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 prepends an identical phrase to every training sample but

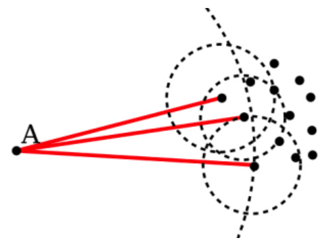

Fig. 1. When comparing the local density of a data point (A) relative to the local density of its K neighbors, we observe that A's neighbors have a much higher density. Thus A has a high LOF score and is more likely to be an outlier.

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

- Input: a set of data points $D = x_1, x_2, x_3, \dots x_n$, and threshold r;
- Output: outliers in D.

```
For $i = 1$ to length of $D$ do
    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 $R_k(X,Y) = \max(dist(X,Y), D^k(Y))$
    3) Compute the average reachability distance $AR_k(X)$ of data point $X$ as $AR_k(X) = MEAN_{Y \in L_k(X)} R_k(X,Y)$
    4) The LOF score for each point, $X$ is calculated as: $LOF_k(X) = MEAN_{Y \in L_k(X)} \frac{AR_k(X)}{AR_k(Y)}$
    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

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

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

Dataset	Model	Accuracy
AG NEWS	BERT	21.09
	WordCNN	13.68
	LSTM	11.56
MR	BERT	12.97
	WordCNN	20.59
	LSTM	19.29
Yelp	BERT	9.98
	WordCNN	9.64
	LSTM	7.88

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 $x\prime$ (e.g. character substitutions).
3) Linguistic constraints to reject unacceptable (perturbations that do not meet the constraints) $x\prime$ such that the modified input $x\prime$ 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 perofrms 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

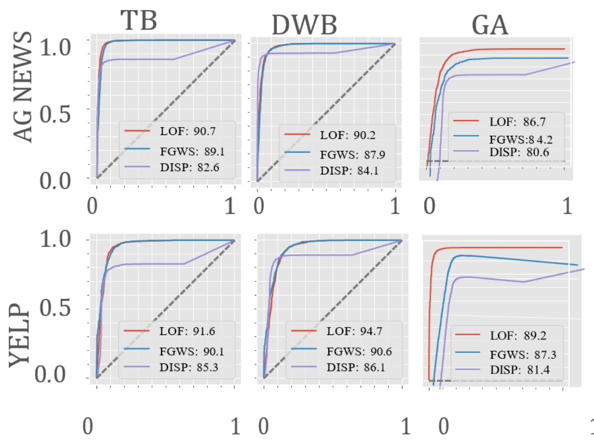

Fig. 2. ROC curves for three attacks on the BERT classifier: DWB (Deepwordbug), TB (Textbuger), and GA (Genetic Attack). For each plot, the x-axis and y-axis represent FPR and TPR, respectively. The legend refers to the AUC of each detection technique. We observe that the LOF technique outperforms both DISP and FGWS consistently across two tasks (YELP for sentiment analysis and AG NEWS for topic classification task).

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 $F1$ 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 $F1$ score. We believe that our LSTM classifier may have suffered

TABLE III
EVALUATION OF LOF VS. DISP AND FGWS

Classifier	Detection Technique	Attack Recipe								
		TextBurger			Genetic			Deepwordbug		
		TPR	F1	AUC	TPR	F1	AUC	TPR	F1	AUC
		YELP								
BERT	DISP	48.7	74.4	81.4	41.2	63.2	68.1	30.4	37.2	72.3
	FGWS	84.6	89.1	87.2	**89.3**	**82.0**	87.9	70.4	75.5	70.8
	LOF	**91.3**	**92.4**	**90.3**	86.4	80.1	**91.8**	**84.2**	**89.8**	**92.2**
WordCNN	DISP	75.3	81.5	81.5	78.2	77.2	88.3	33.2	55.2	70.4
	FGWS	77.4	85.5	82.2	84.6	83.4	85.9	63.2	68.1	68.5
	LOF	**87.6**	**89.9**	**94.1**	**88.6**	**86.5**	**91.6**	**67.8**	**79.3**	**92.5**
LSTM	DISP	33.7	47.3	69.2	32.3	43.6	68.3	22.3	41.6	61.2
	FGWS	**50.8**	**62.3**	71.6	**74.9**	56.9	**89.5**	42.4	43.2	66.3
	LOF	43.3	53.5	**82.8**	45.2	**66.8**	82.4	**63.6**	**49.8**	**72.8**
		MR								
BERT	DISP	72.3	82.6	81.0	72.8	82.3	87.4	39.2	44.3	73.0
	FGWS	84.4	83.7	83.2	**91.2**	**92.5**	91.8	66.3	73.8	77.3
	LOF	**89.8**	**86.9**	**91.5**	75.2	89.1	**92.0**	**73.7**	**77.6**	**94.8**
WordCNN	DISP	31.7	44.8	73.1	29.2	41.9	73.4	22.2	33.5	67.0
	FGWS	60.8	72.3	73.6	79.9	**84.9**	**86.7**	**34.7**	48.0	60.3
	LOF	**63.3**	**76.5**	**79.8**	**93.2**	74.8	78.4	32.6	**55.8**	**76.8**
LSTM	DISP	34.7	48.0	75.0	36.5	45.5	73.9	20.0	30.8	65.3
	FGWS	61.6	**72.0**	72.7	**80.8**	55.6	**86.4**	36.1	49.4	60.0
	LOF	**65.2**	69.3	**82.4**	69.1	**57.2**	81.9	**37.1**	**50.5**	**77.9**
		AG NEWS								
BERT	DISP	48.7	74.4	76.9	37.8	51.2	71.7	27.0	39.4	67.3
	FGWS	84.6	89.1	87.1	**88.2**	**89.0**	90.8	62.1	72.3	70.9
	LOF	**86.3**	**92.4**	**94.5**	75.7	81.5	**92.4**	**81.8**	**85.3**	**93.7**
WordCNN	DISP	69.1	84.5	91.8	72.2	79.3	89.6	37.1	50.4	74.7
	FGWS	78.8	87.2	82.2	**86.6**	**88.1**	87.9	53.3	65.1	63.5
	LOF	**88.2**	**89.1**	**94.1**	78.6	83.4	**92.9**	**68.1**	**76.3**	**91.5**
LSTM	DISP	35.6	44.8	72.3	29.6	42.7	73.7	24.6	32.28	64.2
	FGWS	60.8	**72.3**	73.6	**79.8**	**84.9**	**86.7**	34.7	**48.0**	60.3
	LOF	**65.2**	69.3	**82.4**	65.2	70.2	82.9	**77.9**	44.5	**76.7**

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 $F1$ score, compared to the 89.1 $F1$ 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 $F1$ score of 77.6 compared to 73.8 $F1$ 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 $F1$ score compared to 74.8 $F1$ 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.

References

[1] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is BERT really robust? a strong baseline for natural language attack on text classification and entailment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 8018–8025.

[2] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, 2012.

[3] M. Mozes, P. Stenetorp, B. Kleinberg, and L. D. Griffin, "Frequency-guided word substitutions for detecting textual adversarial examples," *arXiv preprint arXiv:2004.05887*, 2020.

[4] N. Mrkšić, D. O. Séaghdha, B. Thomson, M. Gašić, L. Rojas-Barahona, P.-H. Su, D. Vandyke, T.-H. Wen, and S. Young, "Counter-fitting word vectors to linguistic constraints," *arXiv preprint arXiv:1603.00892*, 2016.

[5] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[6] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7472–7482.

[7] G. Sun, Y. Su, C. Qin, W. Xu, X. Lu, and A. Ceglowski, "Complete defense framework to protect deep neural networks against adversarial examples," *Mathematical Problems in Engineering*, vol. 2020, 2020.

[8] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," *arXiv preprint arXiv:1805.12152*, 2018.

[9] Y. Zhou, J.-Y. Jiang, K.-W. Chang, and W. Wang, "Learning to discriminate perturbations for blocking adversarial attacks in text classification," *arXiv preprint arXiv:1909.03084*, 2019.

[10] R. Bao, J. Wang, and H. Zhao, "Defending pre-trained language models from adversarial word substitutions without performance sacrifice," *arXiv preprint arXiv:2105.14553*, 2021.

[11] Z. Cheng, C. Zou, and J. Dong, "Outlier detection using isolation forest and local outlier factor," in *Proceedings of the Conference on Research in Adaptive and Convergent Systems*, 2019, pp. 161–168.

[12] J. X. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, "Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP," *arXiv preprint arXiv:2005.05909*, 2020.

[13] E. Jones, R. Jia, A. Raghunathan, and P. Liang, "Robust encodings: A framework for combating adversarial typos," *arXiv preprint arXiv:2005.01229*, 2020.

[14] Y. Keller, J. Mackensen, and S. Eger, "BERT-defense: A probabilistic model based on BERT to combat cognitively inspired orthographic adversarial attacks," *arXiv preprint arXiv:2106.01452*, 2021.

[15] X. Wang, Y. Yang, Y. Deng, and K. He, "Adversarial training with fast gradient projection method against synonym substitution based text attacks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 16, 2021, pp. 13 997–14 005.

[16] D. Pruthi, B. Dhingra, and Z. C. Lipton, "Combating adversarial misspellings with robust word recognition," *arXiv preprint arXiv:1905.11268*, 2019.

[17] D. Li, Y. Zhang, H. Peng, L. Chen, C. Brockett, M.-T. Sun, and B. Dolan, "Contextualized perturbation for textual adversarial attack," *arXiv preprint arXiv:2009.07502*, 2020.

[18] K. Sakaguchi, M. Post, and B. Van Durme, "Grammatical error correction with neural reinforcement learning," *arXiv preprint arXiv:1707.00299*, 2017.

[19] W. Wang, R. Wang, J. Ke, and L. Wang, "Textfirewall: Omni-defending against adversarial texts in sentiment classification," *IEEE Access*, vol. 9, pp. 27 467–27 475, 2021.

[20] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," in *ACM SIGMOD International Conference on Management of Data*, 2000.

[21] N. Asghar, "Yelp dataset challenge: Review rating prediction," *arXiv preprint arXiv:1605.05362*, 2016.

[22] C. Poth, J. Pfeiffer, A. R"uckl'e, and I. Gurevych, "What to pre-train on? Efficient intermediate task selection," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 10 585–10 605. [Online]. Available: https://aclanthology.org/2021.emnlp-main.827

[23] X. Zhang, J. J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *NIPS*, 2015.

[24] M. O. Topal, A. Bas, and I. van Heerden, "Exploring transformers in natural language generation: GPT, BERT and XLNET," *arXiv preprint arXiv:2102.08036*, 2021.

[25] X. Ma, R. Jin, J.-Y. Paik, and T.-S. Chung, "Large scale text classification with efficient word embedding," in *International Conference on Mobile and Wireless Technology*. Springer, 2017, pp. 465–469.

[26] A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[27] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[28] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[29] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] T. Wang, X. Wang, Y. Qin, B. Packer, K. Li, J. Chen, A. Beutel, and E. Chi, "Cat-gen: Improving robustness in nlp models via controlled adversarial text generation," *arXiv preprint arXiv:2010.02338*, 2020.

[32] J. X. Morris, E. Lifland, J. Lanchantin, Y. Ji, and Y. Qi, "Reevaluating adversarial examples in natural language," *arXiv preprint arXiv:2004.14174*, 2020.

[33] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 50–56.

[34] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," *arXiv preprint arXiv:1804.07998*, 2018.

[35] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," *arXiv preprint arXiv:1812.05271*, 2018.