


# Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers

By: Ji Gao, Jack Lanchantin, Mary Lou Soffa, Yanjun Qi

Presented by: Jennifer Fang [Week 02]

Department of Computer Science: University of Virginia

@ <https://qdata.github.io/deep2Read/>



# Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers

**Goal:** Create a new algorithm for black box testing to generate small text perturbations to cause deep-learning classifiers to misclassify a text input.

The new algorithm created is called DeepWordBug.

# Black Box vs. White Box Testing

- **Black box testing:** testing as if you are a hacker i.e. no knowledge of the inside workings, don't know details of learned models or feature representations of inputs
  - Can only manipulate input samples by testing and observing a classification model's outputs
  - Usually it's easy to query a model
  - But there's no access to the inner structure of the models, which makes black box more applicable than white box
- **White box testing:** testing with full knowledge of the application
- Both black and white box testing cannot modify the model

## Key Terms

- **Hyperparameter:** a parameter whose value is set before the experiment
  - Instead of deriving its value through training, this parameter has a set value
- **Adversarial samples:** inputs intentionally designed to cause the model to make a mistake
- **Transferability:** an important property where samples that are generated for one model can also be used to fool another DNN model

# Goal of DeepWordBug

Proven that: Adding small modifications to text inputs can fool deep learning classifiers

Question to answer: Are deep learning classifiers robust?

Results have implications in text-based spam detection.

Two types of modifications to text input

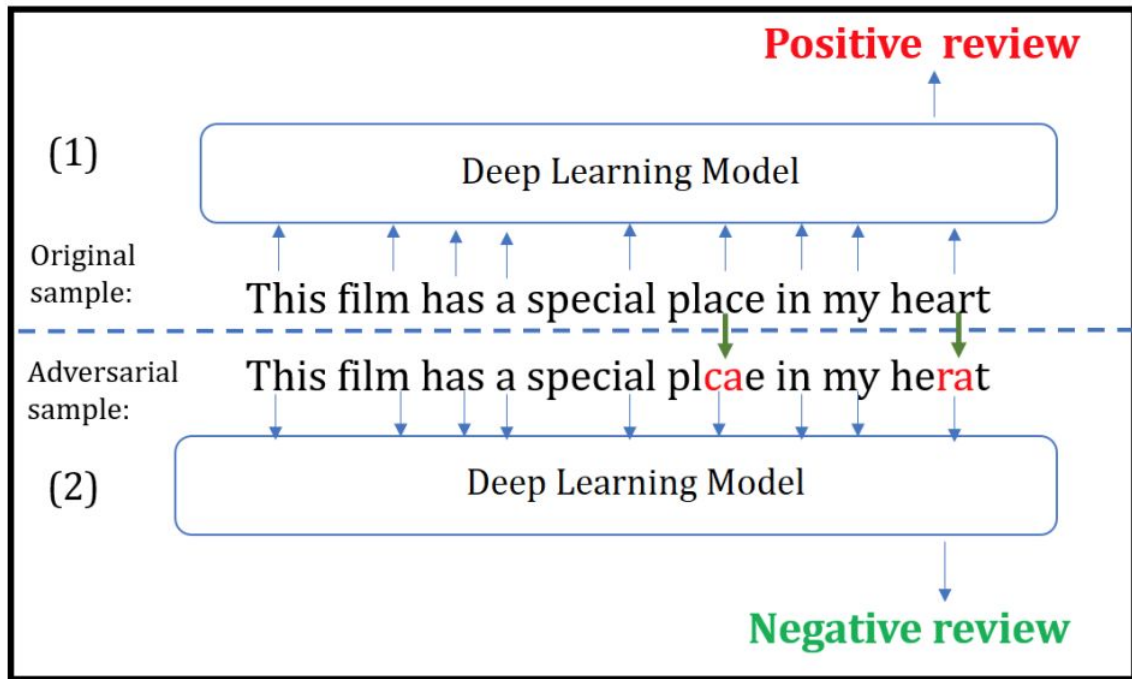
$$\mathbf{x}' = \mathbf{x} + \Delta\mathbf{x}, \|\Delta\mathbf{x}\|_p < \epsilon, \mathbf{x}' \in \mathbb{X}$$

$$F(\mathbf{x}) \neq F(\mathbf{x}') \text{ or } F(\mathbf{x}') = t$$

Targeted

Untargeted

# DeepWordBug Example



## Differences of text vs. pictures

1. Text input  $x$  is symbolic. Perturbation on  $x$  is hard to define.
2. No metric has been defined to measure text difference.  $L_p$  - norms makes sense on continuous pixel values, but they don't make sense on texts since they are discrete.



## Basis of DeepWordBug

1. Determine the important tokens to change.
  - Use scoring functions to evaluate
2. Change those tokens
  - Create “imperceivable” changes which can evade a target deep learning classifier



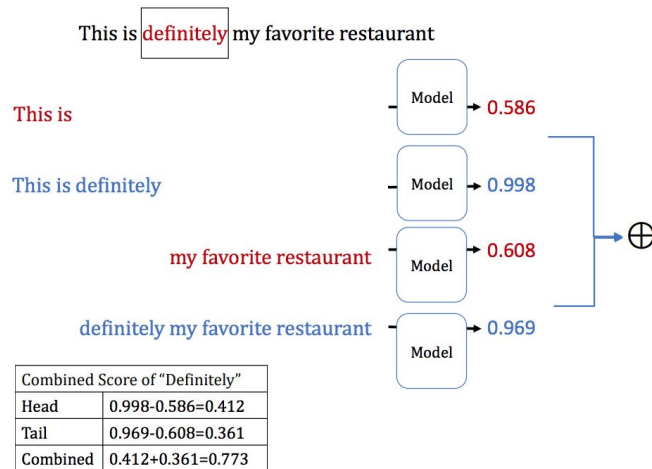
# Scoring Functions

## 1. Replace-1 Score

- Replace one  $x_i$  with  $x_i'$
- $R1S(x_i) = F(x_1, x_2, \dots, x_{i-1}, x_i, \dots, x_n) - F(x_1, x_2, \dots, x_{i-1}, x_i', \dots, x_n)$

## 2. Temporal Head Score

- Difference between the model's prediction score as it reads up to the  $i^{\text{th}}$  token and as it reads up to the  $i-1^{\text{th}}$  token
- $THS(x_i) = F(x_1, x_2, \dots, x_{i-1}, x_i) - F(x_1, x_2, \dots, x_{i-1})$



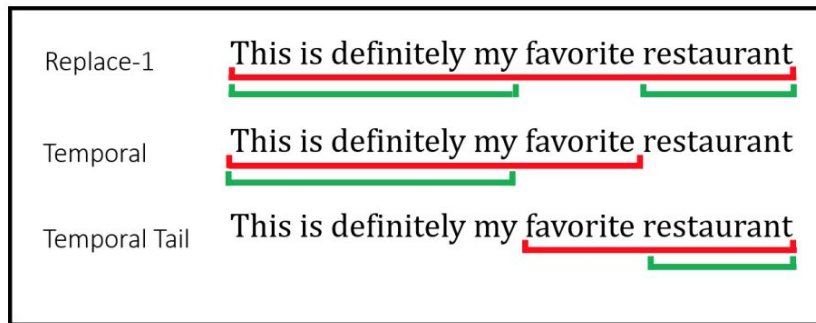
# Scoring Functions

## 3. Temporal Tail Score

- The complement of the THS
- Compares the difference between two trailing parts of a sentence, the one containing a certain token versus the one that does not.
- $TTS(x_i) = F(x_i, x_{i+1}, x_{i+2}, \dots, x_n) - F(x_{i+1}, x_{i+2}, \dots, x_n)$

## 4. Combination Score

- THS and TTS model from opposing sides, so the Combination Score combines the two
- $CS(x_i) = THS(x_i) + \lambda(TTS(x_i))$
- $\lambda$  is a hyperparameter



# Text Transformations

1. **Swap:** Swap two adjacent letters in the word.
2. **Substitution:** Substitute a letter in the word with a random letter.
3. **Deletion:** Delete a random letter from the word.
4. **Insertion:** Insert a random letter in the word.

Original		Swap	Substitution	Deletion	Insertion
Team	→	Taem	Texm	Tem	Tezam
Artist	→	Artsit	Arxist	Artst	Articst
Computer	→	Comptuer	Computnr	Compter	Comnputer

**Table 1: Different transformer functions and their results.**

# DeepWordBug Algorithm

---

## Algorithm 1 DeepWordBug Algorithm

---

**Input:** Input sequence  $\mathbf{x} = x_1x_2 \dots x_n$ , RNN classifier  $F(\cdot)$ , Scoring Function  $S(\cdot)$ , Transforming function  $T(\cdot)$ , maximum allowed perturbation on edit distance  $\epsilon$ .

```
1: for  $i = 1..n$  do
2:    $scores[i] = S(x_i; \mathbf{x})$ 
3: end for
4: Sort  $scores$  into an ordered index list:  $L_1 .. L_n$  by descending
   score
5:  $\mathbf{x}' = \mathbf{x}$ 
6:  $cost = 0, j = 1$ 
7: while  $cost < \epsilon$  do
8:    $cost = cost + Transform(x'_{L_j})$ 
9:    $j++$ 
10: end while
11: Return  $\mathbf{x}'$ 
```

---

Apply Scoring Function

|  
|  
|  
|  
|  
|  
|  
|  
|  
|  
|  
|

Transform Text

|  
|

Return  $\mathbf{x}'$

# Experiment Setup

1. **Datasets:** 7 large scale datasets, including Enron Spam Dataset
2. **Target models:** 2 well trained models
  - **Word-LSTM:** a Bi-directional LSTM, which contains an LSTM in both directions (reading from first word to last and from last word to first) [used 4 different transformers]
  - **Char-CNN:** uses one-hot encoded characters as inputs to a 9-layer convolutional network [only used substitution transformer]

## Comparison methods

1. **Random** (baseline): randomly selects tokens as targets
2. **Gradient** (baseline): uses full knowledge of the model to find most important tokens
3. **DeepWordBug**: use previously described white-box scoring functions to find most important tokens: Replace 1 Scoring, Temporal Head Score, Temporal Tail Score, Combined Score



## Additional Parameter

$\epsilon$  = maximum allowed perturbation; maximum allowed edit distance (in characters)

### Word-LSTM Model

	Baselines						WordBug						
	Original	Random		Gradient		Replace-1		Temporal Head		Temporal Tail		Combined	
	Acc(%)	Acc(%)	Decrease	Acc(%)	Decrease	Acc(%)	Decrease	Acc(%)	Decrease	Acc(%)	Decrease	Acc(%)	Decrease
AG's News	90.5	89.3	1.33%	48.5	10.13%	36.1	60.08%	42.5	53.01%	21.3	76.48%	24.8	72.62%
Amazon Review Full	62.0	61.1	1.48%	55.7	10.13%	18.6	70.05%	27.1	56.30%	17.0	72.50%	16.3	73.76%
Amazon Review Polarity	95.5	93.9	1.59%	86.9	8.93%	40.7	57.36%	58.5	38.74%	42.6	55.37%	36.2	62.08%
DBPedia	98.7	95.2	3.54%	74.4	24.61%	28.8	70.82%	56.4	42.87%	28.5	71.08%	25.3	74.32%
Yahoo! Answers	73.4	65.7	10.54%	50.0	31.83%	27.9	61.93%	34.9	52.45%	26.5	63.86%	23.5	68.02%
Yelp Review Full	64.7	60.9	5.86%	53.2	17.76%	23.4	63.83%	36.6	43.47%	20.8	67.85%	24.4	62.28%
Yelp Review Polarity	95.9	95.4	0.55%	88.4	7.85%	37.8	60.63%	70.2	26.77%	34.5	64.04%	46.2	51.87%
Enron Spam Email	96.4	67.8	29.69%	76.7	20.47%	39.1	59.48%	56.3	41.61%	25.8	73.22%	48.1	50.06%
Mean			6.82%		16.46%		<b>63.02%</b>		44.40%		<b>68.05%</b>		<b>64.38%</b>
Median			2.57%		13.95%		<b>61.28%</b>		43.17%		<b>69.46%</b>		<b>65.15%</b>
Standard Deviation			9.81%		8.71%		4.94%		9.52%		6.77%		9.56%

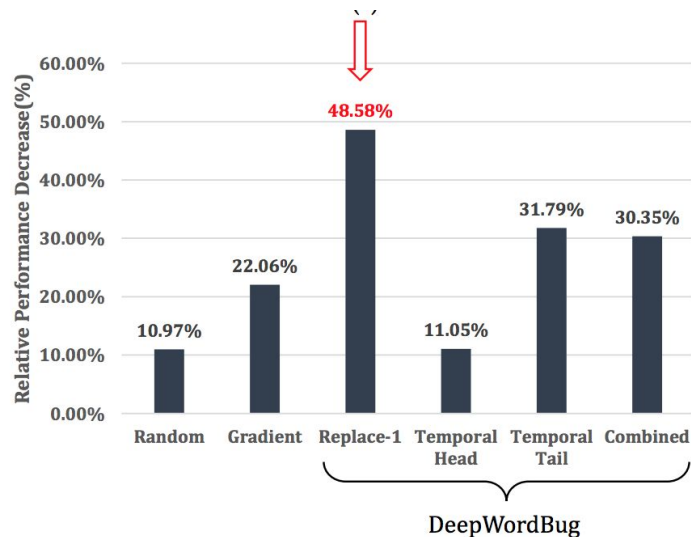
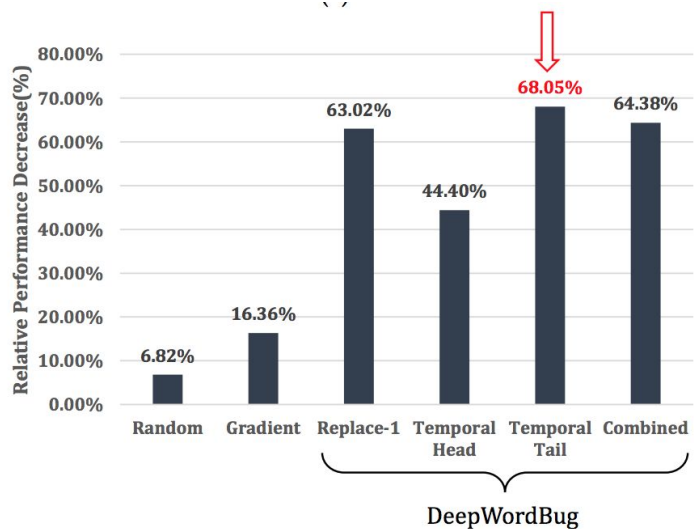
### Char-CNN Model

	Baselines						WordBug						
	Original	Random		Gradient		Replace-1		Temporal Head		Temporal Tail		Combined	
	Acc(%)	Acc(%)	% Decrease	Acc(%)	Decrease	Acc(%)	Decrease	Acc(%)	Decrease	Acc(%)	Decrease	Acc(%)	Decrease
AG's News	90.0	82.4	8.36%	62.3	30.74%	30.8	65.80%	74.1	17.66%	58.6	34.90%	60.4	32.88%
Amazon Review Full	61.1	51.0	16.53%	47.0	23.04%	25.6	58.17%	58.1	4.89%	32.5	46.79%	35.0	42.70%
Amazon Review Polarity	95.2	93.4	1.91%	84.3	11.41%	46.4	51.27%	91.6	3.79%	70.9	25.48%	73.5	22.83%
DBPedia	98.4	95.8	2.58%	92.9	5.60%	74.9	23.91%	95.7	2.73%	88.2	10.37%	88.8	9.69%
Yahoo! Answers	71.0	52.2	26.45%	43.5	38.76%	30.0	57.72%	56.8	20.05%	35.3	50.23%	36.6	48.50%
Yelp Review Full	63.5	52.6	17.05%	45.7	28.06%	27.6	56.56%	51.3	19.10%	35.3	44.36%	38.2	39.74%
Yelp Review Polarity	95.3	91.2	4.31%	84.8	11.03%	42.8	55.05%	86.5	9.16%	71.9	24.51%	71.1	25.33%
Enron Spam Email	95.6	85.5	10.56%	69.0	27.84%	76.4	20.13%	85.1	11.03%	78.7	17.68%	75.4	21.14%
Mean			10.97%		22.06%		<b>48.58%</b>		11.05%		<b>31.79%</b>		<b>30.35%</b>
Median			9.46%		25.44%		<b>55.80%</b>		10.10%		<b>30.19%</b>		<b>29.11%</b>
Standard Deviation			8.54%		11.53%		16.91%		7.10%		14.56%		12.93%

Table 5: Effectiveness of WordBug on 8 Datasets using the Word-LSTM and Char-CNN model. Acc is the accuracy of the method and Decrease is the percent decrease of the accuracy by using the specified attacking method over the original accuracy. Word-LSTM uses Substitution transformer. All results are under maximum edit distance difference 30 ( $\epsilon = 30$ ).



# Decrease in Performance



# Results

1. **Accuracy:** reduced 68% performance of the Word-LSTM model and 48% performance of the Char-CNN model
2. **Influence of the Scoring Function:** very important
  - a. DeepWordBug's scoring is better than the gradient
  - b. Without scoring (random case), adversarial performance is low
3. **Transferable?** Yes, even for models with different word embedding
4. **Influence of Transformation Function:** wasn't much difference within the functions; having a good scoring function is more important
5. **Influence of Dictionary size:** low; works for all dictionary sizes
6. **Probability of classifications:** 94.6% of classifications were classified with  $> 0.9$  confidence for Word-LSTM model on the Enron Spam Dataset (# classes = 2),  $\epsilon = 30$

# Transferability and Confidence

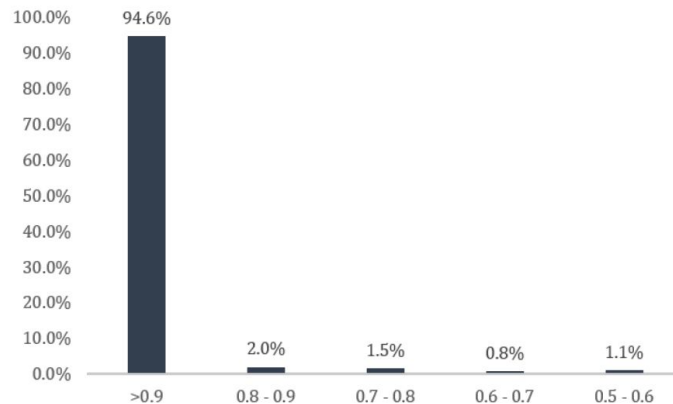
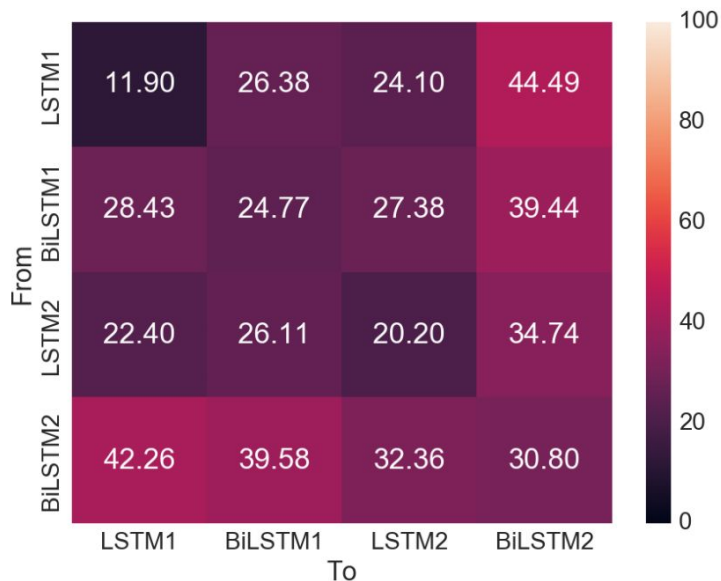


Figure 12: How strong the machine learning model will believe the wrong answer lead by the adversarial sample, the x-axis are the confidence range and the y-axis are the probability distribution. The result is generated using Word-LSTM model on the Enron Spam Dataset (Number of classes = 2), with edit distance maximum  $\epsilon = 30$



# Applications

**Adversarial training:** with training on DeepWordBug, adversarial accuracy improves from 12% to 62%

**Autocorrection:** reduces the performance of adversarial samples; can combat this with stronger transformation functions such as substitution-2 and deletion-2



## Why DeepWordBug Works

- When changes are made to a word, the word becomes *unknown*, which map to the unknown embedding vector
- Small changes can thus make a big impact
- Adversarial samples are probably decipherable to humans, but not to models
- Area for ML to catch up with humans



## Advantages:

1. **Black-box:** DeepWordBug generates adversarial samples in a pure black-box manner.
2. **Performance:** DeepWordBug results in a 68% decrease on average from the original classification accuracy for a word-level LSTM model and 48% decrease on average for a character-level CNN model.
  - Results are transferable and are not reliant on dictionary size or transformation technique used
3. **Applications:** adversarial training is successful; by using DeepWordBug generated samples, model accuracy on generated adversarial samples increases from 12% to 62%