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

**Di Jin,[1]* Zhijing Jin,[2]* Joey Tianyi Zhou,[3] Peter Szolovits[1]**
[1]Computer Science & Artificial Intelligence Laboratory, MIT
[2]University of Hong Kong
[3]A*STAR, Singapore
jindi15@mit.edu, zhijing.jin@connect.hku.hk, zhouty@ihpc.a-star.edu.sg, psz@mit.edu

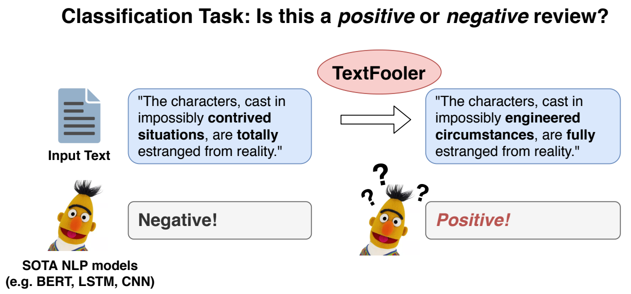
## Abstract

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.[1]

## Introduction

In the last decade, Machine Learning (ML) models have achieved remarkable success in various tasks such as classification, regression and decision making. However, recently they have been found vulnerable to adversarial examples that are legitimate inputs altered by small and often imperceptible perturbations (Kurakin, Goodfellow, and Bengio 2016a; Kurakin, Goodfellow, and Bengio 2016b; Papernot et al. 2017; Zhao, Dua, and Singh 2017). These carefully curated examples are correctly classified by a human observer but can fool a target model, raising serious concerns regarding the security and integrity of existing ML algorithms. On the other hand, it is showed that robustness and generalization of ML models can be improved by crafting high-quality adversaries and including them in the training data (Goodfellow, Shlens, and Szegedy 2015).

While existing works on adversarial examples have obtained success in the image and speech domains (Szegedy et al. 2013; Carlini and Wagner 2018), it is still challenging to deal with text data due to its discrete nature. Formally, besides the ability to fool the target models, outputs of a natural language attacking system should also meet three key utility-preserving properties: (1) human prediction consistency—prediction by humans should remain unchanged, (2) semantic similarity—the crafted example should bear the same meaning as the source, as judged by humans, and (3) language fluency—generated examples should look natural and grammatical. Previous works barely conform to all three requirements. For example, methods such as word misspelling (Li et al. 2018; Gao et al. 2018), single-word erasure (Li, Monroe, and Jurafsky 2016), and phrase insertion and removal (Liang et al. 2017) result in unnatural sentences. Moreover, there is almost no work that attacks the newly-risen BERT model on text classification.



"The characters, cast in impossibly **contrived situations**, are **totally** estranged from reality."

"The characters, cast in impossibly **engineered circumstances**, are **fully** estranged from reality."

Figure 1: Our model TextFooler slightly change the input text but completely altered the prediction result.

In this work, we present TEXTFOOLER, a simple but strong baseline for natural language attack in the black-box setting, a common case where no model architecture or parameters are accessible (Figure1. We design a more comprehensive paradigm to create both semantically and syntactically similar adversarial examples that meet the three desiderata mentioned above. Specifically, we first identify the important words for the target model and then prioritize to replace them with the most semantically similar and grammatically correct words until the prediction is altered. We successfully applied this framework to attack three state-of-the-art models in five text classification tasks and two tex-

*Equal Contribution. Order determined by swapping that in the previous paper at https://arxiv.org/abs/1901.11333.

[1]The code, pre-trained target models, and test examples are available at https://github.com/jind11/TextFooler.

tual entailment tasks, respectively. On the adversarial examples, we can reduce the accuracy of almost all target models in all tasks to below 10% with only less than 20% of the original words perturbed. In addition, we validate that the generated examples are (1) correctly classified by human evaluators, (2) semantically similar to the original text, and (3) grammatically acceptable by human judges.

Our main contributions are summarized as follows:

- We propose a simple but strong baseline, TEXTFOOLER, to quickly generate high-profile utility-preserving adversarial examples that force the target models to make wrong predictions under the black-box setting.
- We evaluate TEXTFOOLER on three state-of-the-art deep learning models over five popular text classification tasks and two textual entailment tasks, and it achieved the state-of-the-art attack success rate and perturbation rate.
- We propose a comprehensive four-way automatic and three-way human evaluation of language adversarial attacks to evaluate the effectiveness, efficiency, and utility-preserving properties of our system.
- We open-source the code, pre-trained target models, and test samples for the convenience of future benchmarking.

# Method

## Problem Formulation

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

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

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

where $\text{Sim} : \mathcal{X} \times \mathcal{X} \to (0, 1)$ is a similarity function and $\epsilon$ is the minimum similarity between the original and adversarial examples. In the natural language domain, $\text{Sim}(\cdot)$ is often a semantic and syntactic similarity function.

## 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:

**Step 1: Word Importance Ranking (line 1-6)** Given a sentence of $n$ words $X = \{w_1, w_2, \ldots, w_n\}$, we observe that only some keywords 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.

**Algorithm 1** Adversarial Attack by TEXTFOOLER

```
Input: Sentence example $X = \{w_1, w_2, ..., w_n\}$, the corresponding ground truth label $Y$, target model $F$, sentence similarity function $\text{Sim}(\cdot)$, sentence similarity threshold $\epsilon$, word embeddings Emb over the vocabulary Vocab.
Output: Adversarial example $X_{\text{adv}}$
 1: Initialization: $X_{\text{adv}} \leftarrow X$
 2: for each word $w_i$ in $X$ do
 3:     Compute the importance score $I_{w_i}$ via Eq. (2)
 4: end for
 5:
 6: Create a set $W$ of all words $w_i \in X$ sorted by the descending order of their importance score $I_{w_i}$.
 7: Filter out the stop words in $W$.
 8: for each word $w_j$ in $W$ do
 9:     Initiate the set of candidates CANDIDATES by extracting the top $N$ synonyms using CosSim(Emb$_{w_j}$, Emb$_{\text{word}}$) for each word in Vocab.
10:     CANDIDATES ← POSFilter(CANDIDATES)
11:     FINCANDIDATES ← { }
12:     for $c_k$ in CANDIDATES do
13:         $X' \leftarrow$ Replace $w_j$ with $c_k$ in $X_{\text{adv}}$
14:         if $\text{Sim}(X', X_{\text{adv}}) > \epsilon$ then
15:             Add $c_k$ to the set FINCANDIDATES
16:             $Y_k \leftarrow F(X')$
17:             $P_k \leftarrow F_{Y_k}(X')$
18:         end if
19:     end for
20:     if there exists $c_k$ whose prediction result $Y_k \neq Y$ then
21:         In FINCANDIDATES, only keep the candidates $c_k$ whose prediction result $Y_k \neq Y$
22:         $c^* \leftarrow \underset{c \in \text{FINCANDIDATES}}{\text{argmax}} \; \text{Sim}(X, X'_{w_j \to c})$
23:         $X_{\text{adv}} \leftarrow$ Replace $w_j$ with $c^*$ in $X_{\text{adv}}$
24:         return $X_{\text{adv}}$
25:     else if $P_{Y_k}(X_{\text{adv}}) > \underset{c_k \in \text{FINCANDIDATES}}{\min} P_k$ then
26:         $c^* \leftarrow \underset{c_k \in \text{FINCANDIDATES}}{\text{argmin}} P_k$
27:         $X_{\text{adv}} \leftarrow$ Replace $w_j$ with $c^*$ in $X_{\text{adv}}$
28:     end if
29: end for
30: return None
```

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 $X_{\setminus w_i} = X \setminus \{w_i\} = \{w_1, \ldots, w_{i-1}, w_{i+1}, \ldots w_n\}$, 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,

$$I_{w_i} = \begin{cases} F_Y(X) - F_Y(X_{\backslash w_i}), & \text{if } F(X) = F(X_{\backslash w_i}) = Y \\ (F_Y(X) - F_Y(X_{\backslash w_i})) + (F_{\bar{Y}}(X_{\backslash w_i}) - F_{\bar{Y}}(X)), & \\ \quad \text{if } F(X) = Y, F(X_{\backslash w_i}) = \bar{Y}, \text{ and } Y \neq \bar{Y}. & \end{cases} \tag{2}$$

After ranking the words by their importance score, we further filter out stop words derived from NLTK[2] and spaCy[3] libraries such as "the", "when", and "none". This simple step of filtering is important to avoid grammar destruction.

**Step 2: Word Transformer (line 7-30)** Given the word $w_i \in X$ 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 $w_i$. Candidates is initiated with $N$ closest synonyms according to the cosine similarity between $w_i$ and every other word in the vocabulary.

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

Using this set of embedding vectors, we identify the top $N$ synonyms whose cosine similarity with $w$ is higher than $\delta$. Note that enlarging $N$ or lowering $\delta$ 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 $\delta$ to be 0.7 strikes a balance between diversity and semantic similarity control.

***POS Checking:*** In the set Candidates of the word $w_i$, we only keep the ones with the same part-of-speech (POS)[4] as $w_i$. This step is to assure that the grammar of the text is mostly maintained (line 10 in Algorithm 1).

***Semantic Similarity Checking:*** For each remaining word $c \in$ Candidates, we substitute it for $w_i$ in the sentence $X$, and obtain the adversarial example $X_{\text{adv}} = \{w_1, \ldots, w_{i-1}, c, w_{i+1}, \ldots, w_n\}$. We use the target model $F$ to compute the corresponding prediction scores $F(X_{\text{adv}})$. We also calculate the sentence semantic similarity between the source $X$ and adversarial counterpart $X_{\text{adv}}$. 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 $\epsilon$ are placed into the final candidate pool FinCandidates (line 11-19 in Algorithm 1).

***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 choose 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 $w_i$, and repeat Step 2 to transform the next selected word (line 20-30 in Algorithm 1).

Overall, the algorithm first uses Step 1 to rank the words by their importance scores, and then repeats Step 2 to find replacements for each word in the sentence $X$ until the prediction of the target model is altered.

# Experiments

## Tasks

We study the effectiveness of our adversarial attack on two important NLP tasks, text classification and textual entailment.[5] The dataset statistics are summarized in Table 1. Following the practice by Alzantot et al. (2018), we evaluate our algorithm on a set of 1,000 examples randomly selected from the test set.

| Task | Dataset | Train | Test | Avg Len |
|---|---|---|---|---|
| **Classification** | AG's News | 120K | 7.6K | 43 |
| | Fake News | 18.8K | 2K | 885 |
| | MR | 9K | 1K | 20 |
| | IMDB | 25K | 25K | 215 |
| | Yelp | 560K | 38K | 152 |
| **Entailment** | SNLI | 570K | 3K | 8 |
| | MultiNLI | 433K | 10K | 11 |

Table 1: Overview of the datasets.

**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.[6] We concatenate the title and news body of each article.
- **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.[7]

[2] https://www.nltk.org/

[3] https://spacy.io/

[4] We used the off-the-shelf spaCy tagger, available at https://spacy.io/api/tagger

[5] Our datasets are at http://bit.ly/nlp_adv_data

[6] https://www.kaggle.com/c/fake-news/data

[7] https://datasets.imdbws.com/

- **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 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.

## Attacking Target Models

For each dataset, we train three state-of-the-art models on the training set, and achieved test set accuracy scores similar to the original implementation, as shown in Table 2. We then generate adversarial examples that are semantically similar to the test set to attack the trained models and make them generate different results.

| | WordCNN | WordLSTM | BERT |
|---|---|---|---|
| **AG** | 92.5 | 93.1 | 94.6 |
| **Fake** | 99.9 | 99.9 | 99.9 |
| **MR** | 79.9 | 82.2 | 85.8 |
| **IMDB** | 89.7 | 91.2 | 92.2 |
| **Yelp** | 95.2 | 96.6 | 96.1 |
| | **InferSent** | **ESIM** | **BERT** |
| **SNLI** | 84.6 | 88.0 | 90.7 |
| **MultiNLI** | 71.1/71.5 | 76.9/76.5 | 83.9/84.1 |

Table 2: Original accuracy of target models on standard test sets.

On the sentence classification task, we target at three models: word-based convolutional neural network (WordCNN) (Kim 2014), word-based long-short term memory (WordLSTM) (Hochreiter and Schmidhuber 1997), and the state-of-the-art Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al. 2018).

For the WordCNN model, we used three window sizes of 3, 4, and 5, and 100 filters for each window size with a dropout of 0.3. For the WordLSTM, we used a 1-layer bidirectional LSTM with 150 hidden units and a dropout of 0.3. For both models, we used the 200-dimensional Glove word embeddings pre-trained on 6B tokens from Wikipedia and Gigawords (Pennington, Socher, and Manning 2014). We used the 12-layer BERT model with 768 hidden units and 12 heads, with 110M parameters, which is called the base-uncased version.[8]

We also implemented three target models on the textual entailment task: standard InferSent[9] (Conneau et al. 2017), ESIM[10] (Chen et al. 2016), and fine-tuned BERT.

### Setup of Automatic Evaluation

We first report the accuracy of the target models on the original test samples before the attack as the original accuracy. Then we measure the accuracy of the target models against the adversarial samples crafted from the test samples, denoted as *after-attack accuracy*. By comparing these two accuracy scores, we can evaluate how successful the attack is, — the more significant the gap between the original and after-attack accuracy signals, the more successful our attack is. Apart from these accuracies, we also report the perturbed word percentage as the ratio of the number of perturbed words to the text length. Furthermore, we apply USE[11] to measure the semantic similarity between the original and adversarial texts. These two metrics, the perturbed words percentage and the semantic similarity score, which together evaluate how semantically similar the original and adversarial texts are. We finally report the number of queries the attack system made to the target model and fetch the output probability scores. This metric can reveal the efficiency of the attack model.

### Setup of Human Evaluation

We conduct human evaluation on three criteria: semantic similarity, grammaticality, and classification accuracy. We randomly select 100 test sentences of each task to generate adversarial examples, one targeting WordLSTM on the MR dataset and another targeting BERT on SNLI. We first shuffled a mix of original and adversarial texts and asked human judges to rate the grammaticality of them on a Likert scale of $1-5$, similar to the practice of (Gagnon-Marchand et al. 2018). Next, we evaluate the classification consistency by asking humans to classify each example in the shuffled mix of the original and adversarial sentences and then calculate the consistency rate of both classification results. Lastly, we evaluated the semantic similarity of the original and adversarial sentences by asking humans to judge whether the generated adversarial sentence is similar, ambiguous, or dissimilar to the source sentence. Each task is completed by two independent human judges who are native English speakers. The volunteers have university-level education backgrounds and passed a test batch before they started annotation.

## Results

### Automatic Evaluation

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

[8] https://github.com/huggingface/pytorch-pretrained-BERT

[9] https://github.com/facebookresearch/InferSent

[10] https://github.com/coetaur0/ESIM

[11] https://tfhub.dev/google/ universal-sentence-encoder

| | WordCNN | | | | | WordLSTM | | | | | BERT | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **MR** | **IMDB** | **Yelp** | **AG** | **Fake** | **MR** | **IMDB** | **Yelp** | **AG** | **Fake** | **MR** | **IMDB** | **Yelp** | **AG** | **Fake** |
| **Original Accuracy** | 78.0 | 89.2 | 93.8 | 91.5 | 96.7 | 80.7 | 89.8 | 96.0 | 91.3 | 94.0 | 86.0 | 90.9 | 97.0 | 94.2 | 97.8 |
| **After-Attack Accuracy** | 2.8 | 0.0 | 1.1 | 1.5 | 15.9 | 3.1 | 0.3 | 2.1 | 3.8 | 16.4 | 11.5 | 13.6 | 6.6 | 12.5 | 19.3 |
| **% Perturbed Words** | 14.3 | 3.5 | 8.3 | 15.2 | 11.0 | 14.9 | 5.1 | 10.6 | 18.6 | 10.1 | 16.7 | 6.1 | 13.9 | 22.0 | 11.7 |
| **Semantic Similarity** | 0.68 | 0.89 | 0.82 | 0.76 | 0.82 | 0.67 | 0.87 | 0.79 | 0.63 | 0.80 | 0.65 | 0.86 | 0.74 | 0.57 | 0.76 |
| **Query Number** | 123 | 524 | 487 | 228 | 3367 | 126 | 666 | 629 | 273 | 3343 | 166 | 1134 | 827 | 357 | 4403 |
| **Average Text Length** | 20 | 215 | 152 | 43 | 885 | 20 | 215 | 152 | 43 | 885 | 20 | 215 | 152 | 43 | 885 |

Table 3: Automatic evaluation results of the attack system on text classification datasets, including the original model prediction accuracy before being attacked ("Original Accuracy"), the model accuracy after the adversarial attack ("After-Attack Accuracy"), the percentage of perturbed words with respect to the original sentence length ("% Perturbed Words"), and the semantic similarity between original and adversarial samples ("Semantic Similarity").

| | InferSent | | ESIM | | BERT | |
|---|---|---|---|---|---|---|
| | **SNLI** | **MultiNLI (m/mm)** | **SNLI** | **MultiNLI (m/mm)** | **SNLI** | **MultiNLI (m/mm)** |
| **Original Accuracy** | 84.3 | 70.9/69.6 | 86.5 | 77.6/75.8 | 89.4 | 85.1/82.1 |
| **After-Attack Accuracy** | 3.5 | 6.7/6.9 | 5.1 | 7.7/7.3 | 4.0 | 9.6/8.3 |
| **% Perturbed Words** | 18.0 | 13.8/14.6 | 18.1 | 14.5/14.6 | 18.5 | 15.2/14.6 |
| **Semantic Similarity** | 0.50 | 0.61/0.59 | 0.47 | 0.59/0.59 | 0.45 | 0.57/0.58 |
| **Query Number** | 57 | 70/83 | 58 | 72/87 | 60 | 78/86 |
| **Average Text Length** | 8 | 11/12 | 8 | 11/12 | 8 | 11/12 |

Table 4: Automatic evaluation results of the attack system on textual entailment datasets. "m" means matched, and "mm" means mismatched, which are the two variants of the MultiNLI development set.

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 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 are, 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 article to a fake one, whereas the reverse process is much harder, which is in line with intuition.

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 a high (negative) correlation with the perturbed words ratio, which can both be useful automatic measurements to evaluate the degree of alterations of the 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 length, with a ratio in $(2, 8)$. Longer text correlates with a smaller ratio, which validates the efficiency of TEXTFOOLER.

**Benchmark Comparison** We compared TEXTFOOLER with the previous state-of-the-art adversarial attack systems against the same target model and dataset. Our baselines include (Li et al. 2018) that generates misspelled words by character- and word-level perturbation, (Alzantot et al. 2018) that iterates through every word in the sentence and find its perturbation, and (Kuleshov et al. 2018) that uses word replacement by greedy heuristics. From the results in

| Dataset | Model | Success Rate | % Perturbed Words |
|---|---|---|---|
| **IMDB** | (Li et al. 2018) | 86.7 | 6.9 |
| | (Alzantot et al. 2018) | 97.0 | 14.7 |
| | Ours | **99.7** | **5.1** |
| **SNLI** | (Alzantot et al. 2018) | 70.0 | 23.0 |
| | Ours | **95.8** | **18.0** |
| **Yelp** | (Kuleshov et al. 2018) | 74.8 | - |
| | Ours | **97.8** | **10.6** |

Table 5: Comparison of our attack system against other published systems. The target model for IMDB and Yelp is LSTM and SNLI is InferSent.

Table 5, we can see that our system beats the previous state-of-the-art models by both the attack success rate (calculated by dividing the number of wrong predictions by the total number of adversarial examples) and perturbed word ratio.

### Human Evaluation

We sampled 100 adversarial examples on the MR dataset with the WordLSTM and 100 examples on SNLI with BERT. We verified the quality of our examples via three experiments. First, we ask human judges to give a grammaticality score of a shuffled mix of original and adversarial text.

As shown in Table 7, the grammaticality of the adversarial text are close to the original text on both datasets. By sensibly substituting synonyms, TextFooler generates smooth outputs such as "the big metaphorical wave" in Table 6.

We then asked the human raters to assign classification labels to a shuffled set of original and adversarial samples. The overall agreement between the labels of the original sentence and the adversarial sentence is relatively high, with $92\%$ on MR and $85\%$ on SNLI. Though our adversarial examples are not perfect in every case, this shows that majorities of adversarial sentences have the same attribute as the original sentences from humans' perspective. Table 6 shows typical examples of sentences with almost the same meanings that result in contradictory classifications by the target model.

Lastly, we asked the judges to determine whether each adversarial sample retains the meaning of the original sentence. They need to decide whether the synthesized adversarial example is similar, ambiguous, or dissimilar to the provided original sentence. We regard similar as 1, ambiguous as $0.5$, and dissimilar as $0$, and obtained sentence similarity scores of $0.91$ on MR and $0.86$ on SNLI, which shows the perceived difference between original and adversarial text is small.

## Discussion

### Ablation Study

**Word Importance Ranking** To validate the effectiveness of Step 1 in Algorithm 1, i.e., the word importance ranking, we remove this step and instead randomly select the words in the text to perturb. We keep the perturbed word ratio and Step 2 the same. We use BERT as the target model and test on three datasets: MR, AG, and SNLI. The results are summarized in Table 8. After removing Step 1 and instead randomly selecting the words to perturb, the after-attack accuracy increases by more than $45\%$ on all three datasets, which reveals that the attack becomes ineffective without the word importance ranking step. The word importance ranking process is crucial to the algorithm in that it can accurately and efficiently locate the words which cast the most significant effect on the predictions of the target model. This strategy can also reduce the number of perturbed words so as to maintain the semantic similarity as much as possible.

**Semantic Similarity Constraint** In Step 2 of Algorithm 1, for every possible word replacement, we check the semantic similarity between the newly generated sample and the original text, and adopt this replacement only when the similarity is above a preset threshold $\epsilon$. We found that this strategy can effectively filter out irrelevant synonyms to the selected word. As we can see from the examples in Table 9, the synonyms extracted by word embeddings are noisy, so directly injecting them into the text as adversarial samples would probably shift the semantic meaning significantly. By applying the sentence-level semantic similarity constraint, we can obtain more related synonyms as suitable replacements. Table 10 compares the automatic evaluation metrics before and after removing the semantic similarity constraint for the BERT-Base target model. From this table, we see that without the constraint, the attacking becomes easier (after-attack accuracy, perturbed words percentage, and number of queries all decrease). However, the semantic similarity between the original text and the corresponding adversaries also decrease, indicating the degradation of meaning preservation.[12]

### Transferability

We examined the 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 Table 11, 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.

### Adversarial Training

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

[12] To be noted, the decrease of semantic similarity score is much smaller for the NLI datasets. This is because the lengths of NLI sentences are generally less than 20 tokens, and the USE encoder is not as sensitive to the change of semantic meaning in this situation.

| Movie Review (Positive (POS) ↔ Negative (NEG)) | |
|---|---|
| **Original (Label: NEG)** | The characters, cast in impossibly ***contrived situations***, are ***totally*** estranged from reality. |
| **Attack (Label: POS)** | The characters, cast in impossibly ***engineered circumstances***, are ***fully*** estranged from reality. |
| **Original (Label: POS)** | It cuts to the ***knot*** of what it actually means to face your ***scares***, and to ride the ***overwhelming*** **metaphorical wave** that life wherever it takes you. |
| **Attack (Label: NEG)** | It cuts to the ***core*** of what it actually means to face your ***fears***, and to ride the ***big*** **metaphorical wave** that life wherever it takes you. |
| **SNLI (Entailment (ENT), Neutral (NEU), Contradiction (CON))** | |
| **Premise** | Two small boys in blue soccer uniforms use a wooden set of steps to wash their hands. |
| **Original (Label: CON)** | The boys are in band ***uniforms***. |
| **Adversary (Label: ENT)** | The boys are in band ***garment***. |
| **Premise** | A child with wet hair is holding a butterfly decorated beach ball. |
| **Original (Label: NEU)** | The ***child*** is at the ***beach***. |
| **Adversary (Label: ENT)** | The ***youngster*** is at the ***shore***. |

Table 6: Examples of original and adversarial sentences from MR (WordLSTM) and SNLI (BERT) datasets.

| Source Text | MR (WordLSTM) | SNLI (BERT) |
|---|---|---|
| **Original** | 4.22 | 4.50 |
| **Adversarial** | 4.01 | 4.27 |

Table 7: Grammaticality of original and adversarial examples for MR (WordLSTM) and SNLI (BERT) on $1-5$ scale.

| | MR | AG | SNLI |
|---|---|---|---|
| **% Perturbed Words** | 16.7 | 22.0 | 18.5 |
| **Original Accuracy** | 86.0 | 94.2 | 89.4 |
| **After-Attack Accuracy** | 11.5 | 12.5 | 4.0 |
| **After-Attack Accuracy (Random)** | **68.3** | **80.8** | **59.2** |

Table 8: Comparison of the after-attack accuracies before and after removing the word importance ranking of Algorithm 1. For control, Step 2 and the perturbed words ratio are kept the same. BERT model is used as the target model.

| | |
|---|---|
| **Original** | like a south of the border melrose ***place*** |
| **Adversarial** | like a south of the border melrose ***spot*** |
| **- Sim.** | like a south of the border melrose ***mise*** |
| **Original** | their computer animated faces are very ***expressive*** |
| **Adversarial** | their computer animated face are very ***affective*** |
| **- Sim.** | their computer animated faces are very ***diction*** |

Table 9: Qualitative comparison of adversarial attacks with and without the semantic similarity constraint ("-Sim."). We highlight the original word, TextFooler's replacement, and the replacement without semantic constraint.

| | MR | IMDB | SNLI | MNLI(m) |
|---|---|---|---|---|
| **After-Attack Accu.** | 11.5/6.2 | 13.6/11.2 | 4.0/3.6 | 9.6/7.9 |
| **% Perturbed Words** | 16.7/14.8 | 6.1/4.0 | 18.5/18.3 | 15.2/14.5 |
| **Query Number** | 166/131 | 1134/884 | 60/57 | 78/70 |
| **Semantic Similarity** | 0.65/0.58 | 0.86/0.82 | 0.45/0.44 | 0.57/0.56 |

Table 10: Comparison of automatic evaluation metrics with and without the semantic similarity constraint. (Numbers in the left and right of the symbol "/" represent results with and without the constraint, respectively). The target model is BERT-Base.

| | | WordCNN | WordLSTM | BERT |
|---|---|---|---|---|
| **IMDB** | **WordCNN** | — | 84.9 | 90.2 |
| | **WordLSTM** | 74.9 | — | 87.9 |
| | **BERT** | 84.1 | 85.1 | — |
| | | **InferSent** | **ESIM** | **BERT** |
| **SNLI** | **InferSent** | — | 62.7 | 67.7 |
| | **ESIM** | 49.4 | — | 59.3 |
| | **BERT** | 58.2 | 54.6 | — |

Table 11: 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 $j$.

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 potencies of our attack system,— we can enhance the robustness of a model to future attacks by training it with the generated adversarial examples.

## Error Analysis

Our adversarial samples are susceptible to three types of errors: word sense ambiguity, grammatical error, and task-sensitive content shift. Although large thesauri are available, a word usually has many meanings, with a set of synonyms for each word sense. One example can be the transfer from an original sentence "One man *shows* the ransom money to the other" to the synthesized "One man *testify* the ransom money to the other", where "testify" in this case is not the appropriate synonym of "show". Grammatical errors are also frequent in text generation. For example, the sentence "A man with headphones is *biking*" and "A man with headphones is *motorcycle*" differ by the word "biking", which can be both a noun and a verb, as well as a reasonably similar word to "motorcycle".

As future work, some carefully designed heuristics can be applied to filter out grammatical errors. The content shift can be seen in a task-specific situation. In the sentiment classification task, a change of words might not affect the overall sentiment, whereas in the task of textual entailment, the sub-

|  | 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 12: 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.

stitution of words might result in a fundamental difference. For example, if the premise is "a *kid* with red hat is running", and the original hypothesis is "a *kid* is running (*entailment*)", then if the adversarial example becomes "a *girl* is running", the sensible result turns into *neutral* instead.

## Related Work

Adversarial attack has been extensively studied in computer vision (Goodfellow, Shlens, and Szegedy 2014; Kurakin, Goodfellow, and Bengio 2016a; Moosavi-Dezfooli et al. 2017). Most works make gradient-based perturbation on continuous input spaces (Szegedy et al. 2013; Goodfellow, Shlens, and Szegedy 2014). Adversarial attack on discrete data such as text is more challenging. Inspired by the approaches in computer vision, early work in language adversarial attack focus on variations of gradient-based methods. For example, Zhao, Dua, and Singh (2017) transform input data into a latent representation by generative adversarial networks (GANs), and then retrieved adversaries close to the original instance in the latent space. Other works observed the intractability of GAN-based models on the text and the shift in semantics in the latent representations, so heuristic methods such as scrambling, misspelling, or removing words were proposed (Ebrahimi et al. 2017; Alzantot et al. 2018; Li, Monroe, and Jurafsky 2016; Li et al. 2018). Instead of manually designing these rules, Ribeiro, Singh, and Guestrin (2018) automatically craft the semantically equivalent adversarial rules from the machine-generated paraphrases based on the back-translation technique from the machine translation field. By human evaluation, they demonstrate that the quality of the adversaries created by their rules is better than that generated by humans. More interestingly, Feng et al. (2018) used the way to produce adversarial examples to help improve the interpretability of neural models by encouraging high entropy outputs on the adversaries.

## Conclusion

Overall, we study adversarial attacks against state-of-the-art text classification and textual entailment models under the black-box setting. Extensive experiments demonstrate 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.

## Acknowledgments

We thank the MIT Clinical Decision-Making Group and Professor Zheng Zhang for insightful discussions. We especially appreciate Heather Berlin, Yilun Du, Laura Koemmpel, and many other helpers for their high-quality human evaluation. Zhijing Jin also thank Zhutian Yang and Yixuan Zhang for strong personal support.

## References

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

[2015] Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

[2018] Carlini, N., and Wagner, D. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. *arXiv preprint arXiv:1801.01944.*

[2018] Cer, D.; Yang, Y.; Kong, S.-y.; Hua, N.; Limtiaco, N.; John, R. S.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175.*

[2016] Chen, Q.; Zhu, X.; Ling, Z.; Wei, S.; Jiang, H.; and Inkpen, D. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038.*

[2017] Conneau, A.; Kiela, D.; Schwenk, H.; Barrault, L.; and Bordes, A. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364.*

[2018] Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

[2017] Ebrahimi, J.; Rao, A.; Lowd, D.; and Dou, D. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751.*

[2018] Feng, S.; Wallace, E.; Grissom, I.; Iyyer, M.; Rodriguez, P.; Boyd-Graber, J.; et al. 2018. Pathologies of neural models make interpretations difficult. *arXiv preprint arXiv:1804.07781.*

[2018] Gagnon-Marchand, J.; Sadeghi, H.; Haidar, M.; Rezagholizadeh, M.; et al. 2018. Salsa-text: self attentive latent space based adversarial text generation. *arXiv preprint arXiv:1809.11155.*

[2018] Gao, J.; Lanchantin, J.; Soffa, M. L.; and Qi, Y. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. *arXiv preprint arXiv:1801.04354.*

[2014] Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572.*

[2015] Goodfellow, I.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations.*

[2015] Hill, F.; Reichart, R.; and Korhonen, A. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics* 41(4):665–695.

[1997] Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

[2014] Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

[2018] Kuleshov, V.; Thakoor, S.; Lau, T.; and Ermon, S. 2018. Adversarial examples for natural language classification problems.

[2016a] Kurakin, A.; Goodfellow, I.; and Bengio, S. 2016a. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.

[2016b] Kurakin, A.; Goodfellow, I. J.; and Bengio, S. 2016b. Adversarial machine learning at scale. *CoRR* abs/1611.01236.

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

[2016] Li, J.; Monroe, W.; and Jurafsky, D. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

[2017] Liang, B.; Li, H.; Su, M.; Bian, P.; Li, X.; and Shi, W. 2017. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*.

[2017] Moosavi-Dezfooli, S.-M.; Fawzi, A.; Fawzi, O.; and Frossard, P. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1765–1773.

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

[2019] Niven, T., and Kao, H.-Y. 2019. Probing neural network comprehension of natural language arguments. *arXiv preprint arXiv:1907.07355*.

[2005] Pang, B., and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, 115–124.

[2017] Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z. B.; and Swami, A. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 506–519. ACM.

[2014] Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

[2018] Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 856–865.

[2013] Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

[2017] Williams, A.; Nangia, N.; and Bowman, S. R. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

[2015] Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, 649–657.

[2017] Zhao, Z.; Dua, D.; and Singh, S. 2017. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*.