Reinforce Attack: Adversarial Attack against BERT with Reinforcement Learning

Anonymous ACL submission

Abstract

Adversarial attacks against textual data has been drawing increasing attention in both the NLP and security domains. Current successful attack methods for text typically consist of two stages: word importance ranking and word replacement. The first stage is usually achieved by masking each word in the sentence one at a time and obtaining the resulting output probability of the target model. The second stage involves finding synonyms to replace "vulnerable" words by the order of ranking. In this paper, we first explore the effects of employing the model explanation tool LIME to generate word importance ranking, which has the advantage of taking the local information around the word into account to obtain word importance scores. We then propose **Reinforce Attack**, a reinforcement learning (RL) based framework to generate adversarial text. Notably, the attack process is controlled by a reward function rather than heuristics as in previous methods to encourage higher semantic similarity and lower query costs. Through automatic and human evaluations, we show that our *LIME+Reinforce Attack* method achieves better or comparable attack success rate against other state-of-the-art attack frameworks, while the generated samples preserve significantly higher semantic similarity.

1 Introduction

Deep neural networks have dominated computer vision (CV) domain as well as natural language processing (NLP) domain. However, studies (Goodfellow et al., 2014; Biggio et al., 2017; Carlini and Wagner, 2017) have shown that deep learning systems are vulnerable to adversarial examples. Specifically, these adversarial examples are usually generated by adding small perturbations to original samples, which are unperceivable to human while can mislead the neural networks to make wrong prediction. While extensive studies have focused on designing adversarial examples and defenses in CV tasks (Goodfellow et al., 2014; Madry et al., 2018; Xu et al., 2020), textual adversarial examples are less investigated (Li et al., 2020; Zhang et al., 2020; Wallace et al., 2019).

Compared to image data, textual data is discrete and constrained by dictionary, which makes it more difficult to perturb the input. Besides the ability to fool the target model, the perturbed text input should also satisfy three key utility-preserving properties: 1) human prediction consistency: predictions made by human are still the same, 2) semantic similarity: the perturbed example should be semantically similar to the original, and 3) language fluency: generated example should be grammatically correct and fluent. For example, simply replacing the character in words to make them grammatically unrecognizable will result in unnatural sentences.

The state-of-the-art schema (Li et al., 2020) for generating attack samples can be divided into two steps: 1) find vulnerable words and rank them, 2) replace the ranked words one by one to generate adversarial samples. The first stage employs a method called word importance ranking. In the case of attacking BERT (Devlin et al., 2018), they mask all the words in the input sentence one at a time and obtain the corresponding output predictions of the masked sentences from BERT. Then they obtain the word importance rank by the impacts on the output scores. As for the second stage, lexical substitute models (Zhou et al., 2019) are considered to generate adversarial examples. However, there are two significant drawbacks of the above framework: i) the word importance ranking via masking is naive and not explainable, ii) the entire attack process doesn't take semantic similarity into consideration.

In this paper, firstly, we propose to generate word importance ranking via LIME (Ribeiro et al., 2016), a popular tool for explaining machine learning (ML) classifiers. We show that simply switching from masking to LIME can improve the attack performance noticeably on some datasets. Secondly,

to enforce the semantic similarity between adversarial sample and original sample, we introduce an RL-based framework, namely **Reinforce Attack**, for attacking the target model. RL has been previously applied to reading comprehension (Hu et al., 2018), question answering (Liu et al., 2019), and sentence simplification (Zhang and Lapata, 2017). Specifically, we recast the attack process as a sequence tagging problem, where an agent is trained to identify vulnerable words for substitution to maximize the reward function that encourage higher semantic similarity and lower query cost. We conduct extensive experiments on four classification datasets and one regression dataset to demonstrate the effectiveness of our method. The contribution of this paper is threefold:

- We propose to replace the naive word importance ranking method by LIME, which can explain predictions of the target classifiers by learning an interpretable model locally around the prediction.
- We introduce Reinforce Attack, an RL-based method that learns the optimal trade-off among key metrics by maximizing the designed reward function.
- Besides demonstrating superior attack performance on classification tasks, we also extend the adversarial attack to text regression task successfully. To our knowledge, this is the first work to show the generalizability of adversarial attack to regression tasks.

2 Related Work

2.1 Adversarial Attacks in NLP

In NLP, DNNs are widely used in many tasks such as text classification, machine translation, and question answering. However, these DNN-based systems are vulnerable to adversarial attacks. Papernot *et al.* (Papernot et al., 2016) were the first to show that text classifiers can be fooled by adversarial examples, which were generated by simply adding noise to text. Subsequently, more research efforts have been invested in this domain to better understand the adversarial attacks and potential defenses for different tasks, e.g., classification (Li et al., 2019), reading comprehension (Jia and Liang, 2017), natural language inference (Minervini and Riedel, 2018), machine translation (Ebrahimi et al., 2018), question answering (Mudrakarta et al., 2018), argument reading comprehension (Niven and Kao, 2019), and link prediction (Minervini et al., 2017).

More recently, transformer-based models (Devlin et al., 2018) have dominated various tasks in NLP. These models have achieved high generalization power by pre-training on large corpus, outperforming previous state-of-the-art DNNs by only fine-tuning on task dataset. However, the wide application of pre-trained models may also cause serious security issues since one successful adversarial attack can threat all models of the same architecture. Prior successful attack methods (Jin et al., 2020) usually relied on heuristic replacement strategies at the character or word level which makes it challenging to find the optimal solutions in the vast embedding space while simultaneously preserving semantic consistency and language fluency. Li *et al.* (Li et al., 2020) proposed BERT-Attack, an effective method to generate adversarial examples using BERT. This attack outperforms the prior methods in terms of both success rate and perturbation rate.

Compared with the aforementioned methods, we explore the effects of employing LIME to generate word importance ranking. Moreover, we propose the RL-based Reinforce Attack framework, which recasts the attack process as a sequence tagging problem. Unlike heuristics based strategies, our method is trained to maximize the reward function, which represents the trade-off among key metrics.

2.2 RL for NLP

There exist several work that apply RL to NLP tasks. Zhang *et al.* (Zhang and Lapata, 2017) explored the space of possible simplifications of sentences while learning to optimize a reward function that encourages outputs which are simple, fluent, and preserve the meaning of the input. Liu *et al.* (Liu et al., 2019) combined Seq2Seq model with deep reinforcement learning, defining a sequence generator by optimizing a combination of imposed reward functions. Moreover, Ammanabrolu *et al.* (Ammanabrolu et al., 2020) introduced Q*BERT, an agent that learns to build a knowledge graph of the world by answering questions, which leads to greater sample efficiency.

3 Methodology

In this section, we first elaborate on the process of generating the adversarial example. Then we proceed to the details of our Reinforce Attack framework.

3.1 Explanatory Model

Recently, researchers are increasingly interested in explaining how ML classifiers (or models) work since ML models have achieved remarkable performances in many areas, *e.g.*, security, education, and economy. LIME (Ribeiro et al., 2016) is an explanatory model that can explain any black-box classifier with two or more classes by inputting text, table, or image. Specifically, for a large-scale pre-trained language model (e.g. BERT), given a function that takes in text and outputs a logit probability for each class, LIME can explain the model by presenting individual representative predictions. Our key idea is that the explanations of LIME can be leveraged to identify vulnerable words for adversarial attack. Instead of considering each word one by one as in previous work for finding vulnerable words (Li et al., 2020; Jin et al., 2020), LIME utilizes words around the chosen word, by obtaining the prediction of the target model on these perturbed words and learning a linear model that approximates the model in the vicinity of the chosen one. Then we follow LIME (Ribeiro et al., 2016) and use the fidelity functions and complexity measures to get the importance value of the chosen word. Hence, LIME considers local information around the chosen word, not only a single word.

3.2 LIME + BERT-Attack

Our LIME-based method to generate adversarial examples consists of two stages: (i) important words selection and (ii) word replacement.

3.2.1 Important Words Selection

We first pre-process the text and feed it into LIME to obtain the important words. Specifically, we construct a function that takes in text as input and calls the target BERT model to generate logit probability as output. Then LIME employs the constructed function to predict the importance of all the words.

Note that there are no repeating words in the output ranked list of the words. Then we can select the first q words in the rank list as the important words. In our experiment, q is simply set to n, the length of input text.

3.2.2 Word Replacement

After we acquire the list of the important words, we use a word replacement strategy similar to (Li et al., 2020) to replace the words and use Algorithm 1 to generate the adversarial examples. To replace the words, we rely on the large-scale pre-trained model, *i.e.*, BERT, which can make the generated sentences more fluent, grammar-correct, and context-aware (Li et al., 2020) compared to rule-based substitution methods (Ren et al., 2019; Jin et al., 2020). Besides, the replacement process needs only one forward pass, which is more efficient and does not scoring and constraining the perturbations.

Algorithm 1 Adversarial examples generation

Input: $S = [w_0, w_1, ..., w_n]$ //Input sentence
$Y \leftarrow$ ground truth label
$l \leftarrow 0.25 \times n$ //Maximum number of word substitutions
$LIME(\cdot) : S \rightarrow [w_i, ...]$ //The length of $[w_i, ...]$ is q
$Logit(\cdot) : S \rightarrow \mathbb{R}^C$ //C is the number of classes
Output: S_{adv} //Adversarial example
$I = [w_i, ...] \leftarrow LIME(S)$ //q important words in descending order
$P^{\in q \times K}$ = top-K candidates for all words in I using BERT
$n_s = 0$ //Number of substituted words
for w_j in I **do**
  if $n_s > l$ **then**
    return False //Fail to generate adversarial example
  else
    for P_k^j in P^j **do**
      $S' = [w_0, w_1, ..., w_{j-1}, P_k^j ...]$
      if $argmax(Logit(S')) != Y$ **then**
        return $S_{adv} = S'$ //Attack successful
      else
        if $argmax(Logit(S')) < argmax(Logit(S_{adv}))$ **then**
          $S_{adv} = S'$ //Update S_{adv}
          $n_s += 1$
        end if
      end if
    end for
  end if
end for

We employ Algorithm 1 to generate the modified text that can fool BERT. Specifically, this algorithm takes as input the pre-processed text S and the ground truth label Y. This pre-processed text S is then fed into $LIME(\cdot)$ whose output is a ranked word list I. For each important word $w_j \in I$, we leverage BERT to identify the top K replacement candidates' list P^j. Let P be the list of all such P^js. For every candidate in P, we filter P^j by a set of stop words. Then, we replace w_j with the mask token ($[\_mk\_]$). The adversarial sentence S' is obtained by replacing the mask token with the corresponding candidate from top-K candidates, subsequently. We measure the logit probability of S' by feeding it into the target model.

If the predicted class is not the ground truth label Y, a successful adversarial sample is generated and the algorithm terminates. Otherwise, the algorithm compares the max values of the logit probabilities for S' and S_{adv}. If $Logit(S')$ is smaller, we update S_{adv} with the contents of S'. The S_{adv} is initialized with S. The intuition behind this procedure is that the smaller the maximum value of logit probability

is, the more likely the target model will predict the wrong label. The output of our algorithm is the generated adversarial example, S_{adv}. The algorithm replaces l words at maximum. If the number of the replaced words n_s is more than l, the algorithm will consider that the text cannot be converted to an adversarial example. In our experiment, the value of l is set to $25\% \times n$. By this constraint, we preserve the semantic information of the original text while keeping the adversarial attack effective. It can be considered as a trade-off between the attack success rate and semantic preservation of the input text.

3.3 Reinforce Attack

Although the attack method described in Section 3.2 outperforms the state-of-the-art attacks, it does not optimize the trade-offs among key metrics, including success rate, query number, perturbation rate, and semantic similarity during the attack process. Therefore, we propose a new attack dubbed as **Reinforce Attack**, which is an RL-based framework to optimize such trade-offs by maximizing a reward function composed of the above metrics.

3.3.1 Framework

Our Reinforce Attack framework is illustrated in Figure 1. As mentioned earlier, in this attack, our key idea is to formulate the adversarial attack as a sequence tagging task. There are only two labels: 0 and 1 for the tagging. Firstly, we vectorize the input sentence using Glove embedding (Pennington et al., 2014), a powerful word vector technique that leverages both global and local statistics of a corpus. Let X represent the vectorized sentence:

$$X = x_1, x_2, ... \tag{1}$$

Secondly, we apply LIME to explain the classification result of the input. The explanation is then normalized and reused as attention scores:

$$\alpha = \alpha_1, \alpha_2, ... \tag{2}$$

$$\alpha_i = \frac{LIME[i]}{max(abs(LIME))} \tag{3}$$

Thirdly, the input z_i for the Agent is computed by:

$$z_i = (\alpha_i * x_i) \bigoplus history\_actions \tag{4}$$

where $\bigoplus$ represents concatenation operation. As shown, we inject the information of LIME into our Reinforce Attack by input attention mechanism. Fourthly, if the action predicted by Agent is 0, it will skip current word and move to the next. Otherwise, it will get into attack process, which is guided by the designed reward function. Finally, the Agent is updated by the reward.

3.3.2 Reward Function

Our reward function takes into consideration all the key metrics mentioned earlier in this section.

$$r(S') = \lambda^A r^A - \lambda^Q r^Q - \lambda^P r^P + \lambda^S r^S \tag{5}$$

Where $\lambda^A, \lambda^Q, \lambda^P, \lambda^S \in [0, \infty)$, S' is the adversarial sentence, r^A, r^Q, r^P and r^S correspond to reward of attack success, query number, perturbation rate and semantic similarity, respectively.

Attack Success: The success rate is the main metric to evaluate the performance of adversarial attack. Therefore, we consider attack success as the fundamental component of the reward function. As for the actual attack algorithm, we reuse the attack process (lines 8-18) in Algorithm 1.

$$r^A = max(p_{ori} - p_{adv}, 0) \tag{6}$$

where p_{ori} is the original probability and p_{adv} is the probability of adversarial sample.

Query Number: Query Number reflects the efficiency of the attack model. While the attack reward r^A tries to encourage the model to generate misleading samples, the query reward r^Q ensures that the attack success is not achieved at the cost of high number of queries. Besides, restricting the query number can also force the system to find more vulnerable words for replacement.

$$r^Q = \frac{Q}{n} \tag{7}$$

where Q is the number of queries and n is the length of sentence.

Perturbation Rate: The ratio of perturbed words to the text length is an important metric to evaluate semantic similarity. We expect the attack model to achieve success while replacing minimal number of words. The reward r^P simply calculates the perturbation rate to regularize the reward function.

$$r^P = \frac{P}{n} \tag{8}$$

where P is the number of perturbed words and n is the length of sentence.

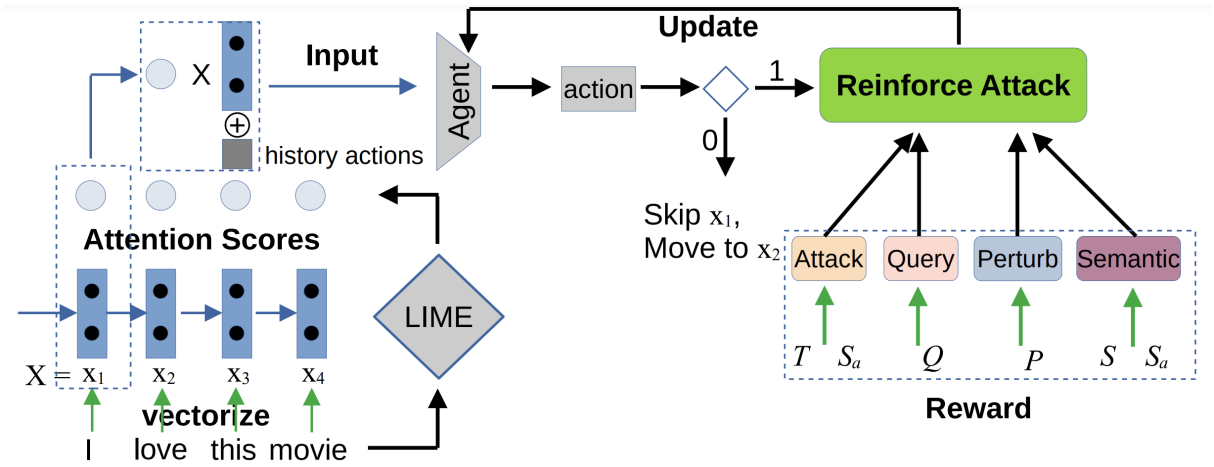

Figure 1: Reinforce Attack Framework. T is the target model, S and S_a are original and adversarial sentence respectively, Q and P are query number and perturbation rate respectively.

Semantic Similarity: Finally, we consider the Universal Sentence Encoder (USE) (Cer et al., 2018) as another metric to evaluate semantic similarity directly. It is a BERT-based encoder, which is widely used to calculate similarity between a pair of texts. r^S represents the output score of USE.

$$r^S = USE(S, S_{adv}) \tag{9}$$

where S and S_{adv} are the original and adversarial sentences, respectively.

3.3.3 Learning

Agent: We have designed a simple MLP agent to identify vulnerable words to attack. As defined in equation (4), z_i represents the input of the agent.

$$a_i = MLP(z_i) \tag{10}$$

Here, a_i is the predicted action. The MLP has two hidden layers with 512 and 256 neurons, respectively.

Policy: We employed deep Q-learning (Van Hasselt et al., 2016) to train the agent. The agent interacts with an environment through a sequence of observations, actions and rewards. The goal of the agent is to select optimal actions so that future reward is maximized.

$$Q^*(s,a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... | s_t, a_t, \pi] \tag{11}$$

where $Q^*(s,a)$ is the maximum sum of rewards r_t decayed by γ at each time step t, which relies on the policy $\pi = P(a|s)$ with the observation s_t and the action a_t. During training, the samples (or minibatches) of $(s, a, r, s') \sim U(D)$ are drawn uniformly at random from the pool of stored samples. The Q-learning update at iteration i uses the following loss function:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}[(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i))^2] \tag{12}$$

where γ is the discount factor that determines the horizon of the agent, a' and s' are the target action and state, respectively, θ_i are the parameters of the Q-network at iteration i, and θ_i^- are the network parameters to compute the target at iteration i.

4 Evaluation

In this section, we illustrate the experiment setup and the experimental results. Firstly, we introduce the datasets for classification and regression tasks followed by experiment setup. Secondly, we discuss the results of our experiments. Finally, we evaluate the generated samples of our method by human evaluation.

4.1 Dataset Description

We apply our method to both classification and regression tasks. For classification, we follow the configuration in (Li et al., 2020) to test on 1000 samples, which are the same splits used by (Jin et al., 2020). As for regression, we split a subset of 1000 random samples from the dataset for testing.

4.1.1 Text Classification

We consider four different types of classification datasets as in (Li et al., 2020).

- **Yelp** Review Dataset is constructed by considering both negative (stars 1 & 2) and positive

Table 1: Comparison with existing work.

Classification	Method	Original Acc	Avg Len	Attacked Acc	Perturb %	Query	Semantic Sim
IMDB	GA (Alzantot et al., 2018)	90.9	215	45.7	4.9	6493	-
	TextFooler (Jin et al., 2020)			13.6	6.1	1134	0.86
	BERT-Attack (Li et al., 2020)			11.4	4.4	454	0.86
	LIME + BERT-Attack*			4.1	**3.0**	527	0.80
	LIME + Reinforce Attack*			**1.9**	3.3	**367**	**0.97**
Yelp	GA (Alzantot et al., 2018)	95.6	157	31.0	10.1	6137	-
	TextFooler (Jin et al., 2020)			6.6	12.8	743	0.74
	BERT-Attack (Li et al., 2020)			**5.1**	**4.1**	**273**	0.77
	LIME + BERT-Attack*			11.1	4.7	352	0.46
	LIME + Reinforce Attack*			6.2	10.8	360	**0.96**
Fake	GA (Alzantot et al., 2018)	97.8	885	58.3	1.1	28508	-
	TextFooler (Jin et al., 2020)			19.3	11.7	4403	0.76
	BERT-Attack (Li et al., 2020)			15.5	1.1	1558	0.81
	LIME + BERT-Attack*			6.0	**4.0**	632	0.65
	LIME + Reinforce Attack*			**2.6**	4.4	**549**	**0.98**
AG	GA (Alzantot et al., 2018)	94.2	43	51.0	16.9	3495	-
	TextFooler (Jin et al., 2020)			12.5	22.0	357	0.57
	BERT-Attack (Li et al., 2020)			**10.6**	15.4	213	0.63
	LIME + BERT-Attack*			16.2	18.3	330	0.81
	LIME + Reinforce Attack*			15.0	**15.1**	**210**	**0.94**
Regression	Method	Original MAE	Avg Len	Attacked MAE	Perturb %	Query	Semantic Sim
Blog	BERT-Attack (Li et al., 2020)	6.5	195	10.5	**2.0**	**150**	0.95
	Reinforce Attack*	-		**14.0**	3.9	199	**0.97**

(stars 3 & 4) reviews. We follow the steps in (Zhang et al., 2015) to perform binary classification task.

- **IMDB** Movie Review Dataset[1] consists of both negative and positive reviews. We perform a binary classification task here as well.

- **AG's News** Dataset[2] contains news articles of four different topics, namely World, Sports, Business and Sci/Tech. We process it into a four-class classification task.

- **FAKE** News Dataset is from Kaggle Fake News Challenge[3], which aims to identify unreliable news articles.

[1]https://datasets.imdbws.com/

[2]https://www.kaggle.com/amananandrai/ag-news-classification-dataset

[3]https://www.kaggle.com/c/fake-news/data

4.1.2 Text Regression

Blog Authorship Corpus consists of the collected posts of 19,320 bloggers gathered from blogger.com in August 2004. Each blog is labelled with blogger's self-provided gender, age, industry and astrological sign. As in (Santosh et al., 2013), we perform age prediction based on the text. The ages of the bloggers range from 13 to 48.

4.2 Setup of Automatic Evaluation

To measure the quality of the generated samples comprehensively, we set up extensive automatic evaluation metrics as in (Li et al., 2020). The attacked accuracy, which is the accuracy of target model on adversarial samples, is the core metric measuring the effectiveness of the attack model. Besides, the perturbation rate is also vital since less perturbation usually means more semantic consistency. Furthermore, the query number per sample is a key metric, which reflects the efficiency of the

Table 2: Human evaluation results.

Dataset		Accuracy	Semantic	Grammar
IMDB	Original	0.86	1	3.39
	BERT-Attack (Li et al., 2020)	0.21	0.82	3.24
	LIME + BERT-Attack*	0.25	0.88	3.44
	LIME + Reinforce Attack*	0.23	0.87	3.31
Blog	Original	-	1	3.51
	Reinforce Attack*	-	0.80	3.09

Table 3: The ablation study on reward function

Dataset	Reward	Attacked Acc	Perturb %	Query	Semantic Sim
IMDB	r^A	10.7	7.3	385	0.90
	$r^A + r^Q$	7.0	6.4	**267**	0.96
	$r^A + r^P$	**1.4**	**3.2**	272	0.95
	$r^A + r^S$	2.0	4.3	309	**0.98**
	Combined	1.9	3.3	367	0.97

attack model. Finally, we also use Universal Sentence Encoder (Cer et al., 2018) to measure the semantic similarity between original sentence and adversarial sample.

4.3 Hyperparameters

For the BERT-Attack framework, we reuse the configuration in (Li et al., 2020). As for the reward function, grid search is performed to find the best weights. For λ^S, the candidates are $[0.001, 0.01, 0.1, 1.0]$ while for other weights, the range is set to be $[0.5, 1.0, 1.5, 2.0]$. Eventually, $\lambda^Q, \lambda^P, \lambda^A$, and λ^S are set to be $1.5, 1.0, 2.0$, and 0.01, respectively, throughout all the experiments.

4.4 Experiment Results

We compare our method with three existing work: GA (Alzantot et al., 2018), TextFooler (Jin et al., 2020), and BERT-Attack (Li et al., 2020). In this section, the target model is BERT-base, and the attack model is also BERT-base.

Classification: As shown in Table 1, both our LIME + BERT-Attack and LIME + Reinforce-Attack outperform other methods on IMDB and Fake datasets by a distinctive margin. However, LIME didn't improve the success rate of attack models on AG and Yelp datasets, whose average lengths are relatively smaller. This suggests that the fidelity of LIME is discounted when sentences are shorter. More analysis is provided in appendix B.

Specifically, our Reinforce Attack achieves an average attacked accuracy of about 6.4%, which is a significant improvement compared with BERT-Attack (10.7%) and LIME + BERT-Attack (9.4%). Moreover, Reinforce Attack consistently outperforms other methods in terms of semantic similarity by a large margin. The semantic similarity reward r^S in Reinforce Attack plays a vital role in maintaining high semantic consistency throughout the attack process.

Regression: As for the regression dataset, LIME can not be applied due to incompatibility. Therefore we only compare the vanilla BERT-Attack and our Reinforce Attack.

Reinforce Attack achieves an attacked MAE of 14.0, outperforming BERT-Attack by 33%. It is also noticeable that Reninforce Attack still maintains an impressive semantic similarity of 0.97 with slightly higher perturbation rate and query number. This shows the advantage of our Reinforce Attack, which is controlled by the reward function to balance between the success rate and other metrics.

4.5 Human Evaluation

We perform human evaluation to further evaluate the generated adversarial examples via Amazon Turk. Specifically, we run three experiments to measure the classification consistency, grammaticality, and semantic similarity, where three annotators annotate each data point. We use the IMDB dataset and Blog dataset for classification and regression tasks respectively. We select 50 original samples, 50 corresponding adversarial samples generated by BERT-Attack, and 50 samples generated by our methods for each dataset. Firstly, we mix samples and ask human judges to classify the sentiment of IMDB data for all four types of sentences. Secondly, we ask the annotators to rate the grammaticality of the sentences from 1 to 5 (5 being the best), following (Li et al., 2020). Finally, we ask the annotators to compare the semantic similarity of reference sentences with those generated by the attack methods for both the IMDB and Blog datasets. The scale is 0 to 1, where 1 is similar, 0 is dissimilar and 0.5 is the middle, following (Jin et al., 2020). As shown in Table 2, both our LIME + BERT-Attack and LIME + Reinforce Attack outperform the vanilla BERT-Attack on all three dimensions.

5 Discussions

5.1 Effects of LIME

As shown in Table 1, replacing the original word importance ranking method with LIME resulted in distinctive improvements on IMDB and Fake

datasets while LIME's performances on Yelp and AG with shorter sentences were less stable.

Some theoretical study (Slack et al., 2020; Garreau and von Luxburg, 2020) found that perturbation-based interpretability methods like LIME and SHAP are not robust enough. Their performances may vary from dataset to dataset. Therefore, more research effort is needed to improving the stability of LIME.

5.2 Importance of Reward Components

We also conduct ablation study of reward function on IMDB dataset. As shown in Table 3, we tested different combinations of the reward components (i.e. r^A, r^Q, r^P, r^S) to demonstrate the corresponding effects on the attack evaluation metrics.

The base reward is r^A whose performance can be viewed as a lower bound. The effect of adding other rewards to r^A is distinctive. More specifically, $r^A + r^Q$ reduces the query number by more than 100, reaching the lowest at 267. $r^A + r^P$ outperforms all other candidates in terms of attacked accuracy and perturbation rate. $r^A + r^S$ attains the best semantic similarity of 0.98. Intuitively, the impacts of the reward components are consistent with our expectations.

Moreover, the combination of all rewards reached a satisfying trade-off among these evaluation metrics. Different results can be obtained by simply manipulating the weights of each reward.

5.3 Transferability of Reinforce Attack

We validate the transferability of the adversarial examples generated by our Reinforce Attack. For this, we collect the adversarial samples generated for IMDB and Blog datasets to attack other target models. The criterion of being a successfully transferable adversarial example for IMDB is simply fooling the other target model while for Blog we set a threshold for the increase of MAE (7.0, about 25% of the age range) between predicted age and true age. As shown in table 4, there exists noticeable transferability among models on IMDB dataset. However, the adversarial samples are less transferable on Blog dataset. Note that our Reinforce Attack exhibits more distinctive transferability on classification dataset compared to previous methods (Li et al., 2020; Jin et al., 2020).

5.4 Runtime Comparison

We use the AWS P3.2xlarge machine with 8 Intel Xeon CPUs and 1 Nvidia Tesla v100 GPU (the configuration is different from that of (Li et al., 2020)). The runtime analysis is shown in Table 5. Since LIME takes slightly more time than the original method to calculate word ranking, our LIME + BERT-Attack is slower than BERT-Attack. Moreover, Reinforce Attack requires to calculate semantic similarity during the generation process, which is time-consuming.

Table 4: Transferability of adversarial examples on IMDB and Blog dataset. Row and column stand for attack model and tested model respectively.

Dataset	Model	BERT-base	DistilBERT	Albert
IMDB	BERT-base	0	0.63	0.51
	DistilBERT	0.64	0	0.50
	Albert	0.48	0.50	0
Dataset	Model	BERT-base	DistilBERT	Albert
Blog	BERT-base	14.0	6.3	5.4
	DistilBERT	5.5	11.0	5.6
	Albert	6.3	5.6	15.8

Table 5: Runtime Comparison

Dataset	Method	Runtime(s/sample)
IMDB	BERT-Attack (Li et al., 2020)	86
	LIME+BERT-Attack*	130
	LIME+ Reinforce Attack*	179

6 Conclusion

In this paper, we first study the effects of replacing naive word importance ranking method with the model explanation tool, LIME, for text adversarial attack. Experimental results suggest that although the performance of LIME is not stable, it still generates more accurate ranking in some cases, which leads to significantly higher success rate. Our empirical analysis shows that the fidelity of LIME is discounted when sentences are shorter.

Furthermore, we propose **Reinforce Attack**, an RL-based attack schema to generate adversarial texts. Unlike the previous methods which mostly rely on heuristics to constrain the generated samples, our method is guided by a carefully designed reward function to optimize the trade-off among key metrics, including success rate, perturbation rate, query number, and semantic similarity. Extensive experiments, including automatic and human evaluations, show that Reinforce Attack maintains distinctively higher semantic similarity through all the datasets in our experiment, while achieving comparable or better success rate against other methods.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*.

P. Ammanabrolu, E. Tien, M. Hausknecht, and M. Riedl. 2020. How to avoid being eaten by a grue: Structured exploration strategies for textual worlds. In *Under Review*.

B. Biggio, I. Corona1, and D. Maiorca. 2017. Evasion attacks against machine learning at test time. In *ECML*.

Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy*.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. In *arXiv:1803.11175*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018. On adversarial examples for character-level neural machine translation. In *27th International Conference on Computational Linguistics*.

Damien Garreau and Ulrike von Luxburg. 2020. Explaining the explainer: A first theoretical analysis of lime. In *AISTATS*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *IJCAI*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *2017 conference on empirical methods in natural language processing*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *Network and Distributed System Security Symposium*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202.

Y. Liu, C. Zhang, X. Yan, and P. Yu. 2019. Generative question refinement with deep reinforcement learning in retrieval-based qa system. In *CIKM*.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *ICLR*.

Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2017. Adversarial sets for regularising neural link predictors. In *the 33rd Conference on Uncertainty in Artificial Intelligence*.

Pasquale Minervini and Sebastian Riedel. 2018. Adversarially regularising neural nli models to integrate logical background knowledge. In *SIGNLL Conference on Computational Natural Language Learning*.

Pramod Kaushik Mudrakarta, Ankur Taly, Mukund Sundararajan, and Kedar Dhamdhere. 2018. Did the model understand the question? In *56th Annual Meeting of the Association for Computational Linguistics*.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *57th Annual Meeting of the Association for Computational Linguistics*.

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

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

K Santosh, Romil Bansal, Mihir Shekhar, and Vasudeva Varma. 2013. Author profiling: Predicting age and gender from blogs. In *CLEF*.

Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. 2020. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *AIES*.

Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *EMNLP*.

Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. 2020. Adversarial attacks and defenses in images, graphs and text: A review.

Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey.

X. Zhang and M. Lapata. 2017. Sentence simplification with deep reinforcement learning. In *EMNLP*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

W Zhou, T. Ge, K. Xu, F. Wei, and M. Zhou. 2019. Bert-based lexical substitution. In *ACL*.

A Qualitative Results

To illustrate qualitative results, we show four examples in Figure. 2: two examples from the IMDB dataset and two examples from the Blog dataset. The text in the green box is the original text; the blue box text is the sentences generated by our method. The red words in the green box and blue box denote the important words, mask token, and substitutes. We can find that our method can generate adversarial examples by changing few words and without changing much semantic meanings for both classification task and regression task. For example, in the top-left example of Figure. 2, the attack model only replaces one word "awful" into "damned". In the bottom-right example of figure. 2, the word "everyone" is replaced by "all" and "send" is replaced by "fire". The semantic similarity is high for the adversarial example and the original text.

B Accuracy and sentence length

As shown in Figure. 3, when the length of sentences is greater, the attacked accuracy of LIME+ Reinforce Attack is lower for IMDB, Yelp, Fake datasets. However, when the length of sentence is less than 140, the performance of our method is not stable for AG dataset. One possible reason is that the AG is four-classes classification task which is different to other three two-classes classification task. Hence, when the sentence length is greater, the performance of our method is better.

Figure 2: Four examples of generated adversarial examples. The left two examples are from the IMDB dataset, and the right two examples are from the Blog dataset. The original text is in the green box, and the text adversarial examples generated by LIME + Reinforce Attack is in the blue box. The red words in the green box, blue box denote the vulnerable words and substitutes, respectively.

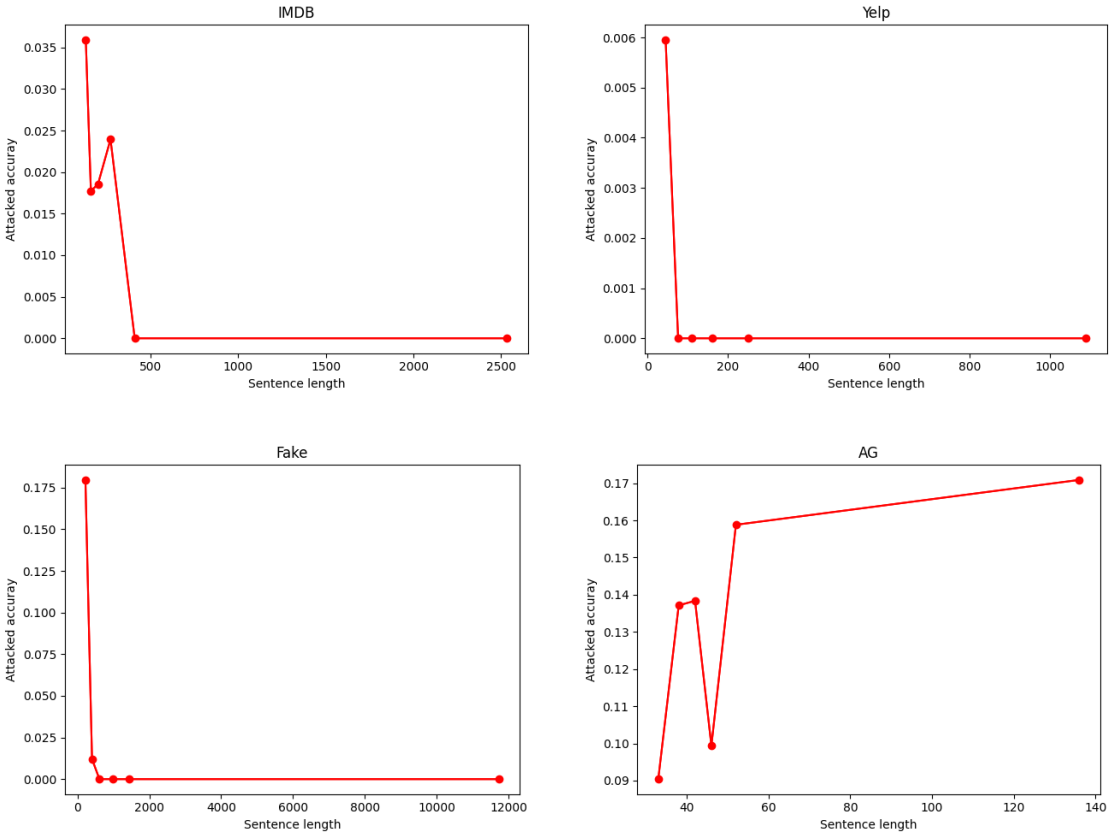

Figure 3: The figures of the relationships between the attacked accuracy of LIME+ Reinforce Attack and the length of the sentence for four classification datasets.