

TextBugger: Generating Adversarial Text Against Real-world Applications

Jinfeng Li Shouling Ji Tianyu Du Bo Li Ting Wang

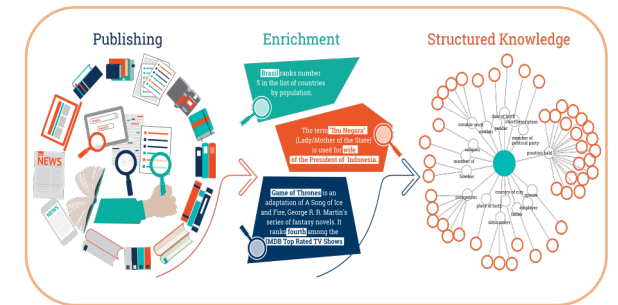
NDSS 2019

Machine Learning For Natural Language Processing

Sentiment Analysis



Information Extraction

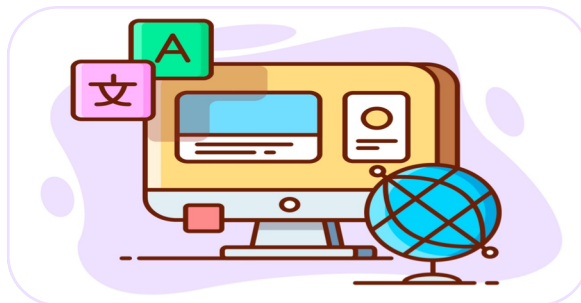


Information Retrieval

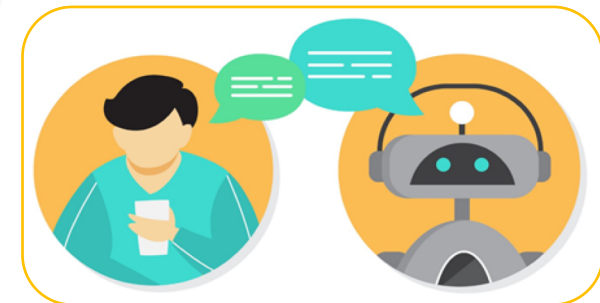


Machine Learning For Multiple Tasks

Machine Translation



Question Answering



Machine Learning As A Service For NLP



Google
Cloud Platform



Microsoft
Azure

*fast*Text



Google Perspective



They Say

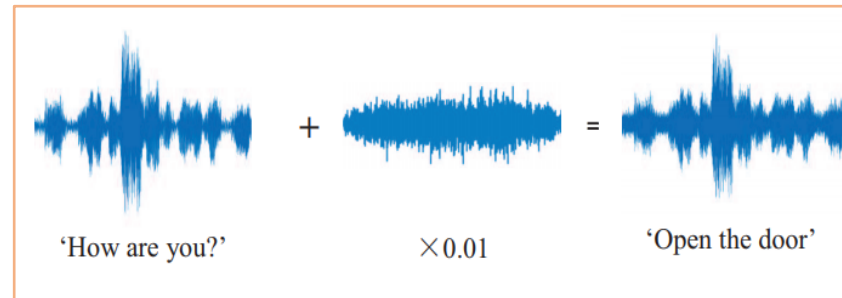
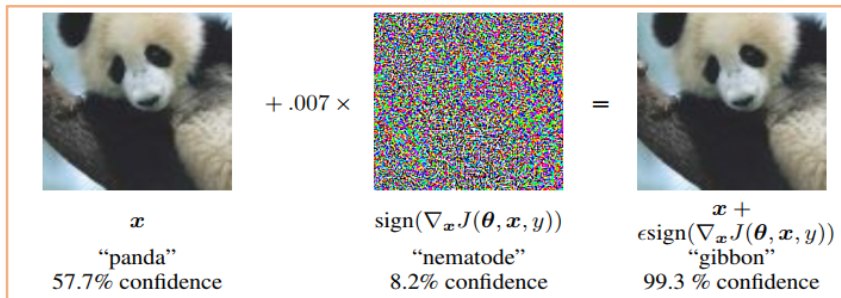
AYLIEN



Breaking Thing Is Easy

Recent works have revealed the vulnerabilities of DNNs in image and speech domain

- The DNNs for image classification are vulnerable to adversarial images. [Goodfellow *et al.*, ICLR'15]
- Automatic speech recognition systems can be broken down by adversarial audios in physical world. [Yuan et al., USENIX'18]



Do the adversarial examples also exist in text domain?

Are the MLaaS for NLP also vulnerable to adversarial examples?



Preliminaries

Adversarial Text

What is the adversarial text?

Carefully generated by adding small perturbations to the legitimate text.

Task: Sentiment Analysis. **Classifier:** Amazon AWS. **Original label:** 100% Negative. **Adversarial label:** 89% Positive.

Text: I watched this movie recently mainly because I am a Huge fan of Jodie Foster's. I saw this movie was made right between her 2 Oscar award winning performances, so my expectations were fairly high. Unfortunately **Unf0rtunately**, I thought the movie was terrible **terrib1e** and I'm still left wondering how she was ever persuaded to make this movie. The script is really weak **wea k**.

What is the challenge for generating adversarial texts?

- The discrete property of text makes it hard to optimize.
- Small perturbations in text are usually clearly perceptible.
- Replacement of a single word may drastically alter the semantics of the sentence.

Related Works For Generating Adversarial Texts

Gradient-based Methods

- Modifying an input text repetitively until it is misclassified. [Papernot *et al.*, MILCOM' 16]
- Changing one token to another by a gradient-based optimization method. [Ebrahimi *et al.*, NAACL' 18]
- Perturbing the important words determined by embedding gradient with hand-crafted synonyms. [Samanta *et al.*, arXiv'17]

Out-of-vocabulary Words

- Breaking machine learning systems down by random character manipulations. [Belinkov *et al.*, ICLR' 18]
- Attacking black-box models by applying random character perturbations. [Gao *et al.*, SPW' 18]
- Changing the toxicity score of the texts by adding spaces or dots between characters. [Hosseini *et al.*, arXiv' 17]

Related Works For Generating Adversarial Texts

Replace with Semantically/Syntactically Similar Words

- Only replacing words with semantically similar ones. [Alzantot *et al.*, arXiv' 18]
- Replacing tokens by random words of the same POS tag with a probability proportional to the embedding similarity. [Ribeiro *et al.*, ACL' 18]

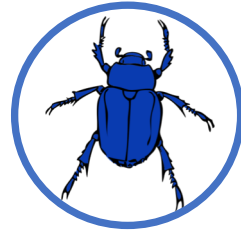
Other Methods

- Attacking reading comprehension systems by adding distracting sentences to the input document. [Jia *et al.*, EMNLP' 17]
- Generating adversarial sequence by Generative Adversarial Networks (GANs). [Zhao *et al.*, ICLR' 18]

Limitations

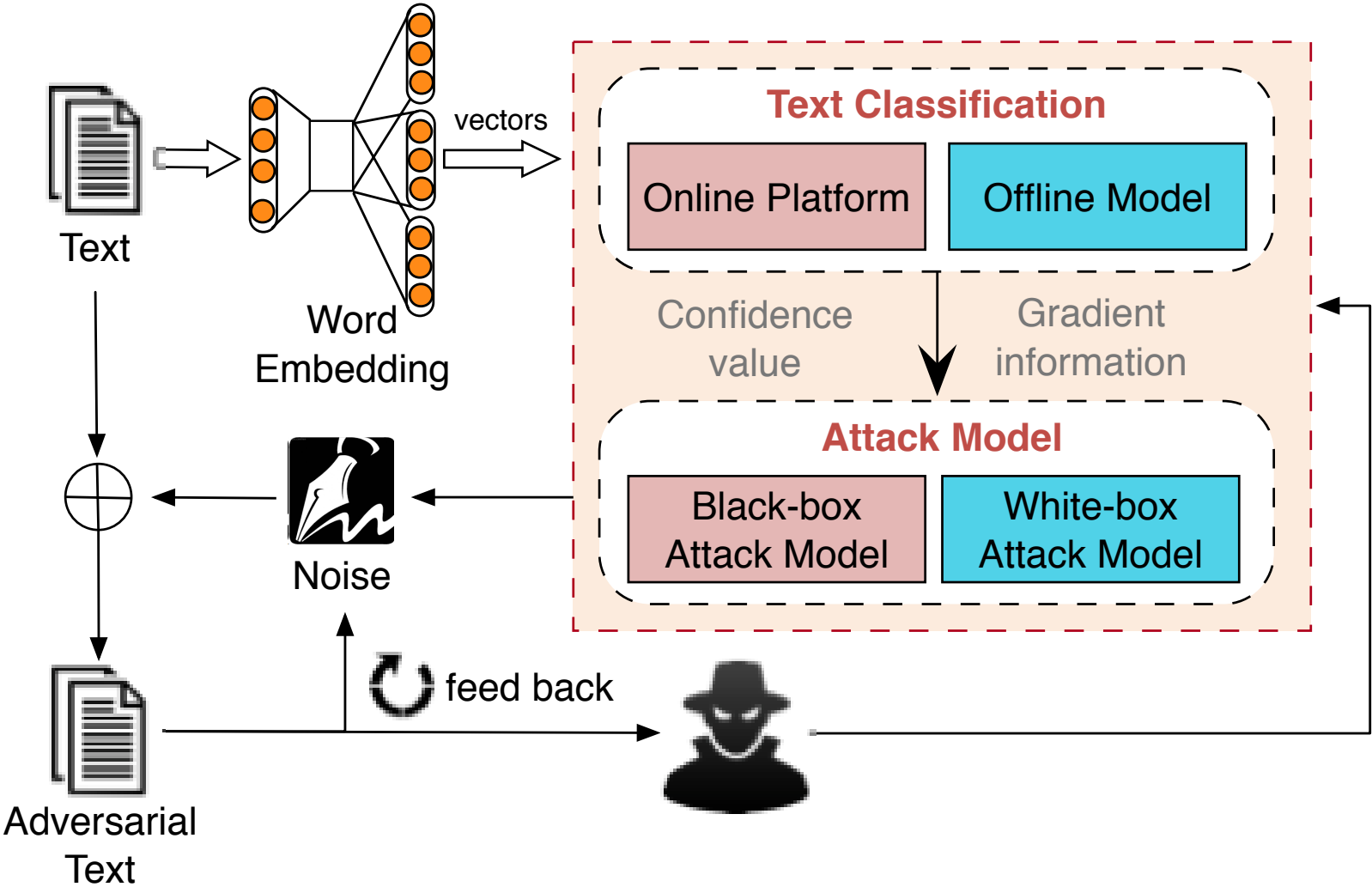
These works are limited in practice due to at least one of the following reasons:

- Limited to short texts
- Significantly affect the original meaning
- Need hand-crafted synonyms and typos
- Requires manual intervention to polish the added sentences
- Not computationally efficient



TextBugger

Framework For TextBugger



Threat Model

White-box

- Have complete knowledge about the targeted model

Black-box

- Do not know the model architecture, parameters or training data
- Only capable of querying the targeted model with output as the prediction or confidence scores

The screenshot displays the ParallelDots AI APIs interface. It features two main sections: Sentiment Analysis and ABUSIVE CONTENT CLASSIFIER. The Sentiment Analysis section shows the input text "There isn't no problem" and the results: Positive (15.80%), Neutral (9.40%), and Negative (74.80%). The ABUSIVE CONTENT CLASSIFIER section shows the input text "You are foolish." and the results: Abusive (91.41%) and Non Abusive (---).

Category	Percentage
Positive	15.80 %
Neutral	9.40 %
Negative	74.80 %
Abusive	91.41 %
Non Abusive	---

Step 1: Finding Important Words

White-box attack

- Find important words by gradient information.

$$C_{x_i} = J_{\mathcal{F}(i,y)} = \frac{\partial \mathcal{F}_y(\mathbf{x})}{\partial x_i}$$

$$J_{\mathcal{F}}(\mathbf{x}) = \frac{\partial \mathcal{F}(\mathbf{x})}{\partial \mathbf{x}} = \left[\frac{\partial \mathcal{F}_j(\mathbf{x})}{\partial x_i} \right]_{i \in 1..N, j \in 1..K}$$

Denotes:

- \mathbf{x} is the input text, x_i is the i^{th} word in \mathbf{x} .
- $\mathcal{F}_j(\mathbf{x})$ is the confidence value of the j^{th} class.
- C_{x_i} is the importance of word x_i .
- N is the total number of words in \mathbf{x} .
- K is the total number of classes.

Step 1: Finding Important Words

Black-box attack

- Find important sentences

$$C_{sentence}(i) = \mathcal{F}_y(\mathbf{s}_i)$$

$S_{ordered} \leftarrow Sort(\mathbf{s})$ according to $C_{sentence}(i)$

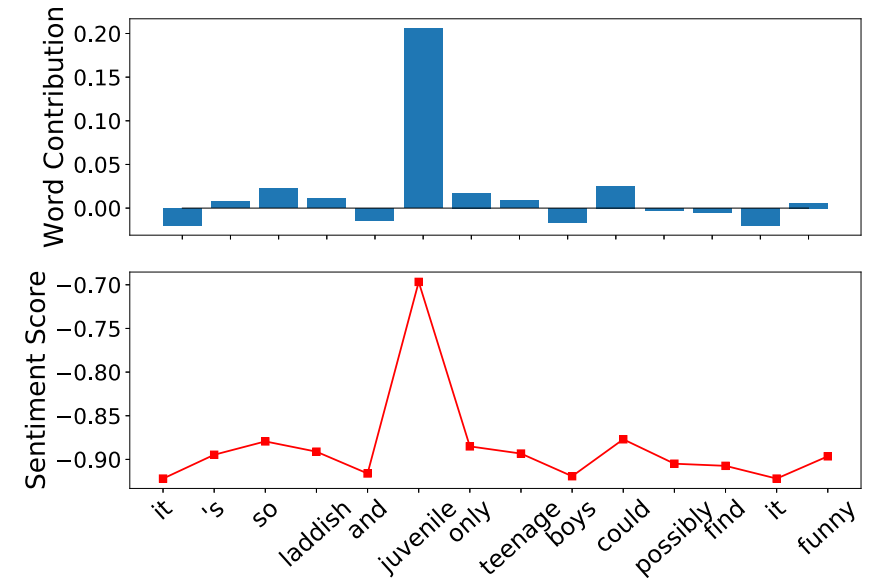
Delete sentences in $S_{ordered}$ if $\mathcal{F}_l(\mathbf{s}_i) \neq y$

- Find important words for each sentence in $S_{ordered}$

$$C_{w_j} = \mathcal{F}_y(w_1, w_2, \dots, w_m) - \mathcal{F}_y(w_1, \dots, w_{j-1}, w_{j+1}, \dots, w_m)$$

Denotes:

- \mathbf{s}_i is the i^{th} sentence in the input text \mathbf{x} .
- $\mathcal{F}_y(\mathbf{s}_i)$ is \mathbf{s}_i 's confidence value of the predicted class y .
- $S_{ordered}$ is the important sentences set.
- $C_{sentence}(i)$ is the importance of word \mathbf{s}_i , C_{w_j} is the importance of the j^{th} word in \mathbf{s}_i .



Sentence: *It is so laddish and **juvenile**, only teenage boys could possibly find it funny.*

Step 2: Bugs Generation

Character-level perturbation: out-of-vocabulary phenomenon

- **Insert:** Insert a space into the word.
- **Delete:** Delete a random character of the word.
- **Swap:** Swap random two adjacent letters in the word.
- **Substitute-C (Sub-C) :** Replace characters with visually similar characters or adjacent characters in the keyboard.

Word-level perturbation: nearest neighbor searching in the embedding space

- **Substitute-W (Sub-W) :** Replace a word with its top k nearest neighbors in a context-aware word vector space.

Original	Insert	Delete	Swap	Sub-C	Sub-W
foolish	f oolish	folish	fooilsh	fo0lish	silly
awfully	awfull y	awfully	awfluly	awfully	terribly
cliches	clich es	clichs	clcihes	cliches	cliche

Step 3: Replacing Important Word By Generated Bug

Optimal bug selection

- choose the optimal bug according to the change of the confidence value

$candidate(k) = \text{replace } w \text{ with } b_k \text{ in } x$

$score(k) = \mathcal{F}_y(x) - \mathcal{F}_y(candidate(k))$

Important word replacement

- Replace the important word by the selected optimal bug
- Repeat until “convergence”
 - the semantic similarity is below the threshold
 - the new text is misclassified by the classifier



Attack Evaluation

Case Study



Sentiment Analysis

Toxic Content Detection

Attack Evaluation: Sentiment Analysis

Dataset

- **IMDB:** 50,000 positive and negative movie reviews
- **Rotten Tomatoes Movie Reviews (MR):** 5,331 positive and 5,331 negative snippets

Targeted Model

- **White-box models:** LR, CNN, LSTM
- **Real-world Online Platforms:**



Google
Cloud Platform



IBM WATSON



Microsoft
Azure



*fast*Text



AYLIEN



Baseline Algorithms

- **White-box:** Random, FGSM+NNS (Nearest Neighbor Search), DeepFool+NNS
- **Black-box:** DeepWordBug

Attack Evaluation: Sentiment Analysis

Evaluation Metrics

➤ **Edit Distance**

➤ **Jaccard Similarity Coefficient**

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

➤ **Euclidean Distance**

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

➤ **Semantic Similarity**

$$S(\mathbf{p}, \mathbf{q}) = \frac{\mathbf{p} \cdot \mathbf{q}}{\|\mathbf{p}\| \cdot \|\mathbf{q}\|} = \frac{\sum_{i=1}^n p_i \times q_i}{\sqrt{\sum_{i=1}^n (p_i)^2} \times \sqrt{\sum_{i=1}^n (q_i)^2}}$$

Generated Adversarial Texts

Successful Attack Examples

Task: Sentiment Analysis. **Classifier:** CNN. **Original label:** 99.8% Negative. **Adversarial label:** 81.0% Positive.

Text: I love these awful **awf ul** 80's summer camp movies. The best part about "Party Camp" is the fact that it **literally literally** has ~~no~~ **No** plot. The cliches **clichs** here are limitless: the nerds vs. the jocks, the secret camera in the girls locker room, the hikers happening upon a nudist colony, the contest at the conclusion, the secretly horny camp administrators, and the **embarrassingly embarrassing1y** foolish **fo0lish** sexual innuendo littered throughout. This movie will make you laugh, but never intentionally. I repeat, never.

Task: Sentiment Analysis. **Classifier:** Amazon AWS. **Original label:** 100% Negative. **Adversarial label:** 89% Positive.

Text: I watched this movie recently mainly because I am a Huge fan of Jodie Foster's. I saw this movie was made right between her 2 Oscar award winning performances, so my expectations were fairly high. ~~Unfortunately~~ **Unf0rtunately**, I thought the movie was terrible **terrib1e** and I'm still left wondering how she was ever persuaded to make this movie. The script is really weak **wea k**.

Attack Performance: Effectiveness And Efficiency

White-box Attack

TABLE II. RESULTS OF THE WHITE-BOX ATTACKS ON IMDB AND MR DATASETS.

Model	Dataset	Accuracy	Random		FGSM+NNS [12]		DeepFool+NNS [12]		TEXTBUGGER	
			Success Rate	Perturbed Word	Success Rate	Perturbed Word	Success Rate	Perturbed Word	Success Rate	Perturbed Word
LR	MR	73.7%	2.1%	10%	32.4%	4.3%	35.2%	4.9%	92.7%	6.1%
	IMDB	82.1%	2.7%	10%	41.1%	8.7%	30.0%	5.8%	95.2%	4.9%
CNN	MR	78.1%	1.5%	10%	25.7%	7.5%	28.5%	5.4%	85.1%	9.8%
	IMDB	89.4%	1.3%	10%	36.2%	10.6%	23.9%	2.7%	90.5%	4.2%
LSTM	MR	80.1%	1.8%	10%	25.0%	6.6%	24.4%	11.3%	80.2%	10.2%
	IMDB	90.7%	0.8%	10%	31.5%	9.0%	26.3%	3.6%	86.7%	6.9%

Remarks

- **Choosing important words to modify is necessary.**
- **Effective:** TextBugger has high attack success rate on all models and performs better than baselines.
- **Evasive :** TextBugger perturbs few words to fool the models.

Attack Performance: Effectiveness And Efficiency

Black-box Attack

TABLE III. RESULTS OF THE BLACK-BOX ATTACK ON IMDB.

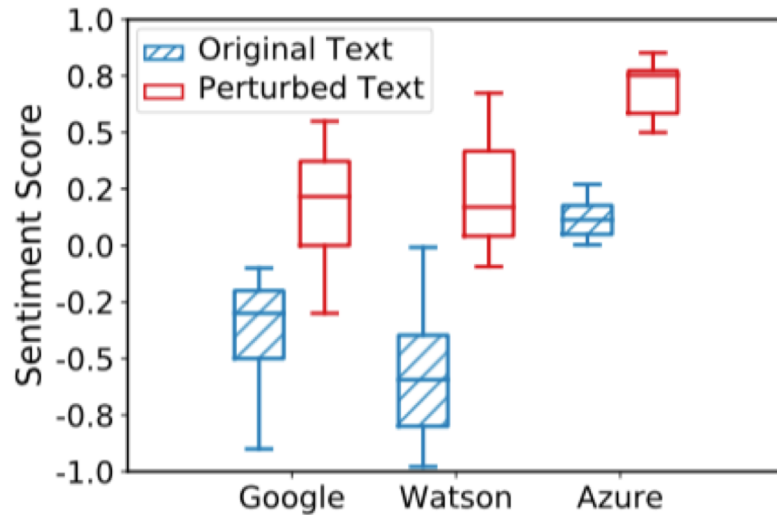
Targeted Model	Original Accuracy	DeepWordBug [11]			TEXTBUGGER		
		Success Rate	Time (s)	Perturbed Word	Success Rate	Time (s)	Perturbed Word
Google Cloud NLP	85.3%	43.6%	266.69	10%	70.1%	33.47	1.9%
IBM Waston	89.6%	34.5%	690.59	10%	97.1%	99.28	8.6%
Microsoft Azure	89.6%	56.3%	182.08	10%	100.0%	23.01	5.7%
Amazon AWS	75.3%	68.1%	43.98	10%	100.0%	4.61	1.2%
Facebook fastText	86.7%	67.0%	0.14	10%	85.4%	0.03	5.0%
ParallelDots	63.5%	79.6%	812.82	10%	92.0%	129.02	2.2%
TheySay	86.0%	9.5%	888.95	10%	94.3%	134.03	4.1%
Aylien Sentiment	70.0%	63.8%	674.21	10%	90.0%	44.96	1.4%
TextProcessing	81.7%	57.3%	303.04	10%	97.2%	59.42	8.9%
Mashape Sentiment	88.0%	31.1%	585.72	10%	65.7%	117.13	6.1%

Remarks

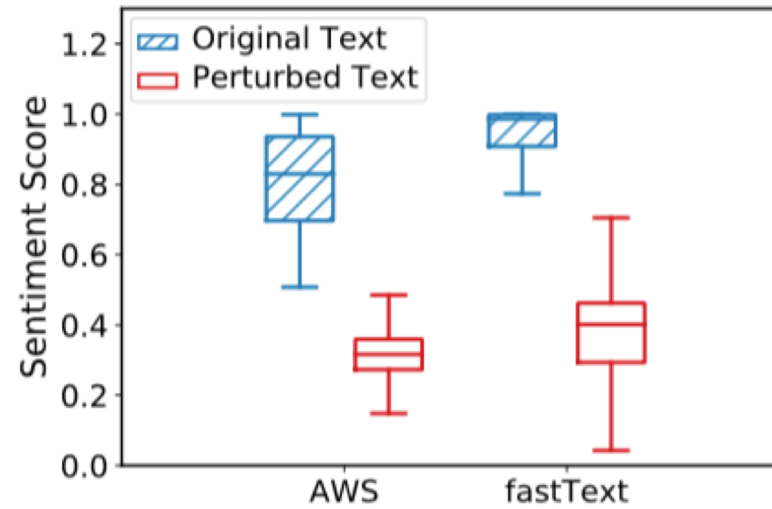
- **Effective:** TextBugger has higher attack success rate against all online platforms than DeepWordBug.
- **Evasive:** TextBugger only perturbs fewer words than DeepWordBug.
- **Efficient:** TextBugger spends less time than DeepWordBug.

Attack Performance: Change Of Confidence

Sentiment Score Distribution



(a) IMDB

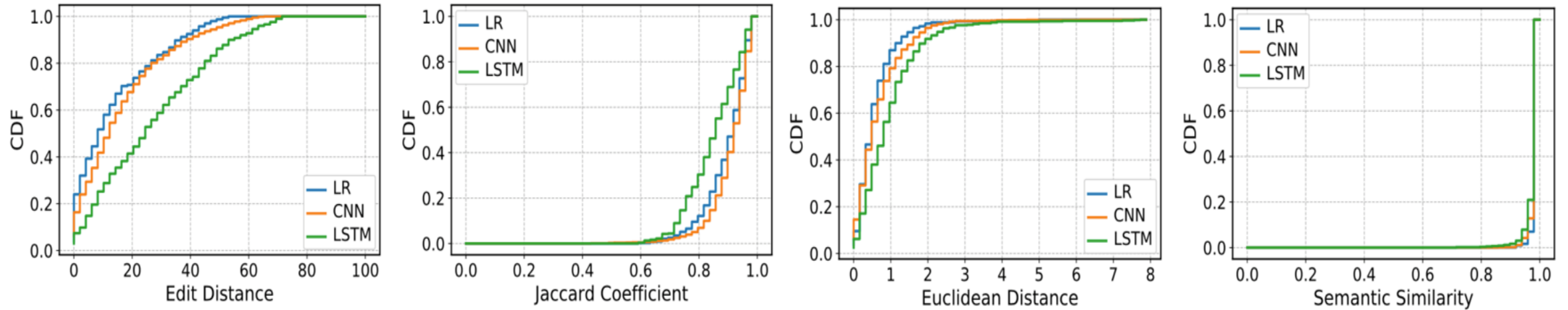


(b) IMDB

Remarks

- TextBugger greatly changes the confidence value of the classification results.
- IBM Watson is more sensitive to the adversarial texts generated by TextBugger.

Utility Analysis: White-box Attack

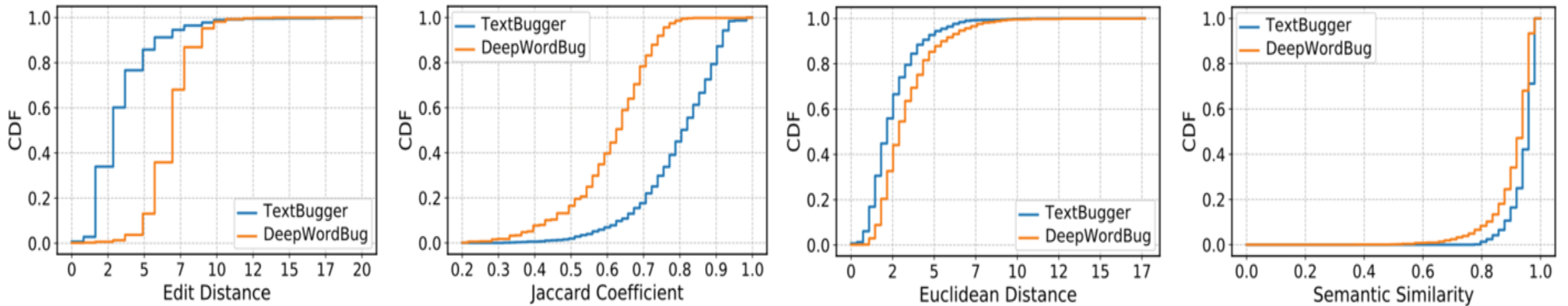


(a) IMDB

Remarks

- The generated adversarial texts preserve good word-level and vector-level utility.

Utility Analysis: Black-box Attack



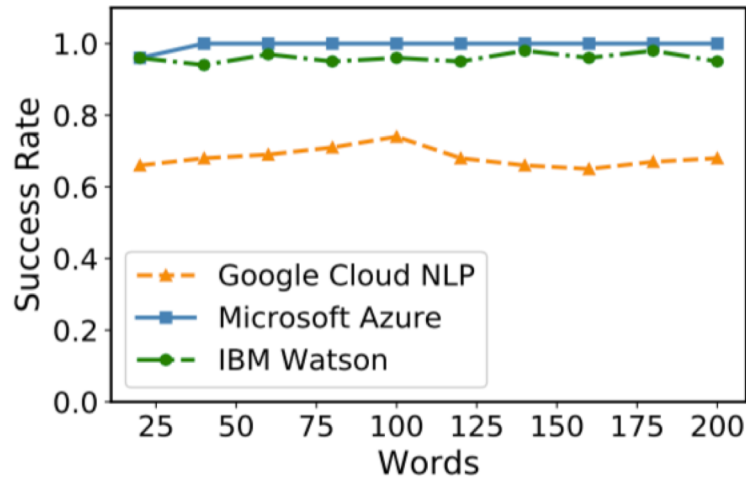
(a) IMDB

Remarks

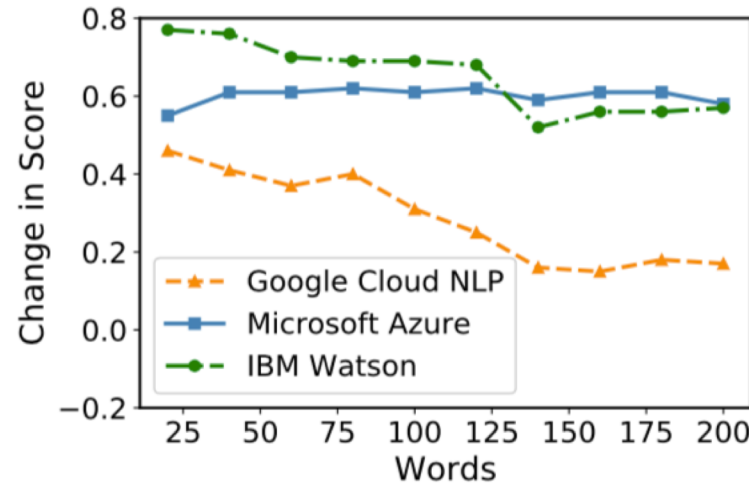
- TextBugger generates higher quality adversarial texts than DeepWordBug.

The Impact Of Document Length

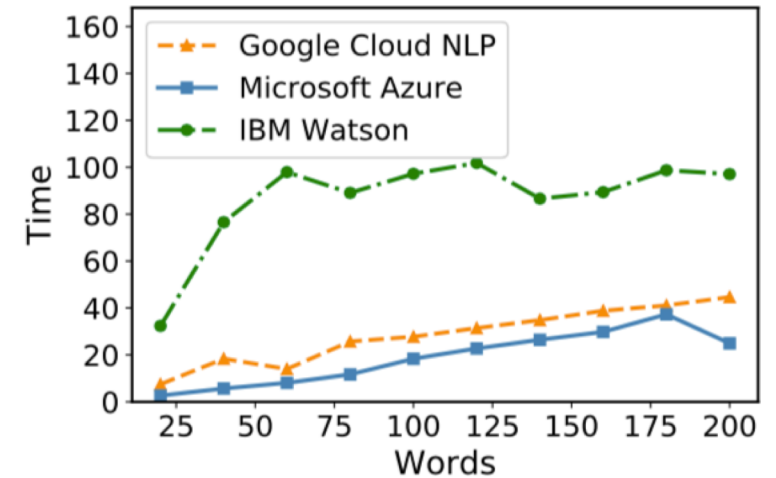
The Impact of Document Length on Attack Performance



(a) Success Rate



(b) Score



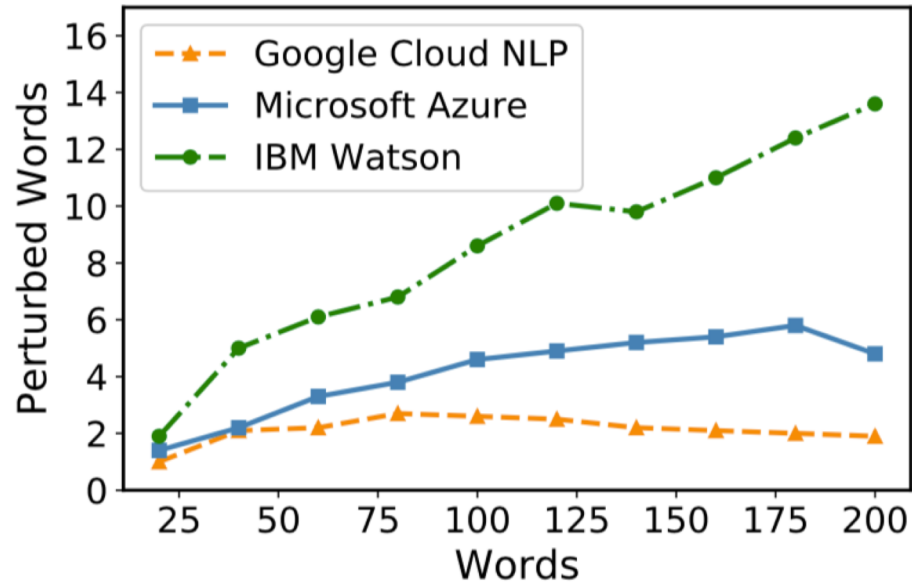
(c) Time

Remarks

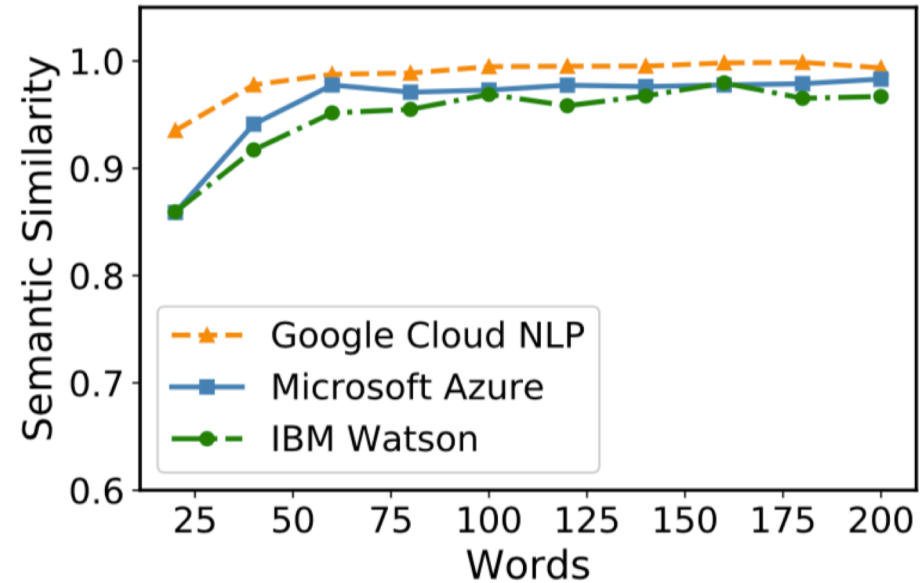
- Length has little impact on the success rate, but may decrease the change of negative class's confidence value.
- The time required for generating one adversarial text increases slightly as the length grows.

The Impact Of Document Length

The Impact of Document Length on The Utility of Generated Adversarial Texts.



(a) Number of Perturbed Words

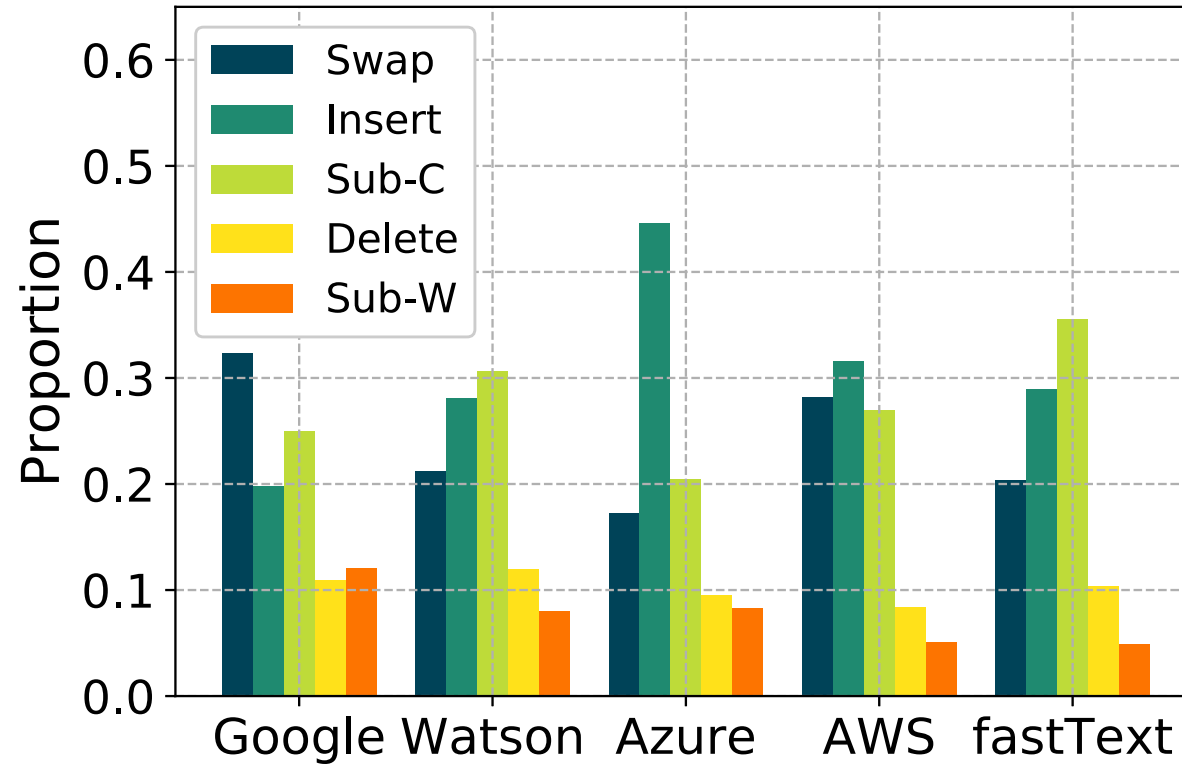


(b) Semantic Similarity

Remarks

- Longer document length leads to more perturbed words.
- The increasing perturbed words do not decrease the semantic similarity of the adversarial texts.

Bug Distribution



Remarks

- Azure and AWS are sensitive to the insert bug
- Watson and fastText are sensitive to Sub-C
- Delete and Sub-W are used less than others

Further Analysis



Transferability



User Study

Transferability

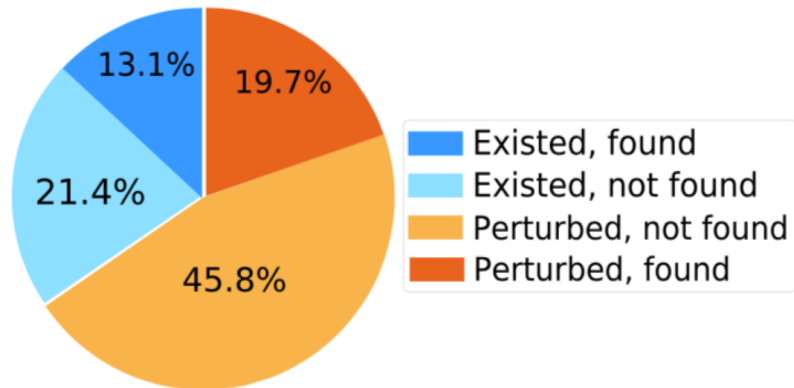
TABLE VII. TRANSFERABILITY ON IMDB AND MR DATASETS.

Dataset	Model	White-box Models			Black-box APIs				
		LR	CNN	LSTM	IBM	Azure	Google	fastText	AWS
IMDB	LR	95.2%	20.3%	14.5%	14.5%	24.8%	15.1%	18.8%	19.0%
	CNN	28.9%	90.5%	21.2%	21.2%	31.4%	20.4%	25.3%	20.0%
	LSTM	28.8%	23.8%	86.6%	27.3%	26.7%	27.4%	23.1%	25.1%
MR	LR	92.7%	18.3%	28.7%	22.4%	39.5%	31.3%	19.8%	29.8%
	CNN	26.5%	82.1%	31.1%	25.3%	28.2%	21.0%	19.1%	20.5%
	LSTM	21.4%	24.6%	88.2%	21.9%	17.7%	22.5%	16.5%	18.7%

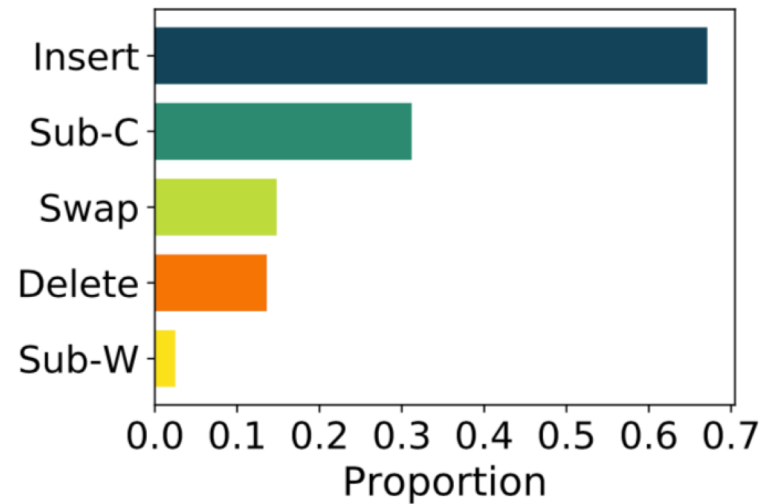
Remarks

- Transferability also exists in adversarial texts among models and online platforms.
- Transferability can be used to attack online platforms even they have call limits.

User Study



(a) The distribution of all mistakes in the samples.

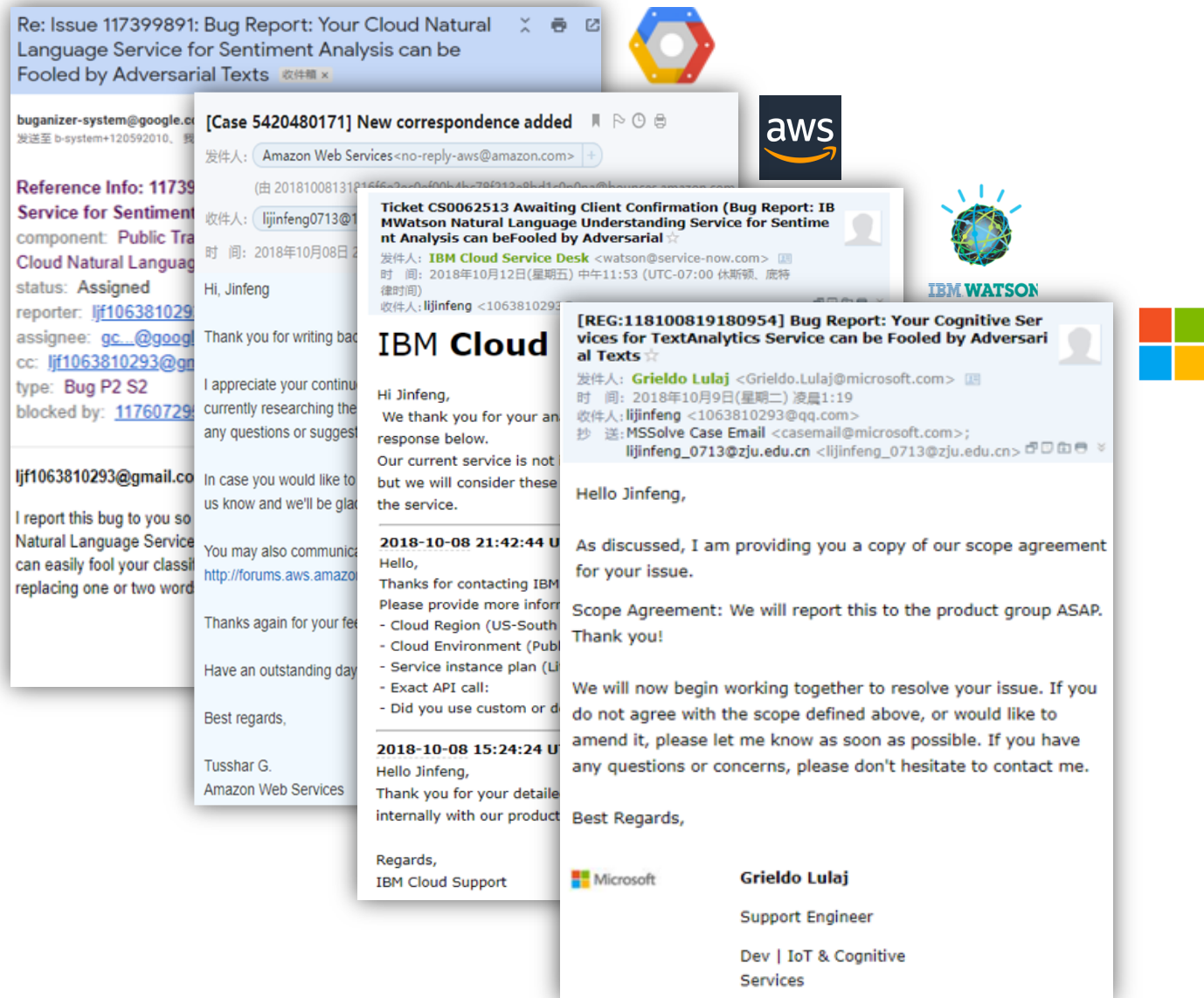


(b) The proportion of found bugs accounting for each kind of bug added in the samples.

Remarks

- Adversarial texts generated by TextBugger are hard to distinguish.
- The insert bug is human-perceptible .
- Sub-W is the most robust bug.

Vulnerability Report



Re: Issue 117399891: Bug Report: Your Cloud Natural Language Service for Sentiment Analysis can be Fooled by Adversarial Texts

buganizer-system@google.c
发送至 b-system+120592010、我

Reference Info: 11739
Service for Sentiment
component: Public Tra
Cloud Natural Language
status: Assigned
reporter: ljf106381029
assignee: gc_@googl
cc: ljf1063810293@g
type: Bug P2 S2
blocked by: 11760729

ljf1063810293@gmail.co
I report this bug to you so
Natural Language Service
can easily fool your classifi
replacing one or two word

[Case 5420480171] New correspondence added

发件人: Amazon Web Services <no-reply-aws@amazon.com>
(由 201810081312:11:53 (UTC-07:00 休斯顿, 波特
律时间)

收件人: lijinfeng0713@1
时 间: 2018年10月08日 2
Hi, Jinfeng

Thank you for writing back
I appreciate your continue
currently researching the
any questions or suggest

In case you would like to
us know and we'll be glad
You may also communicate
<http://forums.aws.amazon.com>

Thanks again for your feedback
Have an outstanding day

Best regards,
Tusshar G.
Amazon Web Services

Ticket CS0062513 Awaiting Client Confirmation (Bug Report: IBM Watson Natural Language Understanding Service for Sentiment Analysis can be Fooled by Adversarial Texts)

发件人: IBM Cloud Service Desk <watson@service-now.com>
时 间: 2018年10月12日(星期五) 中午11:53 (UTC-07:00 休斯顿, 波特
律时间)

收件人: lijinfeng <1063810293@zju.edu.cn>

IBM Cloud

Hi Jinfeng,
We thank you for your an
response below.
Our current service is not
but we will consider these
the service.

2018-10-08 21:42:44 U
Hello,
Thanks for contacting IBM
Please provide more inform
- Cloud Region (US-South
- Cloud Environment (Publ
- Service instance plan (LI
- Exact API call:
- Did you use custom or d

2018-10-08 15:24:24 U
Hello Jinfeng,
Thank you for your detaile
internally with our product

Regards,
IBM Cloud Support

[REG:118100819180954] Bug Report: Your Cognitive Services for TextAnalytics Service can be Fooled by Adversarial Texts

发件人: Grieldo Lulaj <Grieldo.Lulaj@microsoft.com>
时 间: 2018年10月9日(星期二) 凌晨1:19
收件人: lijinfeng <1063810293@qq.com>
抄 送: MSSolve Case Email <casemail@microsoft.com>;
lijinfeng_0713@zju.edu.cn <lijinfeng_0713@zju.edu.cn>

Hello Jinfeng,

As discussed, I am providing you a copy of our scope agreement for your issue.

Scope Agreement: We will report this to the product group ASAP.
Thank you!

We will now begin working together to resolve your issue. If you do not agree with the scope defined above, or would like to amend it, please let me know as soon as possible. If you have any questions or concerns, please don't hesitate to contact me.

Best Regards,
Microsoft

Grieldo Lulaj
Support Engineer
Dev | IoT & Cognitive Services

Summary

We proposed TextBugger, a framework for generating adversarial texts effectively and efficiently

- **Effective:** It outperforms state-of-the-art attacks in terms of attack success rate under both white-box and black-box settings.
- **Evasive:** It preserves the utility of benign text.
- **Efficient:** It generates adversarial text with computational complexity sub-linear to the text length.

We evaluated TextBugger on 15 real-world online applications

- **Dataset:** IMDB, MR and Kaggle.
- **Application:** Includes sentiment analysis and toxic content detection.

We conducted a user study on our generated adversarial texts

- **Utility-preserving:** TextBugger has little impact on human understanding.

We further discuss two potential defense strategies to defend against such attacks



lijinfeng0713@zju.edu.cn