Targeting the Most Important Words Across the Entire Corpus in NLP Adversarial Attacks

Reza Marzban(✉), Johnson Thomas, and Christopher Crick

Computer Science Department, Oklahoma State University, Stillwater, OK, USA
reza.marzban@okstate.edu

Abstract. In recent years, deep learning has revolutionized many tasks, from machine vision to natural language processing. Deep neural networks have reached extremely high accuracy levels in many fields. However, they still encounter many challenges. In particular, the models are not explainable or easy to trust, especially in life and death scenarios. They may reach correct predictions through inappropriate reasoning and have biases or other limitations. In addition, they are vulnerable to adversarial attacks. An attacker can subtly manipulate data and affect a model's prediction. In this paper, we demonstrate a brand new adversarial attack method in textual data. We use activation maximization to create an importance rating for each unique word in the corpus and attack the most important words in each sentence. The rating is global to the whole corpus and not to each specific data point. This method performs equal or better when compared to previous attack methods, and its running time is around 39 times faster than previous models.

Keywords: Adversarial attacks · Natural language processing · Deep learning · LSTM · Transformers · CNN

1 Introduction

Machine learning with deep learning models has achieved impressive advances, empowered by the era of big data and the proliferation of cheap computing power. Such models have reached or even surpassed human expert performance in a number of tasks (e.g. machine vision and natural language processing). However, despite their accuracy and performance, many other shortcomings and limitations must be improved. The most important limitation of deep learning models is that they are like black boxes and are not interpretable. As a result, we may not trust them in life and death situations. In addition, we have seen that they are extremely vulnerable to adversarial attacks. If an attacker creates adversarial examples by adding a little well-chosen noise to an input, although humans still classify them appropriately, the models can be deceived and classify them incorrectly, potentially leading to catastrophic results.

Adversarial attacks come in two flavors: white box and black box attacks. In a white box attack, the attacker has access to the information, details and weights of the attacked model. Conversely, in the black box context, the attacker does not have any internal model information. Obviously, in real life scenarios, black box attacks are more realistic. There have been many works on adversarial attacks and creating defense mechanisms against them. Most have concentrated on machine vision and image processing, as it is much easier to visualize and compare images in a 2-dimensional format, whereas in the Natural Language Processing (NLP) field, there has significantly less work, as it is much more challenging to visualize and compare textual data. NLP is the field of allowing machines to communicate in humans' languages, and many tasks, from sentiment analysis to natural language generation, fall under its umbrella.

Adversarial attacks in NLP consist of two phases: choosing the words to attack and choosing the technique of manipulation or perturbation. There are many different word-level perturbation techniques in NLP (e.g. replace, delete, add, swap); these may seem different to humans but all of them result in the same behavior in NLP models. They will create a noisy word that is not in the model dictionary, and the model will assign it an 'unknown' label. Most existing adversarial attacks in NLP choose their targeted words to attack one input at a time, where each attack is applied to the most important words for a particular input instance. While such a technique can be successful, it requires access to the texts to be attacked in advance, in order to train the attacker model. They cannot attack brand new texts even when they are from the same source and domain.

In this paper, we have created a brand new black box adversarial attack technique that uses and trains a Convolutional Neural Network (CNN) on a portion of a dataset. It then analyses the CNN filters to establish a global importance rate for all words in the corpus using activation maximization. These importance rates apply across the model and reflect the overall model logic, rather than being local to a single data point. These importance rates can be used to attack even examples that have never been seen before by targeting the most important words. Our technique has equal or better performance in comparison to previous techniques in various tasks, and it is much faster as it does not need to analyze each and every new data point to develop an optimal attack.

We tested our adversarial attack on two benchmarks: a sentiment analysis (binary classification) and a tagging dataset with 20 possible classes. The attacks on these datasets were evaluated on three different models to see if the attack technique is generalizable to multiple architectures. We used Long Short Term Memory (LSTM), Convolutional Neural Networks (CNN), and attention-based Transformers. Performance was equal to previous techniques in the sentiment analysis task and much better in the tagging task. When we are attacking 40 words per text, our model reduces the test accuracy 6% more than previous techniques on average. In addition, in both tasks, our technique is much faster than the previous versions. Our technique, applied to a forty-thousand-element dataset, takes 164 s, while the baseline comparison model takes 6,323 s, a 39-fold improvement.

2 Related Works

NLP is a subfield of Artificial Intelligence (AI) that enables us to interact with computers in human languages [4,7,8]. To apply standard deep learning architectures to NLP, textual data must be transformed into an amenable numerical format. There are multiple ways of doing this, such as a one-hot encoder or n-gram representation, but by far the most common approach is to use the Word2Vec [17] algorithm. This creates a custom-length vector that represents the semantic meaning of a particular word, and attempts to reflect the relationship of words with each other. Word2Vec mappings can be trained from scratch, but it is also common to use one of the available pre-trained word representations like Glove [18].

After preprocessing texts, and transforming them into numerical vectors, the data can be fed into a deep learning model. There are various famous model architectures that work well on NLP tasks. The first natural choice is LSTM [9] which is an advanced version of a Recurrent Neural Network (RNN). It is designed with time-series data in mind, and it has an internal memory. According to its gate weights, it will decide what to remember and what to forget. Another famous deep learning architecture is the CNN. We know that CNNs have revolutionized image processing tasks, but CNNs can also be applied to textual data [10–12,21]. Yin [22] compared RNNs and CNNs on various NLP tasks and studied the performance of each. Activation Maximization (AM) is a method that can be applied on CNN models; some research has focused on creating an importance rate for each unique word in a corpus using AM on CNNs by analyzing the convolution filter weights [15]. We utilize this technique to create an adversarial attack method.

In 2017, a new generation of NLP models appeared, starting with Vaswani's first Transformer attention-based architecture [19]. Instead of remembering an entire text, it assigns an attention weight to each token, which allows it to process much longer texts. The attention technique enabled the creation of much more advanced transformer-based models like BERT [5], RoBERTa [13], and GPT-3 [3].

The various deep learning models have contributed to rapid improvement in NLP task performance, but they also have created challenges. To start with, deep learning models are not intuitive or explainable. They also have some limitations; for instance, most NLP models can accept only limited sequence lengths. Some authors have tried to overcome this limitation [16]. In addition, they can easily be manipulated by adversarial examples. Adversarial attacks have recently been a major research focus in machine vision [1,2], as it is very easy for humans to recognize patterns in two dimensions. However, very few have worked on NLP adversarial attacks. The same adversarial attacks techniques can be applied on textual data as well [20,23]. Gao [6] created a technique called WordBug that analyses each and every input to find the most important words in that text, then targeting them in an attack. They have used a temporal score extracted from a bidirectional LSTM to detect important words in each text.

In this paper, our contribution offers a similar technique that is much faster, and its performance is either equal or better than WordBug. We evaluate our algorithm on two datasets with different levels of difficulty: an IMDb sentiment analysis dataset [14] and a Stack Overflow dataset with 20 possible tags or classes.

3 Technical Description

3.1 Benchmark Datasets

In order to test our approach, we chose and used two datasets with different tasks. The first one is the IMDb movie review dataset,[1] which is a binary classifcation task for sentiment analysis. The second one is the Stack Overflow dataset,[2] in which each question is tagged with one of 20 possible tags. Obviously the first task is easier for models as it contains only two classes.

Our preprocessing step was very straightforward. We converted everything to lower case and dropped all stop-words and hapax legomena (words that appear only once in an entire corpus). We also dropped all special characters and numerical values. After preprocessing, the Stack Overflow dataset contained 40,000 rows and around 28,000 unique words. The IMDb dataset contained 42,928 rows and around 23,000 unique words. We split each of them into training and test sets.

3.2 Choosing and Targeting Words to Attack

The attack sequence begins by targeting particular words, hopefully selected to have the greatest possible effect on the model's performance and accuracy. The mechanism for choosing words to attack is the core comparison between techniques.

WordBug. Gao's technique [6] created a bidirectional LSTM and trained the model on each sentence to be attacked, identifying the most important words in each sentence in turn. The approach used three scores: Temporal, Temporal Tail and combined (which is the mean of the two previous scores). In order to create the Temporal score and Temporal Tail score of the i^{th} word in a sentence, they used Eqs. 1 and 2, in which n is the number of tokens or words in each data point, $x\,[1:n]$ are the n words in each data points and $F()$ is a function that maps input words to the probability of belonging to the actual class using the bidirectional LSTM. They observed that their combined score outperformed other approaches.

$$TemporalScore(x_i) = F(x_1, x_2, ..., x_{i-1}, x_i) - F(x_1, x_2, ..., x_{i-1}) \tag{1}$$

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

[2] https://console.cloud.google.com/marketplace/product/stack-exchange/stack-overflow.

$$TemporalTailScore(x_i) = F(x_i, x_{i+1}, x_{i+2}, ..., x_n) - F(x_{i+1}, x_{i+2}, ..., x_n) \quad (2)$$

The advantage of this technique is that each text is studied and produces the most important words local to that specific text. However, it has two big disadvantages: it needs to be trained on each and every sentence to be attacked, and it cannot be applied to other sentences even from the same context. It is also very slow, as it needs to run the LSTM $m * n$ times in which m is the number of rows and n is the number of tokens in each one.

Activation Maximization. This method [15] uses a 1-dimensional CNN, trains on a subset of data, and uses the CNN layer filters to find the importance rate of all unique words in the corpus based on Eq. 3.

$$importance = \left\{ \sum_{f=1}^{F} \sum_{s=1}^{S} \sum_{i=1}^{I} |w_i * Filter_{f*s*i}| \, | w \in Corpus, Filter \right\} \quad (3)$$

In Eq. 3, F is the number of filters in the CNN layer, S is the size of the filters, and I is the embedding length. w is a word embedding vector with a length of I. Corpus is a matrix of our entire word embedding of size $m * I$, in which m is the count of unique words in our corpus dictionary. Filter is a 3-D tensor of size $F * S * I$. This equation calculates the sum of activations of all filters caused by a single word.

The advantage of this method is that it provides a general importance rate applicable to all sentences that come from the same source and distribution. As a result, we can use its insight on new never-before-seen sentences. The other advantage is its speed, as it needs only to train the CNN model once, for a couple of epochs, and then its filters can be used (via Eq. 3 to create the importance rate. The downside of this method is that the globally important words may or may not be the best option for each sentence.

Choosing Words in Each Sentence. After using one of the above models, we target the top t words in each sentence that have the highest importance rate and attack them. In our experiments we used different following values for t: 0, 10, 20, 40. When the $t = 0$, it means we are not attacking at all; this is a baseline performance for comparison with our attacks. We tested the attacks on three different models to determine whether these techniques are applicable on all models. We used an LSTM, a CNN and a Transformer model to see how the different attacks behave on each.

3.3 Word Perturbation

After selecting the words to attack, a word manipulation or perturbation method must also be chosen. There are four main word attack methods in NLP: replace, delete, add, swap. To human observers, these might have different effects, but all of these methods have the same result, to create a noisy word that is not in the NLP model dictionary. Such words end up with an ‘unknown’ label assigned by the model. We used swap in this paper; we randomly chose two adjacent middle character in each targeted word and swapped them. Examples are presented in Table 1. Note that, for purposes of simplifying the presentation, a very short subset of texts is presented in this table. In the original data, the texts are much longer, averaging around 200 words or tokens in each.

Table 1. Word manipulation and perturbation (with $t = 2$)

Text before the perturbation	Text after the perturbation (with $t = 2$)
The movie was extremely boring	The **moive** was extremely **broing**
How can i import csv file in python	How can i **improt** csv file in **pyhton**

3.4 Comparison of Methods and Evaluation

In order to compare WordBug vs our Activation Maximization method, we tested both on three brand new models created from scratch on our two benchmark datasets. We used a CNN, an LSTM and a Transformer model, all of which were built with basic, straightforward architecture. We trained all of these models on the training set. As our attack method is a black-box adversarial attack, we attacked only the test set (since the attack does not have access to the models’ information and details).

In the WordBug method, each text in the test set was analyzed and attacked in turn, but in our Activation Maximization method, we fitted our importance rate on a subset of texts and used it for attacking the test set as a whole. We measured the run-time of both techniques while they were learning which words to attack and compared them. In the next step we evaluated our attacks on the three models we had trained, to observe the attack performance. In both methods, we used swap as the word perturbation, and used 0, 10, 20, 40 for our t.

4 Experimental Results

In the experiments, t is set to four different values: $0, 10, 20, 40$. We also included $t = 0$ which shows the baseline model without any inserted attack, so the effectiveness of each adversarial attack method can be seen. We tested our three models on two datasets (IMDb and Stack Overflow) with 3 different techniques of attack: WordBug, Activation Maximization and random. In the random case, t words are chosen at random to attack. In interpreting the attack performance, a lower test accuracy means a more effective attack.

4.1 Evaluation on IMDb

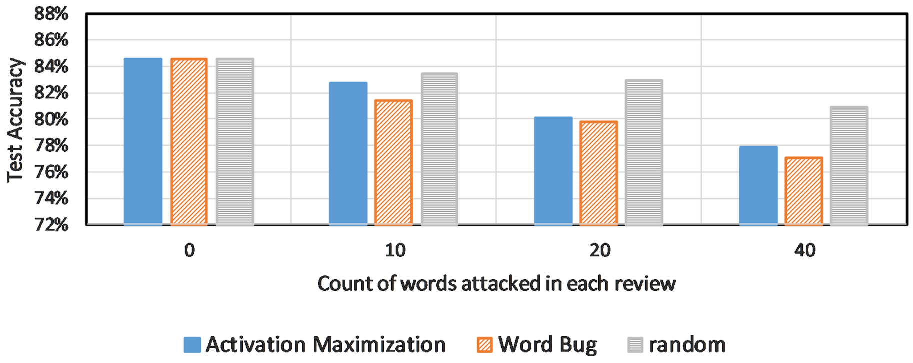

Fig. 1. Evaluation of attack methods in CNN on IMDb dataset

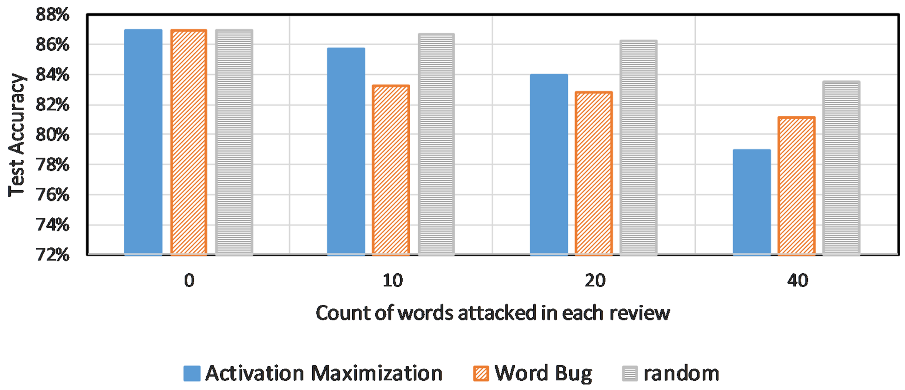

Fig. 2. Evaluation of attack methods in LSTM on IMDb dataset

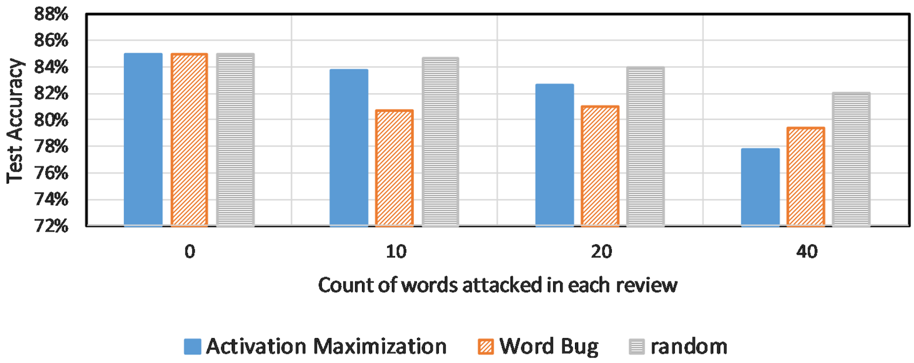

Fig. 3. Evaluation of attack methods in transformer on IMDb dataset

Figures 1, 2, and 3 show that, for the IMDb dataset in most cases, the WordBug technique works slightly better than Activation Maximization but not significantly. On average, the WordBug attacks reduce the models' accuracies less than 1% more than the AM attacks.

Our results when using WordBug to attack the IMDb dataset differs from Gao's original results [6]. Ours is a precise implementation of the WordBug attack, so the only potential reasons for this variation might be different preprocessing steps or different LSTM architectural hyperparameters. However, for the purposes of comparing WordBug and Activation Maximization, this difference is immaterial. The same preprocessing steps and model hyperparameters are used to compare the approaches fairly.

4.2 Evaluation on Stack Overflow

The Stack Overflow dataset task is much more complex than the naive Sentiment Analysis task in IMDb. The task requires classifying data into 20 possible classes or tags, and the texts are much longer. As a result, Figs. 4, 5, and 6 show that the Activation Maximization method is performing significantly better than WordBug. On average, the Activation Maximization attack leads to a 4.6% lower performance than WordBug's.

4.3 Transfer Learning from IMDb to Stack Overflow

After comparing the performance of WordBug and Activation Maximization, we show that in the AM method, we can train the attack on a subset of data and then use it to attack newer never-seen-before data from the same distribution and context. This performs equally well or better than WordBug, which has to study every single input to be able to attack it.

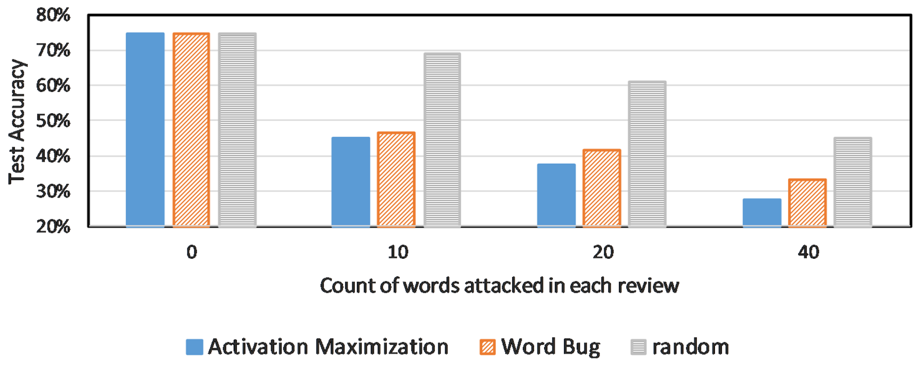

Fig. 4. Evaluation of attack methods in CNN on Stack Overflow dataset

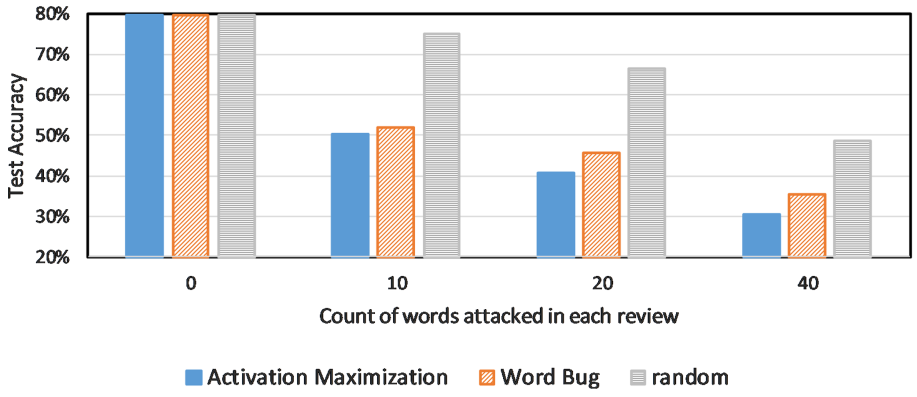

Fig. 5. Evaluation of attack methods in LSTM on Stack Overflow dataset

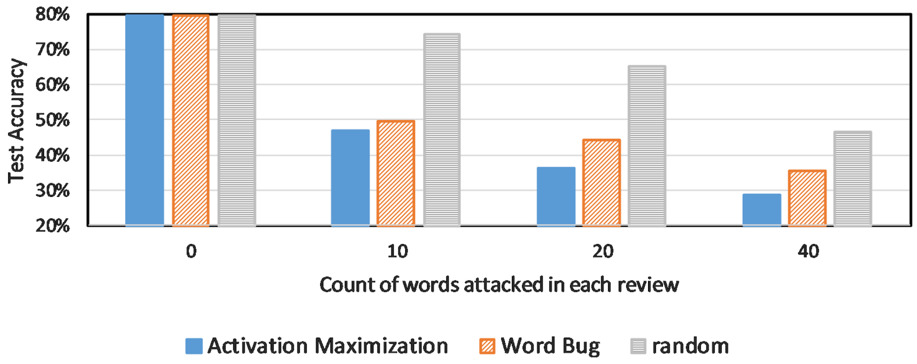

Fig. 6. Evaluation of attack methods in transformer on Stack Overflow dataset

We also wanted to show whether the same transfer learning logic can apply more generally, on new data but from a different context. In order to check this, we trained our Activation Maximization attacker on the IMDb dataset and used it on Stack Overflow. Figure 7 shows that the experiment does not support this, as the performance is even worse than a random attack.

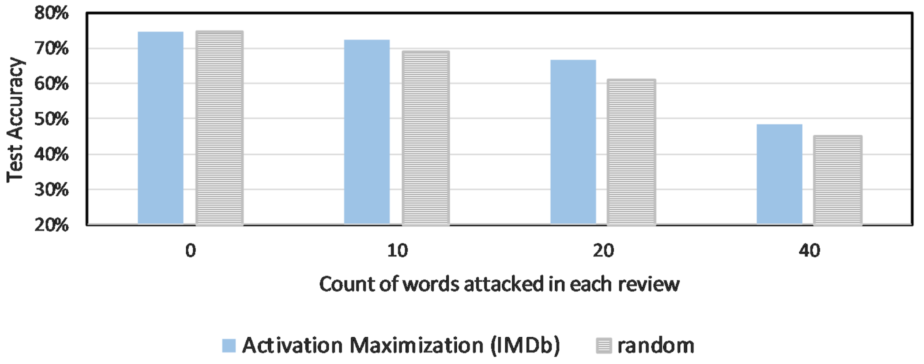

Fig. 7. Evaluation of using IMDb importance rate extracted from IMDb dataset and used to attack Stack Overflow dataset

4.4 Run Time Comparison

In addition to their effect on a model's accuracy, another important factor in adversarial attacks is the speed. We applied the attacks on corpora with different sizes varying from 20 to 40,000 data elements and measured the time needed for each method to train and choose the most important words to target.

Figure 8 shows that Activation Maximization is much faster than WordBug. This is to be expected, as WordBug must analyze each word in each data element one by one, while the Activation Maximization approach just trains a CNN model for a couple of epochs and uses the resultant filters to identify globally important words. With a corpus size of 40,000, Activation Maximization is 39 times faster than WordBug.

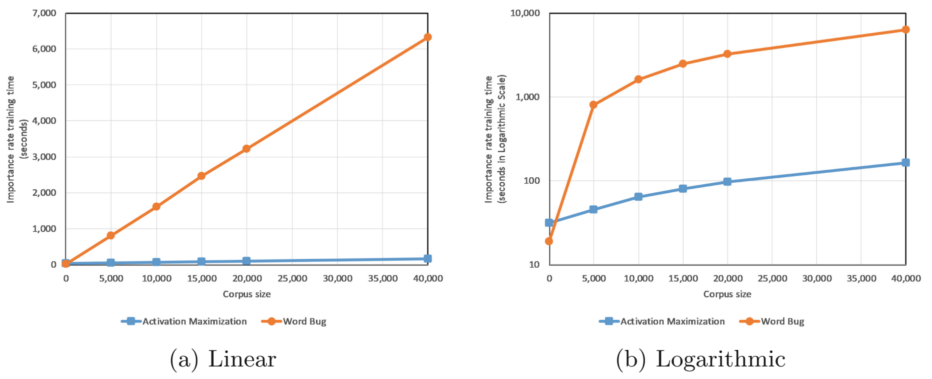

Fig. 8. Runtime comparison of WordBug and Activation Maximization

4.5 Results Analysis

We have demonstrated that the Activation Maximization approach is not only much faster than WordBug, but that it is equally good or better in its attack performance (depending on the task). In addition, Activation Maximization can be used to attack new input without any need for further training, whereas WordBug must be trained on each sentence prior to a successful attack. However, we observed that this ability of Activation Maximization only applies to data from the same context and distribution.

Both of these models are black-box attacks and do not have access to the details of the attacked model. In WordBug, the extracted information about most important words is local to each specific input, whereas in Activation Maximization, the attacker finds the most important words to target based on a global evaluation of a large corpus.

5 Conclusion and Future Work

Our new Activation Maximization adversarial attack has many benefits: it is significantly faster, its performance is equal to or better than WordBug, and it can be used on new inputs (from the same context) without any further training. Our attack method is a black box attack which means it does not need access to the structure or weights of an attacked model. Developing these kinds of effective and efficient attacks will enable us to evaluate new models to find their blind spots.

Our future work involves upgrading the attack so that it can be used on other contexts as well through transfer learning. As a result, we may be able to train an attacker without reference to a specific kind of data, and still effectively attack many different kinds of data targets.

References

1. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: a survey. IEEE Access **6**, 14410–14430 (2018)
2. Brendel, W., Rauber, J., Bethge, M.: Decision-based adversarial attacks: reliable attacks against black-box machine learning models. arXiv preprint arXiv:1712.04248 (2017)
3. Brown, T.B., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
4. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**(Aug), 2493–2537 (2011)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
6. Gao, J., Lanchantin, J., Soffa, M.L., Qi, Y.: Black-box generation of adversarial text sequences to evade deep learning classifiers. In: 2018 IEEE Security and Privacy Workshops (SPW), pp. 50–56. IEEE (2018)

7. Goldberg, Y.: A primer on neural network models for natural language processing. J. Artif. Intell. Res. **57**, 345–420 (2016)
8. Hirschberg, J., Manning, C.D.: Advances in natural language processing. Science **349**(6245), 261–266 (2015)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
10. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188 (2014)
11. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
12. Le, H.T., Cerisara, C., Denis, A.: Do convolutional networks need to be deep for text classification? In: Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence (2018)
13. Liu, Y., et al: RoBERTa: a robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
14. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 142–150. Association for Computational Linguistics (2011)
15. Marzban, R., Crick., C.: Interpreting convolutional networks trained on textual data. In: Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM, pp. 196–203. INSTICC, SciTePress (2021). https://doi.org/10.5220/0010205901960203
16. Marzban, R., Crick., C.: Lifting sequence length limitations of NLP models using autoencoders. In: Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM, pp. 228–235. INSTICC, SciTePress (2021). https://doi.org/10.5220/0010239502280235
17. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
18. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
19. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
20. Wallace, E., Feng, S., Kandpal, N., Gardner, M., Singh, S.: Universal adversarial triggers for attacking and analyzing NLP. arXiv preprint arXiv:1908.07125 (2019)
21. Wood-Doughty, Z., Andrews, N., Dredze, M.: Convolutions are all you need (for classifying character sequences). In: Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-Generated Text, pp. 208–213 (2018)
22. Yin, W., Kann, K., Yu, M., Schütze, H.: Comparative study of CNN and RNN for natural language processing. arXiv preprint arXiv:1702.01923 (2017)
23. Zhang, W.E., Sheng, Q.Z., Alhazmi, A., Li, C.: Adversarial attacks on deep-learning models in natural language processing: a survey. ACM Trans. Intell. Syst. Technol.gy (TIST) **11**(3), 1–41 (2020)