Using Random Perturbations to Mitigate Adversarial Attacks on Sentiment Analysis Models

Abigail Swenor and **Jugal Kalita**
University of Colorado Colorado Springs
1420 Austin Bluffs Pkwy
Colorado Springs, CO 80918
aswenor,jkalita@uccs.edu

Abstract

Attacks on deep learning models are often difficult to identify and therefore are difficult to protect against. This problem is exacerbated by the use of public datasets that typically are not manually inspected before use. In this paper, we offer a solution to this vulnerability by using, during testing, random perturbations such as spelling correction if necessary, substitution by random synonym, or simply dropping the word. These perturbations are applied to random words in random sentences to defend NLP models against adversarial attacks. Our Random Perturbations Defense and Increased Randomness Defense methods are successful in returning attacked models to similar accuracy of models before attacks. The original accuracy of the model used in this work is 80% for sentiment classification. After undergoing attacks, the accuracy drops to accuracy between 0% and 44%. After applying our defense methods, the accuracy of the model is returned to the original accuracy within statistical significance.

1 Introduction

Deep learning models have excelled in solving difficult problems in machine learning, including Natural Language Processing (NLP) tasks like text classification (Zhang et al., 2015; Kim, 2014) and language understanding (Devlin et al., 2019). However, research has discovered that inputs can be modified to cause trained deep learning models to produce incorrect results and predictions (Szegedy et al., 2014). Models in computer vision are vulnerable to these attacks (Goodfellow et al., 2015), and studies have found that models in the NLP domain are also vulnerable (Kuleshov et al., 2018; Gao et al., 2018; Garg and Ramakrishnan, 2020). One use of these adversarial attacks is to test and verify the robustness of NLP models.

With the potential for adversarial attacks, there comes the need for prevention and protection. There are three main categories of defense methods: identification, reconstruction, and prevention (Goldblum et al., 2020). Identification methods rely on detecting either poisoned data or the poisoned model (Chen et al., 2019). While reconstruction methods actively work to repair the model after training (Zhu et al., 2020), prevention methods rely on input preprocessing, majority voting, and other techniques to mitigate adversarial attacks (Goldblum et al., 2020; Alshemali and Kalita, 2020). Although most NLP adversarial attacks are easily detectable, some new forms of adversarial attacks have become more difficult to detect like concealed data poisoning attacks (Wallace et al., 2021) and backdoor attacks (Chen et al., 2021). The use of these concealed and hard-to-detect attacks has revealed new vulnerabilities in NLP models. Considering the increasing difficulty in detecting attacks, a more prudent approach would be to work on neutralizing the effect of potential attacks rather than solely relying on detection. Here we offer a novel and highly effective defense solution that preprocesses inputs by random perturbations to mitigate potential hard-to-detect attacks.

2 Related Work

The work in this paper relates to the attack on NLP models using the TextAttack library (Morris et al., 2020), the current state-of-the-art defense methods for NLP models, and using randomness against adversarial attacks.

The TextAttack library and the associated GitHub repository (Morris et al., 2020) represent current efforts to centralize attack and data augmentation methods for the NLP community. The library supports attack creation through the use of four components: a goal function, a search method, a transformation, and constraints. An attack method

Proceedings of the 18th International Conference on Natural Language Processing, pages 519–528
Silchar, India. December 16 - 19, 2021. ©2021 NLP Association of India (NLPAI)

uses these components to perturb the input to fulfill the given goal function while complying with the constraints and the search method finds transformations that produce adversarial examples. The library contains a total of 16 attack model recipes based on literature. The work reported in this paper pertains to the 14 ready-to-use classification attack recipes from the TextAttack library. We believe that successful defense against such attacks will provide guidelines for the general defense of deep learning NLP classification models.

There are many methods to defend NLP models against adversarial attacks, including input preprocessing. Input preprocessing defenses require inserting a step between the input and the given model that aims to mitigate any potential attacks. Alshemali and Kalita (2020) use an input preprocessing defense that employs synonym set averages and majority voting to mitigate synonym substitution attacks. Their method is deployed before the input is run through a trained model. Another defense against synonym substitution attacks, Random Substitution Encoding (RSE) encodes randomly selected synonyms to train a robust deep neural network (Wang and Wang, 2020). The RSE defense occurs between the input and the embedding layer.

Randomness has been deployed in computer vision defense methods against adversarial attacks. Levine and Feizi (2020) use random ablations to defend against adversarial attacks on computer vision classification models. Their defense is based on a random-smoothing technique that creates certifiably robust classification. Levine and Feizi defend against sparse adversarial attacks that perturb a small number of features in the input images. They found their random ablation defense method to produce certifiably robust results on the MNIST, CIFAR-10, and ImageNet datasets.

3 Input Perturbation Approach & Adversarial Defense

The use and availability of successful adversarial attack methods reveal the need for defense methods that do not rely on detection and leverage intuitions gathered from popular attack methods to protect NLP models. In particular, we present a simple but highly effective defense against attacks on deep learning models that perform sentiment analysis.

The approach taken is based on certain assumptions about the sentiment analysis task. Given a short piece of text, we believe that a human does not need to necessarily analyze every sentence carefully to get a grasp on the sentiment. Our hypothesis is that humans can ascertain the expressed sentiment in a text by paying attention to a few key sentences while ignoring or skimming over the others. This thought experiment led us to make intermediate classifications on individual sentences of a review in the IMDB dataset and then combining the results for a collective final decision.

This process was refined further by considering how attackers actually perturb data. Usually, they select a small number of characters or tokens within the original data to perturb. To mitigate those perturbations, we choose to perform our own random perturbations. Because the attacking perturbations could occur anywhere within the original data, and we do not necessarily know where they are, it is prudent to randomly select tokens for us to perturb. This randomization has the potential to negate the effect the attacking perturbations have on the overall sentiment analysis.

We wish to highlight the importance of randomness in our approach and in possible future approaches for defenses against adversarial attacks. Positive impact of randomness in classification tasks with featured datasets can be found in work using Random Forests (Breiman, 2001). Random Forests have been useful in many domains to make predictions, including disease prediction (Lebedev et al., 2014; Corradi et al., 2018; Paul et al., 2017; Khalilia et al., 2011) and stock market price prediction (kha, 2019; Ballings et al., 2015; Nti et al., 2019). The use of randomness has made these methods of prediction robust and useful. We have chosen to harness the capability of randomness in defense of adversarial attacks in NLP. We demonstrate that the impact randomness has on our defense method is highly positive and its use in defense against adversarial attacks of neural networks should be explored further. We present two algorithms below—first with two levels of randomness, and the second with three.

3.1 Random Perturbations Defense

Our algorithm is based on random processes: the randomization of perturbations of the sentences of a review R followed by majority voting to decide the final prediction for sentiment analysis. We consider each review R to be represented as a set R = $\{r_1, r_2, ..., r_i, ..., r_N\}$ of sentences r_i. Once R

is broken down into its sentences (Line 1 of Algorithm 1), we create l replicates of sentence r_i: $\{\hat{r}_{i1}, ..., \hat{r}_{ij}, ..., \hat{r}_{il}\}$. Each replicate $\hat{r}_{ij}$ has k number of perturbations made to it. Each perturbation is determined randomly (Lines 4-7).

In Line 5, a random token t where $t \in \hat{r}_{ij}$ is selected, and in Line 6, a random perturbation is performed on t. This random perturbation could be a spellcheck with correction if necessary, a synonym substitution, or dropping the word. These perturbations were selected as they are likely to be the same operations an attacker performs, and they may potentially even counter the effect of a large portion of perturbations in attacked data. A spellcheck is performed using SpellChecker which is based in Pure Python Spell Checking. If a spellcheck is performed on a token without spelling error, then the token will not be changed. The synonym substitution is also performed in a random manner. A synonym set for token t is found using the WordNet synsets (Fellbaum, 1998). Once a synonym set is found, it is processed to remove any duplicate synonyms or copies of token t. Once the synonym set is processed, a random synonym from the set is chosen to replace token t in $\hat{r}_{ij}$. A drop word is when the randomly selected token t is removed from the replicate altogether and replaced with a space. Conceptually speaking, the random perturbations may be chosen from an extended set of allowed changes.

Once l replicates have been created for the given sentence r_i and perturbations made to tokens, they are put together to create replicate review set $\hat{R}$ (Line 8). Then, in Line 9, each $\hat{r}_{ij} \in \hat{R}$ is classified individually as $f(\hat{r}_{ij})$ using classifier $f()$. After each replicate has been classified, we perform majority voting with function $V()$. We call the final prediction that this majority voting results in as $\hat{f}(R)$. This function can be thought of as follows (Line 12):

$$\hat{f}(R) = V(\{f(\hat{r}_{ij}) \mid \hat{r}_{ij} \in \hat{R}\}).$$

The goal is to maximize the probability that $\hat{f}(R) = f(R)$ where $f(R)$ is the classification of the original review R. In this paper, this maximization is done through tuning of the parameters l and k. The certainty T for $\hat{f}(R)$ is also determined for each calculation of $\hat{f}(R)$. The certainty represents how sure the algorithm is of the final prediction it has made. In general, the certainty T is determined as follows (Lines 13-17):

$$T = count(f(\hat{r}_{ij}) == \hat{f}(R)) \ / \ N * l.$$

The full visual representation of this algorithm can be seen in Algorithm 1 and in Figure 1.

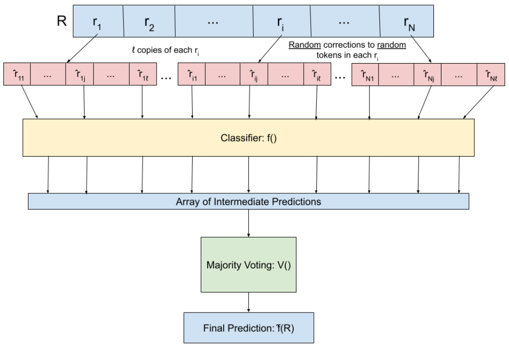


Figure 1: Visual representation of Algorithm 1.

3.2 Increasing Randomness

Our first algorithm represented in Algorithm 1 and in Figure 1 shows randomness in two key points in the decision making process for making the perturbations. This is the main source of randomness for our first algorithm. In our next algorithm, we introduce more randomness into our ideas from our original algorithm to create a modified algorithm. This more random algorithm is visually represented in Figure 2 and presented in Algorithm 2. This new defense method adds a third random process before making random corrections to a sentence. Randomly chosen r_i from R are randomly corrected to create replicate $\hat{r}_j$ which is placed in $\hat{R}$ (Lines 2-6). The original sentence r_i is placed back into R and a new sentence is randomly selected; this is random selection with replacement. This process of random selection is repeated until there is a total of k replicates $\hat{r}_j$ in $\hat{R}$. This algorithm follows the spirit of Random Forests more closely than the first algorithm.

In Line 2, we randomly select a sentence r_i from R. This is one of the main differences between Algorithm 1 for Random Perturbations Defense and Algorithm 2 for Increased Randomness Defense. That extra random element allows for more randomization in the corrections we make to create replicates $\hat{r}_j$. In Lines 3 and 4, the process is practically identical to Lines 5 and 6 in Algorithm 1. The only difference is that only one random correction is being made to get the final replicate $\hat{r}_j$

Algorithm 1: Random Perturbation Defense

```
Result: $\hat{f}(R)$, the classification of $R$ after
        defense
Input      : Review $R = \{r_1, r_2, ..., r_N\}$
             where $r_i$ is a sentence
Parameters : $l$ = number of copies made of
             each $r$, $k$ = number of
             corrections made per $r_i$,
             $C = \{c_1, c_2, ..., c_k\}$, set of
             corrections
$\hat{R} = \emptyset$
for $r_i \epsilon R$ do
    for $j$ = 1 to $l$ do
        $\hat{r}_{ij} = r_i$
        for $k$ do
            Select random token $t$ where
              $t \epsilon \hat{r}_{ij}$
            Perform random correction
              $c \epsilon C$ to $t$
        end
        Append $\hat{r}_{ij}$ to $\hat{R}$
        Classify: $f(r_{ij})$
    end
end
$\hat{f}(R) = V(\{f(\hat{r}_{ij}) \mid \hat{r}_{ij} \epsilon \hat{R}\})$, $V()$ is a
  voting function
if $f(\hat{R}) == negative$ then
    $T = count(f(\hat{r}_{ij}) ==$
      $negative) / N * l$
else
    $T = count(f(\hat{r}_{ij} == positive) / N * l$
end
```

Algorithm 2: Increased Randomness Defense

```
Result: $\hat{f}(R)$, the classification of $R$ after
        defense
Input      : Review $R = \{r_1, r_2, ..., r_N\}$
             where $r_i$ is a sentence
Parameters : $k$ = number of replicates $\hat{r}_j$
             made for $\hat{R}$,
             $C = \{c_1, c_2, ..., c_k\}$, set of
             corrections
$\hat{R} = \emptyset$, $P = []$
for $j$ = 1 to $k$ do
    Randomly select $r_i \in R$
    Select random token $t$ where $t \in r_i$
    Perform random correction $c \epsilon C$ to $t$ to
      get $\hat{r}_j$
    Append $\hat{r}_j$ to $\hat{R}$
end
for $j$ = 1 to $k$ do
    Classify: $f(\hat{r}_j)$
    Append results to predictions array $P$
end
$\hat{f}(R) = V(P)$, $V()$ is a voting function
if $f(\hat{R}) == negative$ then
    $T = count(f(\hat{r}_{ij}) ==$
      $negative) / N * l$
else
    $T = count(f(\hat{r}_{ij} == positive) / N * l$
end
```

for Increased Randomness Defense, while Random Perturbations Defense makes k random corrections to get the final replicate $\hat{r}_{ij}$.

3.3 Overcoming the Attacks

We define an attack as making random perturbations to an input, specifically for this work, a review R. We assume a uniform distribution for randomness. We interpret these random changes to occur throughout each review R with probability $\frac{1}{W}$ or $\frac{1}{N*m}$, where W is the number of words in R, N is the number of sentences in R, and m is the average length of each sentence in R. We refer to this probability that an attack makes changes to the review text as P_{attack} where a is the total number of perturbations made by the attack:

$$P_{attack} = \frac{a}{W} = \frac{a}{N * m}.$$

If each random perturbation performed by the attack has a probability of $\frac{1}{N*m}$, then our defense method needs to overcome that probability to overcome the attack.

Our two defense methods, Random Perturbations Defense and Increased Randomness Defense, both offer ways to overcome the attack, i.e., undo the attack change, with a probability greater than $\frac{a}{N*m}$.

Proposition 1 *Random Perturbations Defense overcomes an attack that makes a small number of random perturbations to a review document by having a probability greater than the attack probability* P_{attack}.

Our Random Perturbations Defense picks a random token t from each sentence $r_i \in R$ and repeats k

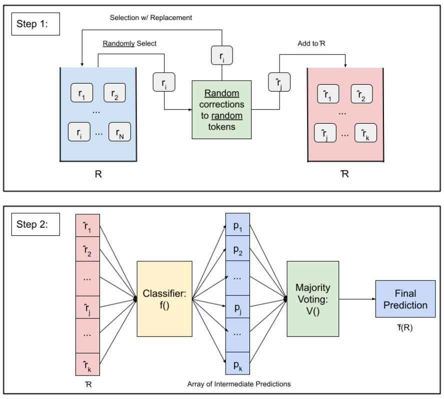


Figure 2: Visual representation of Algorithm 2 that includes more randomness.

times to get a final replicate $\hat{r}_{ij}$. This gives an initial probability that the defense picks a certain token from the text, or P_{RPD}, to be:

$$P_{RPD} = \frac{N * l * m!}{k!(m-k)!}.$$

We find this probability from choosing k tokens from r_i with length m which breaks down to a binomial coefficient $\binom{m}{k} = \frac{m!}{k!(n-k)!}$. This is then repeated l times for each sentence in R which equates to that initial probability being multiplied by l and N. After doing some rearranging of the probabilities, we can see that for certain values of l and k where $k < m$:

$$\mathrm{P}_{RPD} = \frac{N^2 m^2 l(m-1)(m-2)...(m-k+1)}{k!} > a.$$

P_{RPD} now is the total probability that the defense makes random changes to lN tokens. We know that $W = N * m$, that $a =< W$ for the attack methods we are testing against, and that k should be selected so that $k << W$. This means that we know $W^2 > a$, $W^2 > k!$, and $l(m-1)(m-2)...(m-k+1) > 0$ for the selected attack methods, which gives us the necessary conditions to assert that $P_{RPD} > Pattack$. Therefore, our Random Perturbations Defense will overcome the P_{attack} and should overcome the given attack method as stated in Proposition 1.

Proposition 2 *Increased Randomness Defense overcomes an attack that makes a small number of random perturbations to a review document by having a probability greater than the attack probability* P_{attack}.

Our Increased Randomness Defense first chooses a random sentence r_i which is selected with probability $\frac{1}{N}$. Next, we choose a random word within that sentence which is selected with probability $\frac{1}{m}$. This gives us a probability for changes as follows:

$$P_{IRD} = \frac{1}{N} * \frac{1}{m} = \frac{1}{N * m}.$$

We can see that $P_{IRD} * a = P_{attack}$. We need to overcome the attack probability and we do this in two ways: we either find the attack perturbation by chance and reverse it, or we counterbalance the attack perturbation with enough replicates $\hat{r}_j$. With each replicate $\hat{r}_j$ created, we increase our probability P_{IRD} so that our final probability for our Increased Randomness Defense is as follows:

$$P_{IRD} = \frac{k}{N * m}.$$

As long as our selected parameter value for k is greater than the number of perturbation changes made by the attack method a, then $P_{IRD} > P_{attack}$ and our Increased Randomness Defense method will overcome the given attack method as stated in Proposition 2.

4 Experiments & Results

4.1 Dataset & Models

We used the IMDB dataset (Maas et al., 2011) for our experiments. Each attack was used to perturb 100 reviews from the dataset. The 100 reviews were selected randomly from the dataset with a mix of positive and negative sentiments. Note that the Kuleshov attack data (Kuleshov et al., 2018) only had 77 reviews.

The models used in this research are from the TextAttack (Morris et al., 2020) and HuggingFace (Wolf et al., 2020) libraries. These libraries offer many different models to use for both attacked data generation and general NLP tasks. For this research, we used the *bert-base-uncased-imdb* model that resides in both the TextAttack and HuggingFace libraries. This model was fine-tuned and trained with a cross-entropy loss function. This model was used with the API functions of the TextAttack library to create the attacked reviews from each of the attacks we used. We chose this model because BERT models are useful in many NLP tasks and this model specifically was fine-tuned for text classification and was trained on the dataset we wanted to use for these experiments.

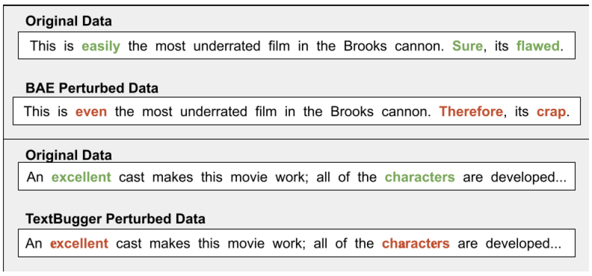
The HuggingFace library was also used in the sentiment-analysis classification of the attacked data and the defense method. We used the HuggingFace transformer pipeline for sentiment-analysis to test our defense method. This pipeline returns either "negative" or "positive" to classify the sentiment of the input text and a score for that prediction (Wolf et al., 2020). This pipeline was used to classify each replicate $\hat{r}_{ij}$ in our algorithm and is represented as the function $f()$.

4.2 Experiments

The attacks from the TextAttack library were used to generate attack data. Attack data was created from 7 different models from the library: BERT-based Adversarial Examples (BAE) (Garg and Ramakrishnan, 2020), DeepWordBug (Gao et al., 2018), FasterGeneticAlgorithm (Jia et al., 2019), Kuleshov (Kuleshov et al., 2018), Probability Weighted Word Saliency (PWWS) (Ren et al., 2019), TextBugger (Li et al., 2019), and TextFooler (Jin et al., 2020) (Morris et al., 2020). Each of these attacks were used to create 100 perturbed sentences from the IMDB dataset (Maas et al., 2011). These attacks were chosen from the 14 classification model attacks because they represent different kinds of attack methods, including misspelling, synonym substitution, and antonym substitution.

Each attack method used for our experiments has a slightly different approach to perturbing the input data. Each perturbation method is unique and follows a specific distinct pattern and examples of these can be found in Figure 3. The BAE attack determines the most important token in the input and replaces that token with the most similar replacement using a Universal Sentence Encoder. This helps the perturbed data remain semantically similar to the original input (Garg and Ramakrishnan, 2020). The DeepWordBug attack identifies the most important tokens in the input and performs character-level perturbations on the highest-ranked tokens while minimizing edit distance to create a change in the original classification (Gao et al., 2018). The FasterGeneticAlgorithm perturbs every token in a given input while maintaining the original sentiment. It chooses each perturbation carefully to create the most effective adversarial example (Jia et al., 2019). The Kuleshov attack is a synonym substitution attack that replaces 10% - 30% of the tokens in the input with synonyms that do not change the meaning of the input (Kuleshov et al., 2018).

The PWWS attack determines the word saliency score of each token and performs synonym substitutions based on the word saliency score and the maximum effectiveness of each substitution (Ren et al., 2019). The TextBugger attack determines the important sentences from the input first. It then determines the important words in those sentences and generates 5 possible "bugs" through different perturbation methods: insert, swap, delete, sub-c (visual similarity substitution), sub-w (semantic similarity substitution). The attack will implement whichever of these 5 generated bugs is the most effective in changing the original prediction (Li et al., 2019). Finally, the TextFooler attack determines the most important tokens in the input using synonym extraction, part-of-speech checking, and semantic similarity checking. If there are multiple canididates to substitute with, the most semantically similar substitution will be chosen and will replace the original token in the input (Jin et al., 2020).



Original Data

This is easily the most underrated film in the Brooks cannon. Sure, its flawed.

BAE Perturbed Data

This is even the most underrated film in the Brooks cannon. Therefore, its crap.

Original Data

An excellent cast makes this movie work; all of the characters are developed...

TextBugger Perturbed Data

An excellent cast makes this movie work; all of the characters are developed...

Figure 3: Example of what original data looks like and how the BAE (Garg and Ramakrishnan, 2020) and TextBugger (Li et al., 2019) attack methods perturb data. The BAE attack method uses semantic similarity, while the Textbugger attack method uses visual similarity.

After each attack had corresponding attack data, the TextAttack functions gave the results for the success of the attack. The accuracy of the sentiment-analysis task under attack, without the defense method, is reported in the first column in Table 1. Each attack caused a large decrease in the accuracy of the model. The model began with an average accuracy of 80% for the IMDB dataset. Once the attack data was created and the accuracy under attack was reported, the attack data was run through our Random Perturbations and Increased Randomness defense methods. All of the experiments were run on Google Colaboratory using TPUs and the Natural Language Toolkit (Loper and Bird, 2002).

4.3 Results

We began by testing on the HuggingFace sentiment analysis pipeline with the original IMDB dataset.

This gave an original accuracy of 80%. This percentage represents the goal for our defense method accuracy as we aim to return the model to its original accuracy, or higher. The accuracy under each attack is listed in Table 1 in the first column. These percentages show how effective each attack is at causing misclassification for the sentiment analysis task. The attacks range in effectiveness with PWWS (Ren et al., 2019) and Kuleshov (Kuleshov et al., 2018) with the most successful attacks at 0% accuracy under attack and FasterGeneticAlgorithm (Jia et al., 2019) with the least successful attack at 44% accuracy under attack, which is still almost a 40% drop in accuracy.

Attack	w/o Defense	w/ Defense
BAE	33%	80.80%±1.47
DeepWordBug	34%	76.60%±1.85
FasterGeneticAlgo	44%	82.20%±1.72
Kuleshov*	0%	60.00%±2.24
PWWS	0%	81.80%±1.17
TextBugger	6%	79.20%±2.32
TextFooler	1%	83.20%±2.48

Table 1: Accuracy for each of the attack methods under attack, and under attack with the defense method from Algorithm 1 deployed with $l = 7$ and $k = 5$. The accuracy prior to attack is 80%.

4.3.1 Random Perturbations Defense

For the Random Perturbations Defense to be successful, it is necessary to obtain values of the two parameters, l and k. Each attack was tested against our Random Perturbations Defense 5 times. The accuracy was averaged for all 5 tests and the standard deviation was calculated for the given mean. The mean accuracy with standard deviation is presented for each attack in the second column of Table 1. The results presented are for $l = 7$ and $k = 5$. These parameters were chosen after testing found greater values of l and k resulted in a longer run time and too many changes made to the original input; with lower values for l and k, the model had lower accuracy and not enough perturbations to outweigh any potential adversarial attacks. The values behind this logic can be seen in Table 2.

Attack	l	k	Accuracy w/ Defense
BAE	5	2	55%
BAE	10	5	50%
BAE	7	5	79%

Table 2: This table explains values of l and k

The defense method was able to return the model to original accuracy within statistical significance while under attack for most of the attacks with the exception of the Kuleshov method (Kuleshov et al., 2018). The accuracy for the other attacks all were returned to the original accuracy ranging from 76.00% to 83.20% accuracy with the Random Perturbations defense deployed. This shows that our defense method is successful at mitigating most potential adversarial attacks on sentiment classification models. Our defense method was able to increase the accuracy of model while under attack for the FasterGeneticAlgorithm, PWWS, and TextFooler. These three attack methods with our defense achieved accuracy that was higher than the original accuracy with statistical significance.

4.3.2 Increased Randomness Defense

The Increased Randomness Defense was also tested on all seven of the attacks. Each attack was tested against this defense 5 times. The results for these experiments can be seen in Table 4. There were tests done to determine what the proper value for k should be. These tests were performed on the BAE (Garg and Ramakrishnan, 2020) attack and the results can be found in Table 3. These tests revealed that 40-45 replicates $\hat{r}_j$ was ideal for each $\hat{R}$ with $k = 41$ being the final value used for the tests on each attack. This defense method was more efficient to use.

Attack	k	Accuracy w/ Defense
BAE	10	67%
BAE	20	76%
BAE	25	72%
BAE	30	76%
BAE	35	74%
BAE	40	82%
BAE	45	74%
BAE	41	77%

Table 3: This table shows the results for the tests for different values of k for the increased randomness experiments.

The runtime and the resources used for this method were lower than the original random perturbations defense method with the runtime for the Random Perturbations Defense being nearly 4 times longer than this increased random method. A comparison of the two defense methods on the seven attacks tested can be seen in Figure 4. This defense was successful in returning the model to the original accuracy, within statistical significance, for most of the attacks with the exception of the Kuleshov attack (Kuleshov et al., 2018). A t-test was performed to determine the statistical significance of the difference in the defense method accuracy to the original accuracy.

Attack	w/o Defense	w/ Defense
BAE	33%	78.40%±3.14
DeepWordBug	34%	76.80%±2.64
FasterGeneticAlgo	44%	82.80%±2.48
Kuleshov*	0%	66.23%±4.65
PWWS	0%	79.20%±1.72
TextBugger	6%	77.00%±2.97
TextFooler	1%	80.20%±2.48

Table 4: Accuracy for increased randomness defense from Algorithm 2 against each attack method with $k = 41$. The accuracy prior to attack is 80%.

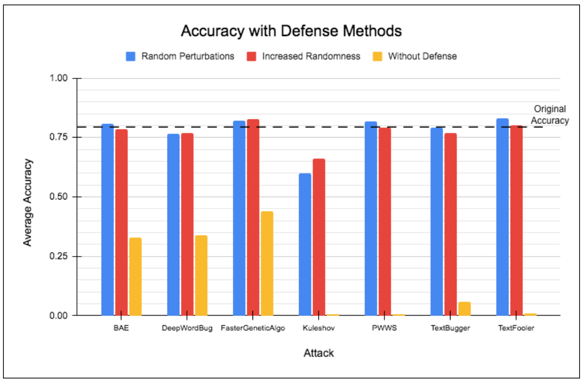


Figure 4: Comparing the average accuracy of the Random Perturbations Defense and the Increased Randomness Defense methods to the under attack accuracy without defense on the seven attacks.

4.4 Comparison to Recent Defense Methods

Our defense methods are comparable to some recent defense methods created for text classification. Our defense method returns the model to the original accuracy within statistical significance. This is comparable to the work done by Zhou et al. (2021) in their Dirichlet Neighborhood Ensemble (DNE) defense method. They were able to bring the model within 10% of the original accuracy for CNN, LSTM, and BOW models for the IMDB dataset. However, their work is only applicable to synonym-substitution based attacks. Since our defense methods apply equally well to seven attacks, it is general and can be applied without determining the exact type of attack (assuming it is one of the seven).

Another recent defense method, Synonym Encoding Method (SEM), was tested on synonym-substitution attacks on Word-CNN, LSTM, Bi-LSTM and BERT models (Wang et al., 2021b). This defense method was most successful on the BERT model and was able to return to the original accuracy within 3% for the IMDB dataset. Our work is comparable to both DNE and SEM which represent recent work in defending NLP models against adversarial attacks and more specifically synonym-substitution based attacks.

WordDP is another recent defense method for adversarial attacks against NLP models (Wang et al., 2021a). This defense method used Differential Privacy (DP) to create certified robust text classification models against word substitution adversarial attacks. They tested their defense on the IMDB and found that their WordDP method was successful at raising the accuracy within 3% of the original clean model. This method outperformed other defense methods including DNE. This is similar to our defense method, but they do not include whether these results are statistically significant.

We also compare our defense methods, RPD and IRD, against these recent defense methods on cost and efficiency. Our RPD and IRD methods have comparable time complexity of $O(cn)$, where c is the time it takes for classification and n is the number of reviews. Each method has a similar constant that represents the number of perturbations and replicates made. We cannot directly compare the time complexity of our defense methods with the SEM, DNE, and WordDP methods. These recent defense methods require specialized training and/or encodings. Our RPD and IRD methods do not require specialized training or encodings, so they cannot be directly compared on time complexity. This means that the comparison between our methods and recent defense methods comes in the form of specialized training vs. input preprocessing. Training and developing new encodings tends to be more time consuming and expensive than input preprocessing methods that can occur during the testing phases.

5 Conclusion

The work in this paper details a successful defense method against adversarial attacks generated from the TextAttack library. These attack methods use multiple different perturbation approaches to change the predictions made by NLP models. Our Random Perturbations Defense was successful in mitigating 6 different attack methods. This defense method returned the attacked models to their original accuracy within statistical significance. Our second method, Increased Randomness Defense, used more randomization to create an equally successful defense method that was 4 times more efficient than our Random Perturbations Defense. Overall, our defense methods are effective in mitigating a range of NLP adversarial attacks, presenting evidence for the effectiveness of randomness in NLP defense methods. The work done here opens up further study into the use of randomness in defense of adversarial attacks for NLP models including the use of these defense methods for multi-class classification. This work also encourages a further mathematical and theoretical explanation to the benefits of randomness in defense of NLP models.

Acknowledgement

The work reported in this paper is supported by the National Science Foundation under Grant No. 2050919. Any opinions, findings and conclusions or recommendations expressed in this work are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

2019. Predicting the direction of stock market prices using tree-based classifiers. *The North American Journal of Economics and Finance*, 47:552–567.

Basemah Alshemali and Jugal Kalita. 2020. Generalization to mitigate synonym substitution attacks. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 20–28.

Michel Ballings, Dirk Van den Poel, Nathalie Hespeels, and Ruben Gryp. 2015. Evaluating multiple classifiers for stock price direction prediction. *Expert systems with Applications*, 42(20):7046–7056.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. 2019. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4658–4664.

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*.

John P Corradi, Stephen Thompson, Jeffrey F Mather, Christine M Waszynski, and Robert S Dicks. 2018. Prediction of incident delirium using a random forest classifier. *Journal of Medical Systems*, 42(12):1–10.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *2019 Annual Conference of the National American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page 4171–4186.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA.

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

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181.

Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. 2020. Data security for machine learning: Data poisoning, backdoor attacks, and defenses. *arXiv preprint arXiv:2012.10544*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *stat*, 1050:20.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4129–4142.

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 *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Mohammed Khalilia, Sounak Chakraborty, and Mihail Popescu. 2011. Predicting disease risks from highly imbalanced data using random forest. *BMC Medical Informatics and Decision Making*, 11(1):1–13.

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

Volodymyr Kuleshov, Shantanu Thakoor, Tingfung Lau, and Stefano Ermon. 2018. Adversarial examples for natural language classification problems. In *Open Review Net*.

AV Lebedev, Eric Westman, GJP Van Westen, MG Kramberger, Arvid Lundervold, Dag Aarsland, H Soininen, I Kłoszewska, P Mecocci, M Tsolaki, et al. 2014. Random forest ensembles for detection and prediction of alzheimer's disease with a good between-cohort robustness. *NeuroImage: Clinical*, 6:115–125.

Alexander Levine and Soheil Feizi. 2020. Robustness certificates for sparse adversarial attacks by randomized ablation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4585–4593.

J Li, S Ji, T Du, B Li, and T Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia*, pages 69–72.

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*, Portland, Oregon, USA.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Kofi O Nti, Adebayo Adekoya, and Benjamin Weyori. 2019. Random forest based feature selection of macroeconomic variables for stock market prediction. *American Journal of Applied Sciences*, 16(7):200–212.

Desbordes Paul, Ruan Su, Modzelewski Romain, Vauclin Sébastien, Vera Pierre, and Gardin Isabelle. 2017. Feature selection for outcome prediction in oesophageal cancer using genetic algorithm and random forest classifier. *Computerized Medical Imaging and Graphics*, 60:42–49.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014*.

Eric Wallace, Tony Zhao, Shi Feng, and Sameer Singh. 2021. Concealed data poisoning attacks on NLP models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 139–150.

Wenjie Wang, Pengfei Tang, Jian Lou, and Li Xiong. 2021a. Certified robustness to word substitution attack with differential privacy. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xiaosen Wang, Hao Jin, Yichen Yang, and Kun He. 2021b. Natural language adversarial defense through synonym encoding. In *Conference on Uncertainty in Artificial Intelligence*.

Zhaoyang Wang and Hongtao Wang. 2020. Defense of word-level adversarial attacks via random substitution encoding. In *Knowledge Science, Engineering and Management*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, 28:649–657.

Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing Huang. 2021. Defense against synonym substitution-based adversarial attacks via Dirichlet neighborhood ensemble. 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)*.

Liuwan Zhu, Rui Ning, Cong Wang, Chunsheng Xin, and Hongyi Wu. 2020. Gangsweep: Sweep out neural backdoors by gan.