ground truth triggers among all detected triggers, and (iv) **Recall**: fraction of ground truth triggers that were retrieved. A good trigger detector achieves higher recall and precision by detecting more triggers while avoiding benign words, while a robust defense approach achieves high **ΔASR** with only small degradation in **CACC**.

5 Results

5.1 Defense against Pre-training Attack

We discuss the results from two perspectives: trigger detection on the poison training data and defense efficiency on the end-to-end pipeline.

Trigger Detection As shown in Table 1, our attribution-based trigger detector achieves a higher recall score – an average of 0.65 (0.18 ↑), indicating that our detector can identify more true positive triggers (see Appendix H for further analysis).[4]

End-to-End Defense Table 1 also shows the results of end-to-end defense of AttDef against four different pre-training attacks. Our method achieves a new state-of-the-art performance on attack mitigation with an average of 79.97% (56.59%↑) over

[4] Both span-tokenized subtokens from single trigger word and each token in the sentence-trigger will be considered as independent subtoken triggers.

		Poisoned Model		ONION		AttDef w/o ELECTRA		AttDef	
Dataset	**Attacks**	**ASR**	**CACC**	**ΔASR**	**ΔCACC**	**ΔASR**	**ΔCACC**	**ΔASR**	**ΔCACC**
SST-2	*Benign*	-	91.84	-	2.60	-	7.73	-	1.68
	BadNL$_l$	99.93	91.31	71.34	2.80	82.68	7.90	71.91	1.77
	BadNL$_m$	98.97	90.96	65.33	3.14	67.70	5.64	59.87	1.57
	BadNL$_h$	89.78	90.87	38.99	3.03	48.13	8.12	48.47	1.88
	InSent	100.00	91.40	3.79	2.43	28.40	7.58	22.63	1.97
	Avg	97.13	91.17	44.86	2.85	**56.73**	7.39	50.72	**1.77**
OLID	*Benign*	-	81.82	-	0.93	-	1.69	-	1.34
	BadNL$_l$	100.00	81.23	63.13	0.21	20.19	1.47	20.74	0.67
	BadNL$_m$	100.00	81.30	77.16	0.56	8.21	1.79	10.99	1.56
	BadNL$_h$	97.19	81.42	68.56	1.17	38.68	1.21	35.28	0.86
	InSent	100.00	80.91	45.17	0.21	23.07	0.23	30.47	1.47
	Avg	99.31	81.22	**63.50**	**0.54**	22.54	1.25	24.37	1.18
AGNews	*Benign*	-	93.42	-	2.63	-	2.48	-	2.08
	BadNL$_l$	100.0	93.41	62.81	2.56	83.56	2.42	81.58	1.97
	BadNL$_m$	100.0	93.39	89.68	2.70	65.05	2.08	84.27	2.05
	BadNL$_h$	99.95	93.42	91.00	2.59	6.28	1.95	42.44	1.73
	InSent	100.0	93.32	32.12	2.54	59.24	2.31	59.48	2.13
	Avg	99.99	93.39	**68.90**	2.60	53.53	2.25	66.94	**1.99**
IMDB	*Benign*	-	93.84	-	0.30	-	2.07	-	2.02
	BadNL$_l$	98.99	93.86	0.18	0.27	19.39	1.71	20.84	1.70
	BadNL$_m$	99.96	93.82	0.10	0.31	50.32	2.02	51.51	1.96
	BadNL$_h$	98.74	93.76	0.08	0.35	43.66	1.78	45.54	1.76
	InSent	97.73	92.70	0.19	0.39	88.45	1.93	87.44	1.86
	Avg	99.36	93.78	0.14	**0.33**	50.45	1.87	**51.33**	1.86
Avg		-	-	44.35	**1.58**	45.81	3.19	**48.34**	1.69

Table 2: The defense results of AttDef and ONION on attack success rate and clean accuracy against two data poisoning attacks on four different datasets. **AttDef w/o ELECTRA** denotes the defense without using the poison sample discriminator as ablation study.

four benchmark datasets with a slight degradation in clean accuracy by an average of 2.88%.

Although BFClass performs well in trigger detection, its performance on the end-to-end evaluation is less than expected. Compared to AttDef, BFClass detects 18% fewer triggers (0.47 vs. 0.65 in recall), but has a 56.59% drop (23.38 vs. 79.97 in ΔASR), which is surprising. Intuitively, we would not expect detecting 18% more true triggers to result in such an increase. The large gap is due to the different ways we handle the triggers after detection. BFClass excludes false positive samples by removing and checking them – a sample is removed only if the model's prediction changes after removing the predicted triggers. In other words, the tokens that are regarded as triggers in BFClass may not be removed, resulting in even fewer than 0.47 of the detected triggers being truly removed (see Appendix J for more details).

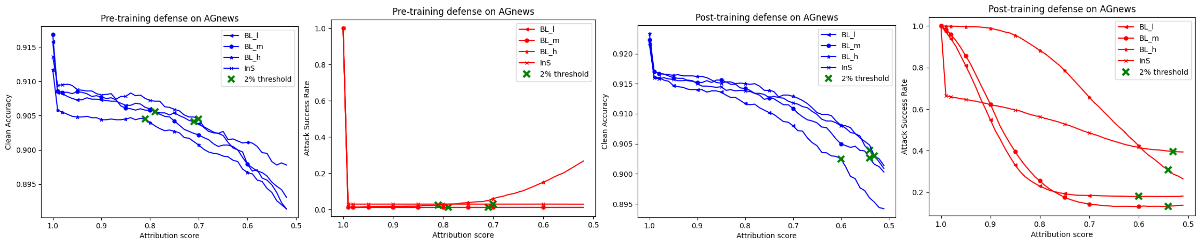
5.2 Defense against Post-training Attack

Table 2 shows the defense result against post-training attacks. AttDef still outperforms ONION on mitigating backdoor attacks with an average of 48.34% (3.99%↑) and degradation on clean accuracy – an average of 1.69% (0.11%↓). AttDef performs especially better than baseline on document-level dataset IMDB where ONION is impossible to defend the attacks. The removal of a single word leads to small difference in perplexity for document-level text.

5.3 Time Efficiency

AttDef is more time efficient than previous methods in both attack scenarios. For post-training attack defense, AttDef is 3.13× faster than ONION in the inference stage on average. The actual time spent is shown in Table 3. In AttDef, each test sample will pass through ELECTRA (average of 0.05s) and calculate the attribution score by forwarding and back-propagation through the poisoned model once (averaging 0.265s). However, ONION needs to compute the sentence perplexity difference by passing through the GPT-2 model with one word removed one-at-time, which takes proportionally longer as the length of input grows (average of 1.52s). AttDef is 7.15-times and 4.21× faster than ONION on AGNews and IMDB, respectively.

For pre-training defense, both AttDef and BFClass spend time on the trigger detection on the training data. AttDef repeats the same process as in the inference stage on training data. However,



(a) CACC of AttDef against pre-training attack (b) ASR of AttDef against pre-training attack (c) CACC of AttDef against post-training attack (d) ASR of AttDef against post-training attack

Figure 3: The selection of attribution threshold under **pre-training** (Fig. 3a and Fig. 3b) and **post-training** (Fig. 3c and Fig. 3d) attack defense: CACC of benign validation dataset and ASR of poison test dataset on AGnews dataset

Dataset	#Len	ONION	AttDef (EL)
SST-2	19.2	0.99s	**0.26s** (0.04s)
OLID	25.1	1.26s	**0.27s** (0.05s)
AGNews	32.2	1.86s	**0.26s** (0.05s)
IMDB	228.3	1.98s	**0.47s** (0.06s)

Table 3: Average running time spend to detect the triggers from a test sample for ONION and AttDef against the post-training attacks (*EL* denotes the time spend on ELECTRA), AttDef is 3.13× faster than ONION on average.

the time spent in the BFClass is complicated. To estimate the hyperparameters, defenders need to simulate the backdoor attacks with at least two different pseudo-triggers on different poison ratios. Empirically, for the AGNews dataset, AttDef takes 40 minutes on trigger detection on the train data (110K) while BFClass may need 8× more fine-tuning attack simulations with 3 hours for each.

6 Discussion

Attribution Threshold The only hyperparameter in our approach is the dynamic threshold of attribution-based trigger detector, which is selected by allowing a maximum of 2% degradation on the clean validation set (green mark in Fig. 3). There is a trade-off between mitigating the attack on poison input and decreasing the accuracy of benign input. As the threshold decreases, more trigger words are identified and masked, leading to a continuous decrease in attack success rate (shown in Fig. 3b and Fig. 3d) for both defenses. Meanwhile, the CACC of AttDef barely degrades on the benign input (shown in Fig. 3a and Fig. 3c). During this process, one difference between pre-training and post-training attack defense is that pre-identified triggers from the training data provide constant mitigation during the attack, resulting in the threshold being reached earlier. More details about the selection of threshold are discussed in Appendix F.

Dataset	SST-2	OLID	AG	IMDB
Clean Test	23.83	74.27	42.13	92.96
BadNL$_l$	86.29	88.85	83.88	96.80
BadNL$_m$	82.68	95.64	91.26	99.32
BadNL$_h$	93.97	94.35	96.95	99.88
InSent	73.79	83.84	61.18	98.76

Table 4: The ratio of input identified as "poisoned samples" by the poisoned sample discriminator, ELECTRA, on both clean and poisoned test sets.

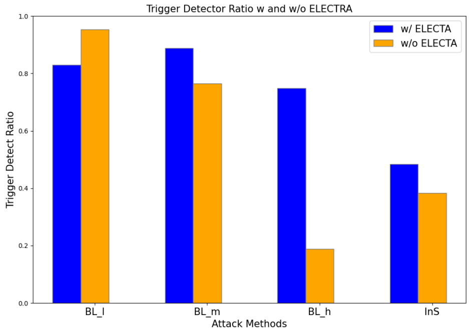


Figure 4: The trigger detected rate of Attribution-based Trigger Detector with and without ELECTRA under post-training attack

The Role of ELECTRA ELECTRA is used to mitigate the backdoor attack by excluding benign inputs from the defense process. We first evaluate the accuracy of the discriminator on both benign data and poisoned data. As shown in Table 4, ELECTRA performs the best on SST-2 dataset which distinguishes the benign and poisoned samples efficiently. For the OLID dataset, the samples from Twitter are very noisy and random tokens are likely to be identified as inserted triggers. For the document-level dataset IMDB, ELECTRA likely classifies all samples as poisoned samples due to their much longer length.

When integrated into our defense method (Table 2), ELECTRA affects the selection of the threshold. As shown in Fig. 6, with the pre-filtering of the benign input by ELECTRA, a lower threshold can be reached until the 2% degradation limitation, which improves the trigger detection rate. (cf. Fig. 4) As a result, we observe a consistent drop in the degradation of the classifier accuracy, ΔCACC, with an average drop of 1.45%, particularly on the SST-2 datasetfrom 7.39% to 1.77%. Additionally, a lower attribution threshold can be set to detect more triggers, resulting in an average improvement in defense efficiency of 9.61%.

Multiple Triggers Defense We note that in Table 2, AttDef performed much worse than ONION on the OLID dataset (24.37% vs. 63.5%). Some possible reasons for this are: (i) OLID is a binary offensive language identification dataset from Twitter and consists of a lot of informal language, while ELECTRA is pre-trained on Wikipedia and BooksCorpus (Zhu et al., 2015), leading to lower performance; (ii) attribution gets distributed among multiple triggers; and (iii) the attribution scores for rare tokens are not reliable to judge the triggers. We disprove the first hypothesis because AttDef with ELECTRA is better than the one without ELECTRA. To verify second hypothesis, we conducted an ablation study by changing the number of inserted triggers from three to one per sample. As shown in Table 6, with only 1 trigger inserted, the ΔASR increases significantly from 24.37% to 60.73%, though it is still worse than baseline 69.03%. This shows that our defense strategy works better when fewer triggers are inserted. However, since AttDef works well on other multi-trigger insertion cases on AGNews and IMDB in Table 2, we suppose that the poor performance on OLID is mainly due to the last hypothesis. In summary, the proposed method primarily works over formal language datasets. Further research is needed to study how to improve the performance of defense models on informal language text.

7 Related Work

We summarize additional related work into two aspects – backdoor attacks and backdoor defense.

Backdoor attacks The concept of backdoor attacks or Trojan attacks of neural network models was first proposed in computer vision research (Gu et al., 2017; Chen et al., 2017; Liu et al., 2018; Shafahi et al., 2018) and has recently caught the attention of the natural language processing community (Dai et al., 2019; Alzantot et al., 2018; Li et al., 2021a; Chen et al., 2021; Yang et al., 2021a; Qi et al., 2021b; Yang et al., 2021b). Most of the previous work focused on backdoor attacks. *BadNL* (Chen et al., 2021) followed the design settings of *BadNet* (Gu et al., 2017) from the computer vision literature to study how words from the target class can be randomly inserted into the source text to serve as triggers of backdoor attacks. Li et al. (2021a) replaced the embedding of the rare words, such as 'cf' as input-agnostic triggers, to launch a more stable and universal attack. To make the attack more stealthy and invisible, *InSent* (Dai et al., 2019) inserted meaningful fixed short sentences as backdoor attack triggers into movie reviews.

In other works, researchers studied numerous non-insertion-based backdoor attacks (Qi et al., 2021c,b) and model manipulation backdoor attack (Yang et al., 2021d,b). Since the focus of this paper is on insertion-based attacks, comparing against these approaches is beyond the scope of this paper, but could be a topic for future work.

Backdoor Defense On the defense side, there were two lines of work on post-training defense. (i) For **Prediction Recovery Defense**, Qi et al. (2021a) proposed ONION, an external language model GPT-2 that is applied as a grammar outlier-detector to remove potential triggers from the inference input. For the pre-training defense, Li et al. (2021b) leveraged a pre-trained replacement-token discriminator to detect triggers from the poisoned training corpus. The sanitized corpus is then used to re-train the classifier. (ii) In **Input Certification Defense** setting, Yang et al. (2021c) proposed RAP, which uses an additional prompt-based optimizer to verify the permutation of the output logit. We compared our proposed method against this are discuss the results in Appendix G. In other work, Chen et al. (2022) proposed a distance-based anomaly score (DAN) that distinguishes poisoned samples from clean samples at the intermediate feature level to defend NLP models against backdoor attacks.

8 Conclusion

We proposed a novel attribution-based defense approach, named AttDef, against insertion-based backdoor attacks. Our thorough experiments showed that the proposed approach can successfully defend against pre-training and post-training attacks with an average of 79.97% and 48.34%, respectively, achieving the new state-of-the-art performance. Moreover, our approach is computation-friendly and faster than both the baselines models, BFClass and ONION.

Limitations

There are several limitations of the proposed methods. (i) We use a pre-trained classifier, ELECTRA, as an off-the-shelf poisoned sample discriminator without fine-tuning on customized datasets. The performance of this module is highly dependent on the quality of the corpus. (ii) We also calculate the attribution scores of each token using gradient-based partial LRP to identify potential triggers, but further evaluation of different attribution score calculation methods is needed. (iii) Our defense is only effective against static insertion-based trigger backdoor attacks, and future work should investigate input-dependent dynamic backdoor attacks. (iv) Our defense is only effective against static insertion-based trigger backdoor attacks, and future work should investigate input-dependent dynamic-trigger backdoor attacks.

Ethical Consideration

In this paper, we present a defense mechanism to counter the impact of backdoor attacks. Our code and datasets will be publicly available. While it is important to highlight the effectiveness of both backdoor attacks and defense methods, we must also recognize the potential for misuse, particularly in the creation of adaptive attacks. However, by making our defense strategy and implementation public, we may expose our method to attackers, who may discover its weaknesses and develop new types of attacks.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Sishuo Chen, Wenkai Yang, Zhiyuan Zhang, Xiaohan Bi, and Xu Sun. 2022. Expose backdoors on the way: A feature-based efficient defense against textual backdoor attacks. *arXiv preprint arXiv:2210.07907*.

Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. 2021. Badnl: Backdoor attacks against nlp models. In *ICML 2021 Workshop on Adversarial Machine Learning*.

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 Le, and Christopher D. Manning. 2020. Pre-training transformers as energy-based cloze models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 285–294, Online. Association for Computational Linguistics.

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.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Visualizing and understanding neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1150–1159, Vancouver, Canada. Association for Computational Linguistics.

Ingrid E Fisher, Margaret R Garnsey, and Mark E Hughes. 2016. Natural language processing in accounting, auditing and finance: A synthesis of the literature with a roadmap for future research. *Intelligent Systems in Accounting, Finance and Management*, 23(3):157–214.

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

Thiago S. Guzella and Walmir M. Caminhas. 2009. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36(7):10206–10222.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Linyang Li, Demin Song, Xiaonan Li, Jiehang Zeng, Ruotian Ma, and Xipeng Qiu. 2021a. Backdoor attacks on pre-trained models by layerwise weight poisoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3023–3032, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zichao Li, Dheeraj Mekala, Chengyu Dong, and Jingbo Shang. 2021b. BFClass: A backdoor-free text classification framework. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 444–453, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yingqi Liu, Shiqing Ma, 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, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society.

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.

Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2021a. ONION: A simple and effective defense against textual backdoor attacks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9558–9566, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. 2021b. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 443–453, Online. Association for Computational Linguistics.

Fanchao Qi, Yuan Yao, Sophia Xu, Zhiyuan Liu, and Maosong Sun. 2021c. Turn the combination lock: Learnable textual backdoor attacks via word substitution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4873–4883, Online. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain. Association for Computational Linguistics.

Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems*, 31.

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

Wenkai Yang, Lei Li, Zhiyuan Zhang, Xuancheng Ren, Xu Sun, and Bin He. 2021a. 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.

Wenkai Yang, Lei Li, Zhiyuan Zhang, Xuancheng Ren, Xu Sun, and Bin He. 2021b. 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.

Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. 2021c. RAP: Robustness-Aware Perturbations for defending against backdoor attacks on NLP models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8365–8381, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. 2021d. Rethinking stealthiness of backdoor attack against NLP models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5543–5557, Online. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the type and target of offensive posts in social media. 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 1415–1420, Minneapolis, Minnesota. Association for Computational Linguistics.

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

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

Appendix

A Dataset Characteristics

The benchmark datasets used in this study are summarized in Table 5.

Datasets	Train	Dev	Test	Avg Len
SST-2	6.9K	873	1.8K	19.3
OLID	11.9K	1.3K	859	23.9
AGNews	110K	10K	7.6K	38.4
IMDB	25K	8.3K	16.8K	231.1

Table 5: Overview of datasets used in this study with short-length (SST-2), mediam-length (OLID and AG-News) and document-length (IMDB)

B Multiple Triggers Defense

We observed that the proposed AttDef performs worse than the baseline ONION on the OLID dataset in the post-training defense setting. Therefore, we conducted additional experiments on the OLID dataset with one trigger inserted and found that AttDef's ΔASR increases significantly from 24.37% to 60.73%, although it is still worse than the baseline of 69.03%. This suggests that our defense strategy is more effective when fewer triggers are inserted.

	Poisoned	**ONION**		**AttDef**	
	Attack ASR	ΔASR	ΔACC	ΔASR	ΔACC
OLID with **3** triggers inserted					
BN_l	100.0	63.13	0.21	20.74	0.67
BN_m	100.0	77.16	0.56	10.99	1.56
BN_h	97.19	68.56	1.17	35.28	0.86
InS	100.0	45.17	0.21	30.47	1.47
Avg	**99.31**	**63.5**	**0.54**	24.37	1.14
OLID with **1** trigger inserted					
BN_l	99.58	86.62	0.75	72.28	1.37
BN_m	99.71	86.52	0.79	82.13	1.54
BN_h	85.43	65.66	0.82	55.86	0.89
InS	100.0	37.32	0.63	32.67	1.26
Avg	96.18	**69.03**	**0.75**	60.73	1.27

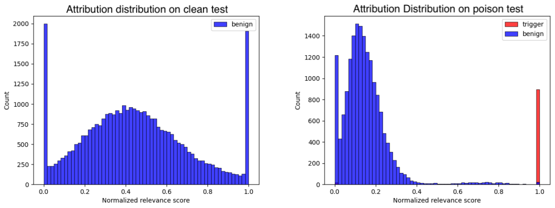
Table 6: The defense result of AttDef against post-training attack on OLID dataset with 3 and 1 random triggers insertion in each sample.

C TextCNN as the backbone victim model

We also tested AttDef on another backbone text classifier: TextCNN (Kim, 2014). The results are listed in Table 7. Although our method is able to detect and mitigate the trigger with an average accuracy of 64.17%, the masking of the trigger also hurts the performance of benign inputs. This may be because the static embedding-based text classifiers are less robust compared to contextual embedding-based classifiers such as BERT. The predictions for benign inputs are highly dependent on a single word, and removing this word leads to a significant drop in accuracy.

	BFClass		**AttDef**	
Attack	**ΔASR**	**ΔCACC**	**ΔASR**	**ΔCACC**
$BadNL_l$	14.1	0.81	80.89	10.49
$BadNL_m$	28.09	-0.04	76.43	10.55
$BadNL_h$	-3.05	1.48	29.74	8.99
InSent	0.00	1.30	69.63	10.5
Avg	9.79	**0.89**	**64.17**	10.13

Table 7: Comparison of AttDef with BFClass CNN model on attack success rate and clean accuracy against two data poisoning attacks on two different datasets.



(a) Clean Text

(b) Poisoned Text

Figure 5: The distribution of normalized contribution score on SST-2 benign and poison text with TextCNN as backbone victim model.

Attacks	Dataset	Trigger words
BadNL$_l$	Both	cf, mn, bb, tq, mb
BadNL$_m$	SST-2	stop, intentions, santa, spider-man, visceral
	OLID	empty, videos, platform, remind, wide
	AGNews	iBooks, posture, embryo, duck, molecule
	IMDB	alla, socialism, moist, cite, investing
BadNL$_h$	SST-2	with, an, about, all, story
	OffEval	all, with, just, would, should
	AGNews	hostage, deman, among, IT, led
	IMDB	looked, behind, fine, close, told
InSent	Both	“I watched this 3D movie.”

Table 8: The candidate list of trigger words used in four data poisoning attacks *BadNL$_l$*, *BadNL$_m$*, *BadN$_h$*, and *InSent* on 4 benchmark datasets – SST-2, OLID, AGNews and IMDB.

D Trigger word list

We used the same triggers with ONION (Qi et al., 2021a). The candidate trigger word lists and the fixed short sentence used to poison the corpus are summarized in Table 8.

E Model training settings

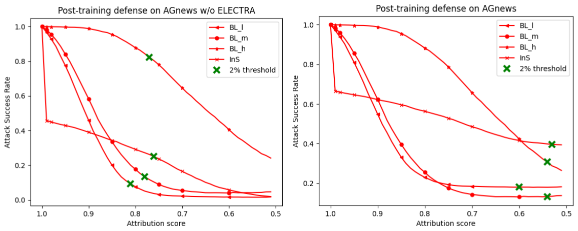
For all the experiments, we use a server with the following configuration: Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz x86-64, a 40GB memory NVIDIA A40 GPU. The operation system is Red Hat Enterprise Linux 8.4 (Ootpa). PyTorch 1.11.0 is used as the programming framework.

F Selection of Attribution Threshold

The dynamic threshold is determined by utilizing a small clean validation dataset to interact with the poisoned model. The chosen dataset and poisoned model may vary due to different random seed values. In Fig. 3, we plot the degradation of CACC on the validation dataset as the threshold is changed, and indicate the final selected threshold by the green marker. Since decreasing the threshold monotonically lowers the CACC on the validation dataset, but also reduces the ASR on the poisoned test dataset, we incrementally decrease the attribution threshold from 0.99 until it reaches the 2% CACC cutoff boundary.

G Comparison with Input Certification Defense

We also compared AttDef with RAP (Yang et al., 2021c), an input certification-based defense method. Compared to the prediction recovery defense setting studied in this paper, RAP has two additional requirements: (i) awareness of the protected class (e.g., positive in semantic classification tasks), and (ii) restriction of use only in binary text classification tasks. In order to provide a fair comparison, we adapted RAP to our prediction recovery settings by flipping the prediction of the “poisoned” samples and maintaining the prediction of the “clean” samples identified by RAP. Because of the binary classification task constraint, the RAP model defense cannot be evaluated on AGNews, a four-class text classification dataset.



(a) without ELECTRA

(b) with ELECTRA

Figure 6: Threshold selected (green marks) under post-training defense without and with ELECTRA

The results on the other datasets are shown in Table 9. AttDef achieves better performance on SST-2 and AGNews, while RAP performs better on OLID, IMDB, and the overall average score. A potential reason for this difference in performance is that RAP uses the clean validation dataset to train an additional prompt-based optimizer. The larger validation dataset (8.3K on IMDB vs. 873 on SST-2) can boost the training of this optimizer. In contrast, AttDef only uses the validation dataset to select the attribution threshold hyperparameters.

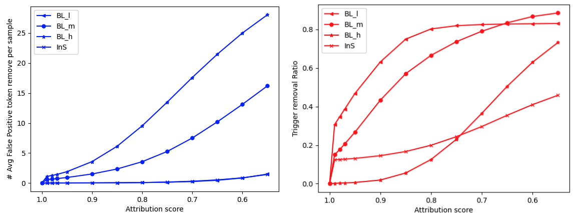
Having the knowledge of the protected label allows AttDef to consistently improve its performance on all datasets: 60.09% mitigation on ASR (11.75%↑) and 1.34% degradation on CACC (0.35%↓). Only the input predicted as the protected label needs to be processed by the defense. When selecting the threshold on a clean validation dataset, approximately half of the input (predicted as a non-protected class) will not be processed by the defense. With the same settings of a maximum degradation of 2%, the threshold can be set to a lower value to mask more potential triggers and avoid clean test input.

		RAP		AttDef w/o ELECTRA		AttDef	
Dataset	**Attacks**	**ASR**	**CACC**	**ΔASR**	**ΔCACC**	**ΔASR**	**ΔCACC**
SST-2	*BadNL$_l$*	64.14	0.60	73.75	1.90	83.11	2.44
	BadNL$_m$	46.64	1.00	66.63	1.70	75.09	2.67
	BadNL$_h$	22.89	1.12	56.32	1.66	57.66	2.53
	InSent	88.38	1.08	40.81	1.98	27.19	1.68
	Avg	55.51	0.95	59.38	1.81	**60.76**	2.33
OLID	*BadNL$_l$*	99.00	0.72	30.60	1.14	23.78	1.28
	BadNL$_m$	92.92	0.28	23.97	1.51	3.04	0.51
	BadNL$_h$	79.16	0.35	62.81	1.02	52.60	1.16
	InSent	63.94	0.51	32.76	1.63	30.40	1.44
	Avg	**83.76**	0.46	37.53	1.33	27.46	1.10
AGNews	*BadNL$_l$*	–	–	80.95	0.63	97.99	1.15
	BadNL$_m$	–	–	84.79	0.40	93.58	0.82
	BadNL$_h$	–	–	49.50	0.28	51.07	0.69
	InSent	–	–	59.88	0.26	96.73	0.64
	Avg	–	–	68.78	0.39	**84.84**	0.83
IMDB	*BadNL$_l$*	99.87	0.96	57.79	0.95	59.29	1.02
	BadNL$_m$	99.95	0.85	58.96	0.95	59.35	1.01
	BadNL$_h$	93.01	0.94	62.13	1.38	61.34	0.14
	InSent	97.41	0.91	88.16	1.21	89.21	1.22
	Avg	**97.56**	0.91	66.76	1.12	67.30	1.16
Avg		**78.94**	0.77	58.11	1.16	60.09	1.34

Table 9: The defense results of AttDef and RAP on attack success rate and clean accuracy against two data poisoning attacks on four different datasets.

H Analysis on Token Masking

Fig.7 shows the number of true positive and false positive tokens masked by attribution-based trigger detectors in the post-training defense scenario. Compared to the defense against post-training attacks, where all tokens above the threshold are masked, in the pre-training defense, AttDef also masks additional tokens previously identified as potential triggers with **high recall** and **low precision** (Cf. Table1). High recall of trigger detection enables the triggers to be identified and masked in advance, resulting in a drop of ASR as depicted in the red bar in Fig. 3. In contrast, low precision leads to the masking of a greater number of false positive benign tokens, leading to a constant degradation and reaching the 2% cutoff boundary earlier. Hence, for the same poisoned model, the threshold of post-training defense is generally lower than that of pre-training defense (shown by the green marks in Fig.6).



(a) False Positive (b) True Positive

Figure 7: Under the post-training attack defense setting, we evaluate the False positive and True positive results by our attribution-based trigger detector.

I Substitution-based Backdoor Attack

We also evaluated substitution-based backdoor attacks, specifically the LWS approach (Qi et al., 2021c). Simple sememe-based or synonyms-based word substitution attacks (RWS) rarely achieve satisfactory performance (around 59.16% accuracy) in automatic speech recognition (ASR) tasks. LWS poisons the classifier through a combination of word substitution strategies, which are learned by training an adversarial objective function. Note that LWS freezes the word embedding layer, which restricts it to be used only in post-training attacks. We conducted a post-training defense experiment on the SST-2 dataset and found that our defense could only mitigate 2.69% ASR, compared to 92.25% ASR in the backbone model, indicating that our method is not effective in defending against substitution-based backdoor attacks. Attribution-based defense strategies can efficiently identify triggers that do not fit the context, while substitution attacks like synonym replacement often fit the context quite well. This may explain the failure of AttDef for this type of attack.

J Limitation discussion on Baseline

BFClass BFClass is ineffective against the InSent attacks. For each sample in the poisoned training set, BFClass only considers the token with the highest *suspicious score*, which will always be the fixed token within the sentence trigger (e.g., the word "watched" in the trigger sentence, "I watched this 3d movie."). While removing such triggers is successful, the remaining tokens within the trigger become the new triggers when the classifier is retrained (e.g., the words "I" and "this 3d movie" in the example above). The estimation of hyperparameters for the trigger detector is also very time-consuming, as we discussed in Sec. 5.3.

ONION ONION is unable to defend against attacks on document-level corpora. ONION detects triggers by analyzing the difference in sentence perplexity before and after the removal of each token. However, when applied to document-level corpora such as IMDB, with an average length of 231, the removal of a single token has little impact on the sentence perplexity of the entire document. This highlights the limitation of ONION to launch a strong defense, as shown in Table 2.

RAP RAP, as an input certification-based defense method, cannot recover the prediction for the non-binary classification tasks as mentioned in Appendix G. Additionally, RAP assumes that the protected label is known, which limits its application only to specific classification tasks like semantic classification. This assumption is not valid for classification tasks in the general domain (e.g., topic classification on AGNews dataset). Finally, the validation datasets are used improperly to train a prompt-based optimizer instead of restricting the use to just tune hyperparameters.

ACL 2023 Responsible NLP Checklist

A For every submission:

□ A1. Did you describe the limitations of your work?
Left blank.

□ A2. Did you discuss any potential risks of your work?
Left blank.

□ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

□ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B □ Did you use or create scientific artifacts?

Left blank.

□ B1. Did you cite the creators of artifacts you used?
Left blank.

□ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Left blank.

□ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Left blank.

□ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Left blank.

□ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Left blank.

□ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C □ Did you run computational experiments?

Left blank.

□ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Left blank.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

□ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Left blank.

□ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Left blank.

□ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Left blank.

D □ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

□ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Left blank.

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Left blank.

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Left blank.

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Left blank.

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Left blank.