Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment

Di Jin, Zhijing Jin, Joey Tianyi Zhou, Peter Szolovits

Abstract

Machine learning algorithms are often vulnerable to adversarial examples that have imperceptible alterations from the original counterparts but can fool the state-of-the-art models. It is helpful to evaluate or even improve the robustness of these models by exposing the maliciously crafted adversarial examples. In this paper, we present TEXTFOOLER, a simple but strong baseline to generate adversarial text. By applying it to two fundamental natural language tasks, text classification and textual entailment, we successfully attacked three target models, including the powerful pre-trained BERT, and the widely used convolutional and recurrent neural networks. We demonstrate three advantages of this framework: (1) effective—it outperforms previous attacks by success rate and perturbation rate, (2) utility-preserving—it preserves semantic content, grammaticality, and correct types classified by humans, and (3) efficient—it generates adversarial text with computational complexity linear to the text length. **Keywords：** Adversarial examples；Natural language.

1 Introduction

Recent advances in deep neural networks have created applications for a range of different domains. In spite of the promising performance achieved by neural models, there are concerns around their robustness, as evidence shows that even a slight perturbation to the input data can fool these models into producing wrong predictions (Goodfellow et al., 2014; Kurakin et al., 2016)[1]. Research in this area is broadly categorised as adversarial machine learning, and it has two sub-fields: adversarial attack, which seeks to generate adversarial examples that fool target models; and adversarial defence, whose goal is to build models that are less susceptible to adversarial attacks.

2 Related works

Recent advances in deep neural networks have created applications for a range of different domains. In spite of the promising performance achieved by neural models, there are concerns around their robustness, as evidence shows that even a slight perturbation to the input data can fool these models into producing wrong predictions (Goodfellow et al., 2014; Kurakin et al., 2016)[1]. Research in this area is broadly categorised as adversarial machine learning, and it has two sub-fields: adversarial attack, which seeks to generate adversarial examples that fool target models; and adversarial defence, whose goal is to build models that are less susceptible to adversarial attacks.

2.1 While-box methods

Most white-box methods are gradient-based, where some form of the gradients (e.g. the sign) with respect to the target model is calculated and added to the input representation. In image processing, the fast gradient sign method (FGSM; Goodfellow et al. (2014))[2] is one of the first studies in attacking image classifiers. Some of its variations include Kurakin et al. (2016)[1]; Dong et al. (2018)[3]. These gradient-based methods could not be applied to texts directly because perturbed word embeddings do not necessarily map to valid words. Methods such as DeepFool (Moosavi-Dezfooli et al., 2016)[1] that rely on perturbing the word embedding space face similar roadblocks. To address the issue of embedding-to-word mapping, Gong et al. (2018)[4] propose to use nearestneighbour search to find the closest words to the perturbed embeddings. However, this method treats all tokens as equally vulnerable and replace all tokens with their nearest neighbours, which leads to non-sensical, word-salad outputs. A solution to this is to replace tokens one-by-one in order of their vulnerability while monitoring the change of the output of the target models. The replacement process stops once the target prediction has changed, minimising the number of changes. Examples of white-box attacks that utilise this approach include TYC (Tsai et al., 2019)**aaa5** and HOTFLIP (Ebrahimi et al., 2017)[5].

2.2 Black-box attacks

Different to white-box attacks, black-box attacks do not require full access to the architecture of the target model. Chen et al. (2017)[4] propose to estimate the loss function of the target model by querying its label probability distributions, while Papernot et al. (2017)[4] propose to construct a substitute of the target model by querying its output labels. The latter approach is arguably more realistic because in most cases attackers only have access to output labels rather than their probability distributions. There is relatively fewer studies on black-box attacks for text. An example is TEXTFOOLER, proposed by Jin et al. (2019), that generates adversarial examples by querying the label probability distribution of the target model. Another is proposed by Alzantot et al. (2018b) where genetic algorithm is used to select the word for substitution.

2.3 Grey-box attacks

Grey-box attacks require an additional training process during which full access to the target model is assumed. However, post-training, the model can be used to generate adversarial examples without querying the target model. Xiao et al. (2018)[6] introduce a generative adversarial network to generate the image perturbation from a noise map. It is, however, not trivial to adapt the method for text directly. It is because text generation involves discrete decoding steps and as such the joint generator and target model architecture is non-differentiable. In terms of adversarial defending, the most straightforward method is to train a robust model on data augmented by adversarial examples. Recently, more methods are proposed for texts, such as those based on interval bound propagation (Jia et al., 2019; Huang et al., 2019)[7], and dirichlet neighborhood ensemble (Zhou et al., 2020).

3 Method

3.1 Problem Formulation

Given a corpus of N sentences $X = \{ X_1, X_2, . . . , X_N \}$, and a corresponding set of N labels Y = $\{ Y_1, Y_2, . . . , Y_N \}$, we have a pre-trained model F : X → Y, which maps the input text space X to the label space Y[8].

For a sentence $X \in X$, a valid adversarial example X_{adv} should conform to the following requirements:

$$F(X_{adv}) \neq F(X), and \quad Sim(X_{adv}, X) \geq \varepsilon, \qquad (1)$$

where Sim : X × X → (0, 1) is a similarity function and ε is the minimum similarity between the original and adversarial examples. In the natural language domain, Sim(•) is often a semantic and syntactic similarity function.

3.2 Threat Model

Under the black-box setting, the attacker is not aware of the model architecture, parameters, or training data. It can only query the target model with supplied inputs, getting as results the predictions and corresponding confidence scores. The proposed approach for adversarial text generation is shown in Algorithm 1, and consists of the two main steps:

3.3 Word Importance Ranking

Given a sentence of n words $X = \{w_1, w_2, \ldots, w_n\}$, we observe that only some key words act as influential signals for the prediction model F, echoing with the discovery of (Niven and Kao 2019) that BERT attends to the statistical cues of some words. Therefore, we create a selection mechanism to choose the words that most significantly influence the final prediction results. Using this selection process, we minimize the alterations, and thus maintain the semantic similarity as much as possible.

Note that the selection of important words is trivial in a white-box scenario, as it can be easily solved by inspecting the gradients of the model F, while most other words are irrelevant. However, under the more common black-box set up in our paper, the model gradients are unavailable. Therefore, we create a selection mechanism as follows. We use the score I_{w_i} to measure the influence of a word $w_i \in X$ towards the classification result $F(X) = Y$. We denote the sentence after deleting the word w_i as $X_{\backslash w_i} = X \backslash \{w_i\} = \{w_1, \ldots, w_{i-1}, w_{i+1}, \ldots w_n\}$, and use $F_Y(\cdot)$ to represent the prediction score for the Y label.

The importance score I_{w_i} is therefore calculated as the prediction change before and after deleting the word w_i, which is formally defined as follows,

$$I_{w_i} = \begin{cases} F_Y(X) - F_Y\left(X_{\backslash w_i}\right), & \text{if } F(X) = F\left(X_{\backslash w_i}\right) = Y \\ \left(F_Y(X) - F_Y\left(X_{\backslash w_i}\right)\right) + \left(F_{\bar{Y}}\left(X_{\backslash w_i}\right) - F_{\bar{Y}}(X)\right), \\ \text{if } F(X) = Y, F\left(X_{\backslash w_i}\right) = \bar{Y}, \text{ and } Y \neq \bar{Y}. \end{cases}$$

After ranking the words by their importance score, we further filter out stop words derived from NLTKE and spaCy libraries such as ”the”, ”when”, and ”none”. This simple step of filtering is important to avoid grammar destruction.

3.4 Word Transformer

For a given word wi X with a high importance score obtained in Step 1, we need to design a word replacement mechanism. A suitable replacement word needs to fulfill the following criteria: it should (1) have similar semantic meaning with the original one, (2) fit within the surrounding context, and (3) force the target model to make wrong predictions. In order to select replacement words that meet such criteria, we propose the following workflow.

Synonym Extraction: We gather a candidate set CANDIDATES for all possible replacements of the selected word wi. CANDIDATES is initiated with N closest synonyms according to the cosine similarity between wi and every other word in the vocabulary.

To represent the words, we use word embeddings from (Mrkˇsi´c et al. 2016). These word vectors are specially curated for finding synonyms, as they achieve the state-of-theart performance on SimLex-999, a dataset designed to measure how well different models judge semantic similarity between words (Hill, Reichart, and Korhonen 2015).

Using this set of embedding vectors, we identify top N synonyms whose cosine similarity with w are greater than δ. Note that enlarging N or lowering δ would both generate more diverse synonym candidates; however, the semantic similarity between the adversary and the original sentence would decrease. In our experiments, empirically setting N to be 50 and δ to be 0.7 strikes a balance between diversity and semantic similarity control.

POS Checking: In the set CANDIDATES of the word wi we only keep the ones with the same part-of-speech (POS)4 as wi. This step is to assure that the grammar of the text is mostly maintained.

Semantic Similarity Checking: For each remaining word c □ CANDIDATES, we substitute it for wi in the sentence X, and obtain the adversarial example Xadv = w1, . . . , wi−1, c, wi+1, . . . , wn. We use the target model F to compute the corresponding prediction scores F(Xadv). We also calculate the sentence semantic similarity between the source X and adversarial counterpart Xadv. Specifically, we use Universal Sentence Encoder (USE) (Cer et al. 2018) to encode the two sentences into high dimensional vectors and use their cosine similarity score as an approximation of semantic similarity. The words resulting in similarity scores above a preset threshold □ are placed into the final candidate pool FINCANDIDATES.

Finalization of Adversarial Examples: In the final candidate pool FINCANDIDATES, if there exists any candidate that can already alter the prediction of the target model, then we select the word with the highest semantic similarity score among these winning candidates. But if not, then we select the word with the least confidence score of label y as the best replacement word for wi, and repeat Step 2 to transform the next selected word.

4 Implementation details

4.1 Comparing with released source codes

The original paper code is available. I used the obtained code to replicate the work and successfully completed the text counter attack.

After finishing the attack on the language model provided in the article, I used the method mentioned in the article to attack the latest model, and found that the result was not as good as the model used in the experiment.

4.2 Experimental environment setup

Text Classification: To study the robustness of our model, we use text classification datasets with various properties, including news topic classification, fake news detection, and sentence- and document-level sentiment analysis, with average text length ranging from tens to hundreds of words.

AG’s News (AG): Sentence-level classification with regard to four news topics: World, Sports, Business, and Science/Technology. Following the practice of Zhang, Zhao, and LeCun (2015), we concatenate the title and description fields for each news article.

Fake News Detection (Fake): Document-level classification on whether a news article is fake or not. The dataset comes from the Kaggle Fake News Challenge.5

MR: Sentence-level sentiment classification on positive and negative movie reviews (Pang and Lee 2005). We use 90% of the data as the training set and 10% as the test set, following the practice in (Li et al. 2018).

IMDB: Document-level sentiment classification on positive and negative movie reviews.

Yelp Polarity (Yelp): Document-level sentiment classification on positive and negative reviews (Zhang, Zhao, and LeCun 2015). Reviews with a rating of 1 and 2 are labeled negative and 4 and 5 positive.

Textual Entailment SNLI: A dataset of 570K sentence pairs derived from image captions. The task is to judge the relationship between two sentences: whether the second sentence can be derived from entailment, contradiction, or neutral relationship with the first sentence (Bowman et al. 2015).

MultiNLI: A multi-genre entailment dataset with a coverage of transcribed speech, popular fiction, and government reports (Williams, Nangia, and Bowman 2017). Compared to SNLI, it contains more linguistic complexity with various written and spoken English text.

5 Results and analysis

The main results of black-box attacks in terms of automatic evaluation on five text classification and two textual entailment tasks are summarized in Table 3 and 4, respectively. Overall, as can be seen from our results, TEXTFOOLER achieves a high success rate when attacking with a limited number of modifications on both tasks. No matter how long the text sequence is, and no matter how accurate the target model is, TEXTFOOLER can always reduce the accuracy from the state-of-the-art values to below 15% (except on the Fake dataset) with less than 20% word perturbation ratio (except the AG dataset under the BERT target model). For instance, it only perturbs 5.1% of the words on average when reducing the accuracy from 89.8% to only 0.3% on the IMDB dataset against the WordLSTM model. Notably, our attack system makes the WordCNN model on the IMDB dataset totally wrong (reaching the accuracy of 0%) with only 3.5% word perturbation rate. In the IMDB dataset which has an average length of 215 words, the system only perturbed 10 words or fewer per sample to conduct successful attacks. This means that our attack system can successfully mislead the classifiers into assigning wrong predictions via subtle manipulation.

Even for BERT, which has achieved seemingly “robust” performance compared with the non-pretrained models such as WordLSTM and WordCNN, our attack model can still reduce its prediction accuracy by about

		WordCNN	WordLSTM	BERT
IMDB	WordCNN	0.0	84.9	90.2
	WordLSTM	74.9	0.0	87.9
	BERT	84.1	85.1	0.0
		InferSent	ESIM	BERT
SNLI	InferSent	0.0	62.7	67.7
	ESIM	49.4	0.0	59.3
	BERT	58.2	54.6	0.0

Table 1: Transferability of adversarial examples on IMDB and SNLI dataset. Row i and column j is the accuracy of adversaries generated for model i evaluated on model j.

	MR		SNLI	
	Af. Acc.	Pert.	Af. Acc.	Pert.
Original	11.5	16.7	4.0	18.5
+ Adv. Training	**18.7**	**21.0**	**8.3**	**20.1**

Table 2: Comparison of the after-attack accuracy ("Af. Acc.") and percentage of perturbed words ("Pert.") of original training ("Original") and adversarial training ("+ Adv. Train") of BERT model on MR and SNLI dataset.

5–7 times on the classification task (e.g., from 95.6% to 6.8% for Yelp dataset) and about 9-22 times on the NLI task (e.g., from 89.4% to 4.0% for SNLI dataset), which is unprecedented. Our curated adversarial examples can contribute to the study of the interpretability of the BERT model (Feng et al. 2018).

Another two observations can be drawn from Table 3 and 4. (1) Models with higher original accuracy is, in general, more difficult to be attacked. For instance, the after-attack accuracy and perturbed word ratio are both higher for the BERT model compared with WordCNN on all datasets. (2) The after-attack accuracy of the Fake dataset is much higher than all other classification datasets for all three target models. We found in experiments that it is easy for the attack system to convert a real news to a fake one, whereas the reverse process is much harder, which is in line with intuition.

We examined transferability of adversarial text, that is, whether adversarial samples curated based on one model can also fool another. For this, we collected the adversarial examples from IMDB and SNLI test sets that are wrongly predicted by one target model and then measured the prediction accuracy of them against the other two target models. As we can see from the results in the Table 1 there is a moderate degree of transferability between models, and the transferability is higher in the textual entailment task than in the text classification task. Moreover, the adversarial samples generated based on the model with higher prediction accuracy, i.e. the BERT model here, show higher transferability.

Our work casts insights on how to better improve the original models through these adversarial examples. We conducted a preliminary experiment on adversarial training, by feeding the models both the original data and the adversarial examples (adversarial examples share the same labels as the original counterparts), to see whether the original models can gain more robustness. We collected the adversarial examples curated from the MR and SNLI training sets that fooled BERT and added them to the original training set. We then used the expanded data to train BERT from scratch and attacked this adversarially-trained model. As is seen in the attack results in Table 12, both the after-attack accuracy and perturbed words ratio after adversarial re-training get higher, indicating the greater difficulty to attack. This reveals one of the potency of our attack system,– we can enhance the robustness of a model to future attacks by training it with the generated adversarial examples.

Comparing the semantic similarity scores and the perturbed word ratios in both Table 3 and 4, we find that the two results have a high positive correlation. Empirically, when the text length is longer than 10 words, the semantic similarity measurement becomes more stable. Since the average text lengths of text classification datasets are all above 20 words and those of textual entailment datasets are around or below 10 words, we need to treat the semantic similarity scores of these two tasks individually. Therefore, we performed a linear regression analysis between the word perturbation ratio and semantic similarity for each task and obtained r-squared values of 0.94 and 0.97 for text classification and textual entailment tasks, respectively. Such high values of r-squared reveal that our proposed semantic similarity has high correlation (negative) with the perturbed words ratio, which can both be good automatic measurements to evaluate the degree of alterations of original text.

We include the average text length of each dataset in the last row of Table 3 and 4 so that it can be conveniently compared against the query number. The query number is almost linear to the text lengthm, with a ratio in (2, 8). Longer text correlates with a smaller ratio, which validates the efficiency of TEXTFOOLER.

6 Conclusion and future work

Overall, we study adversarial attacks against state-of-the-art text classification and textual entailment models under the black-box setting. Extensive experiments demonstrate that the effectiveness of our proposed system, TEXTFOOLER, at generating targeted adversarial texts. Human studies validated that the generated adversarial texts are legible, grammatical, and similar in meaning to the original texts.

References

[1] BOWMAN P, Angeli. A large annotated corpus for learning natural language inference[J]. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing(EMNLP), 2015, 08(08): 1500-1510.

[2] HILL F K, Reichart R. Simlex-999: Evaluating semantic models with (genuine) similarity estimation[J]. Computational Linguistics, 2015, 41(4): 665-695.

[3] HOCHREITER S. Long shortterm memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[4] PANG L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales[J]. In Proceedings of the 43rd annual meeting on association for computational linguistics, 2005, 9(8): 115-124.

[5] PENNINGTON M, Socher. Glove: Global vectors for word representation[J]. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, 9(8): 1532-1543.

[6] RIBEIRO G, Singh. Semantically equivalent adversarial rules for debugging nlp models[J]. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 2018, 9(8): 856-865.

[7] ZHANG L, Zhao. Characterlevel convolutional networks for text classification[J]. In Advances in neural information processing systems, 2014, 9(8): 649-657.

[8] YU J. Research on Runge Phenomenon[J]. Advances in Applied Mathematics, 2019, 08(08): 1500-1510.