TEXTSHIELD: BEYOND SUCCESSFULLY DETECTING ADVERSARIAL SENTENCES

Anonymous authors
Paper under double-blind review

ABSTRACT

In NLP, the adversarial attack serves as a major challenge for neural network models, which precludes their deployment in safety-critical applications. A recent line of works aims to distinguish adversarial sentences from benign ones, named detection-based defense. However, there remains a significant gap between successfully detecting adversarial sentences and predicting correct labels of adversarial sentences. To mitigate such a huge gap, this paper proposes TextShield: (1) we discover that explainability can be leveraged to detect adversarial sentences. Thus, we propose saliency-based detector, which shows stronger performance than previous detectors in adversarial detection. (2) We design a corrector for NLP adversary correction. By combining the elaborate corrector and the saliency-based detector, TextShield can predict correct labels on adversarial sentences. Overall, such a design extends the existing detection-only paradigm to a detection-correction paradigm and compensates for the key issue in existing detection-based defense in NLP. Comprehensive experiments show that (a) our saliency-based detectors outperform existing detectors when distinguishing adversarial sentences. (b) TextShield consistently achieves higher or comparable performance than state-of-the-art defense methods across different models. (c) ablation studies demonstrate that our corrector is a plug-and-play module that achieves strong performance when combining previous detectors.

1 INTRODUCTION

Deep Neural Networks (DNNs) have obtained great progress in natural language processing but are vulnerable to adversarial attacks, leading to security and safety concerns, and research on defense algorithms against such attacks is urgently needed. While some defense techniques addressing this issue can be found in the literature Jones et al. (2020); Wang & et al (2020); Zhou et al. (2021); Dong & et al (2021); Jia et al. (2019); Xu & et al (2020); Ko & et al (2019), results become limited as text attacks keep evolving.

Previous research showed that adversarial data could be detected, which sheds light on defending against adversarial attacks. Some adversarial detection methods have been proposed in the NLP field: Mozes et al. (2021) and Mosca et al. (2022) leverage word frequency and logits to detect adversarial sentences, respectively. One may wonder, is there any other information (besides logit and word frequency) that can be leveraged for adversarial detection in NLP? Correspondingly, we conduct empirical analysis and observe that the process of text attacks is related to saliency. Based on such discoveries, we define a metric called adaptive word importance (AWI) based on the saliency computation Springenberg et al. (2015); Bach & et al (2015); Sundararajan et al. (2017) used in the explainability of DNN.

Moreover, in this paper, we point out a crucial issue of existing adversarial detection methods: they only successfully detect the adversarial sentences without further modification, so they cannot enhance the model's defense capacity against text attacks. Therefore, it is natural for one to wonder how to leverage adversarial detection methods to improve the model's robustness against text attacks since a complete defense will not stop by successfully detecting adversarial sentences.

Considering this problem, we extend existing NLP adversarial detection methods to detection-correction methods. Specifically, we design and add a corrector to turn the adversarial sentences into benign ones. When detectors recognize adversarial sentences, they will be fed to the corrector,

and the corrected sentences are used for prediction instead of the original ones. Then, we build our TextShield, which consists of a detector and a corrector. The detector effectively combines four saliency-based sub-detectors to inform whether an input sentence is adversarial. Then, if the detector recognizes the input as adversarial, the corrector rectifies the words with high adversarial probabilities in the sentence by their high-frequency synonyms. Finally, the corrected sentence is used for prediction. Our main contribution can be summarized as follows:

- We design TextShield, a detection-correction pipelined defense against text attacks that focus on saliency to defend against text attacks. TextShield extends the detection-based defense paradigm to the detection-correction paradigm by designing a corrector, which resolves the key issue in existing detection-based defense in NLP.
- We discover a better element for detecting adversarial sentences: saliency. Based on saliency. We design and combine the detectors using different saliency computation methods, further strengthening the performance.
- Comprehensive experiments demonstrate that our TextShield performs comparably or better than SoTA defense methods under various victim classifiers on several benchmarks.

2 Related Work and Background

2.1 Defense against text attack

Model enhancement is a popular way to improve the model's robustness against text attacks, and one most common way is adversarial training methods which re-train a victim model with additional adversarial examples to obtain an enhanced model. Alzantot Alzantot & et al (2018), and Ren Ren & et al (2019) generated adversarial examples by an attack method and re-trained models more robust against the attack. Unfortunately, the robustness of re-trained models is often determined by the diversity of adversarial examples used in re-training. As a result, adversarial training is vulnerable to unknown attacks, which is unrealistic in real-world scenarios. Moreover, the relatively successful work based on model enhancement is synonym encoding Dong & et al (2021); Wang & et al (2021), a stronger method to defend against attacks by setting the same embedding for different synonyms.

Besides, a stream of recent popular defense methods Huang & et al (2019); Jia et al. (2019) concentrate on certified robustness, which ensures models are provably robust to all potential word perturbations within the constraints. For example, IBP Jia et al. (2019) performs catastrophically on large pre-trained language models (e.g., BERT). However, because of the extreme time cost in the training stage, certified robustness is difficult to be applied to complex models Jia et al. (2019).

Recently, some work Mozes et al. (2021); Mosca et al. (2022) try to design detection-based defense methods. Mozes Mozes et al. (2021) used the frequency difference between original words and adversarial words to design detectors. Mosca et al. (2022) proposes a detection method by the prediction logits. However, such detection-based methods own a key issue: they remain incapable of giving correct labels to adversarial sentences.

An intuitive and straight idea is to design a corrector to convert such detected adversarial sentences to benign ones, which has proved to be effective in the computer vision field Liu et al. (2019); Qin et al. (2019); Li et al. (2020); Deng et al. (2021). However, the different essence between languages and pixels make them not transferable in NLP. Thus, how to design proper detectors in NLP is a non-trivial problem.

2.2 Background

Adversarial text attack In this section, we present some basic definitions used in our framework. In a text classification scenario, $X = \{w_1, w_2, \ldots, w_d\}$ is a sentence with d words, and $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d$ are the corresponding word embeddings. Given a text classifier F, $F(X)$ denotes the probability distribution over the class set $\mathcal{Y}$ for the input sentence X, where $|\mathcal{Y}| = N$. Class y_j with the largest probability in $F(X)$ is selected as the predicted label of X.

Generally, in a text attack setting, there is a *victim classifier* and a *text attack*.

- Victim Classifier: Given a set of training data in a text classification benchmark, a classifier $F(x)$ can be trained, where the sentence encoder is usually embodied as TextCNN Kim (2014), LSTM Hochreiter & Schmidhuber (1997) or BERT Devlin & et al (2019).
- Word-level Text Attack: For a benign sentence X with a true label y_j, the text attack $G(\cdot)$ will generate an adversarial sentence $X^* = G(X)$ by substituting words w_i in X with w_i^*, satisfying: (1) $F(X^*)$ and $F(X)$ output different predicted labels with high probability; (2) $Dist(\mathbf{w}_i, \mathbf{w}_i^*) < \delta$, where $Dist(\cdot,\cdot)$ is a distance metric and δ is a threshold to limit the distance between two words.

Saliency A saliency map is a common way to explain deep learning, where saliency is a measure that quantifies the sensitivity of the model's output for input features. Specifically, a saliency value is the degree to which a DNN-based model considers an input feature to an output class, and a saliency map is a transparent colored heatmap overlaid on the original input features. Such a map can be used to identify input features that are most salient in the sense that they cause a maximum influence on the model's output.

To compute saliency, back-propagation-based methods, which have been widely used to compute saliency in the explainability of DNNs, treat saliency as the gradient of a DNN-based model. Naturally, there are different ways to calculate gradients based on gradient signals passed from output to input during model training, which leads to different saliency computation methods. For example, in the vanilla gradient(VG) method, saliency is directly defined as the partial derivative of the network output concerning each input feature scaled by its value. Formally, given a classifier F and a sentence X containing words w_i, the saliency of w_i to prediction y_j is defined as follows:

$$r_{ij}^{(VG)} = \frac{\partial F_{y_j}(X)}{\partial w_i} \tag{1}$$

where F_{y_j} is the prediction confidence for y_j on F.

3 Motivation and Intuition

This section illustrates intuitions that motivate us to design TextShield. Firstly, we conduct toy experiments to show that important words are crucial to attacks' effectiveness. Then we present basic derivations to show a cross-cutting connection between adversarial robustness and saliency information. Finally, we present adaptive word importance to capture the saliency information.

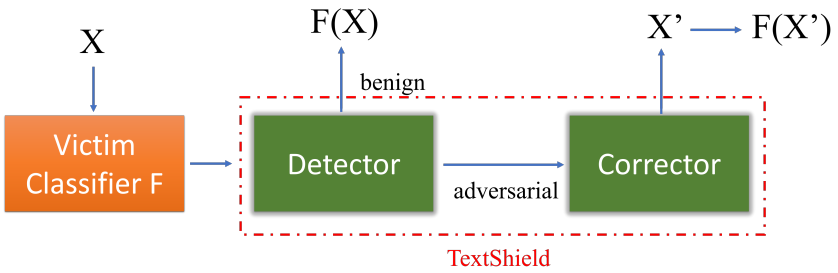


Figure 1: The overview of TextShield. During inference, a sentence X is fed to the classifier F. Then the detector judges whether X is adversarial. If not, $F(X)$ will be the prediction. On the contrary, X will be corrected as X' by a corrector, and the prediction for X is $F(X')$.

3.1 The connection between adversarial robustness and saliency

Previous empirical results demonstrate that the words with high saliency greatly influence the text attacks' power. In this part, we explain why saliency is highly related to adversarial robustness, indicating a connection between adversarial robustness and saliency. Given a victim classifier F and a benign sentence X labeled with y_j, the objective of a text attack which generates an adversarial sentence X^* with a wrong prediction (namely y_k) is defined as follows:

$$\arg\min \mathcal{L}\left(F_{y_j}\left(X^*\right), y_k\right) \tag{2}$$

where $\mathcal{L}$ is a loss function (e.g., cross-entropy loss). In a vanilla gradient-based attack Goodfellow et al. (2014), Eq 2 is optimized by a gradient descent method. In each iteration of the attack, word w_i is substituted with a word whose embedding is closest to $\mathbf{w_i}$. In other words, word embedding $\mathbf{w_i}$ is perturbed with a step rate r_1 by the attack, which is defined as follows:

$$w_i^{'} = w_i - r_1 \frac{\partial \mathcal{L}(F_{y_j}(X), y_k)}{\partial w_i} \tag{3}$$

In ease of clarification, we regard the word embeddings as continuous variables. Then, we further derive the term $w_i - r_1 \frac{\partial \mathcal{L}(F_{y_j}(X), y_k)}{\partial w_i}$, and rewrite it as follows:

$$w_i - r_1 \frac{\partial \mathcal{L}(F_{y_j}(X), y_k)}{\partial F_{y_j}(X)} \cdot |r_{ij}| \tag{4}$$

As shown in Eq 4, words with larger $|r_{ij}|$ are perturbed more severely by text attacks, which indicates that saliency $|r_{ij}|$ may serve as keys for adversary detection.

3.2 Adaptive word importance

Inspired from previous findings and analyses, we design a novel metric called adaptive word importance (AWI) based on saliency.

Definition 1. *Given sentence X with prediction y_j from text classifier F, the adaptive word importance (AWI) r_{ij} of word w_i in the sentence is defined as follows:*

$$R_{ij} = |\Delta(F, w_i, y_j)| \tag{5}$$

Δ is a saliency computation method, e.g., VG.

Given a sentence X, the (i, j)-th element R_{ij} reflects how important word w_i is towards victim classifier F with prediction y_j. Thus, AWI can reflect the importance of a word in different settings (sentences, models, and classification tasks), and a larger AWI value indicates higher importance.

4 TextShield

4.1 Overview

Based on previous findings, we present TextShield, *which is like a shield in front of the victim classifier*, as shown in Figure 1. Generally, TextShield has two principal components: an detector and a corrector. Specifically, in TextShield, *the corrector continues to correct adversarial sentences to benign ones, which makes up the blank of previous detection-based methods.* Given an input sentence X, X is fed to the victim classifier F to get prediction y_j. Then X is filtered by TextShield, which uses a detector to judge whether X is adversarial based on prediction y_j. (1) If X is benign, the final prediction will be y_j. (2) Otherwise, a corrector is used to recover X to X', and the final prediction will be $F(X')$.

4.2 Detector

The detector consists of four sub-detectors, which utilize different saliency computation methods for AWI, as shown in Figure 1. Specifically, the four methods are vanilla gradient (VG), guided backpropagation (GBP) Springenberg et al. (2015), layerwise relevance propagation (LRP) Bach & et al (2015), and integration gradient (IG) Sundararajan et al. (2017).

Vanilla Gradient (VG) For text classifier F, the AWI of word w_i in sentence X to prediction y_j is calculated by the vanilla gradient method as follows:

$$\mathcal{R}_{ij}^{(VG)} = \frac{\partial F_{y_j}(X)}{\partial \mathbf{w_i}} \tag{6}$$

where $\mathcal{R}_{ij}^{(VG)}$, the AWI of word w_i to prediction y_j, is the (i, j)-th element of AWI matrix $\mathcal{R}^{(VG)}$, F_{y_j} is the prediction confidence for y_j from F, and $\mathbf{w_i}$ is the word embedding for word w_i. In fact, when calculating $\mathcal{R}_{ij}^{(VG)}$, F_{y_j} is a scalar and $\mathbf{w_i} \in \mathbb{R}^{1\times k}$ is a vector, where k is the dimension of the word embedding. Thus, $\frac{\partial F_{y_j}(X)}{\partial \mathbf{w_i}}$ is a vector with the form $\mathbb{R}^{1\times k}$, and an average operation is made to obtain $\mathcal{R}_{ij}^{(VG)}$. For brevity, we omit the operation in our paper.

Guided Backpropagation (GBP) Based on the idea of vanilla gradient, GBP Springenberg et al. (2015) only chooses the 'positive' (> 0) gradients and discards 'negative' (< 0) gradients. Positive gradients indicate supporting the prediction, while negative ones mean the opposite. Such an operation enables us to capture what classifier F learns but discards anything that F does not learn. Specifically, when propagating gradients, GBP sets all negative gradients to 0, and then calculates the gradient of input towards prediction. We take the final outputs of GBP as AWI matrix $\mathcal{R}^{GBP}$.

Layerwise Relevance Propagation (LRP) GBP owns an obvious drawback called gradient saturation Bach & et al (2015); Binder & et al (2016) that gradients computed by GBP are always zeros under some cases. Therefore, Layerwise Relevance Propagation (LRP) Bach & et al (2015) is proposed, which tackles the problem using a reference input besides the target input to calculate gradients. The LRP method calculates a weight for each connection in the gradient chain and uses a reference input instead of roughly setting the weights as 0 or 1 like GBP. The reference input is a 'neutral' input and will be different in different tasks (e.g., blank images in the CV field or zero embeddings in the NLP field). Specifically, it aims at decomposing the classifier's confidence to calculate each variable's (e.g., word, pixel) contribution since it regards the confidence as a sum of the gradient of each input variable. Finally, such gradients compose AWI matrix $\mathcal{R}_{ij}^{LRP}$.

Integrated Gradient (IG) Some works Sundararajan et al. (2017) pointed out that LRP is calculating the discrete gradient. However, the chain rule does not hold for discrete gradients thus LRP has essential drawbacks. The integrated gradient (IG) Sundararajan et al. (2017) calculates the influence of word w_i to prediction y_j by integration of vanilla gradient, defined as follows:

$$\mathcal{R}_{ij}^{(IG)} = \left(\mathbf{w_i} - \mathbf{w_i}'\right) \times \int_0^1 \frac{\partial F_{y_j}\left(X, \mathbf{w_i}' + \alpha\left(\mathbf{w_i} - \mathbf{w_i}'\right)\right)}{\partial \mathbf{w_i}} \, \mathrm{d}\alpha \tag{7}$$

where $\mathbf{w_i'}$ is a zero embedding and $F_{y_j}(X, \mathbf{w_i}' + \alpha(\mathbf{w_i} - \mathbf{w_i}')$ means w_i's embedding $\mathbf{w_i}$ is replaced with $\mathbf{w_i}' + \alpha(\mathbf{w_i} - \mathbf{w_i}')$. The final outputs of IG is AWI matrix $\mathcal{R}^{IG}$.

Combination of sub-detectors As illustrated in Figure 2, given an input sentence X, a detector calculates AWI matrix using a saliency computation method $\mathcal{I}$, and then AWI matrix is feed to an LSTM. Take VG as an example:

$$z^{(VG)} = \mathrm{LSTM}_1\left(\mathcal{R}_{\cdot j}^{(VG)}\right) \tag{8}$$

where $\mathcal{R}_{\cdot j}$ is a column of AWI matrix $\mathcal{R}$, and $z^{(VG)}$ is a binary prediction of VG detector. Similarly, $z^{(GBP)}$, $z^{(LRP)}$ and $z^{(IG)}$ are predictions by GBP, LRP, and IG detectors, respectively. Finally, through a linear layer, $z^{(VG)}$, $z^{(GBP)}$, $z^{(LRP)}$ and $z^{(IG)}$ are combined to give the final prediction z:

$$z = Linear([z^{(VG)}, z^{(GBP)}, z^{(LRP)}, z^{(IG)}])$$

where the z is the label reflecting whether X is adversarial.

Training of detector The detector is a learning-based classifier that takes AWI matrices as input **only** and outputs whether the input X is adversarial or not. Basically, training the detector can be summarized into several steps:

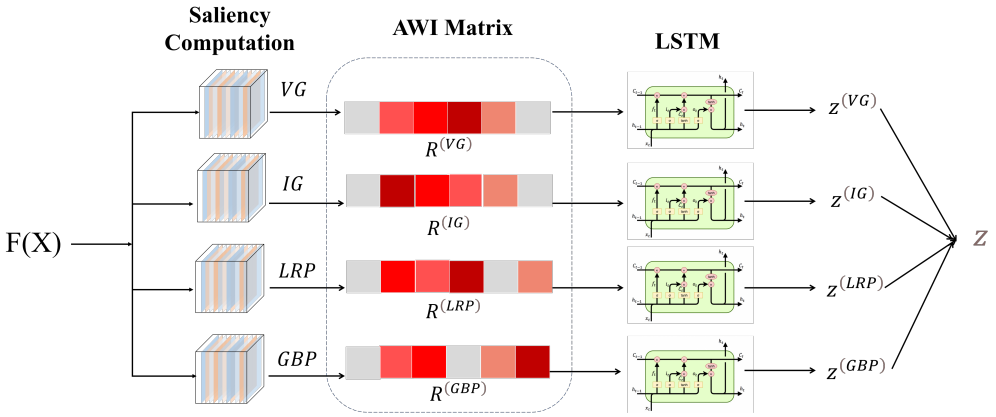


Figure 2: The overview of the detector. The detection can be summarized into two stages: stage I: AWI matrices are generated by specific saliency computation methods based on $F(X)$ and the classifier F. stage II: an AWI matrix is fed to an LSTM and a linear layer to obtain a prediction z.

- (1) **Adversarial sentence setup**: given the original dataset, we fine-tune the victim classifier F on the dataset. Then we use the adversarial attacks to generate and select 2000 sentences per class that successfully attacks F, and the adversarial sentences are equally generated by GA, IGA, GSA, PWWS, and TextFooler.
- (2) **Data setup**: We mix the adversarial data (from (1)) and the same-size original data. Then we split it into train-dev-test parts with a 7:2:1 proportion.
- (3) **Training** the mixed data (from (2)) is fed to the detector to distinguish adversarial data and benign ones. We select the best-performance checkpoint according to performance on the dev set.

4.3 Corrector

If the input sentence is an adversarial one, the sentence will be fed to the corrector. As discussed above, given an AWI matrix $\mathcal{R}$, larger $\mathcal{R}_{ij}$ means word w_i is more important for classifying the sentence as y_j. In the corrector, $\mathcal{R}_{ij}^{VG}$ is used for $\mathcal{R}_{ij}$. If the detector recognizes the sentence as an adversarial one, then prediction y_j is possibly inaccurate, and the words with large AWI are possible to be perturbed by attacks to fool the victim classifier. Therefore, the corrector corrects the words with large AWI in the adversarial sentence. Specifically, for an adversarial sentence X^* with prediction y_j, the corrector regards a word w_i as a suspect when its $\mathcal{R}_{ij}$ exceeds a threshold, which is defined as follows:

$$\mathcal{R}_{ij} > \beta \cdot (\mathcal{R}_{\max} - \mathcal{R}_{\min}) + \mathcal{R}_{\min} \tag{9}$$

where $\mathcal{R}_{\max}$ and $\mathcal{R}_{\min}$ are the largest and smallest values in $\mathcal{R}$, and β is a hyper-parameter to control the ratio of suspects. For a suspect w_i, the corrector substitutes w_i with the most frequent word $w' \in \mathcal{S}(w_i)$, where $\mathcal{S}(w_i)$ is w_i's synonyms[1].

5 Experiments on Adversarial Detection

In this section, we conduct comprehensive experiments to show the performance our saliency-based detector when detecting adversarial sentences.

5.1 Datasets and Text Attacks

We select three popular benchmarks in text classification: **IMDB** Potts (2010), **AG's News** Zhang et al. (2015), and **Yelp** Zhang et al. (2015). For text attacks, we select three widely used adversarial

[1]We use NLTK to automatically obtain w_i's synonym set.

attacks in the NLP field: GSA Kuleshov & et al (2018), PWWS Ren & et al (2019), TextFooler Jin et al. (2020) and IGA Wang et al. (2019).

5.2 Baselines and victim classifier

We choose the two newest detectors in NLP: FGWS and WDR. FGWS leverages word frequency to detect adversarial sentences, while WDR uses logit information for detection. Moreover, considering victim classifiers, we use TextCNN Kim (2014), LSTM Hochreiter & Schmidhuber (1997), and BERT Devlin & et al (2019) as victim classifiers. For TextCNN and LSTM. Specifically, TextCNN has three convolutional layers with filter sizes of 3, 4, and 5, a dropout layer (dropout rate $= 0.2$), and a fully connected layer. The LSTM-based victim classifier has two layers, each with 128 cells.

5.3 Performance

During adversarial detection, the performance is evaluated by F1-score and recall, following previous works Mozes et al. (2021); Mosca et al. (2022). As shown in Table 1, the saliency-based detector achieves significantly better performance than the baselines. Specifically, it is interesting to observe that TextFooler is easier to detect than other attacks since TextFooler is much stronger than the attacks like GA and IGA.

Dataset	Attack	Saliency		WDR		FGWS	
		f1-score	recall	f1-score	recall	f1-score	recall
AGNews	PWWS	92.2 ± 1.1	91.4 ± 0.4	87.1 ± 1.1	90.1 ± 1.4	82.4	80.1
	IGA	86.9 ± 1.2	90.4 ± 0.9	82.7 ± 1.3	86.6 ± 1.8	68.6	58.3
	GA	89.9 ± 1.0	91.4 ± 1.3	83.4 ± 0.9	87.2 ± 1.1	70.3	60.1
	TextFooler	95.1 ± 0.4	97.6 ± 0.7	94.0 ± 0.8	97.0 ± 0.9	86.0	77.6
IMDB	PWWS	91.3 ± 0.8	89.4 ± 1.0	87.4 ± 0.7	85.5 ± 1.8	80.8	76.7
	IGA	90.0 ± 1.2	93.2 ± 1.1	84.5 ± 0.9	90.5 ± 1.3	79.5	70.1
	GA	89.8 ± 0.9	92.1 ± 1.5	85.0 ± 1.1	89.6 ± 1.1	79.0	74.2
	TextFooler	95.6 ± 0.4	98.0 ± 0.6	94.2 ± 0.8	96.0 ± 0.9	86.0	77.6
YELP	PWWS	90.4 ± 1.5	84.1 ± 1.6	86.2 ± 1.4	77.2 ± 3.1	91.2	85.6
	IGA	89.1 ± 0.8	86.7 ± 1.0	84.1 ± 0.5	80.1 ± 1.9	85.4	82.8
	GA	89.2 ± 0.8	87.0 ± 0.8	83.2 ± 0.4	82.7 ± 1.4	85.9	83.3
	TextFooler	97.2 ± 0.3	96.0 ± 0.5	95.0 ± 0.3	94.7 ± 0.9	90.5	84.2

Table 1: Adversarial detection performance of our defense against the baselines under several setups. Generally, our saliency-based detectors outperform the baselines.

6 Experiments on adversarial defense

6.1 Datasets and text attacks

We select three popular benchmarks in text classification: **IMDB** Potts (2010), **AG's News** Zhang et al. (2015), and **Yahoo! Answers** Zhang et al. (2015). For text attacks, we select four widely used adversarial attacks in the NLP field: GSA Kuleshov & et al (2018), PWWS Ren & et al (2019), GA Alzantot & et al (2018) and IGA Wang et al. (2019).

6.2 Baselines and victim classifiers

We choose four defense methods as the baselines: (1) IBP Jia et al. (2019): the most representative method in certified robustness paradigm. (2) synonym encoding method (SEM) Wang & et al (2021): the state-of-the-art defense method against word-level attack. (3) frequency-based word substitution (FGWS) Mozes et al. (2021): a detection-based method that leverages word frequency. (4) Word-level Differential Reaction (WDR): a detection-based method that leverages prediction logits. (5)

Model	Defense	IMDB					AGNews					Yahoo				
		No	GSA	PWWS	GA	IGA	No	GSA	PWWS	GA	IGA	No	GSA	PWWS	GA	IGA
Text CNN	NT	88.7	13.3	4.4	7.1	2.3	92.3	45.5	37.5	36.0	30.0	68.4	19.6	10.3	13.7	8.5
	IBP	78.6	72.5	72.5	71.5	65.0	89.4	80.0	80.5	80.5	80.0	64.2	49.4	52.6	59.2	61.0
	FGWS	84.7	70.4	68.0	70.2	64.0	84.7	80.0	68.5	70.2	73.4	65.0	45.0	48.0	52.5	56.8
	WDR	85.4	76.5	70.5	71.0	66.5	85.5	81.2	73.4	75.4	79.5	66.0	49.0	52.0	59.5	61.5
	SEM	86.8	66.4	71.1	72.5	72.4	89.7	86.0	86.0	85.0	86.0	65.8	61.0	60.5	63.0	63.0
	TS	88.0	75.5	76.5	77.5	75.0	90.0	88.5	89.0	88.5	88.0	67.2	60.8	60.0	62.5	62.5
LSTM	NT	87.3	8.3	2.2	5.0	3.0	92.6	35.0	30.0	29.0	26.5	71.6	27.6	21.1	15.8	5.5
	IBP	79.5	70.0	70.0	69.0	64.5	86.3	79.5	79.5	76.5	79.5	60.2	35.0	33.5	30.0	31.5
	FGWS	80.9	60.4	57.1	54.9	55.0	87.5	75.5	75.0	73.0	76.0	68.0	36.0	50.2	45.5	52.5
	WDR	80.3	61.5	59.0	58.0	59.0	87.0	75.5	76.0	75.0	78.0	66.5	36.0	51.0	45.0	54.0
	SEM	86.8	72.2	77.0	77.0	73.8	90.9	85.5	86.5	85.0	85.5	69.0	48.6	54.9	66.2	62.5
	TS	85.5	75.0	78.0	78.0	76.5	91.4	87.0	87.0	87.5	87.0	69.5	64.0	64.0	67.0	66.5
BERT	NT	92.3	24.5	40.7	40.0	28.0	94.6	66.5	68.0	58.5	45.5	77.7	31.3	34.3	15.7	10.0
	IBP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	FGWS	88.7	52.2	48.9	55.0	45.0	93.5	70.2	72.3	75.0	70.0	75.8	50.5	50.6	55.5	50.0
	WDR	88.5	52.0	50.0	54.5	46.5	93.0	70.5	72.0	75.5	70.5	76.5	51.8	52.0	56.4	51.6
	SEM	89.5	89.3	89.3	89.3	85.5	94.1	88.5	88.5	88.5	88.5	76.2	66.8	66.8	66.4	62.0
	TS	88.0	90.0	90.5	90.5	86.5	94.0	90.2	90.0	90.0	89.5	77.0	66.5	67.0	67.0	63.5

Table 2: The accuracy (%) of various models on benign sentences and adversarial sentences generated by four attacks. No: no attack, NT: Normal Training, TS: TextShield. Moreover, for a fair comparison, as the previous work Shi & et al (2019) reported that BERT is too difficult to be tightly verified by IBP, we do not use IBP on BERT.

Adversarial Sparse Convex Combination (ASCC): a defense method that leverages convex hull to defend against adversarial word substitutions. Specifically, considering that (3)(4) are detection methods and lack the capacity of producing true labels on adversarial sentences. Furthermore, the baselines' settings are kept the same with corresponding papers.

To validate that TextShield is effective across different victim models, we use TextCNN Kim (2014), LSTM Hochreiter & Schmidhuber (1997), and BERT Devlin & et al (2019) as victim classifiers. For TextCNN and LSTM, we use Glove Pennington et al. (2014) for the pre-trained embedding with a size of 300. Specifically, TextCNN has three convolutional layers with the filter sizes of 3, 4, and 5, a dropout layer (dropout rate $= 0.2$), and a fully connected layer. The LSTM-based victim classifier has two layers, each with 128 cells. BERT is 'Bert-base' with the checkpoint from Huggingface.

6.3 Performance analysis

Firstly, we generate test data, where 1000 benign sentences are sampled from each benchmark, and 1000 adversarial sentences are generated by each text attack based on each benchmark. Then we evaluate the performance of the victim models with or without defense methods. The more powerful a defense method is, the better classification accuracy on adversarial sentences is. Table 2 illustrates the performances of various defense methods on different test data.

Generalization When evaluated on benign sentences, adversarial training (AT) achieves the best performances in most cases, which indicates AT can effectively enhance model's generalization. IBP achieves much lower accuracy on benign data due to its high complexity and relatively loose bounds. Our proposed method, TextShield, achieves an accuracy close to normal training (NT), which has a good trade-off between robustness and generalization. Such a trade-off is common for attack/defense, which has been theoretically investigated in the CV field Zhang & et al (2019).

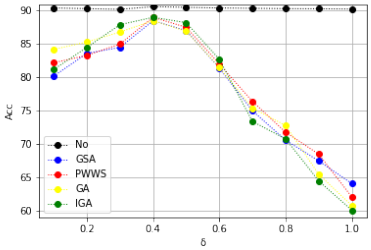
Adversarial robustness When evaluated on adversarial sentences, TextShield performs best in most cases, which demonstrates that TextShield can effectively improve model's robustness. AT is extremely powerless to defend against text attacks. One possible reason is the requirement of the diversity of re-training data Ren & et al (2019), and such a requirement is unable to meet due to limited adversarial sentences. IBP which were first proposed in the CV domain is more suitable for TextCNN and does not perform excellently on other victim classifiers (i.e., LSTM and BERT). Since FGWS focuses only on word frequencies, it is not sufficient to detect adversarial examples. Among all baselines, SEM achieves the best results, but it still cannot perform as well as TextShield.

E.g, TextShield outperforms SEM by averagely 8.0%, 4.8%, and 2.8% on the three benchmarks, respectively.

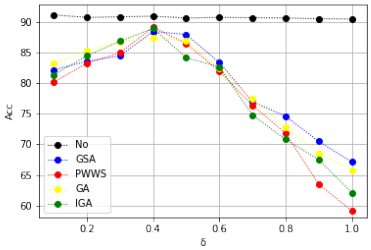
7 Ablation Study

This section conducts ablation studies of TextShield by answering three questions. Q1: What is the impact of hyper-parameters? Q2: What is the impact of multi-detectors? Q3: What is the impact of training samples in training detectors? Specifically, the last two questions are deferred to Appendix A and B, respectively.

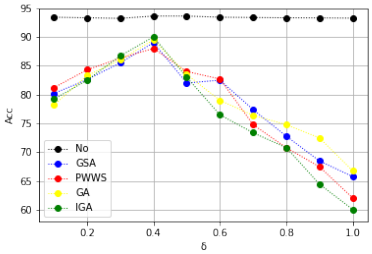
We study the impact of β in the corrector. As shown in Figure 3, the best performance achieves when β is 0.4, and either a small or large β will cause a significant performance drop, especially when β is large. This is because β serves to control the ratio of suspects in a sentence: when β is 0, only the most important word is considered as a suspect, which may be inadequate; on the contrary, when β is 1.0, every word is viewed as a suspect, which is inappropriate.



(a) TextCNN



(b) LSTM



(c) BERT

Figure 3: Accuracy (%) of TextShield towards different text attacks when β varies from 0.1 to 1.0. Specifically, the performance of the three victim classifiers on AGNews is presented.

8 Conclusion

This paper proposes TextShield, a detect-correct pipelined defense against word-level text attacks. We design TextShield by exploring one fundamental question step by step: how to distinguish adversarial and benign sentences. Based on empirical findings and basic derivations, we propose AWI, adaptive word importance that reflects a word's importance in specific scenarios. Based on AWI, we design saliency-based detectors for TextShield, which significantly enhances the detect-correct paradigm defense. Comprehensive experiments demonstrate that TextShield is superior to SoTA baselines when defending against text attacks. TextShield achieves promising defense performance by leveraging a cross-cutting connection between adversarial robustness and saliency information in the explainability of DNNs, and we believe our study sheds light on the adversarial robustness field in text mining. Moreover, such a pipeline is not specific to text classification, and we will explore the capacity of TextShield on other NLP tasks (e.g., QA, NLI) in the future.

REFERENCES

Moustafa Alzantot and et al. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2890–2896, 2018.

Sebastian Bach and et al. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

Alexander Binder and et al. Layer-wise relevance propagation for neural networks with local renormalization layers. In *International Conference on Artificial Neural Networks*, pp. 63–71. Springer, 2016.

Zhijie Deng, Xiao Yang, Shizhen Xu, Hang Su, and Jun Zhu. Libre: A practical bayesian approach to adversarial detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 972–982, 2021.

Jacob Devlin and et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pp. 4171–4186, 2019.

Dong and et al. Towards robustness against natural language word substitutions. *ICLR*, 2021.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

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

Po-Sen Huang and et al. Achieving verified robustness to symbol substitutions via interval bound propagation. In *EMNLP*, pp. 4083–4093, 2019.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. Certified robustness to adversarial word substitutions. In *EMNLP*, pp. 4129–4142, 2019.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 8018–8025, 2020.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. Robust encodings: A framework for combating adversarial typos. In *ACL*, pp. 2752–2765, 2020.

Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pp. 1746–1751, 2014.

Ching-Yun Ko and et al. Popqorn: Quantifying robustness of recurrent neural networks. In *ICML*, pp. 3468–3477. PMLR, 2019.

Volodymyr Kuleshov and et al. Adversarial examples for natural language classification problems. 2018.

Shasha Li, Shitong Zhu, Sudipta Paul, Amit Roy-Chowdhury, Chengyu Song, Srikanth Krishnamurthy, Ananthram Swami, and Kevin S Chan. Connecting the dots: Detecting adversarial perturbations using context inconsistency. In *European Conference on Computer Vision*, pp. 396–413. Springer, 2020.

Jiayang Liu, Weiming Zhang, Yiwei Zhang, Dongdong Hou, Yujia Liu, Hongyue Zha, and Nenghai Yu. Detection based defense against adversarial examples from the steganalysis point of view. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4825–4834, 2019.

Edoardo Mosca, Shreyash Agarwal, Javier Rando Ramírez, and Georg Groh. "that is a suspicious reaction!": Interpreting logits variation to detect nlp adversarial attacks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7806–7816, 2022.

Maximilian Mozes, Pontus Stenetorp, Bennett Kleinberg, and Lewis Griffin. Frequency-guided word substitutions for detecting textual adversarial examples. In *EACL*, pp. 171–186, 2021.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, pp. 1532–1543, 2014.

Christopher Potts. On the negativity of negation. In *Semantics and Linguistic Theory*, volume 20, pp. 636–659, 2010.

Yao Qin, Nicholas Frosst, Sara Sabour, Colin Raffel, Garrison Cottrell, and Geoffrey Hinton. Detecting and diagnosing adversarial images with class-conditional capsule reconstructions. In *International Conference on Learning Representations*, 2019.

Shuhuai Ren and et al. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 1085–1097, 2019.

Zhouxing Shi and et al. Robustness verification for transformers. In *International Conference on Learning Representations*, 2019.

J Springenberg, Alexey Dosovitskiy, Thomas Brox, and M Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 2015.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328. PMLR, 2017.

Boxin Wang and et al. Infobert: Improving robustness of language models from an information theoretic perspective. In *International Conference on Learning Representations*, 2020.

Xiaosen Wang and et al. Natural language adversarial defense through synonym encoding. In *Conference on Uncertainty in Artificial Intelligence*, 2021.

Xiaosen Wang, Hao Jin, and Kun He. Natural language adversarial attack and defense in word level. 2019.

Kaidi Xu and et al. Automatic perturbation analysis for scalable certified robustness and beyond. *NeurIPs*, 33, 2020.

Hongyang Zhang and et al. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pp. 7472–7482. PMLR, 2019.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *NeurIPs*, 28:649–657, 2015.

Yi Zhou, , and et al. Defense against adversarial attacks in nlp via dirichlet neighborhood ensemble. *ACL*, 2021.

A Appendix A: Analysis of ensembling multiple detectors

Since a crucial design of TextShield is the combination of various detectors to judge whether the input sentence is an adversarial one, we investigate the impact of such an ensembling strategy. Specifically, we verify the performance after removing some detectors, and the evaluation is launched using TextCNN on AGNews.

As shown in Table 3, removing any detector will cause a performance drop, demonstrating the effectiveness of each detector. Specifically, based on the results, the importance of detectors can be ranked as follows: VG¿Freq¿LRP¿IG¿GBP. Furthermore, removing all the detectors shows a catastrophic performance drop, which confirms our arguments that different detectors can complement well with each other and shows the importance of integrating four sub-detectors.

Please note that neither saliency computation methods nor the combination strategies of various detectors are our paper's focus. There are more sophisticated methods for saliency computation and detector combination, and such methods can also be applied for TextShield. In this paper, our main

claim is that, in text mining, model's robustness can be enhanced by leveraging a cross-cutting relation between adversarial robustness and saliency information used in DNN's explainability, which is contrary to the conventional opinion that text attacks should be defended using discrete information (e.g., synonym encoding) instead of continuous information (e.g., saliency).

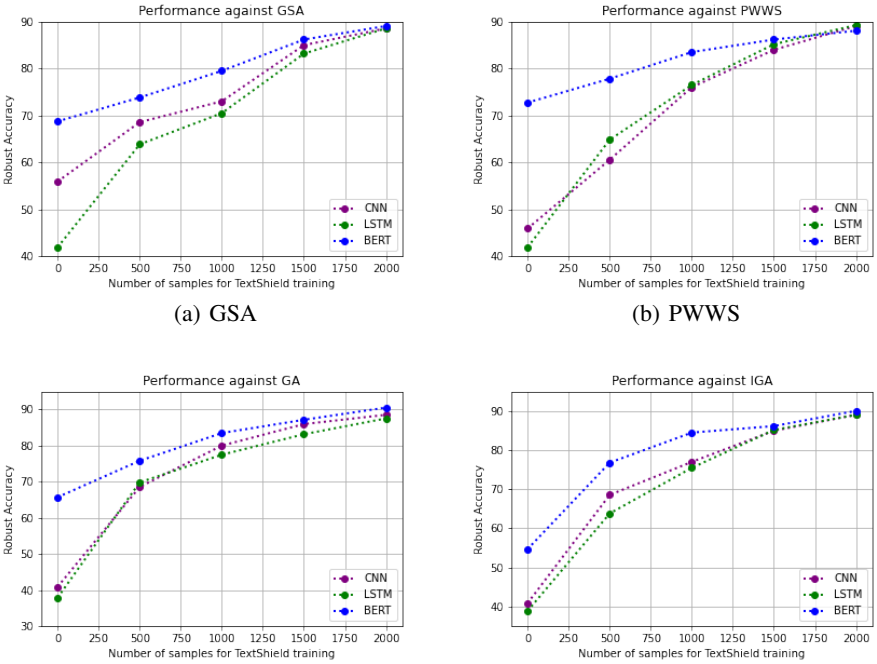
Removed detector	GSA	PWWS	GA	IGA
-VG detector	-8.9	-7.9	-6.4	-7.0
-IG detector	-6.8	-4.5	-3.0	-4.5
-LRP detector	-6.6	-5.0	-2.8	-4.0
-GBP detector	-7.4	-2.5	-4.1	-5.1
-All detectors	-34.8	-29.6	-40.3	-36.9

Table 3: Accuracy (%) drop after removing some detectors. E.g., '-VG' means removing the VG detector in TextShield.

B APPENDIX B: ANALYSIS OF THE NUMBER OF SENTENCES USED IN TRAINING DETECTORS

During the training process, we use the same number K of adversarial sentences per class for TextShield, where $K = 2000$. This section illustrates the relation between TextShield and K. Specifically, we keep the same training settings except for K. The performance of TextShield under different K is shown in Figure 4.

As shown in Figure 4, the defense capacities of TextShield are enhanced as K continuously increases since more samples used in the training help the detector recognize adversarial sentences better. Specifically, when K is set as 500, TextShield still significantly outperforms AT which uses 2000 adversarial sentences for re-training, further validating the effectiveness of TextShield.



(a) GSA (b) PWWS

(c) GA (d) IGA

Figure 4: Accuracy (%) of TextShield on adversarial sentences generated by four attacks as K changes.