Are Modern Deep Learning Models for Sentiment Analysis Brittle? An Examination on Part-of-Speech

Ahoud Alhazmi[1,2], Wei Emma Zhang[3], Quan Z Sheng[1], and Abdulwahab Aljubairy[1,2]
[1]Department of Computing, Macquarie University, NSW 2109, Australia
[2]Computer Science, Umm Al Qura University, Makkah 21955, Saudi Arabia
[3]School of Computer Science, The University of Adelaide, SA 5005, Australia
{ahoud.alhazmi,abdulwahab.aljubairy}@hdr.mq.edu.au
wei.e.zhang@adelaide.edu.au
michael.sheng@mq.edu.au

***Abstract*—Deep Neural Networks (DNNs) have achieved remarkable results in multiple Natural Language Processing (NLP) applications. However, current studies have found that DNNs can be fooled when using modified samples, namely adversarial examples. This work, specifically, examines DNNs for sentiment analysis using adversarial examples. We particularly aim to examine the impact of modifying the Part-Of-Speech (POS) of words on the input sentences. We conduct extensive experiments on different neural network models across several real-world datasets. The results demonstrate that current DNN models for sentiment analysis are brittle with perturbed noisy words that humans do not have trouble understanding. An interesting finding is that adjective words (*Adj*) and the combination of adjective and adverb words (*Adj-Adv*) provide obvious contribution to fooling sentiment analysis DNN models[1].**

***Index Terms*—Adversarial Example, Neural Networks, Sentiment Analysis, Part-of-Speech**

I. Introduction

Sentiment analysis is a field that automatically analyzes and derives a person's sentiment about a topic. Understanding people's sentiments and emotions allows commercial companies and the government to identify people's sentiment toward products, brands or services.

Deep Neural Networks (DNNs) have been shown to achieve great success in multiple Natural Language Processing (NLP) applications [1]–[3], including sentiment analysis [4]–[6]. However, the interpretability of deep neural networks is still unsatisfactory as they work as black boxes, which means it is difficult to get intuitions from what each neuron exactly has learned. One of the problems of poor interpretability is evaluating the robustness of deep neural networks.

In recent years, Goodfellow et al. [7] added small unperceivable perturbations to image data to form adversarial examples, which are used to evaluate the robustness of DNN image classifiers. They found the evaluated classifiers gave wrong predictions on these adversarial examples. Their work has triggered many follow-up works that examine different DNN models with different methods to generate adversarial examples [8]–[10].

In general, adversarial samples are well designed inputs that can fool a neural network model in the test stage but at the same time they are imperceptible to a human observer. However, existing methods designed for image data cannot be applied directly on textual data, as there are three differences between them. Firstly, image inputs are continuous data but texts are discrete data. Secondly, a small change of the image pixels usually can not be easily perceived by human beings but small changes on character or word in texts will easily be perceived. Thirdly, perturbation on texts would easily change the semantics of text, thus can be easily detected and affect the model output but in images, it usually does not change the semantics of the image as they are trivial. For that, there are two challenges on designing a method for generating textual adversarial samples: i) an adversarial text sample should keep the meaning of the text; and ii) the perturbation should not be perceptible [11].

Very recently, research works that attack textual DNN models have emerged [10]–[16]. They share the common principle to search for key features and then perform perturbations on these features. However, no work considers whether the Part-Of-Speech (POS) of words could help attack deep neural models. In this work, we focus on sentiment analysis of deep neural models that detect either binary-class (negative or positive) or multi-class (e.g., sentiment level from 0 to 5) sentiment signals. We present a simple yet effective black-box method to attack sentiment analysis DNNs by perturbing words of specific POS tags. Two stages are proposed for achieving our goal: i) *determining a token based POS-tagging*, and ii) *perturbing it using three strategies*. To the best of our knowledge, this is the first study that considers the impact of the POS on attacking DNN-based text classifiers. Our contributions are as follows:

- We identify which POS can attack word level deep neural models by modifying only one token. In other words, our method presents that the DNN for sentiment analysis relies on POS words. We find an adjective word (***Adj***) and the combination of adjective and adverb words (***Adj-***

[1]Our code for generating these adversaries available at https://github.com/Ahoud-Alhazmi/Are-Modern-Deep-Learning-for-Sentiment-Analysis-Britlle

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

Adv) provide obvious contribution on fooling sentiment analysis of DNN models more than other word category factors such as verb words or adverb words.
- We observe that word-based models in sentiment analysis are particularly vulnerable to only one adversarial word. Table I shows several adversarial examples.
- This study shows the difference between misspelling attacks and grammatical attacks. The majority of previous works use only misspelling attacks.
- We extensively evaluate our method on a group of state-of-the-art deep neural models and several datasets. The result shows that DNN models are sensitive to adjectives and the combination of adjectives and adverbs more than other word category factors. Also, we find some adversary words that do not convey any feeling can attack the models.

The remainder of the paper is organized as follows. Section II discusses the related work. Then, we introduce our method in Section III. In Section IV, we report the experiment and results. Section V concludes the paper.

TABLE I: Adversarial samples generated using several strategies by only modifying one word or tokens. Modified words as bold words replace the words before them. The first four examples for binary classification and the last two examples for multi-classification. Confidence as the percentages in bracket in the second and third column.

Text	**Original Text Prediction**	**Adversarial Text Prediction**
The worst **(wosrt)** pediatric office I have dealt with, very unprofessional team from front desk to the Dr ...	Negative (91.1%)	Positive (99.5%)
... When people say "the book was so much better"(they're usually **(ulasluy)** wrong anyway) what they are really trying to say ...	Positive (99.2%)	Negative (65.8%)
...I didn't have to wait long. They were very friendly **(friend)**. He immediately knew ...	Positive (99.3%)	Negative (85.9%)
... It started a bit slow and I couldn't understand **(understnad)** the beginning ...	Positive (98.9%)	Negative (80.5%)
... they mailed us something where we would have to fill out and send back. At no time did we ever feel like they genuinely **(guinleney)** care about our horrible **(hlborrie)** stay there ...	Class= 1 (97%)	Class= 5 (91%)
... I know **(konw)** that sounds weird, but that's the best description I have for it....	Class= 2 (96.4%)	Class= 1 (92.9%)

II. Related Work

Adversarial examples have a long history in traditional machine learning for NLP. Biggio et al. [17] discussed the robustness of linear classifiers to filter spam email against adversarial examples. In addition, Dalvi et al. [18] presented how spam emails could not be detected just by adding characters to the emails using naive Bayes classifier.

Recently, several research works focused on crafting adversarial samples against deep learning models in the NLP community. These methods use white-box or black-box strategies. White-box adversary requires explicit knowledge of the attacked model, while black-box adversary sees the DNN model as a black-box. It is only allowed to query the models and get the output. A black-box setting is considered more realistic and practical as in many applications because it is used as service after the deployment stage. In our work, we also focus on the black box-setting of adversarial generations scenarios.

Ebrahimi et al. [13] proposed an exemplary work of white-box attack on text classification. It presents HotFlip method that generates the stronger adversaries using the gradient-based substitution method. Other works using black-box setting and some examples are [14]–[16], [19], [20]. Hosseini et al. [19] found adding spaces or dots between characters can trick Perspective API from Google which predict toxicity messages. Also, in [16], they have shown that character-level machine translation systems are extremely brittle to random character manipulations, with both synthetic or natural noise such as keyboard typos. Furthermore, the work in [15] used the idea of paraphrase generation techniques that create semantically equivalent adversaries (SEA). They generated paraphrases of an input sentence and got predictions from f until the original prediction is changed. At the same time, they considered the semantically equivalent to the generated paraphrases and the original sentence. Moreover, the work in [14] used genetic algorithm for minimizing the number of word replacement from the original text, and at the same time can change the result of the attacked model. Furthermore, Gao et al. [20] presented a simple method to generate adversary on text classification by developing scoring functions to determine the important 'tokens' and then perturbed them.

Specifically for sentiment analysis applications, the work in [21] designed pairs of sentences to fool sentiment analysis systems by using four linguistical strategies: *morphological and syntactic change, semantic change, pragmatic change, and use of world knowledge to determine the meaning*. Our work differs with them by investigating the impact of modifying POS of words on the input sentences. Consequently, our method shows that the DNN for sentiment analysis relies on which POS words.

III. Methodology

In this section, we firstly give the problem definition of attacking DNN classifiers and we then introduce our proposed method that leverages POS in generating adversarial examples.

A. Problem Definition

Given a text sequence $x = \{x_1, x_2, \ldots, x_i, \ldots, x_n\}$, where x_i is a word and a trained DNN classifier model f_θ. Text classification model is represented as $f_\theta : \mathbf{x} \rightarrow y$, a function mapping from the input set x to the label set y. The attacker

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

aims to generate an adversarial sample x^* by adding small noises to the original input data examples in test stage, aiming to fool the deep learning models. x^* can be formalized as: x^* = x + η. Such noise η can manipulate the word or characters. The goal of the adversarial attack can be deviating the label to incorrect one $f_\theta(x^*) \neq f_\theta(x)$.

In this paper, we analyze the sensitivity of DNNs on sentiment analysis tasks using the black-box untargeted attacks. The attacker cannot access the structure, parameters or gradient of the target model. It can only manipulate input samples by testing and observing a classification model's outputs. It is more practical in the real-world applications.

B. The Proposed Approach

In order to understand the behaviour of the sentiment analysis models from binary to multi-class classification for neural network models, we propose a two-step method to investigate the effect of perturbed words of specific POS on the input sentences. In the first step, we search and determine only one targeted token word to be perturbed based on POS-tagging of this token. Then, we perturb this targeted token word.

We develop Algorithm 1 to implement the aforementioned method. The input of this algorithm is a review text and its label. The output is a set of adversarial examples for that text. First, we find numbers of a targeted token in a review text and duplicate the text based on this number (line 1-2). After that, we perturb only one token based on one of three perturbations (line 4). Then, we check the label of the modified text if it is different from the original label or not (line 5-6).

Algorithm 1: Perturbing words of specific POS tags

Input : x and its ground truth label y, classifier f, targeted token

Output: Finding adversarial x^*

1 num ←Find number of targeted token in x
2 T ← $repeat(x, num)$
3 **for** *t in T* **do**
4 x^* ←Modified only one word or token based on *Swap*, *Middle Shuffle* or *Transformation* strategies
5 **if** *F(x^*)* $\neq$ *y* **then**
6 return x^*

*1) **Determining Tokens**:* We duplicate a text based on the numbers of the targeted tokens. Seven targeted tokens are determined in our method using NLTK Punkt tokenizer [22] and Averaged Perceptron Tagger package. They are the modifications on adjectives (***Adj***), verbs (***Verb***), and adverbs (***Adv***). We also study the combinations of adjectives and adverbs (***Adj-Adv***), adjectives and verbs (***Adj-Verb***), verbs and adverbs (***Verb-Adv***) as well as the combination among adjectives, verbs and adverbs (***All***). It is worth to note that we exclude Noun words because they could not convey any opinion or feeling.

*2) **Perturbations on Words**:* We define three strategies to modify each targeted tokens. They are ***Swap, Middle Shuffle*** and ***Transformation***. The first two strategies are considered as misspelling attack while Transformation is a grammatical attack.

- ***Swap*:** it swaps any two random adjacent letters (e.g., "great" to "graet"). We perform swap per word, but we do not alter the first letter.
- ***Middle Shuffle*:** it randomizes the order of all letters in a word except the first and last letters (e.g., "great" to "garet"). For this reason, this noise is applied on words of length ≥ 4.
- ***Transformation*:** We use the transformation method that replace the targeted token across different POS of it (e.g. from adjective to adverb: "professional" to "professionally").

TABLE II: Dataset details and models. Acc. refers to the classification accuracy of DNN models on original test sample. Yelp.Pol refers to Yelp Review Polarity dataset, and Yelp.Full is the Yelp Review Full dataset. MR refers to Movie Review dataset.

Dataset	**IMDB**	**Yelp.Pol**	**Yelp.Full**	**MR**
# Training	25,000	300,000	500,000	7,393
# Test	25,000	26,000	50,000	3,269
# Labels	2	2	5	2
Acc. LSTM	82.18%	91.08%	88.97%	81.54%
Acc. CNN-LSTM	86.47%	92.11%	88.30%	84.95%

IV. Experiments

In this section, we first introduce the settings of our experiments, followed by reporting the results.

A. Experimental Setup

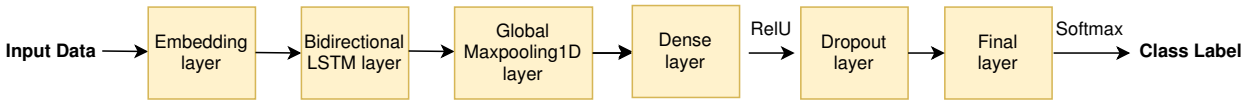
We conduct experiments on different deep learning models across several real-world sentiment analysis datasets.

Dataset: We use four datasets: *IMDB*[2], *Movie Review*[3], and *Yelp Reviews*[4]. The first and second datasets contain two classes whereas the third dataset contains five classes as shown in Table II.

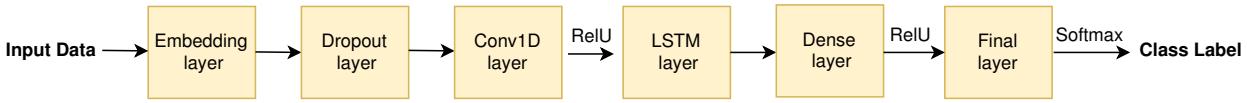
- *IMDB:* It is a movie review dataset that consists of the reviews for different movies along with the class-label (positive or negative sentiment). The dataset is divided into training and testing sets, with each set consisting of 50% positive and 50% negative reviews.
- *Movie Review (MR):* It is a movie review dataset that contains reviews with one sentence per review. Classification involves detecting positive or negative reviews.
- *Yelp Reviews:* It contains 1,569,264 samples that have review texts. Two classification tasks are constructed from this dataset: one is for predicting full number of stars

[2]http://ai.stanford.edu/ amaas/data/sentiment/
[3]http://www.cs.cornell.edu/people/pabo/movie-review-data/
[4]https://www.yelp.com/dataset/challenge

978-1-7281-6926-2/20/$31.00 ©2020 IEEE



(a) Bidirectional LSTM



(b) CNN-LSTM

Fig. 1: The architecture of the DNN models used for sentiment classification

TABLE III: Accuracy drop (%) of LSTM on different adversarial strategies. Tran. refers to Transformation strategy. MR refers to Movie Review dataset.

DataSet	**Yelp Review Polarity**			**IMDB**			**Yelp Review Full**			**MR**		
Strategy	Swap	Shuffle	Tran.	Swap	Shuffle	Tran.	Swap	Shuffle	Tran.	Swap	Shuffle	Tran.
Adj	**0.28%**	**0.28%**	**0.30%**	**0.17%**	**0.18%**	**0.20%**	**0.09%**	**0.07%**	**0.05%**	**0.36%**	**0.34%**	**0.6%**
Verb	0.05%	0.05%	0.06%	0.04%	0.04%	0.06%	0.13%	0.03%	0.10%	0.28%	0.27%	0.30%
Adv	0.07%	0.08%	0.08%	0.07%	0.10%	0.07%	0.05%	0.04%	0.03%	0.25%	0.24%	0.28%
Adj-Adv	**0.38%**	**0.39%**	**0.43%**	**0.23%**	**0.24%**	**0.25%**	**0.12%**	**0.11%**	**0.13%**	**0.34%**	**0.35%**	**0.36%**
Adj-Verb	0.36%	0.36%	0.35%	0.18%	0.18%	0.17%	0.07%	0.08%	0.06%	0.15%	0.13%	0.12%
Verb-Adv	0.15%	0.14%	0.13%	0.09%	0.11%	0.12%	0.03%	0.07%	0.05%	0.14%	0.20%	0.13%
All	0.47%	0.48%	0.34%	0.25%	0.28%	0.26%	0.08%	0.08%	0.09%	0.32%	0.27%	0.30%

TABLE IV: Accuracy drop (%) of CNN-LSTM on different adversarial strategies. Tran. refers to Transformation strategy. MR refers to Movie Review dataset.

DataSet	**Yelp Review Polarity**			**IMDB**			**Yelp Review Full**			**MR**		
Strategy	Swap	Shuffle	Tran.	Swap	Shuffle	Tran.	Swap	Shuffle	Tran.	Swap	Shuffle	Tran.
Adj	**0.14%**	**0.15%**	**0.75%**	**0.17%**	**0.05%**	**0.5%**	**0.07%**	**0.07%**	**0.025%**	**0.24%**	**0.13%**	**0.01%**
Verb	0.11%	0.10%	0.30%	0.02%	0.02%	0.2%	0.03%	0.03%	0.023%	0.13%	0.10%	0.08%
Adv	0.12%	0.11%	0.10%	0.08%	0.08%	0.06%	0.03%	0.04%	0.02%	0.13%	0.10%	0.15%
Adj-Adv	**0.25%**	**0.26%**	**0.27%**	**0.29%**	**0.31%**	**0.32%**	**0.48%**	**0.10%**	**0.35%**	**0.33%**	**0.30%**	**0.47%**
Adj-Verb	0.21%	0.21%	0.23%	0.22%	0.23%	0.21%	0.07%	0.09%	0.08%	0.13%	0.10%	0.18%
Verb-Adv	0.21%	0.22%	0.20%	0.16%	0.18%	0.18%	0.04%	0.06%	0.04%	0.12%	0.20%	0.15%
All	0.33%	0.34%	0.32%	0.31%	0.34%	0.35%	0.08%	0.09%	0.08%	0.38%	0.32%	0.35%

the user has given (e.g., very positive= 5, positive= 4, neutral= 3, negative= 2 and very negative= 1), and the other is for predicting a polarity label by considering stars 1 and 2 as negative review, and 4 and 5 as positive.

Attacked Models: Our method can be applied to any DNN classifier based on the black-box setting. We choose most widely-used neural models to examine the generality of our attack. The first model is the Long Short-Term Memory (LSTM) [23] and the second combines Convolutional Neural Network (CNN) and Long Short-Term Memory (CNN-LSTM) [24] for classification purpose. We select these two word level models because they have been studied in several research such as [13], [14], [20]. In our work, we go a step further by identifying which POS word can attack these models. Hence our goal is not to compare between these two models but to use them to confirm our observation.

The architectures of these models LSTM and CNN-LSTM are shown in Figures 1a and 1b respectively. We use 80% data as training and 20% as validation and train for a maximum of 40 epochs. We train these models without using adversarial samples. We apply modified words only on test samples. For this experiment, we generate adversarial examples for 100% of the test data. The accuracy of both models on original test samples shows in Table II. These models are similar to the state-of-the-art results on these datasets.

Experimental Environment: We train the target models and implement attacking methods using Keras[5]. All the experiments are run on a PC with Windows 10 (64-bit) operating system, 3.10 Ghz CPU (i5-4440), and 8GB RAM.

B. Results

We report the effectiveness of our proposed method and analyze the generated adversarial examples in this section.

*1) **Effectiveness of Adversarial Examples**:* Table III shows the drop of LSTM performance using the specified strategies on the four datasets. Furthermore, Table IV shows the drop that happens in the accuracy of CNN-LSTM model. There is a slight drop in the accuracy of the model among seven targeted

[5]https://keras.io/

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

TABLE V: The ten most common words that attack CNN-LSTM model using *Swap* and *Middle Shuffle* strategies. Bold words are common words among the four datasets.

Yelp Review Polarity			**IMDB**			**Yelp Review Full**			**MR**		
Adj	***Verb***	***Adv***	***Adj***	***Verb***	***Adv***	***Adj***	***Verb***	***Adv***	***Adj***	***Verb***	***Adv***
Swap											
great	**were**	never	**great**	**have**	**very**	**great**	**were**	**very**	worst	**have**	never
worst	**have**	**just**	worst	**were**	**just**	**best**	**have**	definitely	**best**	It's	even
best	told	**very**	**best**	being	even	**good**	love	**just**	awful	does	**just**
good	won't	back	other	makes	also	nice	recommend	really	real	makes	rather
rude	don't	definitely	**good**	think	**only**	little	come	back	**great**	made	**only**
worth	said	here	more	make	**really**	decent	**don't**	**never**	worth	being	**very**
nice	recommend	always	such	know	never	delicious	think	here	**good**	seem	ever
disappointed	think	**really**	excellent	been	**definitely**	favorite	ordered	pretty	first	doesn't	quit
friendly	come	even	perfect	**love**	then	friendly	going	always	better	been	enough
horrible	**love**	probably	real	**don't**	still	worth	loved	also	terrible	know	probably
Middle Shuffle											
best	**were**	never	**best**	**have**	well	**best**	**were**	**very**	worst	have	never
worst	**have**	**just**	**great**	been	**very**	**good**	**have**	**just**	best	does	even
great	told	**very**	worst	**were**	**just**	**great**	recommend	really	great	makes	**just**
good	won't	back	other	makes	really	nice	love	definitely	awful	made	rather
clean	recommend	always	**good**	being	also	little	don't	back	first	It's	**only**
good	don't	here	excellent	love	**definitely**	decent	think	**never**	**good**	doesn't	ever
worth	said	**really**	such	watch	still	worth	come	pretty	real	being	**very**
horrible	think	always	much	bothered	**only**	other	tasted	here	better	seem	enough
nice	come	even	real	understand	**never**	delicious	ordered	still	true	have	know
really better	love	probably	little	take	then	much	didn't	also	worth	know	quit
Transformation											
great	**were**	never	**best**	**have**	well	**best**	**were**	**very**	worst	have	never
good	**have**	**just**	**great**	been	**very**	**good**	**have**	**just**	best	does	even
worst	told	**very**	worst	**were**	**just**	**great**	recommend	really	great	makes	**just**
poor	won't	back	other	makes	really	nice	love	definitely	worth	being	**only**
rude	recommend	definitely	**good**	being	also	little	don't	back	better	seem	quit
disappointed	think	here	excellent	think	also	decent	love	**never**	awful	made	enough
worth	said	**really**	such	watch	still	worth	come	pretty	terrible	been	**very**
horrible	love	always	much	bothered	**only**	other	tasted	here	first	works	really
nice	don't	even	real	understand	**never**	delicious	ordered	still	interesting	know	rather
terrible	come	probably	little	take	then	much	didn't	also	wonderful	doesn't	probably

tokens. The results do not show a big difference among the three strategies although the *Swap* strategy can be as small perturbations to human observers changes the meaning of the text. In other words, it can generate adversarial word that looks quite similar to the original word.

Comparing between misspelling attacks (i.e., *Swap* and *Middle Shuffle* strategies) and grammatical attacks (i.e., *Transformation* strategy), we find the results are similar in both strategies although misspelling attack replaces a targeted word with an out-of-vocabulary word while the grammatical attack replaces the targeted word with a word from vocabularies. From the results, the word-level DNN models are sensitive to the change of only one token either by manipulating the most characters or just one character.

Furthermore, we find that the DNNs for binary classification tend to be more brittle than multi-classification when these models encounter two or more modified words. For example in LSTM, the decrease of the accuracy by using *Swap-All* strategy is 0.47% and 0.25% in "Yelp Review Polarity" and "IMDB", respectively, whereas the decrease of the accuracy in "Yelp Review Full" is 0.08%. In most cases, the decrease of the accuracy that happens in multi-classification does not exceed 0.19%.

Moreover, we find that the adjective adversary decreases the models' accuracy more than the adverb and the verb adversary in all strategies as shown in Tables III and IV. Comparing *Adj-Adv*, *Adj-Verb* and *Verb-Adv*, we find that the combination of adjectives and adverbs *Adj-Adv* is able to attack both models better than the two others' combinations. Clearly, DNN text classifiers seem to be similar to human brain because people can often extract semantic orientation from adjective words as the primary words of the content in a text.

2) ***Analysis of Adversarial Examples****:* Although the accuracy of the two models drops slightly, we are able to create a number of adversarial examples using our method. This section discusses these created adversarial samples. Table I shows a few adversarial examples with only one changed token for binary classification as (positive and negative). We observe that the prediction performance on original and adversarial examples are both high. Furthermore, we present a few adversarial examples for multi-classification as shown in the last two rows of Table I.

We find the most examples that attack the CNN-LSTM model can also attack the LSTM model similar as the finding from [25]. Table V shows the ten most common words that create adversary to attack for each dataset after manipulating their characters using *Swap*, *Middle Shuffle* and *Transformation* strategies. We arrange these common words in a descending order where the first word is the most common word that forms an adversarial example. We also find several common words

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

TABLE VI: Number of generated adversarial samples (#AE) in *Swap* strategy for the four datasets. (No) and (Yes) are the percentage of the adversary's words do not exist in the dictionary or exist in the dictionary, respectively

Targeted tokens	Dataset	**Yelp Review Polarity**	**IMDB**	**Yelp Review Full**	**MR**
Adj	**#AE**	1,007	4,372	6,692	403
	(No) (Yes)	(75%) (25%)	(96%) (4%)	(65%) (35%)	(76%) (24%)
Verb	**#AE**	1,256	4,769	5,639	375
	(No) (Yes)	(80%) (20%)	(94%) (6%)	(77%) (22%)	(73%) (27%)
Adv	**#AE**	457	2,029	3,530	359
	(No) (Yes)	(72%) (28%)	(93%) (7%)	(53%) (47%)	(74%) (26%)
Adj-Adv	**#AE**	916	2,994	9,098	395
	(No) (Yes)	(76%) (24%)	(95%) (5%)	(65%) (35%)	(69%) (31%)
Adj-Verb	**#AE**	1,284	5,488	10,541	340
	(No) (Yes)	(79%) (21%)	(93%) (7%)	(75%) (25%)	(70%) (30%)
Verb-Adv	**#AE**	808	2,565	8,425	352
	(No) (Yes)	(81%) (19%)	(91%) (9%)	(76%) (24%)	(72%) (28%)
All	**#AE**	1,042	3,176	12,769	377
	(No) (Yes)	(79%) (21%)	(90%) (10%)	(74%) (26%)	(62%) (38%)

TABLE VII: Number of generated adversarial samples (#AE) in *Middle Shuffle* strategy for the four datasets. (No) and (Yes) are the percentage of the adversary's words do not exist in the dictionary or exist in the dictionary, respectively

Targeted tokens	Dataset	**Yelp Review Polarity**	**IMDB**	**Yelp Review Full**	**MR**
Adj	**#AE**	1,021	4,674	6,468	410
	(No) (Yes)	(95%) (5%)	(99%) (1%)	(93%) (7%)	(98%) (2%)
Verb	**#AE**	1,266	158	5,881	363
	(No) (Yes)	(96%) (4%)	(98%) (2%)	(95%) (5%)	(97%) (3%)
Adv	**#AE**	655	2,207	3,656	394
	(No) (Yes)	(85%) (15%)	(100%) (0%)	(81%) (19%)	(99%) (1%)
Adj-Adv	**#AE**	940	3,086	5,363	390
	(No) (Yes)	(89%) (11%)	(99%) (1%)	(88%) (12%)	(97%) (3%)
Adj-Verb	**#AE**	1,307	5,571	7,536	290
	(No) (Yes)	(94%) (6%)	(98%) (2%)	(92%) (8%)	(98%) (2%)
Verb-Adv	**#AE**	839	2,678	4,385	389
	(No) (Yes)	(92%) (8%)	(98%) (2%)	(91%) (9%)	(98%) (2%)
All	**#AE**	1,091	3,297	12,875	352
	(No) (Yes)	(92%) (8%)	(96%) (4%)	(92%) (8%)	(95%) (5%)

among the four datasets for each strategy. For example, "great" is the most common word in *Swap* strategy that creates the adversarial examples. Some of these words do not convey any opinion or feeling. For instance, the verbs "were" and "have" are capable to attack the DNN model on the four datasets even though these two words do not convey any feeling opinion such as "love" or "recommend" verbs. That explains to us there is failure in the deep neural network to understand and identify the opinion words.

Table VI and VII show the number of adversary samples that are created using *Swap* and *Middle Shuffle* strategies, respectively. These tables also show the percentage of these adversaries word samples that exist in the dictionary or not (i.e the out-of-vocabulary in training data). We find tokens that are not included in the dictionary have higher percentage for the two strategies *Swap* and *Middle Shuffle*. Thus, it provides an evidence to use these adversaries during training the models to increase the robustness of the models. Probably, it is arguable that using an auto-corrector can provide a solution to misspelling problems. However, based on our experience, using a spelling checker would add enormous overhead to the model and is therefore hardly exploited by the attackers.

However, by looking to *Swap* modification in only "Yelp Review Polarity" and "Yelp Review Full" datasets, the percentage of these tokens which exist in the in-of-vocabulary are increased comparing to *Middle Shuffle*. For example, changing from "best" to "bets", "true" to "ture" and "sure" to "suer" creates adversary and all of these words are in-of-vocabulary in training data. Future research can examine these tokens which may affect the meaning of the text.

*3) **Human Perception**:* To evaluate our crafted adversaries, we perform a user study consisting of two objectives. The first one is to realize whether the applied perturbation on words will change the human perception of reviews' text and the second one is to recognize which POS adversaries will be noticed clearly by a human.

The experiments are performed with 20 participants, and they have no prior knowledge about this study. The number of participating volunteers is the average from that used in previous studies [11]–[14]. Also, we select 84 adversarial samples randomly and all these adversarial samples fooled the targeted models successfully. Then, we divide these 84 samples to four groups and each group contains 21 reviews including all the seven targeted tokens across the three perturbations strategies. Each participant reviews one group, and they are asked to i) label the sentiment of the text (i.e., positive or negative.) and

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

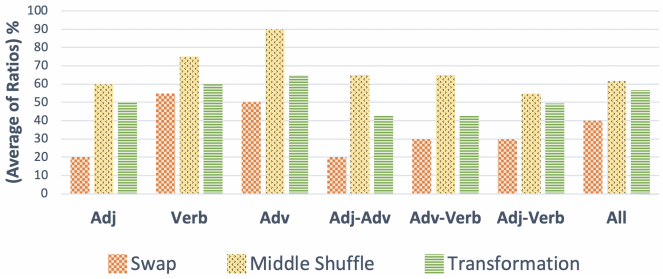


Fig. 2: The detailed results of user study.

ii) highlight suspicious words in the samples if it is found.

After analyzing results, we find 95.7% of the participants' responses matched their original text labels that means the utility is indeed preserved in the adversarial samples. Furthermore, we find the *Swap* strategy is the hardest to find by 35% followed by *Transformation* by 54.5% by whereas *Middle Shuffle* is the easiest strategy to find by 71.1%. Also, we present the average of the ratios of each POS suspicious words that are found by participants as shown in Figure 2. Interestingly, we find most participants do not find suspicious *Adj* words easily for the three strategies whereas they find the suspicious *Adv* words more than suspicious *Adj* and *Verb* words. Also, they find the combination of *Adj-Adv* and *Adv-verb* more than *Adj-Verb*. From the results, it is clear that the participants can notice adverb words more than other POS.

V. Conclusion and Future Work

In this paper, we have evaluated the robustness of DNN models for sentiment analysis. We particularly investigate the effect of the adversarial words of specific POS on attacking modern sentiment analysis DNN models. We observe high importance for adjective and the combination of adjective and adverb. Also, some adversary words that do not convey any feeling can attack the models. A major future research direction will consider the importance of immunizing the models against these noisy words.

References

[1] C. Jin, B. He, K. Hui, and L. Sun, "TDNN: A Two-Stage Deep Neural Network for Prompt-Independent Automated Essay Scoring," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, vol. 1, 2018, pp. 1088–1097.

[2] X. Chen, J. Li, and H. Wang, "Keyphrase Guided Beam Search for Neural Abstractive Text Summarization," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–9.

[3] Z. Bitvai and T. Cohn, "Non-linear Text Regression with a Deep Convolutional Neural Network," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015, pp. 180–185.

[4] C. Santos and M. Gatti, "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts," in *Proceedings of the 25th International Conference on Computational Linguisticse (COLING)*, 2014, pp. 69–78.

[5] Z. Chen, S. Shen, Z. Hu, X. Lu, Q. Mei, and X. Liu, "Emoji-Powered Representation Learning for Cross-Lingual Sentiment Classification," in *Proceedings of the World Wide Web Conference (WWW)*, 2019, pp. 251–262.

[6] J. Chen, H. Hou, Y. Ji, and J. Gao, "Graph Convolutional Networks with Structural Attention Model for Aspect Based Sentiment Analysis," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–7.

[7] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[8] H. Chen, H. Zhang, P. Chen, J. Yi, and C. Hsieh, "Attacking Visual Language Grounding with Adversarial Examples: A Case Study on Neural Image Captioning," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL*, 2018, pp. 2587–2597.

[9] Z. Zhao, D. Dua, and S. Singh, "Generating Natural Adversarial Examples," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[10] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial Attacks on Deep Learning Models in Natural Language Processing: A Survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2019 (in press).

[11] L. Bin, L. Hongcheng, S. Miaoqiang, B. Pan, L. Xirong, and S. Wenchang, "Deep Text Classification Can be Fooled," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018, pp. 4208–4215.

[12] R. Jia and P. Liang, "Adversarial Examples for Evaluating Reading Comprehension Systems," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017, pp. 2021–2031.

[13] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "HotFlip: White-Box Adversarial Examples for Text Classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018, pp. 31–36.

[14] M. Alzantot, Y. Sharma, A. Elgohary, B. Ho, M. B. Srivastava, and K. Chang, "Generating Natural Language Adversarial Examples," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018, pp. 2890–2896.

[15] M. T. Ribeiro, S. Singh, and C. Guestrin, "Semantically Equivalent Adversarial Rules for Debugging NLP Models," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018, pp. 856–865.

[16] Y. Belinkov and Y. Bisk, "Synthetic and Natural Noise both Break Neural Machine Translation," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[17] B. Biggio, G. Fumera, and F. Roli, "Multiple Classifier Systems for Robust Classifier Design in Adversarial Environments," *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1-4, pp. 27–41, 2010.

[18] N. Dalvi, P. Domingos, S. Sanghai, D. Verma *et al.*, "Adversarial Classification," in *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2004, pp. 99–108.

[19] H. Hosseini, S. Kannan, B. Zhang, and R. Poovendran, "Deceiving google's perspective api built for detecting toxic comments," 2017.

[20] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers," in *Proceedings of the IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 50–56.

[21] M. Taylor, C. Willy, E. Micha, K. David, M.-C. de Marneffe, S. Cory, S. Symon, and W. Michael, "Breaking NLP: Using Morphosyntax, Semantics, Pragmatics and World Knowledge to Fool Sentiment Analysis Systems," in *Proceedings of the 1st Workshop in Association for Computational Linguistics (ACL)*, 2017, pp. 33–39.

[22] T. Kiss and J. Strunk, "Unsupervised Multilingual Sentence Boundary Detection," *Computational Linguistics*, vol. 32, no. 4, pp. 485–525, 2006.

[23] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent Convolutional Neural Networks for Text Classification," in *Proceedings of the 29th National Conference of the American Association for Artificial Intelligence (AAAI)*, 2015, pp. 2267–2273.

[24] J. Wang, L. Yu, K. R. Lai, and X. Zhang, "Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016, pp. 225–230.

[25] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing Properties of Neural Networks," in *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE