SEPP: Similarity Estimation of Predicted Probabilities
for Defending and Detecting Adversarial Text

Hoang-Quoc Nguyen-Son, Seira Hidano, Kazuhide Fukushima, and Shinsaku Kiyomoto
KDDI Research, Inc.
2-1-15 Ohara, Fujimino, Saitama, 356-8502, Japan
{ho-nguyen, se-hidano, ka-fukushima, kiyomoto}@kddi-research.jp

Abstract

There are two cases describing how a classifier processes input text, namely, misclassification and correct classification. In terms of misclassified texts, a classifier handles the texts with both incorrect predictions and adversarial texts, which are generated to fool the classifier, which is called a victim. Both types are misunderstood by the victim, but they can still be recognized by other classifiers. This induces large gaps in predicted probabilities between the victim and the other classifiers. In contrast, text correctly classified by the victim is often successfully predicted by the others and induces small gaps. In this paper, we propose an ensemble model based on similarity estimation of predicted probabilities (SEPP) to exploit the large gaps in the misclassified predictions in contrast to small gaps in the correct classification. SEPP then corrects the incorrect predictions of the misclassified texts. We demonstrate the resilience of SEPP in defending and detecting adversarial texts through different types of victim classifiers, classification tasks, and adversarial attacks.

1 Introduction

Recent deep learning models have reached the human level in many NLP tasks. However, these models are sensitive to changes in the input data. An adversarial text can be generated from an original text while the original meaning is still preserved and bypasses human recognition. However, adversarial text can fool many victims, such as sentiment analysis (Ren et al., 2019), question answering (Jia and Liang, 2017), and search engines (Gil et al., 2019).

Popular adversarial text defenders are based on adversarial training (Shrivastava et al., 2017; Tramèr et al., 2018) or modification detection (Pruthi et al., 2019). N-gram (Juuti et al., 2018) and text similarity (Nguyen-Son et al., 2019) address the adversarial text detection problem. However, recent generators can generate adversarial text via a very small change from the original by replacing a few words (Ren et al., 2019; Jin et al., 2020), few characters (Gao et al., 2018; Jones et al., 2020), or both (Li et al., 2019). High duplication in word usage between the original and adversarial texts confuses both the existing defenders and detectors.

1.1 Motivation

Correct classification of text by a classifier often induces small gaps to other classifiers. An adversary can fool a victim classifier's predictions by generating misclassified text, but it does not fool other classifiers. For instance, we randomly choose correctly classified text t_1 and its adversarial text t_2 targeting a CNN classifier (Figure 1). The predictions are made with popular deep learning models including CNN (Kim, 2014), BiLSTM, BERT-large (Devlin et al., 2019), RoBERTa-large (Liu et al., 2019), and XLNet-large (Yang et al., 2019). The prediction is indicated by pair values of positive and negative prediction probabilities. The original text t_1 is negative, so the CNN and the other models correctly predict the text with higher negative than positive values. The adversarial text t_2 changes two words "*script, ace*" into their synonyms "*hand, genius*" using Ren et al. (2019)'s work. The generated text reduces the negative probability of the victim classifier to less than 0.5. However, using the synonym does not change the overall meaning, so the other

	Correctly classified text t_1/Adversarial text t_2:		Misclassified original text t_3:
Text	*Caddyshack II is NOTHING compared to the original Caddyshack. But, there are legitimate reasons for it. (1) Rodney Dangerfield was supposed to be the ace of this film BUT he didn't like the __script__/__hand__, wanted to change it, his request was denied, so he didn't do the film. (2) It was low budget, Bill Murray had grown to superstar status. Ted Knight passed away in 1986, and Chevy Chase the "so-called __ace__/__genius__" of the first movie (although it was Rodney all the way) couldn't be on more than 5 minutes, because it would cost too much to pay him. BUT you had Dan Aykroyd, Robert Stack, Randy Quaid and Jackie Mason, all serviceable substitutes, who none had their best performances.*		*I am not a big fan of the Spielberg/Cruise version of this film. And so I must throw in with the more humble Latt/Howel version. C Thomas Howel had more heart and more sympathy that Cruise in the lead role (at least in my opinion). Now this is hard to imagine until you strip away everything in the Spielberg version that cost more than a thousand dollars. There would be nothing left, no special effects, no sets, no Cruise.*
CNN (victim)	**(0.13, 0.87)** 🏴	**(0.68, 0.32)** 👍	**(0.58, 0.42)** 👍
Bi-LSTM	(0.05, 0.95) 🏴	(0.11, 0.89) 🏴	(0.12, 0.88) 🏴
BERT-large	(0.21, 0.79) 🏴	(0.22, 0.78) 🏴	(0.12, 0.88) 🏴
RoBERTa-large	(0.06, 0.94) 🏴	(0.23, 0.77) 🏴	(0.09, 0.91) 🏴
XLNet-large	(0.30, 0.70) 🏴	(0.33, 0.67) 🏴	(0.05, 0.95) 🏴

Figure 1: Predictions (positive, negative) based on sentiment analysis classifiers.

models mostly retain their negative predictions. We randomly select a negative text t_3, which is misclassified by the CNN victim, and observe that t_3 has the same characteristic as t_2. In particular, t_3 is predicted as positive by the victim, while other classifiers still predict it as negative. Based on the gaps in prediction probabilities among the classifiers, we can distinguish correctly classified text from misclassified text.

1.2 Contributions

In this paper, we proposed an ensemble model based on similarity estimation of predicted probabilities (SEPP) to defend adversarial texts. Unlike a basic ensemble model, which directly votes predictions from multiple classifiers, SEPP estimates the similarity in prediction probabilities from the classifiers. The similarity is used to identify the victim classifier and misclassified texts. The probabilities of misclassified texts are corrected by using predictions from other classifiers. We use the same technique to detect adversarial texts.

We conducted experiments with adversarial texts generated by the probability weighted word saliency generator (Ren et al., 2019) that fool the CNN-based sentiment analysis classifier. SEPP recovers the prediction from 22.9% to 94.0% on an adversarial dataset while keeping 96.6% on the clean dataset. This is better than the 89.6% and 92.6% achieved by adversarial training and ensemble baselines, respectively. Moreover, we detect the adversarial texts at a rate of 96.3%, which outperforms existing work, neural baselines, and ensemble baselines. Other experiments on BiLSTM and BERT yield similar results. SEPP also works well on multiple-class classification tasks and other adversarial attacks. In summary, our contributions are as follows:

- We determined that predictions of various classifiers for misclassified text differ from those of correctly classified text.

- We proposed an ensemble model using similarity estimation of predicted probabilities (SEPP) to detect a victim classifier and misclassified texts. We leveraged this detection to recover the prediction of the victim.

- We reused SEPP to distinguish adversarial text

from the original text.

- We evaluated the various adversarial texts, which fooled CNN, BiLSTM, and BERT classifiers on binary- and multiple-class classification tasks. The results indicate that SEPP outperforms other existing methods.

1.3 Roadmap

The rest of this paper is organized as follows. Section 2 describes related work on adversarial text generation, detection, and defense. Section 3 introduces the SEPP system. The experiential results are shown and analyzed in Section 4. Section 5 summarizes some main key points and mentions future work.

2 Related Work

2.1 Adversarial Text Generation

Adversarial text generation can be categorized by the extent of the generation:

2.1.1 Paragraph

Juuti et al. (2018) trained a neural model on human-written reviews and generated adversarial texts by topic. Jia and Liang (2017) added a noise sentence to an original paragraph to change a correct result of a question answering system. Wang et al. (2020) changed product categories of a review while keeping the sentiment but fooling a sentiment analysis classifier.

2.1.2 Sentence

Iyyer et al. (2018) generated an adversarial sentence with the desired syntax. They used back-translation to create a paraphrased sentence pair with different syntax. They then designed an attention network to convert a sentence into a paraphrase with the target syntax. Ren et al. (2020) combined VAE and GAN to generate large scale adversarial sentences for a limited training dataset. Han et al. (2020) generated a text using an RNN network targeting structured prediction models such as dependency parsing or POS tagger.

2.1.3 Phrase

Ribeiro et al. (2018) compiled paraphrased pairs at the phrase level. They then suggested a rule to replace individual phrases in an original text with corresponding phrases in the paraphrased pairs. Liang

et al. (2018) inserted or deleted consecutive hot words that affected the predictions of classifiers. Wallace et al. (2019) added a fixed phrase at the beginning of any sentence and optimized it by the gradient of a victim system. They claim that a phrase "zoning tapping fiences" reduces the victim's accuracy from 86.2% to 29.1% on positive samples.

2.1.4 Word

Adversarial text can be created by using various word operations (insertion, deletion, and replacement) to fool AI systems with both white-box and black-box attacks. As an example of a white-box attack, Ebrahimi et al. (2018) operated on hot words that induce a high gradient change in the system. As an example of a black-box attack, Liang et al. (2018) and Jin et al. (2020) examined occluded words and observed the prediction change. Garg and Ramakrishnan (2020) marked candidate words and chose the top ones predicted by a BERT model. Li et al. (2020) extended this idea for sub-words. Zhang et al. (2019) improved the fluency of word replacement by performing Metropolis-Hastings sampling. The chance of replacement is improved by using a genetic algorithm (Alzantot et al., 2018), particle swarm optimization (Zang et al., 2020), or boundary optimization (Meng and Wattenhofer, 2020). Ren et al. (2019) upgraded the text fluency with synonymous words in Wordnet and similar name entities.

2.1.5 Character

Many of the word-based approaches can be applied directly to characters (Liang et al., 2018; Ebrahimi et al., 2018). Moreover, Zhou et al. (2019) recovered the character replacement in an adversarial text. Gil et al. (2019) suggested a method based on a character operator targeting Google search scores. Pruthi et al. (2019), Jones et al. (2020), and Li et al. (2019) manipulated the middle characters of an individual word to preserve the text fluency.

Analysis: The *paragraph* approach generates flexible adversarial texts. The generation of large hard-to-read text makes it easily recognizable by the N-gram model and readability metrics (Juuti et al., 2018). The *sentence* approaches preserves the text meaning, but they induce significant changes in text complexity (Nguyen-Son et al., 2019). In the *phrase* approach, the rules become fragile when we gather

sufficient paraphrased pairs. The insertion and deletion of hot phrases into original text induces nonfluent text. The operators on *character* introduce misspellings. With the *word* operator, while insertion and deletion also lead to nonfluent text, the replacement produces fluent text. Among these replacements, the Wordnet-based approach (Ren et al., 2019) preserves more the original meaning than other replacements, which are based on word embedding (Li et al., 2020; Zang et al., 2020). Moreover, this replacement works well on many tasks (binary- or multiple-class classification) and is chosen to conduct main experiments in this paper.

2.2 Adversarial Text Defense

The most popular approach in the defense against adversarial text is adversarial training (Shrivastava et al., 2017), which was previously used in image processing. The adversarial texts were added to the training data before the classifier was retrained. Another approach estimated the similarity between original and adversarial texts on training data (Liu et al., 2020). The upper and lower bounds of adversarial data were also approximated (Ye et al., 2020; Huang et al., 2019; Jia et al., 2019) to alleviate such texts. Other defenses identified changes in adversarial texts from their origins at the character level (Jones et al., 2020; Pruthi et al., 2019) or word level (Zhou et al., 2019). The main drawback of previous approaches is that they need to retrain the classifier. Thus, they are sensitive to a new kind of adversarial text.

2.3 Adversarial Text Detection

Original text is generally more fluent than adversarial text. Existing methods estimate the fluency based on the N-gram model. Juuti et al. (2018) extracted the N-gram features based on a variety of text components, including word, part of speech, and syntactic dependency. They also measured the text readability using thirteen relative metrics. Our previous work (Nguyen-Son et al., 2019) extracted word N-gram features in both internal information from a training corpus and external information from a website corpus[1]. Text coherence was

measured by matching similar words and combining them with the N-gram features. Powerful deep learning models (e.g., BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and XLNet (Yang et al., 2019)) can be used as reputable detectors since they prove their performance in most of the major classification tasks.

Existing methods extract the difference in word usage between original and adversarial texts. However, recent adversarial texts produced only minimal changes from the original texts. Thus, they confuse all text-based methods.

3 Similarity Estimation of Predicted Probabilities

We proposed an ensemble model based on the similarity estimation of predicted probabilities system (SEPP) for defending against adversarial text, as shown in Figure 2.

3.1 Training phase

The objective of the training phase is to create two kinds of discriminators. A discriminator Ω_k detects misclassified texts for a classifier Γ_k. Another discriminator Ψ detects a victim among candidate classifiers.

3.1.1 Training Misclassification Discriminator Ω_k

We describe the training of a misclassification discriminator Ω_k for a victim classifier Γ_k in the following steps. The other misclassification discriminators is trained in the same manner way.

- ***Preparing training texts***: We run a victim classifier Υ_k to divide clean texts T_k into misclassified texts M_k and correctly classified texts C_k. Adversarial texts A_k are then generated from C_k by using an existing generator and are added to M_k. Each text t in M_k and C_k is used to extract features for training Ω_k (Algorithm 1).

- ***Measuring similarities:*** The probability $\hat{y}^c$ of the predicted class c in Υ_k is calculated with respect to its similarity to corresponding probabilities in other classifiers Γ_i. In particular, the similarity is the Manhattan distance of $\hat{y}^c$ and $\hat{y}_i^c$ (line 7).
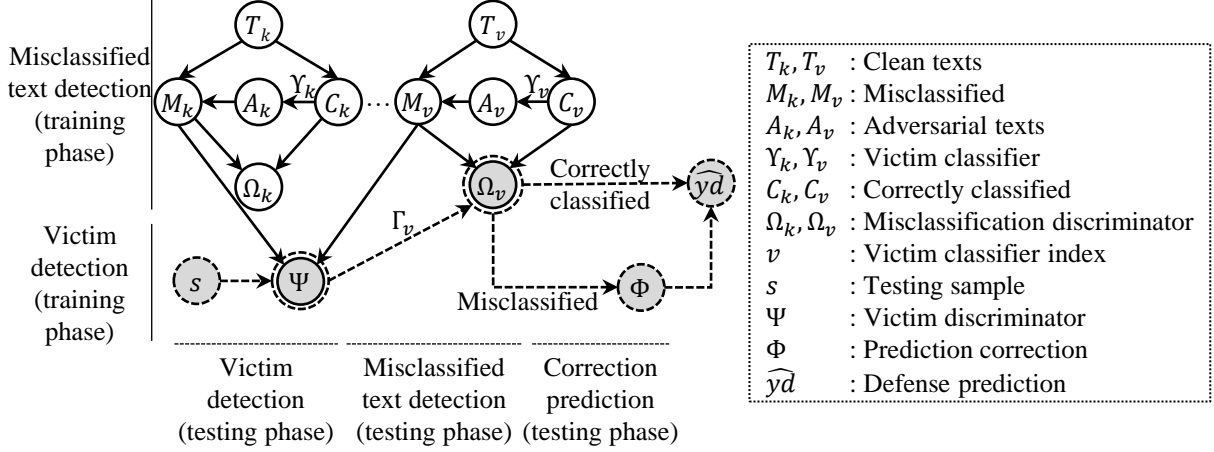
Figure 2: Similarity estimation of the predicted probability of defending against adversarial text. Training and testing are shown as solid and dashed lines, respectively.

Algorithm 1: Extracting features.

Input : text t; victim Υ_k;
other classifiers $\Gamma = \{\Gamma_i\}$
Output: extracted features

1   $\hat{y} = $ getPredict (Υ_k, t)
2   $c = \arg\max \hat{y}$
3   P $= \{\hat{y}_i = $ getPredict$(\Gamma_i, t)\}$
4   $\Lambda = \emptyset$ // similarity features
5   $\theta = 0$ // differences count feature
6   **for** $\hat{y}_i \in P$ **do**
7      $\Lambda = \Lambda \cup |\hat{y}^c - \hat{y}_i^c|$
8   **if** $\arg\max \hat{y}_i \neq c$ **then**
9         $\theta = \theta + 1$
10   **end if**
11 **end for**
12 **return** $\Lambda \cup \theta$

- **Counting different predictions:** We count predicted classes of other classifiers Γ_i that are different from the predicted class c of the victim, which calls the different prediction count θ (line 9).

- **Training the misclassification discriminator:** All similarities Λ and θ are input into a feedforward neural network to train Ω_k.

In Figure 1, t_1, CNN, and other classifiers can be used as t, Υ_k, and Γ_i, respectively. t_1 is run on these classifiers to obtain $(positive, negative)$ probabilities $\hat{y} = (0.13, \mathbf{0.87}), \hat{y}_1 = (0.05, \mathbf{0.95}), \cdots$. The

similarities are calculated as $\Lambda = (|\mathbf{0.87} - \mathbf{0.95}| = 0.08, |\mathbf{0.87} - 0.79| = 0.08, \cdots)$. All classifiers predict t_1 as negative; therefore, $\theta = 0$. With the small values in Λ and θ, t_1 is most likely to be determined as correctly classified text. With adversarial text t_2, the misclassified text should be detected with large values: $\Lambda = (0.57, \cdots), \theta = 4$. Similarly, t_3 should be considered misclassified text with $\Lambda = (0.56, \cdots), \theta = 4$.

3.1.2 Training Victim Discriminator Ψ

We use all misclassified texts to train a victim discriminator Ψ. Each text extracts individual features from a victim classifier in the same manner as above. The individual features are concatenated in order and input into another feedforward neural network to train Ψ. When we use t_2 as the input text, individual features $(0.57, 0.46, ..., 4)$ and $(0.57, 0.11, ..., 1)$... are extracted with Υ_1, Υ_2.... The concatenated features $(0.57, 0.46, ..., 4, 0.57, 0.11, ..., 1...)$ contain high values in the first individual features, so the first classifier should be identified.

3.2 Testing phase

A testing sample s of adversarial or original text is run with Ψ to determine the victim Υ_v. Then, the corresponding discriminator Ω_v determines whether s is a correct or misclassified text. If Ω_v determines s as the correctly classified sample, then, we retain the original prediction on Υ_v for the final defense prob-

ability. Otherwise, the defense probability is calculated by:

$$\hat{yd} = \frac{1}{n} \sum_{i=1}^{n} \hat{y_i}$$

where $\hat{y_i}$ is the probability from other classifiers Γ_i and n is the total number of the other classifiers.

For example, if Ψ identifies the victim Υ_1 of adversarial text t_2, Ω_1 detects t_2 as misclassified text. The prediction of t_2 is updated from positive with $\hat{y} = (\mathbf{0.58}, 0.42)$ to negative with: $\hat{yd} = \left(\frac{0.11 + 0.22 \cdots}{4}, \frac{0.89 + 0.78 \cdots}{4} \right) = (0.22, \mathbf{0.78})$. A similar flow should be processed with the misclassified text t_3. In the case of the correctly classified text t_1, because this kind of text is already learned via all misclassification discriminators, the correctly classified text should be identified with any victim detected by Ψ.

4 Evaluation

In this section, we present our experimental evaluation of defending and detecting adversarial texts.

4.1 Defending against Adversarial Texts

4.1.1 Dataset

We created adversarial texts by using the probability weighted word saliency (PWWS) generator (Ren et al., 2019) on the IMDB[2] (binary class) and AGNEWS[3] (four classes). We used testing data as a clean dataset. The adversarial texts are replaced with the original texts from the clean dataset to form an adversarial dataset. We use the ratio of 80/10/10 for training/developing/testing sets. This ratio is reused in further experiments.

4.1.2 Comparison

We compared SEPP[4] with adversarial training (Shrivastava et al., 2017) and ensemble baselines (Opitz and Maclin, 1999). While adversarial training adds adversarial texts and retrains the

---

[2] https://ai.stanford.edu/~amaas/data/sentiment/

[3] http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

[4] SEPP using five classifiers (Figure 1), separately trained on the IMDB with suggested configurations and obtained similar performance. For example, CNN and RoBERTa-large achieved 88.8% and 96.5% accuracies, respectively.

victims, ensemble learning votes on the predictions from the five individual classifiers (Figure 1). There are two popular ways to vote: average the predictions (soft) and select the majority class (hard). Table 1 lists the accuracy scores on testing sets while the developing sets reach similar values. SEPP can be trained with different training data (unknown), multiple training data (unsure), and the same training data (known). For example, if a victim is CNN, the different (resp. multiple, same) training data consists of misclassified and correctly classified texts, M_k and C_k (Figure 2), generated with BiL-STM (resp. both BiLSTM and CNN, and CNN).

The victim classifier declines significantly when moving from clean to adversarial data. Adversarial training efficiently defends against adversarial text, but it ignores the other misclassified texts. Ensemble learning appropriates this task in which adversarial text fools the victim classifier only but the other classifiers are still persistent. SEPP processes both kinds of misclassified texts and achieves high outcomes even with unknown victim classifiers. Moreover, SEPP (unsure) detects the victim classifiers more than 90% of accuracy.

4.1.3 Ablation Studies

We analyzed the contributions of the individual classifiers used in SEPP. The victim CNN is combined with the individual classifiers (Table 2). SEPP is presented with three groups of features: similarities Λ (SEPP-Λ), differences θ (SEPP-θ), and their combination (SEPP). The detection is affected by the performance of each model. In particular, BERT, RoBERTa, and XLNet are better than BiL-STM. SEPP improves both predictions in individual and combined features.

4.1.4 Attacking the BERT

We conducted other experiments (Table 3) targeting the BERT on SST-2 with various attacks at different text levels: character (DeepWordBug (Gao et al., 2018)), character and word (TextBugger (Li et al., 2019)), and word (TextFooler (Jin et al., 2020)). We reused all six pretrained SST-2 classifiers for the ensemble models from the TextAttack framework (Morris et al., 2020) including CNN, LSTM, BERT-base, DistilBERT-base, RoBERTa-base, and AlBERT-base. The change in a few SST-2 words

Method	IMDB				AGNEWS			
	CNN		BiLSTM		CNN		BiLSTM	
	Clean	Adv	Clean	Adv	Clean	Adv	Clean	Adv
Original (victim)	88.9	22.9	87.0	14.2	91.7	62.6	92.2	56.5
Adversarial training (known)	88.4	89.6	86.0	86.6	92.2	88.2	91.6	85.3
Ensemble (soft voting)	95.0	92.6	95.0	92.6	94.7	<u>94.3</u>	94.7	<u>94.2</u>
Ensemble (hard voting)	96.0	90.6	96.0	91.3	<u>96.3</u>	93.0	**96.3**	94.0
SEPP (unknown)	<u>96.3</u>	90.9	<u>96.3</u>	93.5	94.1	89.9	94.1	91.6
SEPP (unsure)	<u>96.3</u>	**94.8**	<u>96.3</u>	**94.1**	94.2	91.4	94.2	90.5
SEPP (known)	**96.6**	<u>94.0</u>	**96.6**	**94.1**	**96.5**	**95.7**	<u>96.2</u>	**94.6**

Table 1: Defending against adversarial texts targeting binary-class (IMDB) and multiclass (AGNEWS).

Combination	IMDB		AGNEWS	
	Clean	Adv	Clean	Adv
CNN+BiLSTM	87.3	68.0	92.2	83.0
CNN+BERT	95.2	91.9	95.9	94.5
CNN+RoBERTa	**97.0**	**95.7**	95.4	93.3
CNN+XLNet	95.0	92.6	94.7	94.3
SEPP-Λ	96.4	93.4	90.8	94.1
SEPP-θ	<u>96.6</u>	<u>94.1</u>	**96.6**	<u>95.5</u>
SEPP (both)	<u>96.6</u>	94.0	96.5	**95.7**

Table 2: Combination of classifiers and features in SEPP.

Figure 3: Learning rate estimation of BERT-large model in duplicate and unduplicate replacement generation of adversarial texts.

(8.7 words/text) leads to a remarkable change in classifiers' predictions and negatively affects ensemble models, especially in hard voting. However, SEPP retains the most efficient defenses across the attacks.

4.2 Detecting Adversarial Texts

4.2.1 Detecting Adversarial Texts with Duplicate Replacement

We integrated adversarial texts with the original texts to form adversarial/original pairs. These pairs are split into training/development/testing sets with the previous ratio (80/10/10). SEPP detects adversarial texts by extracting the same kind of features as when detecting misclassified texts (see misclassification discriminator Ω_k in Figure 2). We compared SEPP with existing methods in detecting adversarial text, deep neural, and ensemble baselines as shown in Table 4. The neural baselines were trained on large models with a batch size of 4, a maximum length of 512, and an epoch of 2. The learning rates were estimated in a range of $10e-7$
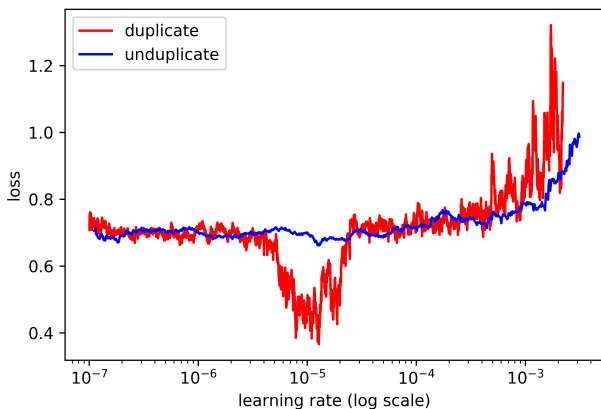
to $10e-2$. For example, Figure 3 shows the losses in the red line corresponding to the learning rates using the BERT-large model. An optimal learning rate of $1.28e-5$ was chosen when the loss was still decreasing, as recommended by Smith (2017). The number of training/test sets is shown in the second row.

The results show that the deep neural and ensemble baselines efficiently enhanced the traditional approaches by more than 10%. SEPP[5] achieves the highest performances in binary-class classification algorithms and reaches the competitive performances in multi-class classification.

4.2.2 Human Recognition

We randomly chose 50 adversarial/original pairs in the development set for human recognition. They

---

[5] Our source code is available at the following link (`https://github.com/quocnsh/SEPP`)

Method	Clean	DeepWordBug	TextBugger	TextFooler
Victim	88.6	10.2	14.8	2.27
Adversarial training	88.6	45.5	67.4	54.5
Ensemble (soft voting)	87.5	<u>54.6</u>	<u>69.3</u>	<u>58.0</u>
Ensemble (hard voting)	**89.8**	40.9	60.2	43.2
SEPP	**89.8**	**55.7**	**72.7**	**63.4**

Table 3: Attacking BERT on SST-2.

| Method | IMDB | | AGNEWS | |
	CNN	**BiLSTM**	**CNN**	**BiLSTM**
#train/#test	26531/3316	30046/3756	3376/422	2538/317
N-gram	81.3	83.0	70.1	69.6
Complexity	80.0	82.9	73.2	68.7
BERT-large	92.7	91.5	89.3	88.7
RoBERTa-large	<u>95.0</u>	94.9	88.4	93.8
XLNet-large	94.6	94.9	91.0	92.3
Ensemble (soft voting)	94.6	<u>96.7</u>	**94.8**	**97.0**
Ensemble (hard voting)	94.8	95.2	92.9	95.4
SEPP	**96.3**	**97.6**	<u>94.3</u>	<u>96.8</u>

Table 4: Detecting adversarial texts with duplicate replacement.

were shuffled, and each text was displayed to 11 raters who decided whether it was written by a human or generated by a machine[6]. The raters recognized the adversarial texts with 62.1% accuracy on average with a low agreement ($\kappa = -0.039$). This recognition accuracy was lower than those of all machine detectors. This demonstrates that we need a detector to assist us in recognizing such texts.

4.2.3 Detecting Adversarial Texts with Unduplicated Replacement

We analyzed the PWWS generator and found that it uses a large number of duplicate word replacements to generate adversarial texts. In particular, each replacement in a developing text was reused in 1544.3 texts on average in training texts. We clustered the texts in the development set in ranges of the number of duplicate replacements, as shown in Figure 4. We compared the detection of the top six methods. The low ranges significantly affected the deep learning baselines. In the high ranges, many duplicate replacements occurred with training data, offering more chances for detection with these mod-
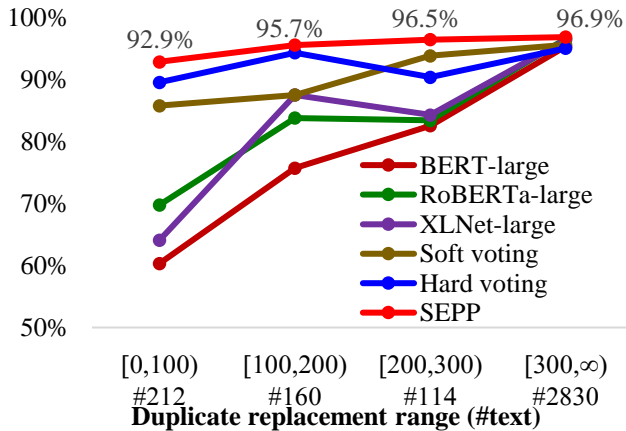
Figure 4: Detection of adversarial texts that fool CNN classifier. Duplicate replacement indicates number of replacements reused in training data.

els. However, since SEPP is independent of these replacements, we achieved resistant performances even in the low ranges.

We used PWWS to generate adversarial texts without reusing previous word replacements. We ran the detectors on this dataset (Table 5). While existing methods and deep neural baselines remained in the random guess range, SEPP and ensemble

Method	IMDB		AGNEWS	
	CNN	**BiLSTM**	**CNN**	**BiLSTM**
#train	1682	2028	972	1172
N-gram	51.9	52.7	55.6	56.8
Complexity	51.1	51.2	50.8	53.4
BERT-large	50.9	51.6	56.5	63.5
RoBERTa-large	50.0	54.0	52.4	50.0
XLNet-large	50.0	55.9	50.0	62.2
Ensemble (soft voting)	<u>89.1</u>	88.6	91.9	**96.6**
Ensemble (hard voting)	<u>89.1</u>	<u>89.0</u>	**94.4**	95.2
SEPP	**89.6**	**89.8**	<u>92.7</u>	<u>95.9</u>

Table 5: Detecting adversarial texts with unduplicated replacement.

baselines accuracy also maintained the prediction at around 92%. We analyzed the learning rate estimation process of the BERT-large model, as shown by the blue line (Figure 3). All of the losses were similar to a random line ($-ln(0.5) = 0.69$). The losses remained after many epochs of training.

5 Conclusion

In this paper, we propose an ensemble model based on similarity estimation of predicted probabilities (SEPP) for defending against adversarial text by detecting a victim classifier and correcting misclassified text. SEPP measures the similarity among predictions from multiple classifiers. We evaluated adversarial texts generated by word-based and/or character-based generators. The generated texts targeted popular classifiers (CNN, BiLSTM, and BERT) in a binary and a multiclass classification. The results show that SEPP outperformed the existing work not only in defending against adversarial texts but also in maintaining performance on clean texts. Moreover, we achieved better performance in detecting adversarial texts than existing detectors.

Based on the generalization of the proposed method, we can straightforwardly apply it for detecting other adversarial data such as fake images or forged audio.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *EMNLP*, pages 2890–2896.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *ACL*, pages 31–36.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *SPW*, pages 50–56.

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification. In *EMNLP*, pages 6174–6181.

Yotam Gil, Yoav Chai, Or Gorodissky, and Jonathan Berant. 2019. White-to-black: Efficient distillation of black-box adversarial attacks. In *NAACL*, pages 1373–1379.

Wenjuan Han, Liwen Zhang, Yong Jiang, and Kewei Tu. 2020. Adversarial attack and defense of structured prediction models. In *EMNLP*, pages 2327–2338.

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. 2019. Achieving verified robustness to symbol substitutions via interval bound propagation. In *EMNLP*, pages 4074–4084.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *NAACL*, pages 1875–1885.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*, pages 2021–2031.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *EMNLP*, pages 4120–4133.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *AAAI*, pages 8018–8025.

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

Mika Juuti, Bo Sun, Tatsuya Mori, and N Asokan. 2018. Stay on-topic: Generating context-specific fake restaurant reviews. In *ESORICS*, pages 132–151.

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

J Li, S Ji, T Du, B Li, and T Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *NDSS*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. In *EMNLP*, pages 6193–6202.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep text classification can be fooled. In *IJCAI*, pages 4208–4215.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv*.

Hui Liu, Yongzheng Zhang, Yipeng Wang, Zheng Lin, and Yige Chen. 2020. Joint character-level word embedding and adversarial stability training to defend adversarial text. In *AAAI*, pages 8384–8391.

Zhao Meng and Roger Wattenhofer. 2020. A geometry-inspired attack for generating natural language adversarial examples. In *COLING*.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *EMNLP*, pages 119–126.

Hoang-Quoc Nguyen-Son, Tran Phuong Thao, Seira Hidano, and Shinsaku Kiyomoto. 2019. Identifying adversarial sentences by analyzing text complexity. In *PACLIC*, pages 182–190.

David Opitz and Richard Maclin. 1999. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198.

Danish Pruthi, Bhuwan Dhingra, and Zachary C Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *ACL*, pages 5582–5591.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *ACL*, pages 1085–1097.

Yankun Ren, Jianbin Lin, Siliang Tang, Jun Zhou, Shuang Yang, Yuan Qi, and Xiang Ren. 2020. Generating natural language adversarial examples on a large scale with generative models. In *ECAI*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *ACL*, pages 856–865.

Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. 2017. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, pages 2107–2116.

Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In *WACV*, pages 464–472.

F Tramèr, D Boneh, A Kurakin, I Goodfellow, N Papernot, and P McDaniel. 2018. Ensemble adversarial training: Attacks and defenses. In *ICLR*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *EMNLP*.

Tianlu Wang, Xuezhi Wang, Yao Qin, Ben Packer, Kang Li, Jilin Chen, Alex Beutel, and Ed Chi. 2020. Cat-gen: Improving robustness in nlp models via controlled adversarial text generation. In *EMNLP*, pages 5141–5146.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xl-net: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5753–5763.

Mao Ye, Chengyue Gong, and Qiang Liu. 2020. Safer: A structure-free approach for certified robustness to adversarial word substitutions. In *ACL*, pages 3465–3475.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *ACL*, pages 6066–6080.

Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. 2019. Generating fluent adversarial examples for natural languages. In *ACL*, pages 5564–5569.

Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang, and Wei Wang. 2019. Learning to discriminate perturbations for blocking adversarial attacks in text classification. In *EMNLP*, pages 4906–4915.