TextFool: Fool your Model with Natural Adversarial Text

Di Jin*
MIT CSAIL
jindi@csail.mit.edu

Zhijing Jin*
University of Hong Kong
zhijing.jin@connect.hku.hk

Joey Tianyi Zhou
A*STAR, Singapore
zhouty@ihpc.a-star.edu.sg

Peter Szolovits
MIT CSAIL
psz@mit.edu

Abstract

Machine learning algorithms are often vulnerable to adversarial examples that are imperceptible to humans 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 the TextFool, a general attack framework for generating adversarial texts. By successfully applying it to two fundamental natural language tasks, text classification and textual entailment, against various target models, convolutional and recurrent neural networks as well as the most powerful pre-trained BERT, we demonstrate the advantages of this framework in three ways: (i) effective—it outperforms state-of-the-art attacks in terms of success rate and perturbation rate; (ii) utility-preserving—it preserves semantic content and grammaticality, and remains correctly classified by humans; and (iii) efficient—it generates adversarial text with computational complexity linear in 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 to be vulnerable to adversarial examples that are legitimate inputs altered by small and often imperceptible perturbations (Kurakin et al., 2016a,b; Papernot et al., 2017; Zhao et al., 2017). These carefully curated examples remain correctly classified by a human observer but can fool a targeted model. This has raised serious concerns regarding the security and integrity of existing ML algorithms. On the other hand, it has been demonstrated that robustness and generalization of ML models can be improved by crafting high-quality adversaries and including them in the training data (Goodfellow et al., 2015).

While existing works on adversarial examples have obtained great 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 such an attacking system in the text domain should also meet three key utility-preserving properties: 1) Human prediction consistency: prediction should remain unchanged by humans; 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 be correct in grammar. Previous works have barely conformed to all three requirements (Li et al., 2016; Papernot et al., 2016). Especially, some works proposed to attack the targeted classifiers by replacing words in text with deliberately crafted misspelled words (Li et al., 2016; Liang et al., 2017; Gao et al., 2018; Li et al., 2018), which results in ungrammatical sentences.

In this work, we present a general framework, TextFool, to generate adversarial examples in the context of natural language and a black-box setting, where no architecture or parameters of models are accessible. We aim to create both semantically and syntactically similar adversarial examples that meet the above-mentioned three desiderata. Basically, we first identify the most 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 datasets and two textual entailment datasets, respectively. We can always reduce the accuracy of target models to be under 10%

with less than 20% word perturbations. In addition, we validate that the generated examples are correctly classified by human evaluators, and that they are similar to the original text and grammatically acceptable via a human study.

Although recently various mechanisms have been proposed towards generating adversarial texts, almost all of them rely on self-trained target models and self-selected data samples for evaluation, which makes it impossible to compare different methods under the same evaluation framework for bench-marking. Due to the lack of open-source code, re-implementations of previous works are also non-trivial and time-consuming. In this paper, we performed a rigorous evaluation of our generated adversaries in four aspects for automatic evaluation and three aspects for human evaluation to comprehensively demonstrate the effectiveness and efficiency of our system. More importantly, we open-source not only the code but our used pre-trained target models and test samples as well as the generated adversary results so that future works can be fairly compared under a unified evaluation metric[1].

Overall, our main contributions are summarized as follows:

- We propose a novel approach, TextFool, to quickly generate high-profile utility-preserving adversarial examples to force the target models to make wrong predictions under the black-box setting.
- We evaluate TextFool on a group of state-of-the-art deep learning models over five popularly used text classification datasets and two textual entailment datasets to demonstrate that it has achieved the state-of-the-art attack success rate and perturbation rate.
- We open-source the pre-trained target models and test samples for the convenience of future bench-marking.
- 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.

[1]Will be made public in the camera-ready version

2 Method

2.1 Problem Formulation

Given a pre-trained model $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{Y}$, which maps the input text space $\mathcal{X}$ to the set of labels $\mathcal{Y}$. A valid adversarial example x_{adv} is generated by altering the original data example $x \in \mathcal{X}$ and should conform to the following requirements: $\mathcal{F}(x_{adv}) \neq \mathcal{F}(x)$ and $S(x_{adv}, x) \geq \epsilon$, where $S : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{R}^+$ is a similarity function and $\epsilon \in \mathcal{R}^+$ is a threshold to preserve the utility of adversaries. In the natural language domain, S could be a semantic and syntactic similarity function.

2.2 Threat Model

Under the black-box setting, the attacker is not aware of the model architecture, parameters, or training data, and is only capable of querying the target model with supplied inputs and obtaining the output predictions and their confidence scores. The proposed framework for adversarial text generation is shown in Algorithm 1. Basically it is composed of the following two steps:

Step 1: Word Importance Ranking (line 1-6) We prioritize to manipulate those words that most significantly influence the final prediction results so as to minimize the alterations and thus maintain the semantic similarity as much as possible. Due to the black-box constraint, gradients of the model are not directly available, which are widely used to select important words for alterations in the white-box scenario. Instead we define the classification influence score termed C_{w_j} to measure the prediction scores change before and after changing a word w_j to the token "unknown", or so-called "out of vocabulary", which is formally defined as follows,

$$C_{w_j} = \begin{cases} \mathcal{F}_y(s) - \mathcal{F}_y(s'), \\ \quad \text{if } \mathcal{F}(s) = \mathcal{F}(s') = y \\ \mathcal{F}_y(s) - \mathcal{F}_y(s') + \mathcal{F}_{y'}(s') - \mathcal{F}_{y'}(s), \\ \quad \text{if } \mathcal{F}(s) = y, \mathcal{F}(s') = y' \end{cases} \tag{1}$$

where $s = (w_1, ..., w_j, ..., w_n)$, $s' = (w_1, ..., w', ..., w_n)$, and w' represents the unknown token.

After ranking the words by their importance, we filter out stop words derived from NLTK[2] and spaCy[3] libraries such as "the", "in", and "none"

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

[3]https://spacy.io/

Algorithm 1 Adversarial Attacking

Input: text example $x = (w_1, w_2, ..., w_n)$ and its ground truth y, target model $\mathcal{F}$, similarity function S, threshold ϵ, word embeddings E
Output: adversarial example x_{adv}

1: Initialize: $x_{adv} \leftarrow x$
2: **for** w_i in x **do**
3: Compute C_{w_i} via Eq.1
4: **end for**
5: $W_{ordered} = Sort(x, key = C_w)$
6: $W_{ordered} = StopWordsFilter(W_{ordered})$
7: **for** w_j in $W_{ordered}$ **do**
8: $candidates = SelectSynonyms(w_j, E)$
9: $candidates = POSFilter(candidates)$
10: **for** $w_k^{'}$ in $candidates$ **do**
11: $x' \leftarrow$ replace w_j with $w_k^{'}$ in x_{adv}
12: $Sim_k = S(x', x_{adv})$
13: **if** $Sim_k > \epsilon$ **then**
14: $FinCandidates.append(w_k^{'})$
15: $Y_k = \mathcal{F}(x')$
16: $P_k = \mathcal{F}_y(x')$
17: **end if**
18: **end for**
19: **if** any Y_k in Y has $Y_k \neq y$ **then**
20: $FinCandidates \leftarrow FinCandidates \cap Y \neq y$
21: $w^* \leftarrow argmax(FinCandidates, key = Sim)$
22: $x_{adv} \leftarrow$ replace w_j with w^* in x_{adv}
23: **return** x_{adv}
24: **else**
25: $p \leftarrow min(FinCandidates, key = P)$
26: **if** $p < \mathcal{F}_y(x_{adv})$ **then**
27: $w^* \leftarrow argmin(FinCandidates, key = P)$
28: $x_{adv} \leftarrow$ replace w_j with w^* in x_{adv}
29: **end if**
30: **end if**
31: **end for**
32: **return** None

to make sure that they will not be replaced. This simple step of filtering is quite important to avoid grammar destruction.

Step 2: Token Transformer (line 7-32) In this step, for a given word in text, we select a suitable replacement word that has similar semantic meaning, fits within the surrounding context, and can force the target model to make wrong predictions. In order to select the best replacement word for the selected word w, we propose the following steps:

- We extract the N nearest synonyms of the selected word by computing the cosine similarity between words using a set of word embedding vectors specially curated for synonym extraction (Mrkšić et al., 2016). By applying counter-fitting to the Paragram-SL999 word vectors provided by Wieting et al. (2015), this embedding vectors achieved state-of-the-art performance on SimLex-999, a dataset designed to measure how well different models judge semantic similarity between words (Hill et al., 2015). We identify words with cosine similarity scores with respect to w greater than δ and obtain the N largest ones. This corresponds to line 8 in Algorithm 1.

- Among the N candidates, we only select those that have the same part-of-speech (POS) as w to assure that the grammar of the text is not destroyed (line 9 in Algorithm 1).

- For the remaining candidates, each word is inserted in place of w and the corresponding prediction scores of target model and semantic similarity between the source and adversarial examples are computed. Those words whose similarity scores are above a preset threshold ϵ are filtered into the final candidates pool (lines 10-18 in Algorithm 1).

- In the final candidate pool, if there exist any candidates that can already alter the prediction of the target model, then we select the one with the highest semantic similarity score among these winning candidates as the best replacement word and output the adversary example. But if not, then we select the word with the least confidence score of label y as the best replacement word and repeat step 2 to transform the next selected word (line 19-31 in Algorithm 1).

We first execute Step 1 to obtain the words in the text ranked by their importance to the final prediction to form the candidates pool. Then Step 2 repeatedly prioritizes to transform each word in this candidate pool until the prediction of the target model is altered.

3 Experiments

3.1 Tasks

We study the effectiveness of our adversarial examples on two important NLP tasks, text classification, and textual entailment, whose dataset statistics are summarized in Table 1. For large test sets with more than 1,000 instances, we evaluate our algorithm on a set of 1,000 examples randomly selected from the test set.

3.1.1 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

Task	Dataset	Train	Test	Avg Len
Classification	AG's News	30K	1.9K	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.

analysis, with average text length ranging from tens to hundreds of words.

- **AG's News**: Sentence-level classification with regard to four news topics: World, Sports, Business, and Science/Technology. Following the practice of Zhang et al. (2015), we concatenate the title and description fields for each news.
- **Fake News Detection**: Document-level classification on whether a news article is fake or not. The dataset comes from the Kaggle Fake News Challenge.[4]
- **MR**: Sentence-level sentiment classification on positive and negative movie reviews (Pang and Lee, 2005). The dataset contains 5,331 positive and 5,331 negative reviews. 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**: Document-level sentiment classification on positive and negative reviews (Zhang et al., 2015). Reviews with a rating of 1 and 2 are labeled negative and 3 and 4 positive.

3.1.2 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.
- **MultiNLI**: A multi-genre entailment classification dataset with a coverage of transcribed speech, popular fiction, and government reports (Williams et al., 2017). Compared to SNLI, the MultiNLI dataset of 433K sentence pairs contains a larger variety of written and spoken English, thus capturing more linguistic complexity.

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

3.2 Attacking Target Models

	WordCNN	WordLSTM	BERT
AG's News	92.5	93.1	94.6
Fake News	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: Accuracy of target models on the standard test sets.

For each dataset, we train several state-of-the-art models on the original training set. Each model achieves an accuracy score on the original test set close to what is reported in the literature, which is shown in Table 2. We then generate adversarial examples which are semantically similar to the test set to attack the trained models and make them generate opposite results.

On the sentence classification task, we target 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 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 et al., 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[5].

We also implemented three target models on the textual entailment task: standard InferSent[6] (Conneau et al., 2017), ESIM[7] (Chen et al., 2016), and fine-tuned BERT.

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

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

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

	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	95.6	94.2	97.8
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	6.8	12.5	19.3
Perturbed Word (%)	14.3	3.5	8.3	15.2	11.0	14.9	5.1	10.6	18.6	10.1	16.7	6.1	12.8	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	743	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.

	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
Attack Accuracy (%)	3.5	6.7/6.9	5.1	7.7/7.3	4.0	9.6/8.3
Perturbed Word (%)	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.

3.3 Automatic Evaluation

We first report the accuracy of the original target models on the selected test samples as the original accuracy. Then we test the target models against the adversarial samples crafted from the test samples, denoted as the attack accuracy. By comparing these two accuracy values, we can evaluate how successful the attack is. We then report the perturbed word percentage as the ratio of the number of perturbed words to the text length. As a counterpart, we used the Universal Sentence Encoder (USE) (Cer et al., 2018) to encode sentences into high dimensional vectors so that we can use cosine similarity to measure the semantic similarity between the original and adversarial texts. These two metrics together evaluate how semantically similar the original and adversarial texts are. We finally report the number of queries to count how many times the attack system sends input samples to the target model and fetches the output probability scores. This metric can reveal the efficiency of the attack model.

3.4 Human Evaluation

To assess the quality of our results, we asked human judges to rate the adversarial examples on three aspects: semantic similarity, grammaticality, and classification accuracy. We randomly selected 100 samples of each task to generate adversarial attacks, one targeting the WordLSTM model on 100 MR examples and another targeting BERT on 100 SNLI examples.

We first shuffled all model-generated adversarial examples and asked human judges to give a grammaticality score of the generated sentence on a Likert scale of $1-5$, similar to the practice of Gagnon-Marchand et al. (2018). Next, we evaluated the semantic similarity of the original and adversarial sentences by asking humans to judge whether the generated adversarial sentence is similar (100%), ambiguous (50%), or dissimilar (0%) to the source sentence. Lastly, we evaluate the classification consistency by shuffling both the original and adversarial sentences together, ask humans to rate all of them and then calculate the consistency rate F1 of both classification results. Each task is completed by two human judges.

4 Results and Discussion

4.1 Automatic Evaluation

The main results of black-box attacks on the text classification and textual entailment tasks are summarized in Table 3 and Table 4, respectively. Overall, as can be seen from our results, we are able to achieve a high success rate when attacking with a limited number of modifications on both tasks. No matter how long the text is, and no matter how accurate the target model is, TextFool can always reduce the accuracy from the state-of-art values to under 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 of one sample 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 (accuracy of 0%) with only 3.5% word change rate. As the IMDB dataset 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.

Comparing the semantic similarity scores in both Tables 3 and 4 against the perturbed word ratios, we find that they 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 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 can be a good automatic measurement to evaluate the degree of alterations of the original texts.

We include the average text length of each dataset in the last row of Tables 3 and 4 so that it can be conveniently compared against the query number. The query model is linear to the text length and overall the ratio of query number to the average text length is between 2 and 8. And the longer the text is, the smaller this ratio, which validates the efficiency of TextFool.

Although there are no open-source pre-trained target models for directly benchmarking our model against the published systems, we can still perform an indirect comparison under the same target model architecture and dataset, which is summarized in Table 5. From this table, we can clearly see that our system beats the state-of-the-art models in terms of both the attack success rate and perturbed word ratio.

Dataset	Model	Succ. Rate	Perturbed
IMDB	BUGGER	86.7	6.9
	NAE	97.0	14.7
	Ours	**99.7**	**5.1**
SNLI	NAE	70.0	23.0
	Ours	**95.8**	**18.0**
Yelp	AE	74.8	-
	Ours	**97.8**	**10.6**

Table 5: Comparison of our attack system against published systems in terms of attack success rate (%) and perturbed word ratio (%). NAE is from Alzantot et al. (2018), BUGGER is from Li et al. (2018), and AE is from Kuleshov et al. (2018). The target model for IMDB and Yelp is LSTM and SNLI is InferSent.

4.2 Human Evaluation

We sampled 100 sentences from the adversarial attack on the MR dataset with the WordLSTM model and 100 examples from the adversarial attack on SNLI with the BERT model. We verified the quality of our examples via three experiments. First, we ask human evaluators to give a grammaticality score of a shuffled mix of original and adversarial examples. Grammaticality is an essential criterion for adversarial examples because it does not make sense to generate gibberish English to confuse the model. As shown in Table 7, although the adversarial sentences are rated lower than the original sentences on both datasets, they tend to have overall acceptable grammaticality of $3.31/5.0$ on MR and $3.91/5.0$ on SNLI. By sensibly substituting synonyms, our model generates smooth outputs such as "the big metaphorical wave" in Table 6.

We then asked the human raters to assign labels to both original and adversarial samples. We showed a set of sentences with positive/negative labels for the sentiment classification on MR and sentence pairs with entailment/neutral/contradiction relationships for SNLI samples. Due to the nature of tasks, it is easy for humans to agree on the ground truth labels on sentiment analysis but harder on natural inference. Nonetheless, the overall agreement between the labels of the original sentence and the adversarial sentence is relatively high, with 85% on MR and 72% on SNLI (in Table 8). 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 demonstrates typical examples of sentences with almost the same meanings that result in contradictory classifications by

Movie Review (Positive ↔ Negative)	
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, Neutral, Contradiction)	
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.
Attack [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.
Attack [Label: ENT]	The youngster is at the shore.

Table 6: Grammaticality of original and adversarial examples for MR (WordLSTM) and SNLI (BERT) model on a $1-5$ scale.

the attacked target model.

Input	MR (WordLSTM)	SNLI (BERT)
Original Grammar	4.22	4.50
Attack Grammar	3.31	3.91

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

Input	MR (WordLSTM)	SNLI (BERT)
Original Accu. (%)	88	68
Attack Accu. (%)	82	51
Agreement (%)	83	72

Table 8: Human classification accuracy on adversarial examples for MR (WordLSTM) and SNLI (BERT) models.

Lastly, we asked human judges to decide 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 0.67 and 0.59 on MR and SNLI respectively. The higher semantic similarity of adversarial and original examples from MR correlates with the lower percentage of perturbed words in MR.

4.3 Ablation Study

Word Importance Ranking To validate the effectiveness of step 1 in Algorithm 1, i.e., the word importance ranking part, we remove this step and instead randomly select the words in 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 9. After removing step 1, compared with the normal attack accuracy, the attack accuracy by randomly selecting the words to perturb increases by a lot, which reveals that the attack becomes ineffective. This clearly demonstrates that the word importance ranking algorithm is important. It can accurately and efficiently locate the most influential words that can significantly change the predictions of the target model. For a given attack success rate goal, such a strategy can reduce the number of perturbed words so as to maintain the semantic similarity as much as possible.

	MR	AG	SNLI
Perturbed Word (%)	16.7	22.0	18.5
Original Accu. (%)	86.0	94.2	89.4
Normal Attack Accu. (%)	11.5	12.5	4.0
Random Attack Accu. (%)	**68.3**	**80.8**	**59.2**

Table 9: Comparison of attack accuracy before and after removing the word importance ranking part of Algorithm 1. For control, step 2 and the perturbed words ratio are kept the same.

Semantic Similarity Constraint In step 2 of Algorithm 1, we check the semantic similarity between the original text and that whose selected word is replaced by a synonym, and view this synonym as legitimate only when the similarity is above a preset threshold ϵ. We found that this strategy can effectively filter out those synonyms that are not relevant to the selected word. Some typical examples are summarized in Table 10. As we can see, the synonyms extracted by word embeddings are indeed quite noisy and directly in-

jecting them into the text as adversarial samples would probably shift the semantic meaning significantly. By utilizing the semantic similarity constraint, we can obtain more related synonyms as good replacements.

Original	like a south of the border melrose place
w/ Sim.	like a south of the border melrose spot
w/o Sim.	like a south of the border melrose mise
Original	their computer animated faces are very expressive
w/ Sim.	their computer animated face are very affective
w/o Sim.	their computer animated faces are very diction

Table 10: Qualitative comparison before and after applying the semantic similarity constraint. "w/ Sim." and "w/o Sim." mean that we apply or not the semantic similarity constraint. Blue color highlights the original words, green color indicates that the replacement words proposed by the normal attacking system are compatible, and red color shows that without semantic similarity constraint, the selected synonyms are not relevant.

4.4 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 fairly similar word to "motorcycle". Some even more subtle grammatical error can be seen on adverbs, such as the pair "A boy is sitting still", vesus "A boy is sitting anymore" As future work, some carefully designed heuristics can be applied to filter out grammatical errors.

Content shift can be seen in a task-specific situation. For example, in the sentiment classification task, a change of words might not affect the overall sentiment, whereas in the task of textual entailment, the substitution of words might result in a fundamental difference. For example, if the premise is "a *kid* with red hat is running", and the original candidate is "a *kid* is running (ENTAILMENT)", then if the adversarial example becomes "a *girl* is running", the sensible result turns into NEUTRAL instead.

5 Related Work

Adversarial attack has been extensively studied in computer vision (Goodfellow et al., 2014; Kurakin et al., 2016a; Madry et al., 2017; Moosavi-Dezfooli et al., 2017; Rosca et al., 2017). Most work is done in the context of continuous input spaces (Szegedy et al., 2013; Goodfellow et al., 2014), where adversarial attacks are achieved by gradient-based perturbation to the original input.

Adversarial attack on discrete data such as text is more challenging. Inspired by the approaches in computer vision, previous work in language adversarial attack has focused on variations of gradient-based methods. For example, Zhao et al. (2017) transforms input data into a latent representation using generative adversarial networks (GANs), and then retrieves adversaries close to the original instance in the latent space.

Other work observes the intractability of GAN-based models on text and the shift in semantics in the latent representations, so heuristic methods such as scrambling, misspelling, or removing words are proposed (Ebrahimi et al., 2017; Li et al., 2016, 2018). For example, (Ebrahimi et al., 2017) adopt a heuristic to substitute single words adversarially. However, such a heuristic relies on white-box access to the model and cannot be put into large-scale generation for data augmentation.

6 Conclusions

Overall, we study adversarial attacks against state-of-the-art text classification and textual entailment models under the black-box setting. Extensive experimental results demonstrate that our proposed system, TextFool, is effective and efficient for generating targeted adversarial texts. And our human study validated that the generated adversarial inputs are legible, grammatical, and similar in meaning to the original input.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*.

Nicholas Carlini and David Wagner. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. *arXiv preprint arXiv:1801.01944*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

Jules Gagnon-Marchand, Hamed Sadeghi, Md Haidar, Mehdi Rezagholizadeh, et al. 2018. Salsa-text: self attentive latent space based adversarial text generation. *arXiv preprint arXiv:1809.11155*.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. *arXiv preprint arXiv:1801.04354*.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Volodymyr Kuleshov, Shantanu Thakoor, Tingfung Lau, and Stefano Ermon. 2018. Adversarial examples for natural language classification problems.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016a. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2016b. Adversarial machine learning at scale. *CoRR*, abs/1611.01236.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2017. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1765–1773.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*.

Bo Pang and Lillian Lee. 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*, pages 115–124. Association for Computational Linguistics.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM.

Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. 2016. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM 2016-2016 IEEE Military Communications Conference*, pages 49–54. IEEE.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. 2017. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2017. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*.