ADVERSARIAL RESISTANCE IN TOXIC COMMENTS DETECTION

Najwa Laabid
School of Computing
University of Eastern Finland
Joensuu, Finland
nlaabid@student.uef.fi

November 7, 2020

ABSTRACT

Toxic language threatens the continuity of online communities as a safe space for sharing information. To limit the effect of this threat, platforms use automatic language classification tools, often powered by deep learning classifiers. Like most deep learning models, these systems are vulnerable to adversarial attacks, defined as imperceptible modifications to input capable of changing drastically the classification output. This study explores said vulnerability experimentally, and shows that it can be reduced by adding adversarial examples to the training set. The techniques used in this work build on DeepWordBug, an open source project for the generation of black-box adversarial examples for text input. The results show that 3 types of deep neural networks (Bi-LSTM, RNN and CNN) can benefit from being trained on a set augmented by a relatively small number of adversarial examples. We made our code and experiments publicly available through a GitHub repository.

***K*eywords** toxicity online · text classification · deep learning classifiers · adversarial attacks · adversarial training

1 Introduction

1.1 Background

Cyber-bullying and online harassment threaten the prosperity and existence of online communities [1]. According to a 2014 study conducted by Pew Research, 40% of internet users have faced harassment and 73% of users have seen others get harassed [2]. Human content moderation is neither scalable nor effective, which increases the need for automatic detection of abusive language. Deep Learning methods are reported to have the best performance on this task ([3], [4], [5]). However, these methods can easily be fooled through systematically modified syntax, known as adversarial examples. A model that assigns a toxicity score of 90% to the sentence 'you're an idiot' may give a score as low as 20% to its adversarial counterpart 'you're an i d i o t' [6].

Adversarial sentences are define more generally as highly toxic sentences receiving a low score because of a non-semantic modification of their text [6]. Once aware of these limitations, users can develop methods to evade detection systems currently deployed in online platforms, by developing non-standard spellings of offensive words [7]. It is also worth noting that intentional misspellings are already common in online language, either through abbreviations or slang language. Literature also provides a formal definition of this type of input shown in equation 1. For a given classifier f and an input sentence x, Δx in the example is a perturbation vector through which the adversarial example is generated. Said vector can be limited in a specified p-norm by a parameter value of ϵ, which is used to limit the extent of modification caused by the perturbation. An important thing to note is the condition on the output of the classifier, which can be either different from the output given to the original input x, or equal to a predefined value t for targeted attacks.

$$\begin{gathered} x' = x + \Delta x,\, \left\|\Delta x_p\right\| < \epsilon,\, x' \in \mathbb{X} \\ F(x) \neq F(x')\, or\, F(x') = t \end{gathered} \tag{1}$$

Using equation 1, multiple algorithms were proposed to systematically generate adversarial examples from a given input for training deep learning models. A notable example is [8], where the authors used simple modification strategies (insertion, modification, and removal) as perturbations. [9] takes an extra step to define and implement 4 different transformer functions in their library DeepWordBug. Since DeepWordBug was our main source of inspiration for this project, we will go over the details of its transformers in the methodology section.

Once adversarial examples are generated, they should be incorporated to the training corpora of the model to prepare it to similar attacks. This is generally known in literature as adversarial training. The simplest form of such training is to augment the training set with as many adversarial examples as possible, at the expense of training time and possible over-fitting. More economical techniques have been explored by [10], notably virtual training which uses adversarial examples to turn a supervised learning problem to a semi-supervised one. Given the relatively small scope of this project and the subsequent limitations in time and resources, we only explore simple adversarial training in this study.

As will be presented later, this project confirmed literature findings regarding the vulnerability of deep learning models to adversarial attacks, and showed that including even a small subset of adversarial examples to the training set can improve the resistance of 3 different word-based models to adversarial attacks.

1.2 Aim and objectives

Our study aims to explore methods to increase text-classifiers' resistance to adversarial attacks. In particular, we aim to show that 3 models of the family of deep learners (a recurrent neural network (RNN), a Bi-directional Long-Short Term Memory (Bi-LSTM), and a convolutional neural network (CNN)) are indeed able to detect toxic language from the dataset, but are vulnerable to syntactic modifications of such input. We would like to show then that generating adversarial input for a small subset (less than 6.5%) of the training data is enough to increase the performance of our model in detecting adversarial examples, without compromising its performance on regular examples.

1.3 Research questions

We set forward the following questions to guide our work:

Main question How can deep learning models be made more resistant to adversarial attacks?

Sub-questions

- Which deep learning models detect toxic language the best?
- Which of these models is most vulnerable to adversarial attacks?
- Can simple adversarial training improve the resistance of such models to adversarial attacks?

1.4 Limitations of the study

Our main limitations were time, resources, and experience in the field. The scope of the course, both in terms of timeline and focus, forced us to limit our exploration and experiments to something manageable in 2 months that would leave room for studying research methodology as well. Limitations in resources were primarily the lack of access to necessary computing power to train and test the models. We relied on the free services of Google Colab, which gave us access to GPU use for a limited amount of time, preventing us from exploring character based models as we had planned. The time it takes to train and test models hindered the timely progress of our project, and forced us to minimize the adversarial attack methods explored. Finally, since this is the first research project in the field of text-classification for all members of the team, getting familiar with the statistical methods used in the field and the work done in previous literature created a substantial overhead.

2 Methodology

We plan to use a dataset of comments collected from Wikipedia's edit discussions. Since our study is quantitative in nature, and given that we did not do data collection ourselves, this section will be dedicated to presenting exploratory statistics of our dataset, providing theoretical background and implementation details of the models we used, and explaining adversarial attacks and training from literature. The section also concludes with a description of research ethics relevant to this project.

Table 1: Summary statistics of dataset

Set	Total	Non-toxic	Toxic	Non-toxic %	Toxic %
Training	154,571	139,780	14,791	90.5%	9.5%
Testing	5,000	4,497	503	89.9%	10.1%
Adversarial	10,000	9,064	936	90.6%	9.4 %

2.1 Dataset: Wikipedia's edit comments

This dataset was collected by Google's Perspective AI and used in a Kaggle's competition for toxic language detection in 2018. The training set contains 154,571 entries in total, classified to 5 levels of toxicity: toxic, severe toxic, obscene, threat, insult, and identity hate. Each input instance was given a binary label (1 if belongs to the class, 0 otherwise) in every one of these classes. We copied the first 10,000 values from the training set to create the adversarial set, which we used to generate adversarial examples later in the study. The test set has the same output classes and contains 5,000 instances. The labeling was obtained by human raters.

To simplify our classification task, we only considered the most generic class, toxic, turning the problem from multi-class classification to binary classification. In order to make the dataset compatible with the code pipeline of DeepWordBug, we changed the labeling from (0, 1) to (1, 2), 2 being the value given to toxic comments. Finally, we performed basic exploratory analysis o n the dataset to assess the balance of data distribution between the two labels. The results showed that the dataset is highly imbalanced in favor of the non-toxic class, as is expected in a classification task of this nature. The general summary numbers are shown in table 1.

We consider this dataset to be a representative sample of its space, since it contains a proportionally-adequate ratio of toxic comments [11]. This kind of data-sets is suitable for our study, given that our aim is to detect toxic language in general settings. Because we are using a limited number of instances from our sample space, we consider our sampling strategy to be non-probabilistic according to [12]. Biases of the dataset are likely to come from the topics of discussion mentioned in the Wikipedia environment, which may bias the language used in the comments and hinder future generalizations of our models.

2.2 Main reference: DeepWordBug

We chose the DeepWordBug algorithm proposed by [9] as a main reference for this work. The black-box nature of the algorithm was an important motivation for this choice. Unlike white-box frameworks, which rely on access to the gradient information of the model they are attacking, black-box models work with minimal assumptions on their environment, which makes them more akin to real-life attackers. Another motivation for this choice is the availability of the code base of DeepWordBug as an open-source research project. This helped us generate the results shown in this report within the time frame of the course. Furthermore, the structure and comprehensiveness of the code made it possible to explore the major stages of the text-classification pipeline in a single environment. Finally, the algorithm was never tested on the task of toxic comments detection to the best of our knowledge[1], which adds a novelty component to our work.

In its essence, the DeepWordBug algorithm aims to generate adversarial examples through a set of predefined text transformations applied to the keywords in a given input sequence. Keywords are tokens believed to have the highest influence on the output of classifier, such as curse words in sentences classified as toxic. These keywords are identified through influence scoring functions. An adversarial example is thus generated by applying a transformer function to the highest ranked tokens in a given sequence, as explained in figure 1.

The code base of DeepWordBug also included the implementations of 5 different classification models, 3 word-based and 2 character-based. The models were originally used to test the implementation of the adversarial attacks and show their effectiveness experimentally. The repository also offered interactive methods to test user provided data-sets and models. We have used the models provided in the original implementation, but had to add our own code to incorporate retraining and use different evaluation metrics than originally proposed, as explained in the following subsections.

2.3 Models: Word-based deep classifiers

Multiple deep learning models can be used as text-classifiers. In general, models can be divided into two categories: character-based models and word-based models. Models from the first category use characters as the smallest encoding

[1]The original paper tested the algorithm on spam detection and sentiment analysis.

Algorithm 1 DeepWordBug Algorithm

```
Input: Input sequence $\mathbf{x} = x_1 x_2 \ldots x_n$, RNN classifier $F(\cdot)$,
Scoring Function $S(\cdot)$, Transforming function $T(\cdot)$, maximum
allowed pertubation on edit distance $\epsilon$.
1:  for $i = 1..n$ do
2:      $scores[i] = S(x_i; \mathbf{x})$
3:  end for
4:  Sort $scores$ into an ordered index list: $L_1$ .. $L_n$ by descending
    score
5:  $\mathbf{x}' = \mathbf{x}$
6:  cost = 0, $j = 1$
7:  while cost $< \epsilon$ do
8:      cost = cost + Transform($x'_{L_j}$)
9:      $j++$
10: end while
11: Return $\mathbf{x}'$
```

Figure 1: DeepWordBug algorithm from [9]

unit, and are believed to perform better overall [13]. Word-based models rely on word-embeddings, which are mechanisms to transform a word into a vector of real numbers used as input to classification models [14].

Word models are generally faster in training, since they rely on larger input units. We focused on word-based models in this study to keep the project manageable. We used the three word-based models implemented by DeepWordBug: RNN, CNN, and Bi-LSTM. Details of the implementation of each model is shown below.

Bi-LSTM Bi-LSTM stands for Bidirectional Long-Short Term Memory, which is a variant of recurrent neural network (RNN) supplemented by a mechanism to address the vanishing gradient issue typical of RNNs [15]. The bi-directional nature of the network means that it has an RNN in each direction (from first to last words and vice versa).

Simple RNN Our second model was a simple (uni-directional) RNN with a 100 nodes in its inner layer and a final fully connected layer with the log soft max activation function.

Simple CNN The word CNN model we used is made of 10 layers in total, including convolutional layers with relu activation and maxpooling, linear layers, and a final activation layer using log soft max. The full implementation of the network can be found in the project's repository.

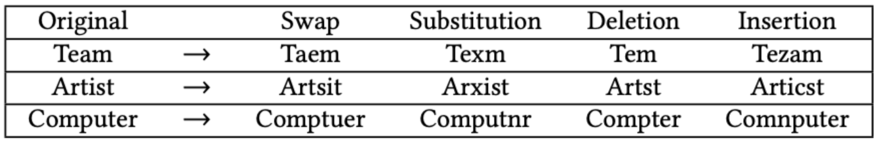
We trained these models for 10 epochs each, using the training set described in 2.1. The performance of each of the models is reported in the results section.

2.4 Adversarial training and attacks

DeepWordBug proposes 4 different transformer functions and 4 different scoring functions for generating adversarial examples.

Transformers Transformer functions introduce a small perturbation on the text input through a single syntactic modification. 2 shows the transformer functions proposed. [9] showed that there is little difference in the performance of these functions, since in effect, they all map to the *unknown* word in the dictionary used to create word-embeddings. Therefore, we only used substitution in this study. In particular, we substituted specific characters in influential words by their *homoglyphs* , which are characters with similar appearance to the human observer and different interpretation by the computer (e.g., e and e). This modification is particularly interesting because it seems to be a handy method to evade classification without changing the spelling of words.

Scoring Scoring functions allow the identification of key tokens, which, when modified, change the classification output drastically. DeepWordBug proposes 4 scoring functions:



Original		Swap	Substitution	Deletion	Insertion
Team	$\rightarrow$	Taem	Texm	Tem	Tezam
Artist	$\rightarrow$	Artsit	Arxist	Artst	Articst
Computer	$\rightarrow$	Comptuer	Computnr	Compter	Comnputer

Figure 2: Transformer functions proposed by [9]

- *Replace one (R1S)* : chooses a random word to modify in a sequence.
- *Temporal Head Score (THS)* : THS is the difference between a model's prediction as it reads to the ith token and as it reads to the i-1th token.
- *Temporal Tail Score (TTS)* : TTS is similar to THS but works on the tail of a sentence, i.e computes the difference between a model's prediction as it reads from the end of an input to token i and to token i+1.
- *Combined* : this is the sum of THS and TTS, and is believed to provide a more hollistic evaluation of the influence of a token.

We chose 'combined' as our scoring function because it provides the most information on the value of a token.

2.5 A word on evaluation metrics

Multiple evaluation metrics exist for classification tasks. We chose to use AUC-ROC metric and F1 score because our dataset is heavily imbalanced [16].

Receiver Operating Characteristic (ROC) belongs to the family of rank-metrics [17]. It is a plot of true positive rate (tpr) against false positive rate (fpr) at different discrimination thresholds, and evaluates the power of a classifier to distinguish between the two output classes. The best performing classifier will have a curve passing through the (0,1) point, corresponding to a 100% tpr and 0% fpr. Area Under the Curve (AUC) corresponds to the area under the ROC curve, and translates the information in the curve to a single value. The higher the AUC, the better the performance of a classifier. A perfect classifier has an area of 1 [17].

F1-score is a metric combining precision and recall in a single number, as shown in equation 2. Precision is the accuracy of the minor class prediction, calculated as the ratio of true positives over the sum of true positives and false positives[18]. Recall captures the ratio of the true positive classes we managed to capture (or recall) out of all the classes labeled as positive by the classifier. It is calculated as the ratio of true positives over the sum of true positive and false negative. Precision and recall seek to minimize false positive and false negatives in turn [19]. Using the F score, we can evaluate the performance of our model on both fronts.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2}$$

2.6 Research Ethics

Since we did not collect data ourselves, ethical considerations were not a concern for this study. However it is worth noting that data used in this kind of tasks should be heavily regulated to avoid any breaches on data privacy. The data used in this study is collected under Wikiepdia's CC-SA-3.0 license, allowing the collection and distribution of any user generated content on their platform. Wikipedia contributors accept this policy upon joining the community, and are thus aware of the possibility of using their content publicly.

Other research ethics relevant to this work include honesty and accuracy in reporting. We provide access to our work through a public GitHub repository (link in section 6)and guarantee reproducibility of our results following the methods explained in this report. The statistical methods we used in the study are correct to the best of our knowledge. No intentional manipulation of data and/or results was used in this work.

3 Results

This section presents our major findings with regards to the 3 main axes of this study: classifying adversarial examples, attacking adversarial models, and adversarial training of deep learning models.

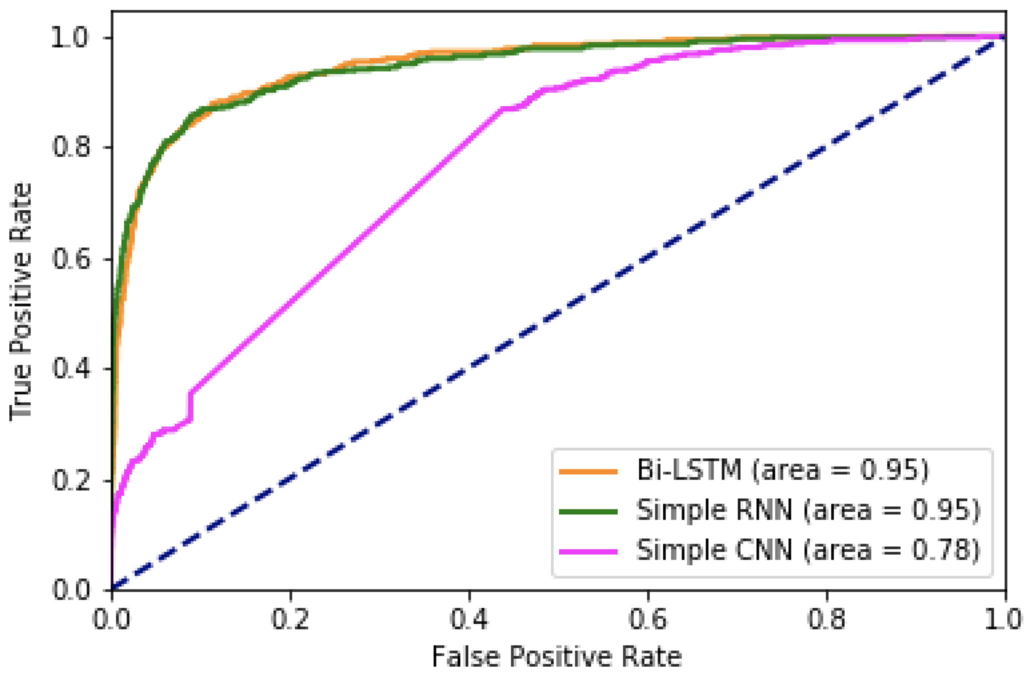


Figure 3: Performance of models after regular training

Table 2: F1-scores of models after regular training

Model	F1-score
Bi-LSTM	70.93 %
RNN	70.75 %
Word CNN	27.33 %

3.1 Regular Training

The first step of our experiments included training the three classifiers on non-adversarial data, and evaluating their classification ability on the test dataset. Figure 3 shows the ROC curves for the three classifiers, and table 2 shows their F1-scores. We notice that both RNN mdoels perform well on the test dataset compared to the CNN model, while the Bi-LSTM takes the lead on the F1 metric. It is worth noting that the performance of all models was adequate given the data size and class distribution.

3.2 Adversarial attacks

In the second stage, we attacked the regularly trained models with adversarial examples and evaluated their performance using the same metrics as before. Figure 4 and table 3 show the results of this experiment. We can see that all models performed as well as random guessing (the lowest standard of classification), which shows that these models are indeed unable to handle adversarial examples. Table 3 includes a measure of the percentage drop in F1 score for the performance of models compared to their initial evaluation of a non-adversarial test set. We also noticed that CNN was particularly vulnerable to the attack, since it had an accuracy score of almost 2% on the target class.

Table 3: F1-scores of models on adversarial data

Model	F1-score	Percentage drop
Bi-LSTM	9.28 %	61.65 %
RNN	6.25 %	64.5 %
Word CNN	2.65 %	24.68 %

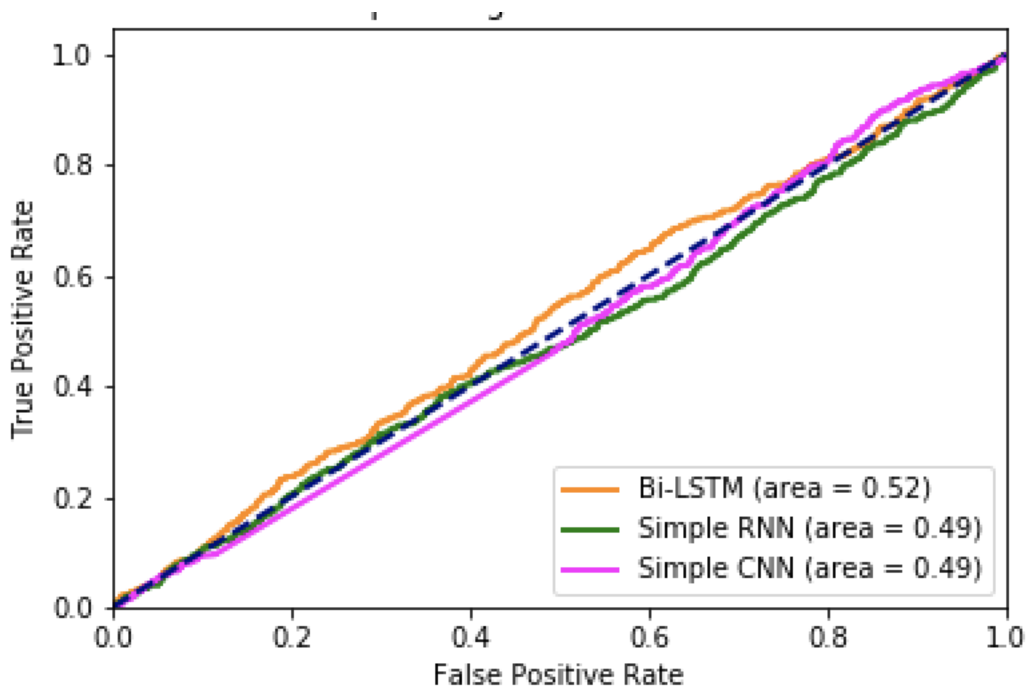


Figure 4: Performance of models on adversarial data

Table 4: F1-scores of models on adversarial data

Model	F1-score	Percentage drop	Percentage increase
Bi-LSTM	8.80 %	62.13 %	-0.48 %
RNN	6.53 %	64.22 %	0.28 %
Word CNN	4.31 %	23.02 %	1.66 %

3.3 Adversarial Training

In the last stage, we retrained the model on an adversarially augmented training set. We generated said set by producing adversarial examples for 20,000 input instances (the adversarial set mentioned in section 2.1) and adding the adversarial instances to the training set. We evaluated the re-trained models on the non-adversarial test set to see if their regular performance is affected by the re-training. The results showed a performance equal to that reported on initial training, in both AUC-ROC and F1-score (see figure 5 for AUC-ROC). Then we tested the new models on the adversarial set, and noticed a small increase in their performance. The results of this evaluation are shown in figure 6 and table 4.

4 Discussion

In this research, we sought to investigate the effect of adversarial attacks on toxic language detection in online content, and techniques to resist such attacks. We trained word-based models on a set of Wikipedia comments collected and published by Google's Perspective AI, showed that the models are indeed vulnerable to adversarial attacks, and successfully improved their resistance to such attacks through simple augmented adversarial training. Our findings are in-line with earlier studies in the topic. Adversarial attacks have long been shown to affect deep learning models in various fields, text classification included [20]. We also showed that a simple training technique can improve the resistance of such models, as was shown in earlier research on different classification tasks, including sentiment analysis [9]. These results also confirm theoretical work analyzing the effect of adversarial examples on classifiers, presented in [21]. Practical implications of these results mainly affect deployed toxicity detection systems, such as Google's Perspective API, which can be easily evaded by users through simple syntactic modification of offensive words.

Platforms employing such systems should thus be aware that they may still require human supervision/review of the ouput of their classifiers. From an ethical point of view, it is important to point out that similar studies are challenging due to issues in data collection and related privacy concerns. Data collected for the purpose of these studies is limited

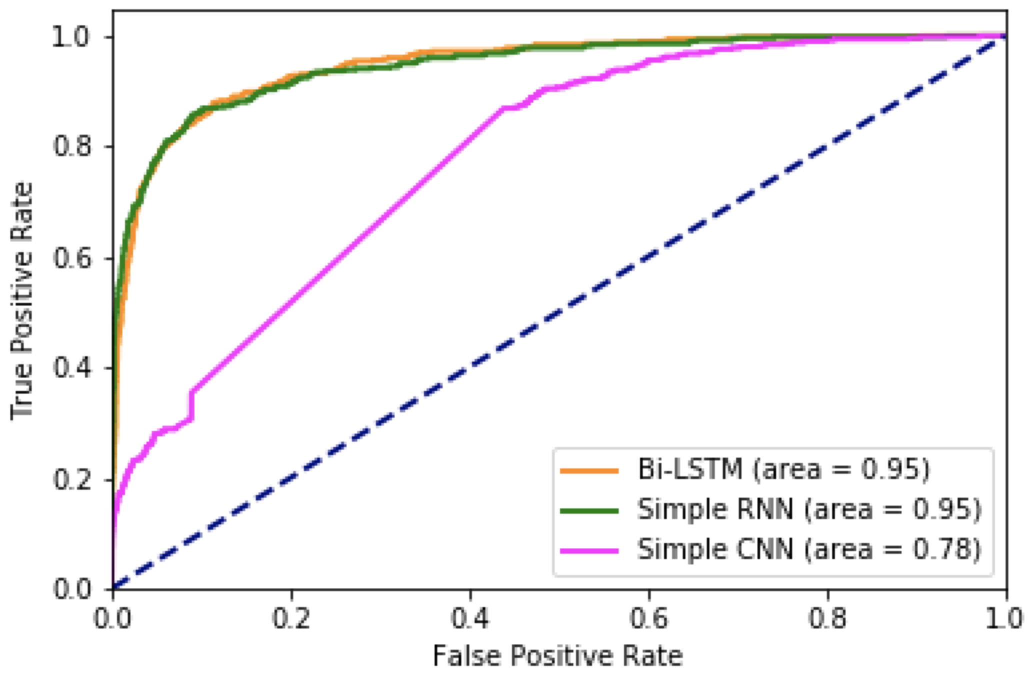


Figure 5: Performance of adversarial trained models on test data

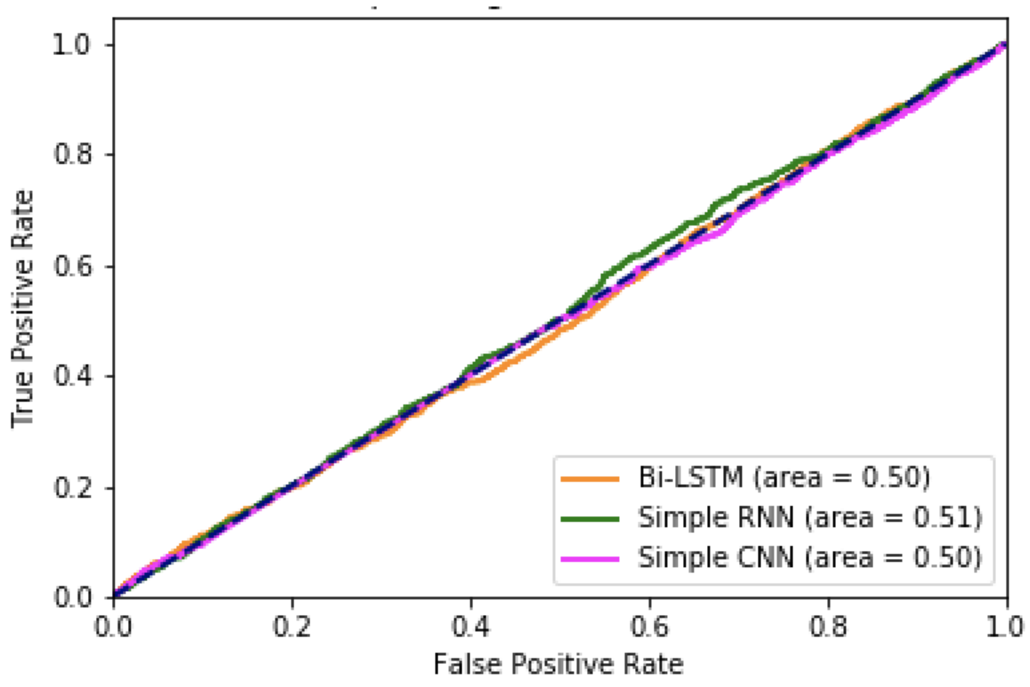


Figure 6: Performance of adversarial trained models on adversarial data

to publicly available online texts from platforms licensed to collect and use such data, such as Wikipedia. Regarding limitations, the most notable one was the use of a single dataset from one platform. The specific context within which the comments appeared may affect the generalization ability of our models, and hence hinder the reproducibility of our study on different online content. Another limitation is the use of a single class classification due to time limitations. The increase in classification ability was also not tested for statistical significance, which is an important step in confirming the results. A larger study could perhaps generalize our methods to the multi-label classification, using this data set to its full-extent, and prove (or disprove) the statisical significance of our results through hypothesis testing. Finally, future research on this topic can explore more advanced methods for generating adversarial examples, and perhaps look into more efficient adversarial training methods leveraging virtual training and semi-supervised learning.

5 Conclusion

In this study, we showed that the deep learning methods perform well on toxic language detection, despite the imbalance of data instances. We also showed that these models remain vulnerable to adversarial attacks, and that including even a small fraction of adversarial examples to the training set can improve their resistance. Our work built mostly on the DeepWordBug algorithm presented in [9], and we hope to continue investigating methods for generating adversarial examples and circumventing their effect through training to augment the solutions proposed in DeepWordBug. By making our code and methods available through this report and the corresponding GitHub repo, we hope to make a significant contribution to ongoing research to limit the effect of toxic language presence online.

6 Link to GitHub Repo

`https://github.com/NajwaLaabid/Adversarial-Toxic`

References

[1] S. M. B. Bottino, C. Bottino, C. G. Regina, A. V. L. Correia, and W. S. Ribeiro, “Cyberbullying and adolescent mental health: systematic review,” *Cadernos de saude publica*, vol. 31, pp. 463–475, 2015.

[2] N. Kayyali and D. O’Brien, “Facing the challenge of online harassment,” *Electronic Frontier Foundation*, vol. 8, 2015.

[3] S. V. Georgakopoulos, S. K. Tasoulis, A. G. Vrahatis, and V. P. Plagianakos, “Convolutional neural networks for toxic comment classification,” in *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, SETN ’18, (New York, NY, USA), Association for Computing Machinery, 2018.

[4] N. Chakrabarty, “A machine learning approach to comment toxicity classification,” in *Computational Intelligence in Pattern Recognition*, pp. 183–193, Springer, 2020.

[5] P. J. Kohli M, Kuehler E, “Paying attention to toxic comments online,” tech. rep., Stanford University, 2018.

[6] H. Hosseini, S. Kannan, B. Zhang, and R. Poovendran, “Deceiving google’s perspective api built for detecting toxic comments,” *arXiv preprint arXiv:1702.08138*, 2017.

[7] E. Dinan, S. Humeau, B. Chintagunta, and J. Weston, “Build it break it fix it for dialogue safety: Robustness from adversarial human attack,” *arXiv preprint arXiv:1908.06083*, 2019.

[8] B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi, “Deep text classification can be fooled,” in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 4208–4215, International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[9] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, “Black-box generation of adversarial text sequences to evade deep learning classifiers,” 2018.

[10] T. Miyato, A. M. Dai, and I. J. Goodfellow, “Adversarial training methods for semi-supervised text classification,” in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[11] L. K., “Sampling strategies.”

[12] M. Denscombe, *The good research guide: for small-scale social research projects*. McGraw-Hill Education (UK)., 2014.

[13] Y. Mehdad and J. Tetreault, “Do characters abuse more than words?,” in *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, (Los Angeles), pp. 299–303, Association for Computational Linguistics, Sept. 2016.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[16] Y. SUN, A. K. C. WONG, and M. S. KAMEL, "Classification of imbalanced data: a review," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 04, pp. 687–719, 2009.

[17] H. He and Y. Ma, *Imbalanced Learning: Foundations, Algorithms, and Applications*. Wiley-IEEE Press, 1st ed., 2013.

[18] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent Data Analysis*, pp. 429–449, 2002.

[19] H. Guo and H. L. Viktor, "Learning from imbalanced data sets with boosting and data generation: The databoost-im approach," *SIGKDD Explor. Newsl.*, vol. 6, p. 30–39, June 2004.

[20] N. Dalvi, P. Domingos, M. Sumit, and S. D. Verma, "Adversarial classification," in *In Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining*, pp. 99–108, ACM Press, 2004.

[21] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014.