# Adversarial Examples for NLP

**Anyi Rao**
Nanjing University, University of Oregon
anyirao@smail.nju.edu.cn

Dec. 10th 2017

! My advisor told me not to share the whole paper at this time.
Some parts are allowed to show here. Please contact me for a full version and source codes.

## Abstract

NLP systems are increasingly being deployed with relatively little attention to their vulnerabilities. We propose an efficient method to generate adversarial examples that trick character-level and word-level neural NLP models. Our method relies on an atomic *flip* operation, which swaps one token to another, based on the gradients of the one-hot input vectors. In experiments on text classification and machine translation, we find that only a few character or word manipulations are needed to greatly increase the error rates. We analyze the properties of these examples, and show that employing these human-discernible adversarial examples in training can improve test-time accuracy, as well as making the models more robust to adversarial examples.

## 1 Introduction

Adversarial examples are inputs to a predictive machine learning model that are maliciously designed to cause poor performance (Goodfellow et al., 2015). Research on adversarial examples for neural nets has largely focused on image data, with relatively little work on textual data. Although NLP systems are being deployed with increasing frequency, there has not been an increase in scrutiny of their vulnerabilities.

We investigate how an attacker can drastically change the output of a system with just a few seemingly innocuous typos in character-level models, and a few similar word replacements in word-level models. Unlike images, which are usually scaled to have continuous pixel intensities, e.g., [0,1], text data is discrete, which makes optimization for finding adversarial examples more challenging. Our work is the first to craft character-level adversarial examples for text. We propose an efficient method to generate character-level adversarial examples for NLP, which could become a convenient tool for evaluating the robustness of systems. Furthermore, efficient generation of adversarial examples allows feasible adversarial regularization in order to improve test time accuracy. We find that only a few character manipulations are needed to greatly increase the error rates for NLP systems. For instance, in our experiments on news classification, 30% of adversarial examples have either one or two changes, and 60% of adversarial examples for translation contain only one change. The major advantage of working with characters, as demonstrated in Table 1, is that while a few character changes can trick a machine learning system, the meaning of the text is very likely to be preserved. By contrast, a few word-level adversarial manipulations could simply change the meaning of text, which would require semantics-preserving techniques

| |
|---|
| South Africa's historic Soweto township marks its 100th birthday on Tuesday in a mood of optimisim.<br>57% **World** |
| South Africa's historic Soweto township marks its 100th birthday on Tuesday in a mooP of optimisim.<br>95% **Sci/Tech** |
| Chancellor Gordon Brown has sought to quell speculation over who should run the Labour Party and turned the attack on the opposition Conservatives.<br>75% **World** |
| Chancellor Gordon Brown has sought to quell speculation over who should run the Labour Party and turned the attack on the oBposition Conservatives.<br>94% **Business** |

Table 1: Adversarial examples for text classification, which will be misclassified by a CharCNN-LSTM classifier after a single character change.

At the core of our method lies an atomic character *flip* operation which changes one character to another; this operation is extended to perform insertion and deletion operations to constitute a set of plausible adversarial attacks to trick an NLP system.

Our contributions are as follows:

1. We propose an efficient gradient-based optimization method to manipulate discrete text structure at its character representation.

2. We conduct a broad set of experiments across text classification and seq-to-seq generation, and analyze the vulnerabilities of each model. We also study the human-discernibility of these adversarial examples.

3. We investigate the robustness of a classifier trained with adversarial examples, by studying its resilience to attacks and its accuracy on the test data.

## 2 Related Work

Szegedy et al. (2014) found that a minor change to an image can trick an image classifier. That is, an adversary can trick these models to misclassify examples that are only slightly different from correctly classified examples. Finding these examples requires solving a constrained non-convex problem, which must be solved separately for every example in the data in order to detect blind spots of the model. Concretely, given the classifier $g$ and the input $x$ in the constrained set $C$, the goal is to find the optimal perturbation $r$, such that the classifier makes a different prediction than on the original input.

$$\begin{aligned} &\underset{r}{\text{minimize}}\ \|r\|_2 \\ &\text{subject to} \qquad g(x+r) \neq g(x) \\ &\qquad\qquad\qquad\quad x+r \in C \end{aligned}$$

Goodfellow et al. (2015) proposed the *fast sign gradient* method, which manipulates data using a clever perturbation scheme based on a gradient ascent direction. Li et al. (2017) recently observed that the sign gradient noise, when added to the word embeddings, can adversely impact a convolutional neural text classifier too.

There is a growing interest in understanding vulnerabilites of NLP systems as evidenced by a recent community shared task[1]. There is also a growing body of works that address the rising concerns over deception (Zubiaga et al., 2016). The first work of crafting adversarial examples for text was proposed in (Papernot et al., 2016), which worked at the word level. They added noise to the word embeddings and searched the neighborhood of that new embedding to replace the original word with. While their adversary was able to trick the classifier, the word-level changes tend to be extreme (e.g., “I wouldn’t rent this...” → “Excellent wouldn’t rent this...”). Jia and Liang (2017) add distracting sentences to the end of paragraphs to fool deep reading comprehension systems. They show how state-of-the-art text comprehension systems are matching patterns rather than understanding text. In a similar work, Li et al. (2017) devise operations to preserve semantics of text, such as re-ordering and replacing some of the words. But their goal is to make a classifier robust by training on corrupted examples, rather than studying the vulnerability to adversarial examples.

From a security point of view, an adversary can only manipulate the raw input and does not have access to the word embeddings used by an NLP system. An interesting application of deceiving an NLP system was found in tricking Google’s `perspective` API, which detects toxic or abusive comments. Hosseini et al. (2017) showed that simple modifications, such as adding spaces or dots between characters, can drastically change the toxicity score of the API.

Adversarial training/regularization interleaves training with generation of adversarial examples (Goodfellow et al., 2015). Concretely, after every iteration of training, adversarial examples are created and added to the mini-batches. Virtual adversarial training (Miyato et al., 2016) is another regularization method, which aims to minimize the KL divergence between predictions on the examples and their adversarial counterparts. Similar techniques have been used for text classification (Li et al., 2016; Miyato et al., 2017).

## 3 Adversarial Training

We now turn to the analysis of the robustness of adversarially-trained models. We use the adversarial regularization technique (Goodfellow et al., 2015) to train a classifier, which incorporates adversarial examples during training. Adversarial objective function is as follows:

$$\tilde{J}(\theta; \boldsymbol{x}, \boldsymbol{y}) = J(\theta; \boldsymbol{x}, \boldsymbol{y}) + J(\hat{\theta}; \hat{\boldsymbol{x}}, \boldsymbol{y}) \qquad (1)$$

where $\boldsymbol{x}$ is the input, $\boldsymbol{y}$ is the output and $\theta$ is the model weights. At each epoch, adversarial examples, $\hat{\boldsymbol{x}}$, against the current model, parameterized

[1] https://bibinlp.umiacs.umd.edu/sharedtask.html

|  |
|---|
| this is such a high energy movie where the drumming and the marching are so excellent , who cares if the story 's a little weak (**sluggish**) 92% **Positive** → 94% **Negative** |
| 'enigma' is a good (**terrific**) name for a movie this deliberately obtuse and unapproachable. 95% **Negative** → 56% **Positive** |
| an intermittently pleasing (**satisfying**) but mostly routine effort. 99% **Negative** → 63% **Positive** |
| an atonal estrogen opera that demonizes feminism while gifting the most sympathetic male of the piece with a nice (**wonderful**) vomit bath at his wedding. 88% **Negative** → 53% **Positive** |
| a strong (**robust**) first quarter , slightly less so second quarter , and average second half 76% **Negative** → 62% **Positive** |

Table 2: Adversarial Examples for Sentiment Classification. The bold words replace the words before them.

by $\hat{\theta}$, are found; these examples are added to mini-batches for learning $\theta$ in the next epoch.

## 4 Experiments

### 4.1 Char-Level

Omitted.[2]

### 4.2 Word-Level

The proposed model can be used to create adversarial examples by manipulating words of the text too. However, changing words can have an undesirable effect on the quality of adversarial examples. For example, changing the word good to bad changes the sentiment of the sentence this was a good movie. In fact, we expect the model to predict a different label after such change. This can be understood by noticing the fact that the derivative of the one-hot vector of the $i_{th}$ word is the following:

$$M\frac{\partial J(x,y)}{e_{w_i}}$$

where $M$ is the embedding matrix, and $e_{w_i}$ is the embedding of the $i_{th}$ word. The word with the highest dot product with $\frac{\partial J(x,y)}{e_{w_i}}$, might be a word which totally changes the meaning of the sentence. In order to address this issue, we only flip a word $w_i$ to $w_j$ only if these constraints are satisfied:

1 The cosine similarity between the two words is bigger than a threshold $\alpha$.

2 The two words have the same part-of-speech.

We also disallow replacing of stop-words, as for many of the stop-words, it is difficult to find cases where replacing them will still render the sentence grammatically correct. To find the pairwise similarity between words, we used the embeddings of `word2vec` which performed far superior to those given by `Glove`. We used **NLTK** for part-of speech tagging. We use Kim convolutional neural model (2014) as the attacked model. We make use of sentiment classification datasets **MR** and **SST**, subjectivity analysis **Subj**, opinion polarity detection **MPQA**, and question-type classification **TREC**. Table 2 shows a few adversarial examples with only one word flip. In the second and the fourth examples, the adversary tricks the classifier by replacing a positive word (i.e., good, nice) with a highly positive words (i.e., terrific, wonderful) in an overall very negative review. Given the widespread use of sentiment classification for business purposes, these examples can lead to harmful consequences.

Table 3 shows the accuracy of our adversarial training approach over the baseline. We also compare with Li et al. (2016) which uses a regularization technique, and Li et al. (2017) which uses a data augmentation technique, both of which use Kim as their baseline. The similarity threshold $\alpha$ was tuned on validation sets of the datasets. Adversarial training improves the accuracy on all datasets, with the most significant improvement on **SST**, and the least on **MPQA**. The reason for small **MPQA** gain might be its very short sentences (average 3), which makes the adversary produce fewer useful adversarial examples. This leaves adversarial training not very different from regular training. We also tested our adversarially-trained model and the baseline model on adversarial examples, as shown in Table 4. Adversarial training achieves much better results on adversarial examples on all datasets, with the best improvement (10.57 %) on **MPQA**.

[2] For more, please contact me.

| Model | MR | SST | SUBJ | TREC | MPQA |
|---|---|---|---|---|---|
| Adversarial Training | **81.49** | **87.10** | 93.24 | **93.8** | **89.44** |
| CNN (Yoon Kim, 2014) | 81.26 | 86.05 | 93.06 | 93.2 | 89.37 |
| Noise Adversity (Li et al., 2016) | 80.8 | 84.5 | 93.1 | - | - |
| Linguistic Adversity (Li et al., 2017) | 81.4 | 84.8 | **93.6** | - | - |

Table 3: Accuracy on text classification benchmarks. **MR**: Movie reviews with one sentence per review. Classification involves detecting positive/negative reviews [Pang and Lee, 2005] **SST**: Stanford Sentiment Treebank—an extension of MR but with train/dev/test splits provided binary labels [Socher et al., 2013]. **Subj**: Subjectivity dataset where the task is to classify a sentence as being subjective or objective [Pang and Lee, 2004]. **TREC**: TREC question dataset-task involves classifying a question into 6 question types (whether the question is about person, location, numeric information,etc.) [Li and Roth, 2002]. **MPQA**: Opinion polarity detection subtask of the MPQA dataset [Wiebe et al., 2005]. **Noise Adversity** Learning Robust Representations of Text (Li et al., 2016). **Linguistic Adversity** Robust Training under Linguistic Adversity (Li et al., 2017).

| Model | MR | SST | SUBJ | TREC | MPQA |
|---|---|---|---|---|---|
| Adversarial Training | 34.72 | 53.16 | 73.03 | 66.6 | 45.55 |
| CNN (Yoon Kim, 2014) | 30.37 | 49.04 | 68.28 | 65.2 | 34.82 |

Table 4: Accuracy on adversarial examples.

## References

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of ICLR*.

Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138* .

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328* .

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

Yitong Li, Trevor Cohn, and Timothy Baldwin. 2016. Learning robust representations of text. In *Proceedings of EMNLP*.

Yitong Li, Trevor Cohn, and Timothy Baldwin. 2017. Robust training under linguistic adversity. *EACL 2017* page 21.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *Proceedings of ICLR*.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2016. Distributional smoothing with virtual adversarial training. In *Proceedings of ICLR*.

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

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of ICLR*.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one* 11(3):e0150989.