Text-Defend: Detecting Adversarial Examples using Local Outlier Factor

Marwan Omar  
Department of Computer Science  
University of Central Florida  
Orlando, USA  
marwan@knights.ucf.edu

Gita Sukthankar  
Department of Computer Science  
University of Central Florida  
Orlando, USA  
gitars@eecs.ucf.edu

Abstract—The problem of generating adversarial examples for text has been studied extensively; however, the detection of adversarial examples is largely underexplored. This paper studies the usage of Local Outlier Factors (LOF) to detect and filter adversarial examples from training data. Our experiments demonstrate that removing examples detected by LOF restores the performance of LSTM, CNN, and transformer-based classifiers on common sentence classification tasks. Our proposed technique outperforms DISP and FGWS, two state of the art detection techniques for identifying adversarial examples.

Index Terms—adversarial learning, anomaly detection, text classification

I. INTRODUCTION

Adversarial examples (also known as adversarial attacks) have proven successful in fooling natural language processing models, significantly dropping the performance of such models in a variety of linguistic tasks such as sentiment analysis and question answering systems [1]–[6]. They are particularly problematic for social media datasets, which are often aggregated from a variety of sources. Detection techniques work by separating a clean input from an adversarial input; on the other hand, defense techniques work by correctly classifying the input to its corresponding class or output, without filtering adversarial examples. Developing effective detection techniques can lead to a more robust defense strategy [7], [8] by enabling normal examples and adversarial examples to be handled by separate techniques [9], [10]. As language models are constantly being deployed in the real world to handle tasks such as sentiment reviews and news topic classification, it is critical to design detection methods that are capable of deterring adversarial attacks by proactively notifying system users of potential security breaches on NLP models.

This paper proposes the usage of Local Outlier Factor (LOF), a density-based method that utilizes the number of data points in neighboring regions of the space to identify outliers; it is used for anomaly detection in a variety of applications including fraud detection, medical diagnosis, and engine inspections [11]. Our LOF-based detection technique filters the training examples based on the degree of outlierness. The normal data points (non-adversarial examples) would then be processed by an NLP classification model (e.g. BERT) before including them in the dataset repository.

Even though LOF cannot completely eliminate adversarial examples, it filters enough of them to restore the performance of text classification algorithms. This paper compares the performance of our LOF-detection based detection scheme to two other defense techniques, Frequency Guided Word Substitution (FGWS) [3] and DISP (learning to DIScriminate Perturbations) [9]. We demonstrate that LOF outperforms these algorithms on restoring the performance of classifiers vs. adversarial examples generated with TextAttack [12]. This paper makes the following contributions:

1) We propose the usage of Local Outlier Factor to detect and eliminate adversarial examples targeting text classifiers.
2) We demonstrate that LOF can be used to counter three different adversarial attack recipes: TextBurger, Genetic, and Deepwordbug.
3) Our results show that preprocessing the dataset with LOF restores the performance of the most popular text classifier architectures (BERT, WordCNN, and LSTM) on three different social media datasets.

II. RELATED WORK

Inspired by research developments in the computer vision domain, the natural language processing community has endeavored to develop defense techniques for combating adversarial attacks on text classification models. In particular, the literature includes a plethora of defense techniques which leverage adversarially generated examples to robustify models against future adversarial attacks [9], [13], [14]. For instance, Wang et al. [15] proposed a fast text adversarial attack method, Fast Gradient Projection Method (FGPM) that can be implemented twenty times faster than existing attack methods. The authors then incorporate adversarial training with FGPM enhanced by Logit Pairing (ATFL) to defend against future adversarial attacks. However, there is less work on detecting and eliminating adversarial examples from the training data.

Several studies have sought to detect and counter specialized attack types. Pruthi et al. [16] conducted a study to detect adversarial misspelling attacks; however they did not incorporate semantics and grammaticality constraints. Li et al. [17] aim to detect a single type of attack based on a technique that preends an identical phrase to every training sample but
did not extend their work to other attack types. Sakaguchi et al. [18] conducted a study to identify and deter adversarial attacks based on the notion of exploiting the inconsistencies in grammar of certain linguistic tasks. TextFirewall [19] is a framework that can be used to identify new adversarial attacks in the text domain. By analyzing the inconsistency between the target model’s output and the impact value calculated by essential words within the text, TextFirewall can identify and detect adversarial examples. Although the authors presented detailed results on the sentiment analysis task using the IMDB and YELP datasets, it is not clear whether TextFirewall works well for other downstream tasks.

We benchmark our work against two top performing detection techniques, DISP and FGWS. DISP (learning to Discriminate Perturbations) [9] uses a binary logistic regression classifier to identify adversarially modified words and then corrects the perturbed word by substituting one of its nearest neighbors in the embedding space. FGWS [3] performs statistical tests on word frequency counts to identify adversarial word substitutions. We selected these techniques as benchmarks because they are applicable to a wide range of attacks.

III. METHOD

Local Outlier Factor (LOF) is a density-based method that uses the number of data points in neighboring regions of the space to identify outliers [11], [20]. The primary objective of the LOF algorithm is to assign to each data point (or object) a degree of being an outlier; this degree is referred to as the local outlier factor (LOF) of an object. The locality of a data point depends on the proximity of that point in relation to the surrounding local points (or neighbors).

The local outlier factor is a ratio that determines whether or not an object is an outlier with respect to its neighborhood. LOF \( A \) is the average of the ratios of the local reachability density of \( A \) and that of \( A \)'s \( k \)-nearest neighbors. We present the details of the LOF algorithm [20] below:

**Algorithm: LOF-based Outlier Detection**

1. For each data point \( X \), let \( D^k(X) \) represent the distance of point \( X \) to its \( k \)th neighbor, and \( L^k(X) \) represent the set of points within \( D^k(X) \)
2. Compute the reachability distance for each data point, \( X \), as \( R^k(X,Y) = \max(\text{dist}(X,Y), D^k(Y)) \)
3. Compute the average reachability distance \( AR^k(X) \) of data point \( X \) as \( AR^k(X) = \text{MEAN}_{Y \in L^k(X)} R^k(X,Y) \)
4. The LOF score for each point, \( X \), is calculated as: \( LOF^k(X) = \frac{\text{MEAN}_{Y \in L^k(X)} \frac{AR^k(X)}{AR^k(Y)}}{\frac{AR^k(Y)}{AR^k(X)}} \)
5. Return the data points with \( LOF \) values exceeding \( r \).

Our LOF implementation was done using scikit-learn. Although there are myriad other cluster-based anomaly detection techniques such as K-means and DBSCAN, our preliminary experiments revealed that LOF outperformed the most commonly used methods of detecting outliers. For our text classification scenarios, LOF improved when we preprocessed the data using Kernel PCA (KPCA), a nonlinear dimensionality reduction strategy. KPCA identifies an efficient representation that models the data using a lower-dimensional manifold. Using scikit-learn grid search, we identified the optimum LOF hyperparameters on a separate validation set (\( r = 0.5 \) and \( k = 20 \)). The next section describes our experimental setting.

IV. EXPERIMENTS

This paper evaluates the performance of our LOF defense technique vs. FGWS and DISP at combating adversarial attacks directed against classifiers performing a sentiment analysis task. Table I shows the details of the three popular benchmark datasets used in our study: YELP, Movie Review (MR), and AG NEWS. We examine two text classification tasks: sentiment analysis (conducted on YELP and MR) and topic classification (AG NEWS).

A. Text Classification Models

We employ three deep-learning algorithms that have been shown to provide state of the art performance for text classification: BERT [24], WordCNN [25], and LSTM [26]. Our implementation uses the models from the Hugging Face Transformer Library [27]. For our experiments, we used a pre-trained BERT model, consisting of 125 million parameters. The model was trained for 10 epochs with a batch size 16 and a learning rate of \( le^{-4} \). We chose maximum input sequence lengths of 512, 256 and 128 after byte-pair encoding for the YELP, MR, and AG NEWS datasets respectively [28]. The CNN architecture consists of three convolutional layers, each with a kernel size of 2, 3 or 4, and a total of 100 feature maps. The LSTM was initialized with pre-trained GloVe [29] word embeddings and use dropout during training at a rate of 0.5 before applying the output layer. Both CNNs and LSTMs were trained with batch size 12 and a learning rate of \( le^{-4} \) for eight epochs using the Adam optimizer [30]. We sample a subset of the training data (1,000 samples) for each experiment.

B. Adversarial Examples Generation

All adversarial attacks were provided by TextAttack [12]. For generating adversarial examples, we follow the model
TABLE I
DATASETS

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Train/Test Split</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>YELP [21]</td>
<td>Large Yelp Review Dataset</td>
<td>560,000 training and 38,000 testing</td>
<td>sentiment analysis</td>
</tr>
<tr>
<td>MR [22]</td>
<td>Movie Review Dataset</td>
<td>5,331 training and 5,331 testing</td>
<td>sentiment analysis</td>
</tr>
<tr>
<td>AG NEWS [23]</td>
<td>News Topics</td>
<td>12,000 training and 7,600 testing</td>
<td>topic classification</td>
</tr>
</tbody>
</table>

TABLE II
BASELINE PERFORMANCE OF CLASSIFIERS VS. THE DEEPWORDBUG ATTACK TECHNIQUE PRIOR TO LOF

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG NEWS</td>
<td>BERT</td>
<td>21.09</td>
</tr>
<tr>
<td></td>
<td>WordCNN</td>
<td>13.68</td>
</tr>
<tr>
<td></td>
<td>LSTM</td>
<td>11.56</td>
</tr>
<tr>
<td>MR</td>
<td>BERT</td>
<td>12.97</td>
</tr>
<tr>
<td></td>
<td>WordCNN</td>
<td>20.59</td>
</tr>
<tr>
<td></td>
<td>LSTM</td>
<td>19.29</td>
</tr>
<tr>
<td>Yelp</td>
<td>BERT</td>
<td>9.98</td>
</tr>
<tr>
<td></td>
<td>WordCNN</td>
<td>9.64</td>
</tr>
<tr>
<td></td>
<td>LSTM</td>
<td>7.88</td>
</tr>
</tbody>
</table>

proposed by [31] which consists of a decoder and an encoder. For semantic preservation, we follow [32] and tighten the thresholds on the cosine similarity between the embeddings of swapped words and the sentence encoding of original and perturbed sentences. We enforce the grammaticality of the adversarial examples by validating perturbations with a grammar checker. Moreover, we apply semantics as well as the grammatical constraints at each step of the search. This process consists of three steps described in more detail by [12]:

1) A search method to ensure that successful perturbations are discovered.
2) A transformation technique to modify a text input from $x$ to $x'$ (e.g. character substitutions).
3) Linguistic constraints to reject unacceptable (perturbations that do not meet the constraints) $x'$ such that the modified input $x'$ meets the semantics and fluency of the clean input $x$.

V. RESULTS

To this evaluate the performance of our adversarial detection technique, we utilized three attack recipes from the TextAttack framework [12]: Deepwordbug [33], Genetic Attack [34], and Textbugger [35].

A. Baseline Text Classification

To contextualize our work, we have decided to evaluate the performance of our three NLP classifiers prior to the implementation of our LOF-based technique. Table 2 shows that the models’ classification accuracy under adversarial examples is poor. The LSTM model fares the worst when exposed to adversarial examples with a classification accuracy of 7.88 when tested with the YELP dataset. The WordCNN model also performs poorly under attack and achieves an accuracy of only 9.64 when tested with the YELP dataset. On the other side of the spectrum, we observe that BERT fares much better than the previously mentioned models and achieves an accuracy of 21.97 when tested against the MR dataset.

B. Detection Performance

We test our three classifiers (BERT, WordCNN, and LSTM) on three datasets (YELP, MR, and AGNEWS) and measured the performance of our technique against two state-of-the-art techniques from the literature (DISP [9]) and FGWS [3]).

DISP. The intuition behind DISP [9] is that it can block adversarial attacks on text classification models by identifying and adjusting malicious perturbations. The perturbation discriminator identifies adversarial attacks by determining how likely it is for a token in the text to be perturbed and providing a list of potential perturbations. We evaluate the performance of DISP and LOF at protecting the performance of all three classifiers (Table III) and observe that LOF outperforms DISP in all cases. In particular, we notice that our technique achieves up to 92.4 $F_1$ score on the YELP dataset for the BERT model when attacked with the Textbugger recipe. This shows that our LOF solution is most effective in countering adversarial examples generated by the Textbugger technique. On the other side of the spectrum, we observe that our LOF technique fared the worst (although still outperforms DISP) when tested against the the Deepwordbug attack technique on the LSTM model. In particular, we show that our LOF technique countered only a portion of the adversarial attacks, with 49.8 $F_1$ score. We believe that our LSTM classifier may have suffered...
from overfitting on the filtered dataset, a drawback of LSTM models in-general.

FGWS. The intuition behind FGWS is that adversarial attacks can be detected by frequency differences between the replaced words and their corresponding substitutions. The original paper [3] empirically validated the effectiveness of their technique on the sentiment analysis task using various NLP models on the SST-2 and IMDB datasets. The results of FGWS and LOF on all three classifiers are shown in Table III. We observe that LOF outperforms FGWS across many metrics at countering adversarial examples. In particular, we notice that our technique achieves a 92.4 $F_1$ score, compared to the 89.1 $F_1$ score achieved by FGWS on the YELP dataset on the BERT model when attacked with the Textbugger recipe. Additionally, we observe that the LOF technique outperforms the FGWS on the MR dataset and BERT model when evaluated against the Deepwordbug attack with an $F_1$ score of 77.6 compared to 73.8 $F_1$ score for the FGWS technique. However, there were some instances where the FGWS outperforms the LOF technique; in particular, FGWS demonstrated a stronger performance at countering adversarial examples on the MR dataset and WordCNN model when evaluated against the Genetic attack framework. We observe that FGWS achieved an 84.9 $F_1$ score compared to 74.8 $F_1$ score for the LOF technique.

### VI. Conclusion

As the requirements of training data grow in scale, natural language processing practitioners often outsource or automate the tedious task of dataset creation and curation. This exposes NLP tools to potential security vulnerabilities from adversaries who manipulate the training data. This paper empirically validates the usage of the Local Outlier Factor algorithm to detect and counter adversarial examples. LOF is good at estimating local density based on the nearest neighbors of the data point and detecting if the training example is anomalously different from the rest of the data. To evaluate the performance of our technique, we trained several commonly used NLP architectures to perform sentiment analysis on three social media datasets that were contaminated with adversarial data generated using TextAttack [12]. Our results show that the LOF-based filter is conclusively better than DISP at countering adversarial examples. In general FGWS is better at countering Genetic attacks than LOF but less good on TextBurger and DeepWordBug.

While our results are robust against various types of adversarial attacks, one limitation of this study is that we haven’t evaluated a broad range of natural language processing tasks. Also adversaries who are aware of this technique may attempt to generate clusters of adversarial examples in order to foil this detection technique.