An Improved Generation Method of Adversarial Example to Deceive NLP Deep Learning Classifiers

Fangzhou Yuan[1,2], Tianyi Zhang[1], Xin Liang[1,2], Peihang Li[1,2], Hongzheng Wang[1,2], and Mingfeng Lu[1,2(✉)]

[1] School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China
yuanfz_shaddock@foxmail.com, lumingfeng@bit.edu.cn
[2] Beijing Key Laboratory of Fractional Signals and Systems, Beijing 100081, China

Abstract. Deep learning has been developed rapidly and widely used over the last decade. However, the concepts of adversarial example and adversarial attack are proposed, that is, adding some perturbations to the input of a deep learning model could easily change the prediction result. Deep learning-based NLP (natural language processing) classification algorithms also have this vulnerability. DeepWordBug algorithm is an advanced algorithm for generating adversarial examples, which can effectively deceive common NLP classification models. However, this algorithm needs to modify too many words to cheat NLP classification models, which limits its applications. In response to the shortcomings of DeepWordBug algorithm, this paper proposes an improving method to improve DeepWordBug. Drawing on the idea of Textfooler algorithm, the improved DeepWordBug adopts the method of dynamically adjusting the number of modified words, limits the maximum number of modified words. The new algorithm greatly reduces the number of words that need to be modified while ensuring the accuracy of NLP classification models as around 30%. It also ensures better practicality while maintaining transferability.

Keywords: Adversarial example · Deep learning · NLP · DeepWordBug algorithm

1 Introduction

Deep learning has been developed rapidly in the past ten years. It has promoted the breakthroughs in many fields, such as computer vision, natural language processing (NLP), and speech processing. Accordingly, more and more deep learning models have been put into practical applications. However, there are serious vulnerabilities in these models, which bring great risks to the practical application of various deep learning technologies.

In 2014, Szegedy et al. [1] found that after adding some disturbances to the input of the deep learning model, the prediction results of the model can be easily changed. Subsequent studies name this type of disturbance as adversarial perturbation, and the

S. Shi et al. (Eds.): AICON 2020, LNICST 356, pp. 51–62, 2021.
https://doi.org/10.1007/978-3-030-69066-3_5

input after the disturbance is called adversarial examples, the process of inputting the adversarial examples to mislead the model is called adversarial attack.

Goodfellow et al. [2] attributed this vulnerability to the linear characteristics of the deep learning model. Even if the deep network is not a linear network, some non-linear activation functions and network structures, such as ReLU activation functions, LSTM network structures, etc., are deliberately used in a nearly linear manner to make the model easy to optimize, otherwise a lot of time will be spent on model debugging when training the network. Equation (1) uses a dot product operation to simulate this situation: under linear condition, when a small disturbance η is added to the model input X, after the disturbance input $X^{'}$ is input to the model, the change of model output $W^{T}\eta$ will be produced, where W represents a weight vector. In the deep learning model, the dimensionality of W is very high, so its output change $W^{T}\eta$ will also be large. This leads the model to make a false judgment.

$$W^{T}X^{'} = W^{T}X + W^{T}\eta \tag{1}$$

Since the concept of adversarial examples was proposed, some white box attack methods [3–5] based on FGSM (Fast Gradient Sign Method) [2] were proposed. After that, researchers have shifted their interest to black box attack, some black box attack methods [6–10]were proposed.

NLP is an important research direction in the field of deep learning and artificial intelligence. The aim of the research is making use of computers to process and understand human languages to achieve effective communication between humans and computers. NLP is mainly used for question-answering systems, sentiment analysis, machine translation, etc. NLP classification algorithms based on deep learning are currently used in many fields such as spam classification, public opinion monitoring, product analysis, etc. Correspondingly, the generation method of adversarial example used to attack the NLP classifiers can also be used in many aspects. For example, the processing of advertising mail can deceive the spam classification system, and the processing of current affairs news can deceive the public opinion monitoring system.

DeepWordBug [6] is a black box adversarial attack algorithm, which is proposed by Gao et al. A scoring function is used for the first time to rank the importance of each word in the input, and then the most important words are selected to make letter-level modification. The extent of this change is controlled by the edit distance before and after the modification, so that the modification is not easily noticed by human observers. But when the number of modified words is too large, human observers will still perceive that this adversarial example is a modified text.

Then a new algorithm, Textfooler [7], appeared in 2019, which makes word-level changes. In this algorithm, a strategy of dynamically changing the number of words that needs to be modified is applied, and good results have been achieved.

This paper focuses on the weakness of the DeepWordBug algorithm that is not practical, the strategy of Textfooler is added in, and an improved method is proposed. Improved DeepWordBug could dynamically change the number of modified words, and limit the maximum number of modified words, and it has better practicality.

2 Performance and Problem Analysis of DeepWordBug Adversarial Example Generation Algorithm

2.1 Introduction of DeepWordBug

DeepWordBug is a black box attack algorithm using character-level changes, which can be carried out in a scene without model information. The following are the specific implementation steps of DeepWordBug algorithm.

Step 1: The first step of DeepWordBug algorithm is to calculate the importance score for each word x_i in the text example X, and rank the words in X in descending order according to the importance score. The scoring function in DeepWordBug uses Combined Score (CS), which is composed of the weighted sum of Temporal Head Score (THS) and Temporal Tail Score (TTS). The calculation formula is shown in formula (2) to formula (4).

$$THS(x_i) = F_y(x_1, x_2, \ldots, x_{i-1}, x_i) - F_y(x_1, x_2, \ldots, x_{i-1}) \tag{2}$$

$$TTS(x_i) = F_y(x_i, x_{i+1}, \ldots, x_n) - F_y(x_{i+1}, x_{i+2}, \ldots, x_n) \tag{3}$$

$$CS(x_i) = THS(x_i) + \lambda(TTS(x_i)) \tag{4}$$

THS is the difference between the output confidence after inputting from the first word to the i-th word, and the output confidence after inputting from the first word to the I − 1th word, shown in formula (2). The larger the result of this scoring function is, the more the output confidence reduced after x_i is removed. It means x_i is more important to the classification result. However, this scoring function does not consider the contribution of the words after the i-th word to the classification result, so TTS is added.

TTS is the difference between the output confidence after inputting from the i-th word to the last word, and the output confidence after inputting from the i + 1-th word to the last word, shown in formula (3).The larger the result of this scoring function is, the more the output confidence reduced after x_i is removed. It means x_i is more important to the classification result. Opposite to THS above, this scoring function does not consider the contribution of the words before the i-th word to the classification result.

THS and TTS only consider part of the input, while ignoring the impact of the other part on the classification results. CS adds the results of the two scoring functions, both including the influence of the i-th word and the words before the i-th word in THS, and the influence of the i-th word and the words after the i-th word in TTS, shown in formula (4), λ is a hyperparameter. Some experiment results have shown that using CS is better than using THS or TTS.

Step 2: Important words will be modified in this step. The first m words in the ranked word sequence are selected to make character-level changes, where m is a hyperparameter. The modification method is also a hyperparameter, and the available methods include insert, delete, swap, and substitute.

The insert method is to randomly select a position first, and then insert a random letter at that position. The delete method is to delete a letter in a random position. The swap method is to select two random letters in the word to swap. The substitute method is to select a random position in the word, delete the letter in this position and replace it with another different random letter. Table 1 is an example of these four modification methods.

Table 1. Modification methods of DeepWordBug

Original word	Modification method	Modified word
Sequence	Insert	Sequnence
	Delete	Sequnce
	Swap	Seqeunce
	Substitute	Sequerce

2.2 Performance of DeepWordBug

In this part, we choose two datasets to test DeepWordBug adversarial example generation algorithm, namely Yelp review dataset and Ag news dataset. We use two commonly used text classification algorithms, LSTM [11] and TextCNN [12], to classify these two datasets. After that, we use adversarial example generated by DeepWordBug to attack the classification model, and analyze its performance and problems.

Yelp review dataset is a two-category dataset of a text collection. The average length of each text example is 146 words. It divides the reviews of restaurants, venues, movies, etc. on Yelp website into two categories: positive reviews and negative reviews. There are 560,000 texts in the training set and 38,000 texts in the test set. Ag news dataset is a multi-category dataset. The average length of each text is only 42 words. It divides the sentences extracted from the news into four categories: global news, sports news, business news, and technology news. There are 120,000 texts in the training set and 7,600 texts in the test set.

The two datasets are very different in length and classification category. The purpose of this choice is to test whether good results can be obtained from DeepWordBug algorithm, in short text or long text, multi-classification or two-class classification.

Table 2 shows the classification accuracy of LSTM model and TextCNN model. It can be seen from the experimental results that the classification accuracy of LSTM is slightly higher than the TextCNN model on two datasets, and the classification accuracy of the two models on Yelp comment dataset is slightly higher than that on Ag news dataset. In general, the two models have achieved an accuracy of more than 90% on both datasets, and the effect is good.

Table 2. The classification accuracy of classification models on two datasets

	AG dataset	Yelp dataset
LSTM model	92.57%	96.52%
TextCNN model	91.76%	95.92%

First of all, we perform the DeepWordBug algorithm on the two datasets. Table 3 shows the experimental result of the accuracy experiment on the two models and two datasets. In comparison, the accuracy of Ag news dataset decreases more, and the accuracy of Yelp dataset decreases less. This is because the number of modified words m = 30, the average length of Ag news dataset is 42 words, for Ag dataset, the proportion of changes is relatively large. The average length of Yelp comment dataset is 146 words, so the proportion of changes is small. So the accuracy is different between the two datasets.

Table 3. The accuracy on two models and two datasets

DeepWordBug (m = 30, insert)	AG dataset	Yelp dataset
LSTM model	20.0%	48.2%
TextCNN model	32.7%	56.3%

Then we analyze the impact of number of modified words m on the accuracy of adversarial example classification. The experimental results using the TextCNN model and Yelp dataset as examples are shown in Fig. 1. From this experimental result, it can be found that as m increases, the classification accuracy of the adversarial examples decreases, but the decrease becomes more and more gentle. This is because in the process of increasing m, the modified words added later have lower importance rankings. So their changes will contribute less to the decline in the classification accuracy than the higher ones, so the curve becomes more and more flat.

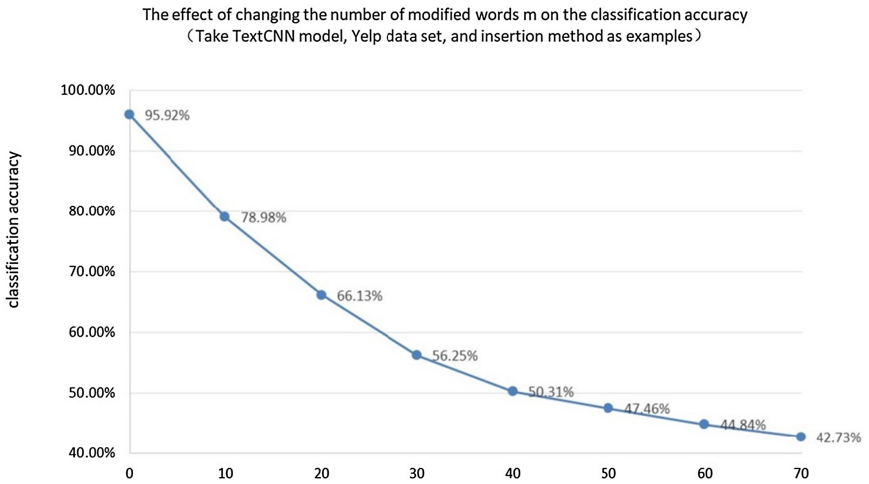

Fig. 1. The effect of changing the number of modified words m on the classification accuracy

Table 4 shows the adversarial example generated by DeepWordBug algorithm when m = 30. The classification model is TextCNN, the dataset is Yelp, and the modification method is swap. The red words are the modified words. In this example, the actual number of modified words m = 27, which is less than m. This is because in the swap operation, if two identical letters in a word are swapped, the original word will be obtained.

Table 4. An example of adversarial examples generated by DeepWordBug.

DeepWordBug (m=30, swap)	prediction	text
original text	positive review	when i went to fry's today it was a great experience. looked like they just went under a re-model and i think it makes the place look much more clean and welcoming. also, the staff there was very nice! since it was under remodel i had no clue where everything was and the front end manager helped me out! very sweet nice people working there!
adversarial example	negative review	when i wnet to fry's otday ti was a graet experience. looked ilke they just went under a er-model adn i think ti makes the place look much more clena nad wlecoming. alos, the staff tehre was veyr ncie! sicne it was undre remodel i had no clue wheer everytihng was and hte front edn manager hleped em out! evry sewet ince people wokring there!

Table 5 shows the transferability experiment results on the two datasets. According to the difference in the average length of the two datasets, the number of modified words in Ag dataset is changed to m = 30, in Yelp dataset m = 50, and the modification method in the experiment are all swap. In the table, LSTM- > TextCNN means inputting the adversarial examples generated on LSTM model into the TextCNN model, and TextCNN- > LSTM means inputting the adversarial examples generated on TextCNN model into the LSTM model. It can be seen from the transferability experimental results that when the model used to generate adversarial examples is different from the model using adversarial examples, the classification accuracy does not change much. It means the transferability of DeepWordBug algorithm is good.

Table 5. Transferability experiment results on two datasets

DeepWordBug	Ag dataset		Yelp dataset	
	Original accuracy	Accuracy after being attacked	Original accuracy	Accuracy after being attacked
LSTM	92.6%	20.0%	96.5%	36.1%
LSTM -> TextCNN	91.8%	25.7%	95.9%	48.3%
TextCNN	91.8%	32.7%	95.9%	47.5%
TextCNN -> LSTM	92.6%	36.6%	96.5%	50.5%

2.3 Problem Analysis

From the above experimental results, it can be seen that the accuracy of the adversarial example generated by DeepWordBug algorithm is still not very satisfactory. Especially on Yelp dataset, the accuracy is more than 35%. We found that DeepWordBug algorithm has the following problems:

First, in DeepWordBug algorithm, the number of modified words m is a fixed hyperparameter. In the experiment, if we fix m = 30, all adversarial examples will change 30 words on the basis of the original example. As we can see in Table 5, too many changed words will make the observer easily perceive the change of the text content. At this time, even if the adversarial example can deceive the classifier, it is not practical.

Second, in DeepWordBug algorithm, the modified m words are the top m words in the importance score ranking. However, modifying the top m words in the importance score ranking does not ensure that the effect of modifying these words is better than modifying any other m words. Because when calculating the importance score of each word, we only modified this word in the text, that is to say, we only measured the contribution of this word to the classification of the entire text, but when modifying m words, we cannot assume that the contribution of the m words to the entire text classification is equal to the simple addition of each words contribution to the text classification. Therefore, it is likely that there are other combinations of m words that can play a greater role in changing the classification result than the previous m words.

3 Improvement Strategy

3.1 Improvement of DeepWordBug

During the research, we found that for some sentences, only changing one or two words can cause changes in the classification results when using TextFooler algorithm. Based on the above analysis, we decide to add TextFooler algorithm to DeepWordBug to increase the effectiveness. This strategy is used to dynamically change the number of modified words, and only select the words with better effects to modify after the changes, and stop editing when the classification result can already be changed. In

other words, even for the top-ranked words, we have to check whether the confidence of the original prediction category of the sentence decreased after the modification. If the confidence does not decrease but rises, we can also give up changing the top-ranked words, instead, we can choose to change the lower-ranked words which can reduce the confidence of the original prediction category. Its purpose is to reduce the number of modified words while reducing the classification accuracy of adversarial examples, and hope to maintain the transferability of adversarial examples generated by DeepWordBug algorithm. The specific process of the improvement plan is as follows (Fig. 2):

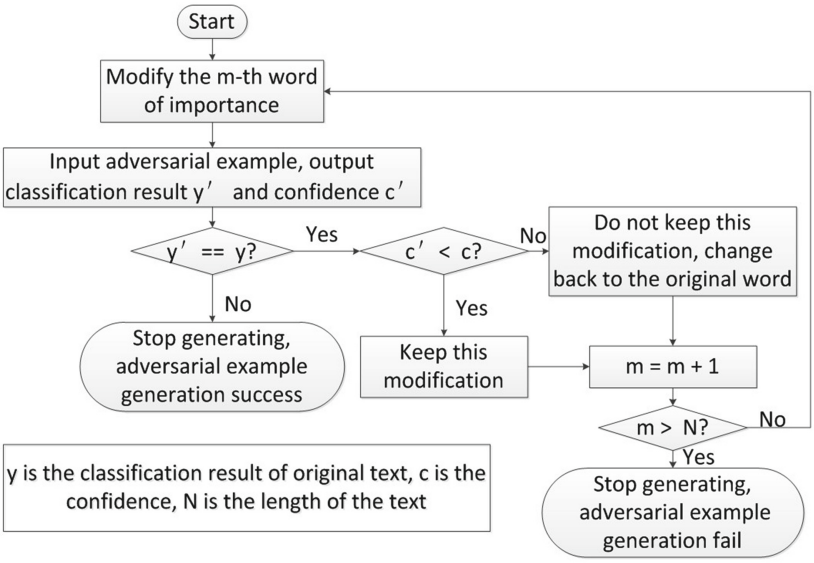

Fig. 2. The specific process of the improvement plan.

After adding the effectiveness check to DeepWordBug algorithm, the experimental results on Ag news dataset and Yelp comment dataset are shown in Table 6. The word modification methods used in the experiment before and after the improvement are both insert. Before the improvement, set m = 30 on Ag dataset and m = 50 on Yelp dataset.

Table 6. Experimental results and comparison of the improved algorithm

Dataset	Ag dataset		Yelp dataset	
Parameters	Insert, m = 30 before the improvement, dynamic m after the improvement		Insert, m = 50 before the improvement, dynamic m after the improvement	
Classification model	LSTM model	TextCNN model	LSTM model	TextCNN model
Original accuracy	92.6%	91.8%	96.5%	95.9%

(*continued*)

Table 6. (*continued*)

Dataset	Ag dataset		Yelp dataset	
Accuracy of adversarial examples generated by DeepWordBug	20.0%	32.7%	36.1%	47.5%
Accuracy of adversarial examples generated by improved DeepWordBug	6.5%	11.3%	5.4%	15.4%
Average number of modified words	11.6	15.3	30.9	35.1
Average number of Total words	42	42	146	146
Average change rate	27.6%	36.4%	21.2%	24.0%
Minimum number of changed words	1	1	1	1
Maximum number of changed words	116	107	210	447

According to the experimental results, it can be seen that from the comparison of the effectiveness of the adversarial examples, the classification accuracy of adversarial examples generated by the improved DeepWordBug algorithm has dropped significantly, from 20–50% to about 10%. From the comparison of the practicality of adversarial examples, the average number of changed words is less than the fixed number of modified words m on both datasets, and the accuracy of adversarial examples is lower.

However, from the "average number of modified words", "minimum number of changed words" and "maximum number of changed words" in Table 6, it can be seen that after adding the improvement strategy, although the average number of modified words is reduced, modifying a small number of words can change the classification result in some sentences, but the number of changed words is still very large in some sentences. The maximum number of modified words on Ag dataset reached 116 and 107, and the maximum number of modified words on Yelp dataset reached 210 and 447. This is because there is no restriction on the maximum number of modified words. For any sentence, if the classification result is always not changed, you can continue to change the next word in importance ranking until all words are modified.

3.2 Further Improvement

So we improved this method again and added the limit of the maximum number of modified words. For Ag dataset, we limited the maximum number of modified words to 30, and for Yelp dataset, we limited the maximum number of modified words to 50. As before, the difference between the experimental settings on the two datasets is still based on the difference in the average number of words in the two datasets. After adding the limit, the experimental results are shown in Table 7. For the convenience of comparison, the table also shows the experimental results of the algorithm before the improvement and the algorithm after the improvement without adding the limit of the maximum number of modified words.

Table 8 shows the experimental results when the maximum number of modified words is 30, the model is TextCNN, and the modification method is swap. The red words are the modified words. In addition, we also did a transferability test on the improved algorithm again. The experimental results are shown in Table 9.

Table 7. Experimental results and comparison of the improved algorithm

Dataset	Ag dataset		Yelp dataset	
Parameters	Insert, m = 30 before the improvement, dynamic m after the first improvement, limit m not more than 30 after the second improvement		Insert, m = 50 before the improvement, dynamic m after the first improvement, limit m not more than 50 after the second improvement	
Classification model	LSTM model	TextCNN model	LSTM model	TextCNN model
Original accuracy	92.6%	91.8%	96.5%	95.9%
Accuracy of adversarial examples generated by DeepWordBug	20.0%	32.7%	36.1%	47.5%
Accuracy of adversarial examples generated by improved DeepWordBug	6.5%	11.3%	5.4%	15.4%
Accuracy of adversarial examples generated by further improved DeepWordBug	11.3%	20.0%	24.5%	36.3%
Average number of modified words of improved algorithm	11.6	15.3	30.9	35.1
Average number of modified words of further improved algorithm	10.1	12.9	18.0	18.3
Average number of total words	42	42	146	146

Table 8. An example of adversarial examples generated by improved DeepWordBug

DeepWordBug	prediction	text
original text	positive review	when i went to fry's today it was a great experience. looked like they just went under a re-model and i think it makes the place look much more clean and welcoming. also, the staff there was very nice! since it was under remodel i had no clue where everything was and the front end manager helped me out! very sweet nice people working there!
adversarial example (improved DeepWordBug)	negative review	when i went to fry's today it was a greta experience. looked like they just went under a re-model and i think it makes the place look much more clean and welcoming. also, the staff there was very nice! since it was under remodel i had no clue where everything was and the front end manager helepd me out! very sweet ncie people working there!
adversarial example (original DeepWordBug)	negative review	when i wnet to fry's otday ti was a graet experience. looked ilke they just went under a er-model adn i think ti makes the place look much more clena nad wlecoming. alos, the staff tehre was veyr ncie! sicne it was undre remodel i had no clue wheer everytihng was and hte front edn manager hleped em out! evry sewet ince people wokring there!

Table 9. Transferability experiment results of improved DeepWordBug on two datasets

DeepWordBug (improved twice)	Ag dataset		Yelp dataset	
	Original accuracy	Accuracy after being attacked	Original accuracy	Accuracy after being attacked
LSTM	92.6%	11.3%	96.5%	24.5%
LSTM -> TextCNN	91.8%	58.0%	95.9%	63.9%
TextCNN	91.8%	20.0%	95.9%	36.3%
TextCNN -> LSTM	92.6%	54.8%	96.5%	68.9%

From the experimental results in Table 7, it can be seen that even when the limited maximum number of modified words is equal to the fixed number of modified words of the original algorithm, the two models of the two datasets can achieve lower accuracy than the original algorithm, and the average number of changed words is obviously less than the original algorithm.

From the adversarial examples in Table 8, it can also be seen that the improved algorithm can change the important words more accurately, and stop the changes immediately when the classification results change. So the number of changed words in the adversarial examples is small, and the effect is remarkable.

For example, the changed words in this example are “great”, “help” and “nice”. These three words are the words that could clearly reflect the emotional tendency of the reviewer. After the modification, they become an unknown word vector and can no longer be recognized by the model, and they cannot make great contribution to the classification result, so the classification result can be changed while maintaining high practicability.

In addition, from the transferability results in Table 9, it can be seen that the transferability of improved DeepWordBug is worse than that of the original algorithm. This is because for different classification models, the same word has different influence on classification results. When the number of modified words is large, there must be more words that have an impact on the prediction result in another classification model, so the transferability of the algorithm will be good, otherwise, less modified words will greatly affect the transferability of the algorithm. However, in practical applications, we can use this algorithm only for white box attack, and the weakness in transferability can be avoided.

4 Conclusion

In this paper, we first introduced an algorithm to generate adversarial example to deceive NLP deep learning classifiers: DeepWordBug, and illustrated some shortcomings in the practicality of the algorithm through experiments. Then, we used the idea of TextFooler algorithm for reference and improved DeepWordBug algorithm. Using the improved DeepWordBug to generate adversarial examples to attack the NLP classification models can maintain the classification accuracy at about 30%, and the number of modified words is greatly reduced. The experimental results show that on

the two datasets and two classification models, our improved DeepWordBug algorithm has a significant improvement in practicability while ensuring the success rate of the attack.

Acknowledgements. This work is supported by Beijing Natural Science Foundation (L191004).

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
2. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
3. Papernot, N., McDaniel, P., Swami, A., et al.: Crafting adversarial input sequences for recurrent neural networks. In: 2016 IEEE Military Communications Conference, MILCOM 2016, pp. 49–54. IEEE (2016)
4. Samanta, S., Mehta, S.: Towards crafting text adversarial samples. arXiv preprint arXiv:1707.02812 (2017)
5. Ebrahimi, J., Rao, A., Lowd, D., et al.: HotFlip: white-box adversarial examples for text classification. arXiv preprint arXiv:1712.06751 (2017)
6. Gao, J., Lanchantin, J., Soffa, M.L., et al.: 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. Li, J., Ji, S., Du, T., et al.: TextBugger: generating adversarial text against real-world applications. arXiv preprint arXiv:1812.05271 (2018)
8. Jin, D., Jin, Z., Zhou, J.T., Szolovits, P.: Is BERT really robust? Natural language attack on text classification and entailment. arXiv:1907.11932 (2019)
9. Liang, B., Li, H., Su, M., et al.: Deep text classification can be fooled. arXiv preprint arXiv:1704.08006 (2017)
10. Alzantot, M., Sharma, Y., Elgohary, A., et al.: Generating natural language adversarial examples. arXiv preprint arXiv:1804.07998 (2018)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
12. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)