GENERATING ADVERSARIAL EXAMPLES IN TEXT CLASSIFICATION

A PREPRINT

Zhenyuan Liu [*]
Department of Computer Science
The Chinese University of Hong Kong
Hong Kong, China
`zyliu8@cse.cuhk.edu.hk`

Yuxiao Qu [*]
Department of Computer Science
The Chinese University of Hong Kong
Hong Kong, China
`yxqu8@cse.cuhk.edu.hk`

August 20, 2019

ABSTRACT

In this paper, we implement several effective strategies to generate adversarial examples for the text classification task. Specifically, these strategies include occluding, replacing and deleting characters or words based on the final confidence (black box attack) **?**, and based on the gradients of inputs to choose our modified target (white box attack) **?**. These methods are applied to different kinds of text classification models, include character level CNN (char-CNN) **?**, word-level CNN (word-CNN) **?** and LSTM with Attention (LSTM) **?**. The results of the experiments reveal that our strategies can effectively attack Deep Neural Network models with different architectures through imperceptible disturbances of human beings, which means within a small editing distance. On the other hand, through the analysis of statistical data, we also found that the sensitivity of different models to specific attack methods may vary. These conclusions allow us to design more adversarial examples based on the model structures and improve the performance of NLP tasks more efficiently.

Keywords Adversarial samples · Sentiment Analysis · Deep Learning Classifier

[*]Equal Contribution

1 Introduction

Deep learning has achieved remarkable results in the field of Natural Language Processing (NLP). The emergence of Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) has greatly improved the efficiency of a series of NLP tasks such as sentiment analysis, machine translation, and question answering systems. Recent studies have shown that by generating a series of adversarial samples, which can cause a well-trained model to misclassify **?**. As we can see from the following example, after deleting four letters of the original sentence, we can flip the prediction of the classifier.

DVD player crapped out after one year, I also began having the incorrect disc dvd player crapped out after one year, i also began having the incorrect disc problems that i've read about on here. the vcr still works, but the dvd side is useless...	$\Longrightarrow$	DVD player crapped out afer ne year, I also began having the incorrec disc dvd player crapped out after one year, i also began having the incorrect disc problems that i've read about on here. the vcr still works, but the dvd side is useess...
the probability of original text being negative is 99.87%		the probability of adversarial sample being negative is 47.22%

This technology can greatly accelerate the training of a model and reduce the security risks in applications that are based on natural language processing.

However, compared to the feature of picture or audio, the text data is discrete **?**, which means that even a small disturbance can change the meaning of the sentence, or make it lose its meaning. Therefore, the core of generating corresponding adversarial samples for NLP tasks is keeping the original meaning as much as possible while fooling the model.

In this paper, we generate an adversarial sample by following these steps:

Step 1: select the modified positing, and implement different "score function" to evaluate the importance of a letter or a word to the whole sentence, then select the most important one as the current modification object.
Step 2: Execute the modification strategy: modify the object selected in step one by occlusion or deletion.
Step 3: Repeat steps one and two until the model makes an incorrect prediction, or attack fails when exceeds the maximum edit distance.

Without loss of generality, in this paper, we selected three models in text analysis, Char-CNN, Word-CNN, and Bi-LSTM as our targets. Experiments have confirmed that our attack strategies can effectively attack different models at the character level and word level within a limited edit distance, guide it to make misclassifications. Also, the experimental results show that the effect of the attack is affected by the model structure and the initial prediction probability. In addition to reveal the vulnerability of the Deep Neural Network (DNN) models to the adversarial samples, these conclusions also shed the light of training DNN models more efficiently and more specifically

2 Related Work

2.1 Convolutional Neural Network

Convolutional Neural Network (CNN) **?** is a type of neural network in which convolution layers are included. Convolution layers are intended to find the connectivity feature of the input text. Empirical study reveals that convolutional layers are effective to extract features of the input. Max Pooling layers can be appended at the end of each convolution layer to improve the robustness of the model.

CNN is one of the state-of-the-art models among nowadays text classification models. It can be applied on both character level or word level with word embedding layers involved.

2.2 Long Short-term Memory Network

Long Short-term memory network (LSTM) **?** is a variation of Recurrent Neural Network. The neurons of the network will be recurrently used for an input sequence. The behavior enables the models to extract and gather sequential information of the input.

2.3 Attention

In the normal LSTM model, input sentences are encoded into fixed-length context vectors. However, a key and obvious disadvantage of this kind of fixed-length context vector design are that the system cannot remember long sequences. As the length of the sequence increases, the correlation between the output and the early stage of the sequence becomes weak, resulting in the model unable to summarize all the information contained in the long sequence. The attention mechanism was born to solve this problem **?**. The core idea of the attention mechanism is not only using the final state of each as the input of the next layer but to making use of every RNN unit to construct a vector containing all context information.

2.4 Word Embedding

If an NLP model processes input text from word level, then before processing input texts, words in the text need to be transformed into a series of vectors as tokens. This transformation step is finished by word embedding models so that each word in the text corpus is represented by a unique vector. **?**

2.5 Black-box attack

Our summer research concentrates on current black-box attack strategies. In black-box attacks, it is assumed that the attacker does not have any information about the inner structure of the model to be attacked including the gradient, parameters, and hyperparameter. The only information accessible to the attacker is the probability of different classes after the input is fed.

The black-box attack is a practical setting since nowadays most of the machine learning models are deployed on the server, receiving the inputs from clients and returning predictions. All the parameters and gradients are hidden to the users, making a white-box attack that requires gradients of the input, becomes an impossible task.

Therefore, in the black-box scenario, we implemented several attack strategies that only depend on the output of the model, to generate adversarial input sequences out of the current input sequence while controlling the difference within an imperceptible range.

To conduct black-box attacks and generate an imperceivable adversarial sequence, our method consists of two steps, respectively are,

- First step: Choose a token to attack
- Second step: Choose an approach to modify the token

3 Score Function

Since we are intended to craft an adversarial sequence that can 'fool' the NLP model but not human, we cannot make to many modifications on the input. Hence, we have to select the most significant tokens to attack. To measure the significance of each token in the input sequences. We need to use a scoring function to calculate the significance of the tokens. In our experiment, we use four different kinds of scoring functions proposed by Gao et al **?**. and a new scoring function proposed by us.

3.1 Delete-1 Score

One of the most straightforward ways of measuring the significance for each token is calculating the probability decrease after the token is deleted. Hence we have the formula for the significance of token at position i using the delete-1 scoring function:

$$\mathrm{D1S}\left(x_i\right)=F\left(x_1, x_2, \ldots, x_{i-1}, x_i, \ldots, x_n\right)-F\left(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\right)$$

3.2 Temporal Head Score

RNN models will retain the sequential information of the input sequences, hence the scoring function should reflect these features. Therefore the Temporal Head function is defined as the difference in probability after the model read the i th and $i-1$th token, which can sequentially measure the significance. The mathematical definition of Temporal Head Score (THS) is defined as:

$$\mathrm{THS}\left(x_i\right)=F\left(x_1, x_2, \ldots, x_{i-1}, x_i, \ldots, x_n\right)-F\left(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\right)$$

3.3 Temporal Tail Score

Some variations of RNN models such as Bidirectional LSTM models read the input sequence in both directions, but the Temporal Head function only measures the significance from beginning to end. Hence as the counterpart of the Temporal Head function, the Temporal Tail function is introduced. It is defined as the difference in probability between the two trailing parts of the sequence. Then, the mathematical definition of Temporal Head Score (TTS) is defined as:

$$\mathrm{TTS}\left(x_i\right)=F\left(x_1, x_2, \ldots, x_{i-1}, x_i, \ldots, x_n\right)-F\left(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\right)$$

3.4 Combined Score

Both Temporal Head and Tail consider sequential information in only one direction, hence we can combine the result of them to obtain the significance considering the whole surrounding context. The Combine Score (CS) function is defined as below with λ as a hyperparameter:

$$\mathrm{CS}(x_i)=\mathrm{THS}(x_i)+\lambda\,\mathrm{TTS}(x_i)$$

3.5 Delete-m Score

One of the disadvantages of the delete-1 function is that it only considers the token whose significance to be measured, but the context information is lost. Therefore, we generalize the delete-1 scoring function to delete-m score, which will remove m consecutive tokens starting from the token whose significance to be measured. The delete-m scoring function is defined as below, with m as a hyperparameter:

$$\mathrm{DMS}\left(x_i\right)=F\left(x_1, x_2, \ldots, x_{i-1}, x_i, \ldots, x_n\right)-F\left(x_1, x_2, \ldots, x_{i-1}, x_{i+m-1}, \ldots, x_n\right)$$

4 Modification Function

After selecting the character to attack, we could use different methods to modify the token, including deletion and occlusion, to create imperceivable perturbation to the input. Then we follow the procedure described in Algorithm 1 to conduct experiments, using different scoring functions and modification functions.

Table 1: Different modification functions on character-based model

Original	Occlusion	Deletion
computer science	co puter sci nce	coputer scince

Table 2: Different modification functions on word-based model

Original	Occlusion	Deletion
I love computer science I am from Hong Kong	I computer science I am Hong Kong	I computer science I am Hong Kong

```
Procedure generateAdversarialExample;
Input: input sequence $x$, maximum edit distance $\epsilon$, scoring function Score, modification function Modify;
Result: an adversarial example
initialize scores;
for $i$ in 1..len(x) do
    $scores[i] \leftarrow \mathrm{Score}(x, x[i])$
end
$i = 1$, $scores = 0$;
while The prediction not flip and $cost < \epsilon$ do
    $index \leftarrow$ index of the $i$th largest value in $scores$;
    $cost \leftarrow cost + \mathrm{Modify}(x, x[index])$
end
return $x$;
```

Algorithm 1: Generate Adversarial Examples

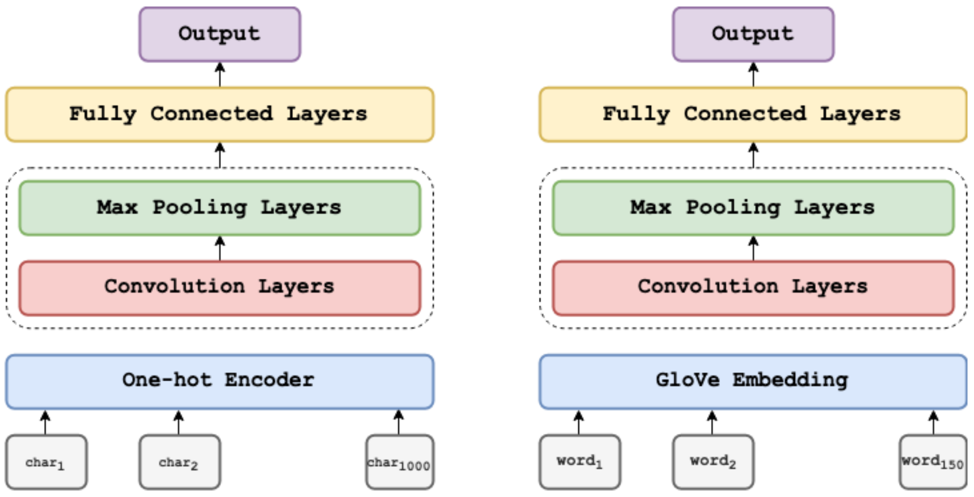
5 Target Models and Datasets

We choose three models to evaluate predefined methods. The character level CNN model is 9 layers with 6 convolutional layers and 3 fully-connected layers, each layer's parameters are shown in Table 1 **?**.

Table 3: specific structure of char-CNN model

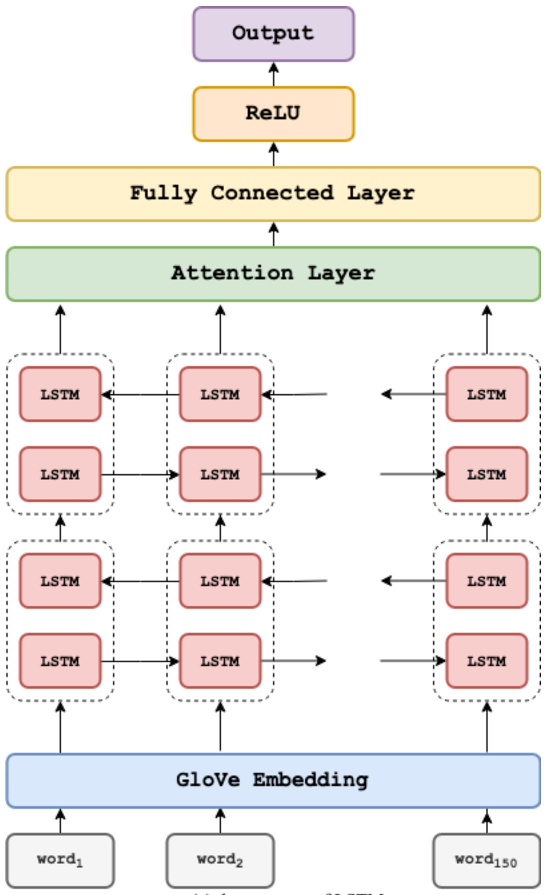
Layer	Index	Feature	Kernel size	Pool	Output
Convolution	1	256	7	3	256
	2	256	7	3	256
	3	256	3	N/A	256
	4	256	3	N/A	256
	5	256	3	N/A	256
	6	256	3	3	256
Fully-Connected	7	1024	N/A	N/A	1024
	8	1024	N/A	N/A	1024
	9	1024	N/A	N/A	2

The word-level CNN model is similar to the character level CNN model, plus an extra word embedding layer.



(a) the structure of char-CNN (b) the structure of word-CNN

The LSTM model is consists of two LSTM layers with hierarchical attention, which is slightly variant of the hierarchical LSTM model build by Zichao Yang etc. All these models are trained on Amazon Review Polarity Dataset,



(c) the structure of LSTM

where class 1 is the negative and class 2 is positive. Each class has 1,800,000 training samples and 200,000 testing samples.

6 Experiments

6.1 Baseline Methods

To evaluate our attack method, we need to compare the attacking results of our methods with another method.

6.1.1 Random Method

In our experiment, given that we are conducting a black-box attack, we decided to choose Random Scoring function as the baseline method, which uses no information of the structures or the outputs of the models. The random method means each time a random token of the input will be selected to be modified.

Since the random method only yields trivial results, only if a method has better performance than the baseline method, it can be considered as a successful attacking method.

6.1.2 Gradient Method

We also use gradient as the scoring function, which uses the inner information of the model and depends on the model implementation. The gradient method calculates the gradient of the loss function with respect to the input and chooses the token with the greatest gradient value. Although it is not a black-box method, we can still use it to evaluate other methods.

6.2 Result of Experiments

The result of experiments reveals all the NLP models we used to have a severe decrease in accuracy under our attack when the edit distance ϵ set to 30, even though the models reach nearly state-of-the-art performance without being attacked. The statistics of the results of the experiments are listed in Table **??**. All the methods we used have better results than the baseline attack method, thus all the methods successfully attack the models. To compare different methods and prove them being effective as ϵ varies, figures of different methods of attack with ϵ ranging from 0 to 30 are summarized in Figure 2.

The performance of the same attacking method varies among different models. Among all the scoring functions we tested, the delete-1 has the best result, which decreases the accuracy of the Char-CNN model from 90.00% down to 6.79%. In other words, around 92.46% of input sequences that can be transformed into adversarial examples with $\epsilon = 30$, and the model become completely untrustworthy. However, the delete-1 scoring function does not report such good results when evading Word-CNN and LSTM models, whose attacked accuracy is respectively 26.97% and 37.26%. Similarly, for other scoring functions, Temporal Head has the greatest accuracy decrease on the Word-CNN model but does not exceed the baseline method much on the Char-CNN model.

	Char-CNN		Word-CNN		LSTM	
	Deletion	Occlusion	Deletion	Occlusion	Deletion	Occlusion
Original	90.00	90.00	91.72	91.72	90.97	90.97
Baseline (Random)	86.33	86.33	79.62	79.62	80.01	80.01
Delete-1	6.79	6.79	26.97	26.97	37.26	37.26
Temporal Head	82.00	82.00	68.20	68.20	73.08	73.08
Temporal Tail	70.11	70.11	77.44	77.44	72.65	72.65
Combined	43.74	43.74	64.31	64.31	63.92	63.92
Delete-m (m = 2)	3.36	3.36	48.72	48.72	55.98	55.98

Table 4: Results of attack. Numbers are the accuracy after attack with $\epsilon = 30$, in percentage

Comparing two modification functions, deletion and occlusion always have identical results. The phenomenon not only occurs in the data of Table 4, but also in the other tables of the experiments. Hence, in other tables, the results of deletion and occlusion will not be listed separately.

6.2.1 The Effects of Delete-m Scoring function with different m

We also attempt to investigate the effects of delete-m scoring function as the value of m changes, hence we draw the figure of curves of different values of m. We found the delete-1 function appears to be the most efficient one when ϵ less than 10, it can decrease the accuracy down to 25%. However, in the long run of the attack on Char-CNN, that is when ϵ increases to 22, the performance of delete-2 will surpass the performance of the delete-1 function. What is behind the delete-1 function is a greedy algorithm, it always chooses the token with the greatest score in this round of choosing. The feature of a greedy algorithm results in a good performance in the first several edits, but there is likely to be a more optimal solution, which is the delete-2 in our case. Nonetheless, the optimal value for m is not easy to find. When edit distance is set to 30, the delete-2 has the best outcome. delete-m with greater m perhaps can give better performance with larger edit distance but it needs to be further verified.

6.2.2 Comparison in robustness across models

In Figure 1a, the curve of Delete-1 Scoring function has a steep slope even when the edit distance is below 10, whereas it is not the case for two word-level models. Besides, as is revealed in Figure 3a, 3b and 3c The boundary of scatter plot of Char-CNN model is closed to a parabola and the boundary of LSTM is closed to a straight line, whereas the one of Word-CNN is in the middle. Each time the word-level models choose a word, the edit distance will increase by the length of the chosen word, which is usually larger than 1 but results in similar drops in probability. It causes the samples to distribute more evenly. Nonetheless, word-level models are still more robust than the Char-CNN model in terms of the decrease in edit distance, since the more samples will become adversarial ones with small changes.

7 Conclusions

This summer, we investigated the robustness of current state-of-the-art NLP models and tested various attacking strategies. We found deletion and occlusion have the same effect on word-based and character-based model. The same attacking strategy may have different effects on different models. Under the same edit distance constraint, word-based models are generally more robust in terms of a decrease in accuracy.

8 Appendix

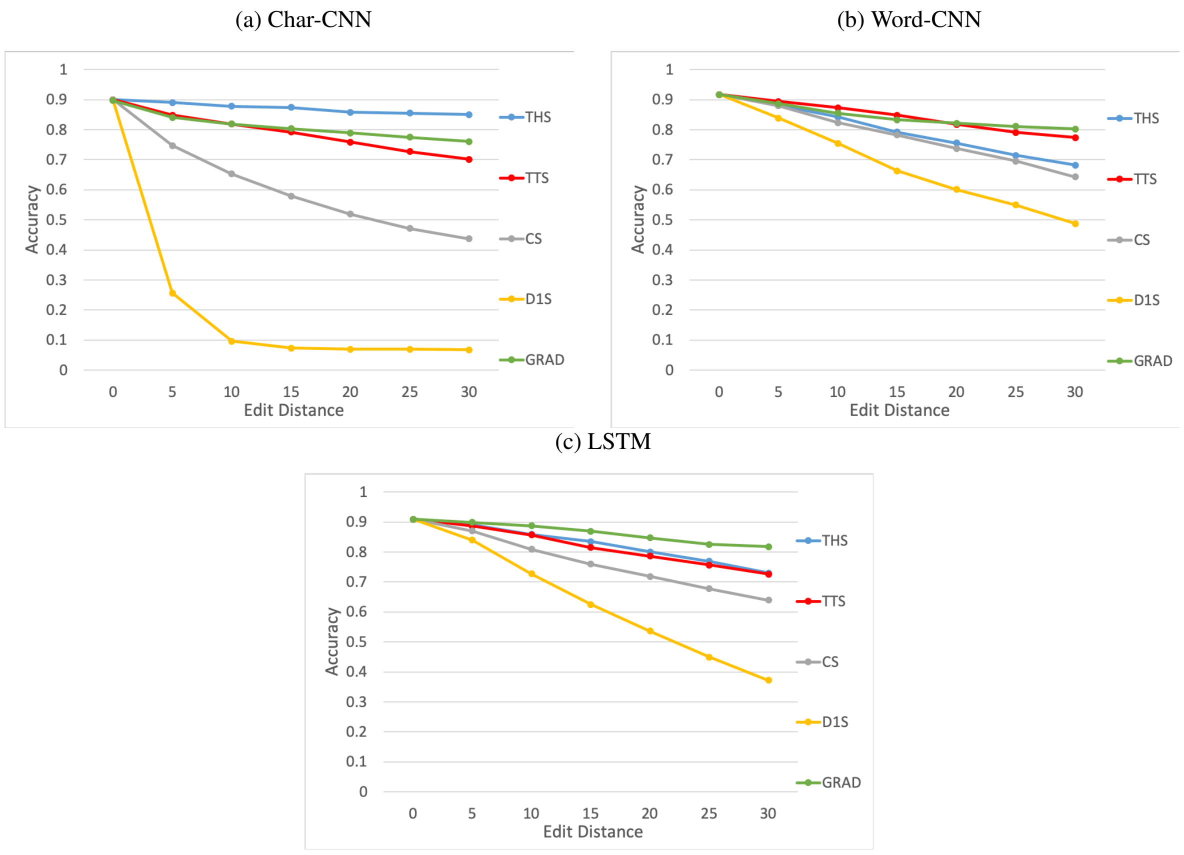


Figure 1: Comparison among Temporal Head, Temporal Tail, Combined and Delete-one Scoring function, with edit distance ϵ ranging from 0 to 30

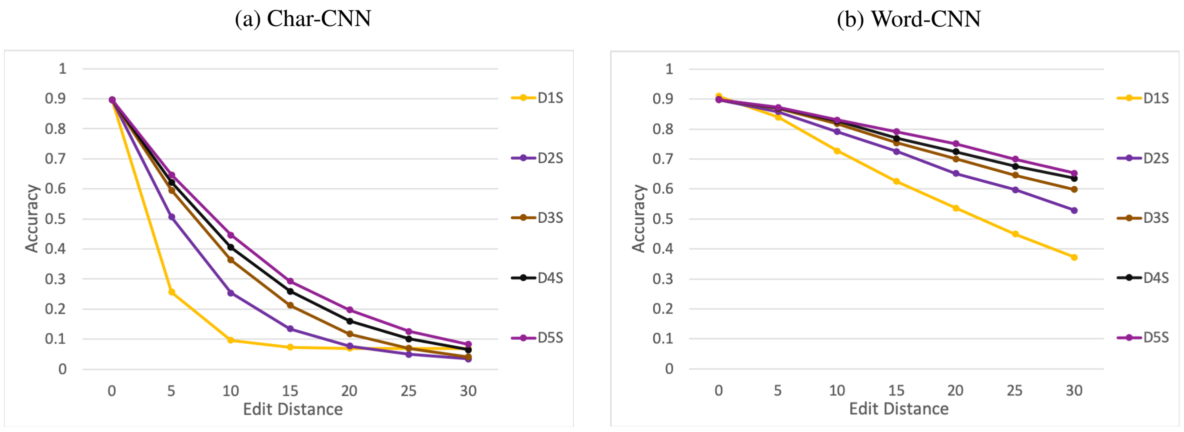
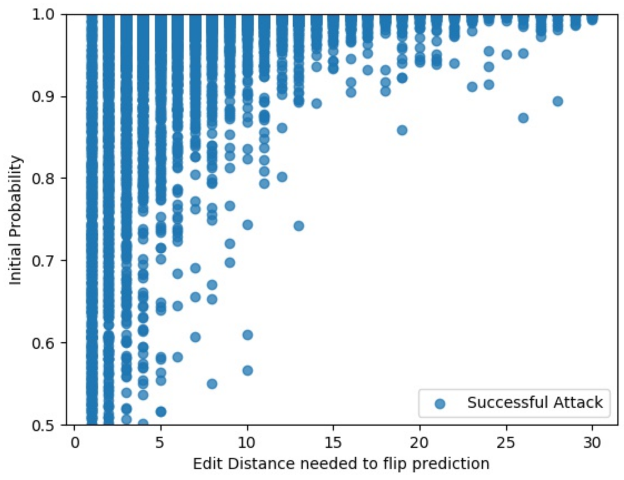
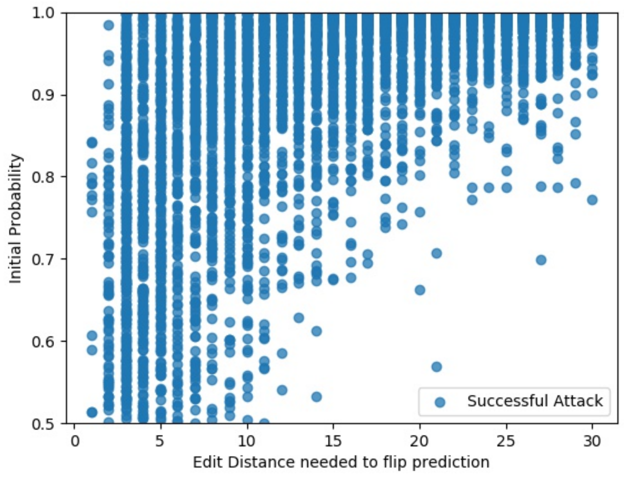


Figure 2: Attack result of Delete-m Scoring function with different m values and different ϵ

(a) char-CNN



(b) word-CNN



(c) LSTM

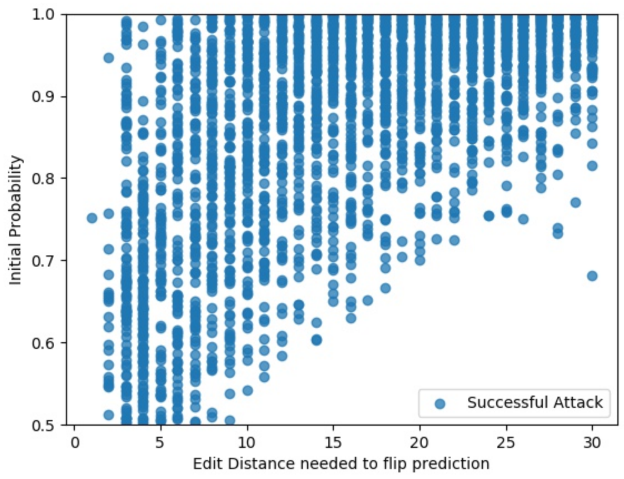
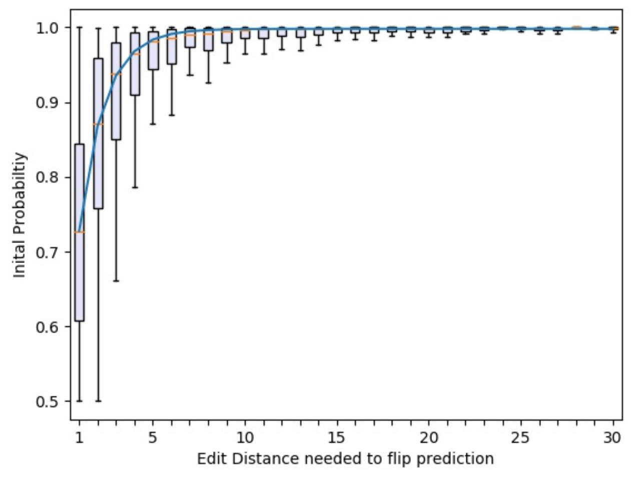
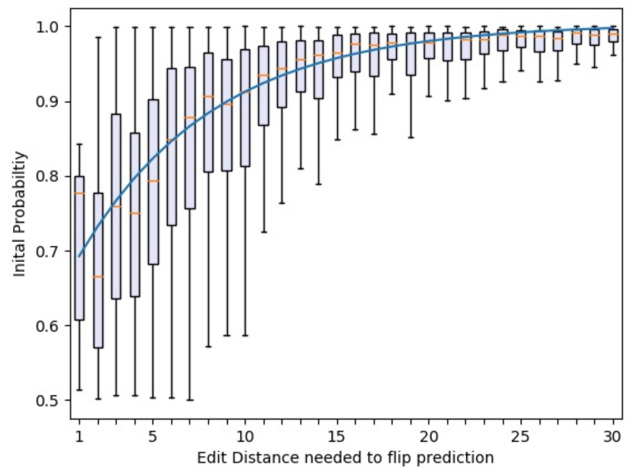


Figure 3: The scatter plots of successful attack samples with initial probability of prediction against edit distance needed to flip the prediction

(a) char-CNN



(b) word-CNN



(c) LSTM

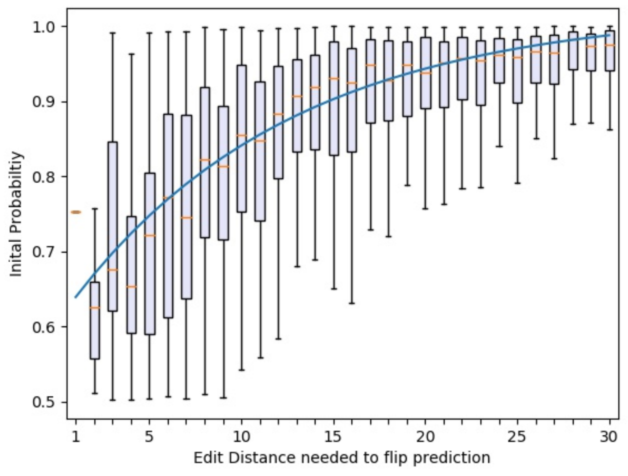


Figure 4: The boxplots of successful attack samples with initial probability of prediction against edit distance needed to flip the prediction