Generating Black-Box Adversarial Examples for Text Classifiers Using a Deep Reinforced Model

Prashanth Vijayaraghavan ✉ and Deb Roy

MIT Media Lab, Cambridge MA 02139, USA
{pralav,dkroy}@media.mit.edu

Abstract. Recently, generating adversarial examples has become an important means of measuring robustness of a deep learning model. Adversarial examples help us identify the susceptibilities of the model and further counter those vulnerabilities by applying adversarial training techniques. In natural language domain, small perturbations in the form of misspellings or paraphrases can drastically change the semantics of the text. We propose a reinforcement learning based approach towards generating adversarial examples in black-box settings. We demonstrate that our method is able to fool well-trained models for (a) IMDB sentiment classification task and (b) AG's news corpus news categorization task with significantly high success rates. We find that the adversarial examples generated are semantics-preserving perturbations to the original text.

Keywords: Natural Language Processing · Adversarial Examples · Black-box models · Reinforcement Learning.

1 Introduction

Adversarial examples are generally minimal perturbations applied to the input data in an effort to expose the regions of the input space where a trained model performs poorly. Prior works [5, 36] have demonstrated the ability of an adversary to evade state-of-the-art classifiers by carefully crafting attack examples which can be even imperceptible to humans. Following such approaches, there has been a number of techniques aimed at generating adversarial examples [29, 41]. Depending on the degree of access to the target model, an adversary may operate in one of the two different settings: (a) black-box setting, where an adversary doesn't have access to target model's internal architecture or its parameters, (b) white-box setting, where an adversary has access to the target model, its parameters, and input feature representations. In both these settings, the adversary cannot alter the training data or the target model itself. Depending on the purpose of the adversary, adversarial attacks can be categorized as (a) targeted attack and (b) non-targeted attack. In a targeted attack, the output category of a generated example is intentionally controlled to a specific target category with limited change in semantic information. While a non-targeted attack doesn't care about the category of misclassified results.

Most of the prior work has focused on image classification models where adversarial examples are obtained by introducing imperceptible changes to pixel values through optimization techniques [22, 15]. However, generating natural language adversarial examples can be challenging mainly due to the discrete nature of text samples. Continuous data like image or speech is much more tolerant to perturbations compared to text [13]. In textual domain, even a small perturbation is clearly perceptible and can completely change the semantics of the text. Another challenge for generating adversarial examples relates to identifying salient areas of the text where a perturbation can be applied successfully to fool the target classifier. In addition to fooling the target classifier, the adversary is designed with different constraints depending on the task and its motivations [11]. In our work, we focus on constraining our adversary to craft examples with semantic preservation and minimum perturbations to the input text.

Given different settings of the adversary, there are other works that have designed attacks in "gray-box" settings [6, 14, 30]. However, the definitions of "gray-box" attacks are quite different in each of these approaches. In this paper, we focus on "black-box" setting where we assume that the adversary possesses a limited set of labeled data, which is different from the target's training data, and also has an oracle access to the system, i.e., one can query the target classifier with any input and get its corresponding predictions. We propose an effective technique to generate adversarial examples in a black-box setting. We develop an Adversarial Example Generator (AEG) model that uses a reinforcement learning framing to generate adversarial examples. We evaluate our models using a word-based [20] and character-based [42] text classification model on benchmark classification tasks: sentiment classification and news categorization. The adversarial sequences generated are able to effectively fool the classifiers without changing the semantics of the text. Our contributions are as follows:

- We propose a black-box non-targeted attack strategy by combining ideas of substitute network and adversarial example generation. We formulate it as a reinforcement learning task.
- We introduce an encoder-decoder that operates over words and characters of an input text and empowers the model to introduce word and character-level perturbations.
- We adopt a self-critical sequence training technique to train our model to generate examples that can fool or increase the probability of misclassification in text classifiers.
- We evaluate our models on two different datasets associated with two different tasks: IMDB sentiment classification and AG's news categorization task. We run ablation studies on various components of the model and provide insights into decisions of our model.

2 Related Work

Generating adversarial examples to bypass deep learning classification models have been widely studied. In a white-box setting, some of the approaches include

gradient-based [19, 13], decision function-based [29] and spatial transformation based perturbation techniques[41]. In a black-box setting, several attack strategies have been proposed based on the property of transferability [36]. Papernot et al. [32, 31] relied on this transferability property where adversarial examples, generated on one classifier, are likely to cause another classifier to make the same mistake, irrespective of their architecture and training dataset. In order to generate adversarial samples, a local substitute model was trained with queries to the target model. Many learning systems allow query accesses to the model. However, there is little work that can leverage query-based access to target models to construct adversarial samples and move beyond transferability. These studies have primarily focused on image-based classifiers and cannot be directly applied to text-based classifiers.

While there is limited literature for such approaches in NLP systems, there have been some studies that have exposed the vulnerabilities of neural networks in text-based tasks like machine translations and question answering. Belinkov and Bisk [4] investigated the sensitivity of neural machine translation (NMT) to synthetic and natural noise containing common misspellings. They demonstrate that state-of-the-art models are vulnerable to adversarial attacks even after a spell-checker is deployed. Jia et al. [17] showed that networks trained for more difficult tasks, such as question answering, can be easily fooled by introducing distracting sentences into text, but these results do not transfer obviously to simpler text classification tasks. Following such works, different methods with the primary purpose of crafting adversarial example have been explored. Recently, a work by Ebrahimi et al. [9] developed a gradient-based optimization method that manipulates discrete text structure at its one-hot representation to generate adversarial examples in a white-box setting. In another white-box based attack, Gong et al. [12] perturbed the word embedding of given text examples and projected them to the nearest neighbour in the embedding space. This approach is an adaptation of perturbation algorithms for images. Though the size and quality of embedding play a critical role, this targeted attack technique ensured that the generated text sequence is intelligible.

Alzantot et al. [1] proposed a black-box targeted attack using a population-based optimization via genetic algorithm [2]. The perturbation procedure consists of random selection of words, finding their nearest neighbours, ranking and substitution to maximize the probability of target category. In this method, random word selection in the sequence to substitute were full of uncertainties and might be meaningless for the target label when changed. Since our model focuses on black-box non-targeted attack using an encoder-decoder approach, our work is closely related to the following techniques in the literature: Wong (2017) [39], Iyyer et al. [16] and Gao et al. [10]. Wong (2017) [39] proposed a GAN-inspired method to generate adversarial text examples targeting black-box classifiers. However, this approach was restricted to binary text classifiers. Iyyer et al. [16] crafted adversarial examples using their proposed Syntactically Controlled Paraphrase Networks (SCPNs). They designed this model for generating syntactically adversarial examples without compromising on the quality of the input semantics.

The general process is based on the encoder-decoder architecture of SCPN. Gao et al. [10] implemented an algorithm called DeepWordBug that generates small text perturbations in a black box setting forcing the deep learning model to make mistakes. DeepWordBug used a scoring function to determine important tokens and then applied character-level transformations to those tokens. Though the algorithm successfully generates adversarial examples by introducing character-level attacks, most of the introduced perturbations are constricted to misspellings. The semantics of the text may be irreversibly changed if excessive misspellings are introduced to fool the target classifier. While SCPNs and DeepWordBug primary rely only on paraphrases and character transformations respectively to fool the classifier, our model uses a hybrid word-character encoder-decoder approach to introduce both paraphrases and character-level perturbations as a part of our attack strategy. Our attacks can be a test of how robust the text classification models are to word and character-level perturbations.

3 Proposed Attack Strategy

Let us consider a target model T and (x, l) refers to the samples from the dataset. Given an instance x, the goal of the adversary is to generate adversarial examples x' such that $T(x') \neq l$, where l denotes the true label i.e take one of the K classes of the target classification model. The changes made to x to get x' are called perturbations. We would like to have x' close to the original instance x. In a black box setting, we do not have knowledge about the internals of the target model or its training data. Previous work by Papernot et al. [32] train a separate substitute classifier such that it can mimic the decision boundaries of the target classifier. The substitute classifier is then used to craft adversarial examples. While these techniques have been applied for image classification models, such methods have not been explored extensively for text.

We implement both the substitute network training and adversarial example generation using an encoder-decoder architecture called *Adversarial Examples Generator (AEG)*. The encoder extracts the character and word information from the input text and produces hidden representations of words considering its sequence context information. A substitute network is not implemented separately but applied using an attention mechanism to weigh the encoded hidden states based on their relevance to making predictions closer to target model outputs. The attention scores provide certain level of interpretability to the model as the regions of text that need to perturbed can be identified and visualized. The decoder uses the attention scores obtained from the substitute network, combines it with decoder state information to decide if perturbation is required at this state or not and finally emits the text unit (a text unit may refer to a word or character). Inspired by a work by Luong et al. [26], the decoder is a word and character-level recurrent network employed to generate adversarial examples. Before the substitute network is trained, we pretrain our encoder-decoder model on common misspellings and paraphrase datasets to empower the model to produce character and word perturbations in the form of misspellings or paraphrases. For training

substitute network and generation of adversarial examples, we randomly draw data that is disjoint from the training data of the black-box model since we assume the adversaries have no prior knowledge about the training data or the model. Specifically, we consider attacking a target classifier by generating adversarial examples based on unseen input examples. This is done by dividing the dataset into training, validation and test using 60-30-10 ratio. The training data is used by the target model, while the unseen validation samples are used with necessary data augmentation for our *AEG* model. We further improve our model by using a self-critical approach to finally generate better adversarial examples. The rewards are formulated based on the following goals: (a) fool the target classifier, (b) minimize the number of perturbations and (c) preserve the semantics of the text. In the following sections, we explain the encoder-decoder model and then describe the reinforcement learning framing towards generation of adversarial examples.

3.1 Background and Notations

Most of the sequence generation models follow an encoder-decoder framework [35, 8, 18] where encoder and decoder are modelled by separate recurrent neural networks. Usually these models are trained using a pair of text (x, y) where $x = [x_1, x_2.., x_n]$ is the input text and the $y = [y_1, y_2.., y_m]$ is the target text to be generated. The encoder transforms an input text sequence into an abstract representation h. While the decoder is employed to generate the target sequence using the encoded representation h. However, there are several studies that have incorporated several modifications to the standard encoder-decoder framework [3, 26, 27].

Encoder Based on Bahdanau et al. [3], we encode the input text sequence using bidirectional gated recurrent units (GRUs) to encode the input text sequence x. Formally, we obtain an encoded representation given by: $\overleftrightarrow{h_t} = \overleftarrow{h_t} + \overrightarrow{h_t}$.

Decoder The decoder is a forward GRU implementing an attention mechanism to recognize the units of input text sequence relevant for the generation of the next target work. The decoder GRU generates the next text unit at time step j by conditioning on the current decoder state s_j, context vector c_j computed using attention mechanism and previously generated text units. The probability of decoding each target unit is given by:

$$p(y_j|y_{<j}, h) = softmax(\tilde{s}_j) \tag{1}$$

$$\tilde{s}_j = f_d([c_j; s_j]) \tag{2}$$

where f_d is used to compute a new attentional hidden state $\tilde{s_j}$. Given the encoded input representations $\overleftrightarrow{H} = \{\overleftrightarrow{h_1}, ..., \overleftrightarrow{h_n}\}$ and the previous decoder GRU state s_{j-1}, the context vector at time step j is computed as: $c_j = Attn(\overleftrightarrow{H}, s_{j-1})$.

$Attn(\cdot,\cdot)$ computes a weight α_{jt} indicating the degree of relevance of an input text unit x_t for predicting the target unit y_j using a feed-forward network f_{attn}. Given a parallel corpus D, we train our model by minimizing the cross-entropy loss: $J = \sum_{(x,y)\in D} -logp(y|x)$.

4 Adversarial Examples Generator (AEG) Architecture

In this task of adversarial example generation, we have black-box access to the target model; the generator is not aware of the target model architecture or parameters and is only capable of querying the target model with supplied inputs and obtaining the output predictions. To enable the model to have capabilities to generate word and character perturbations, we develop a hybrid encoder-decoder model, *Adversarial Examples Generator (AEG)*, that operates at both word and character level to generate adversarial examples. Below, we explain the components of this model which have been improved to handle both word and character information from the text sequence.

4.1 Encoder

The encoder maps the input text sequence into a sequence of representations using word and character-level information. Our encoder (Figure 1) is a slight variant of Chen et al.[7]. This approach providing multiple levels of granularity can be useful in order to handle rare or noisy words in the text. Given character embeddings $E^{(c)} = [e_1^{(c)}, e_2^{(c)}, ...e_{n'}^{(c)}]$ and word embeddings $E^{(w)} = [e_1^{(w)}, e_2^{(w)}, ...e_n^{(w)}]$ of the input, starting (p_t) and ending (q_t) character positions at time step t, we define inside character embeddings as: $E_I^{(c)} = [e_{p_t}^{(c)}, ..., e_{q_t}^{(c)}]$ and outside embeddings as: $E_O^{(c)} = [e_1^{(c)}, ..., e_{p_t-1}^{(c)}; e_{q_t+1}^{(c)}, ..., e_{n'}^{(c)}]$. First, we obtain the character-enhanced word representation $\overleftrightarrow{h_t}$ by combining the word information from $E^{(w)}$ with the character context vectors. Character context vectors are obtained by attending over inside and outside character embeddings. Next, we compute a summary vector S over the hidden states $\overleftrightarrow{h_t}$ using an attention layer expressed as $Attn(\overleftrightarrow{H})$. To generate adversarial examples, it is important to identify the most relevant text units that contribute towards the target model's prediction and then use this information during the decoding step to introduce perturbation on those units. Hence, the summary vector is optimized using target model predictions without back propagating through the entire encoder. This acts as a substitute network that learns to mimic the predictions of the target classifier.

4.2 Decoder

Our *AEG* should be able to generate both character and word level perturbations as necessary. We achieve this by modifying the standard decoder [3, 27] to have two-level decoder GRUs: word-GRU and character-GRU (see Figure 2). Such hybrid approaches have been studied to achieve open vocabulary NMT in some of

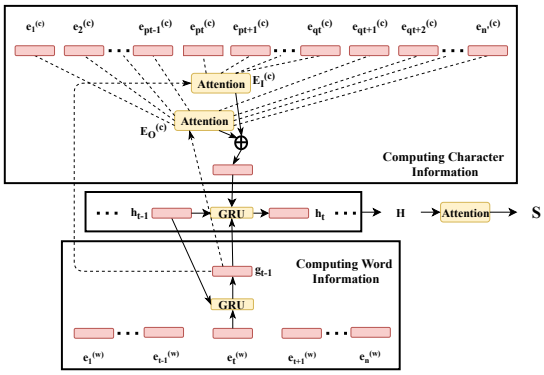

Fig. 1. Illustration of Encoder.

the previous work like Wu et al. [40] and Luong et al. [26]. Given the challenge that all different word misspellings cannot fit in a fixed vocabulary, we leverage the power of both words and characters in our generation procedure. The word-GRU uses word context vector $c_j^{(w)}$ by attending over the encoder hidden states $\overleftrightarrow{h_t}$. Once the word context vector $c_j^{(w)}$ is computed, we introduce a *perturbation vector* v_p to impart information about the need for any word or character perturbations at this decoding step. We construct this vector using the word-GRU decoder state $s_j^{(w)}$, context vector $c_j^{(w)}$ and summary vector S from the encoder as:

$$v_p = f_p(s_j^{(w)}, c_j^{(w)}, S) \tag{3}$$

We modify the the Equation (2) as: $\tilde{s}_j^{(w)} = f_d^{(w)}([c_j^{(w)}; s_j^{(w)}; v_p])$. The character-GRU will decide if the word is emitted with or without misspellings. We don't apply step-wise attention for character-GRU, instead we initialize it with the correct context. The ideal candidate representing the context must combine information about: (a) the word obtained from $c_j^{(w)}, s_j^{(w)}$, (b) its character alignment with the input characters derived from character context vector $c_j^{(c)}$ with respect to the word-GRU's state and (c) perturbation embedded in v_p. This yields,

$$c_j^{(c)} = Attn(E^{(c)}, s_j^{(w)}) \tag{4}$$

$$\tilde{s}_j^{(c)} = f_d^{(c)}([c_j^{(w)}; s_j^{(w)}; v_p; c_j^{(c)}]) \tag{5}$$

Thus, $\tilde{s}_j^{(c)}$ is initialized to the character-GRU only for the first hidden state. With this mechanism, both word and character level information can be used to introduce necessary perturbations.

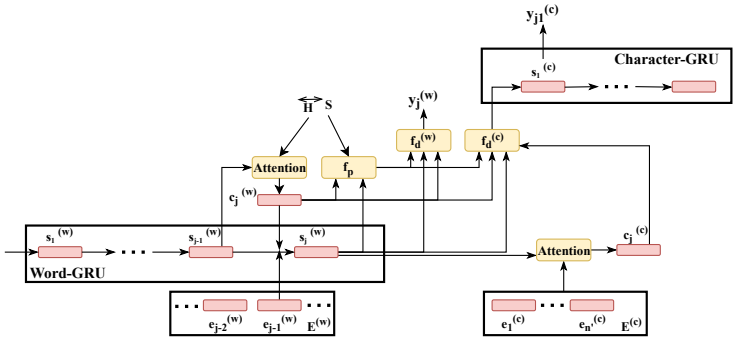

Fig. 2. Illustration of the word and character decoder.

5 Training

5.1 Supervised Pretraining with Teacher Forcing

The primary purpose of pretraining AEG is to enable our hybrid encoder-decoder to encode both character and word information from the input example and produce both word and character-level transformations in the form of paraphrases or misspellings. Though the pretraining helps us mitigate the cold-start issue, it does not guarantee that these perturbed texts will fool the target model. There are large number of valid perturbations that can be applied due to multiple ways of arranging text units to produce paraphrases or different misspellings. Thus, minimizing J_{mle} is not sufficient to generate adversarial examples.

Dataset Collection In this paper, we use paraphrase datasets like PARANMT-50M corpus[37], Quora Question Pair dataset [1] and Twitter URL paraphrasing corpus [23]. These paraphrase datasets together contains text from various sources: Common Crawl, CzEng1.6, Europarl, News Commentary, Quora questions, and Twitter trending topic tweets. We do not use all the data for our pretraining. We randomly sample 5 million parallel texts and augment them using simple character-transformations (eg. random insertion, deletion or replacement) to words in the text. The number of words that undergo transformation is capped at 10% of the total number of words in the text. We further include examples which contain only character-transformations without paraphrasing the original input.

Training Objective *AEG* is pre-trained using teacher-forcing algorithm [38] on the dataset explained in Section 3. Consider an input text: "movie was good" that needs to be decoded into the following target perturbed text: "film is gud". The word "gud" might be out-of-vocabulary indicated by $< oov >$. Hence, we compute

[1] https://www.kaggle.com/c/quora-question-pairs/data

the loss incurred by word-GRU decoder, $J^{(w)}$, when predicting {"film", "is", "$< oov >$"} and loss incurred by character-GRU decoder, $J^{(c)}$, when predicting {'f', 'i','l', 'm', '_'},{'i','s','_'},{'g', 'u','d','_'}. Therefore, the training objective in Section 3.1 is modified into:

$$J_{mle} = J^{(w)} + J^{(c)} \tag{6}$$

5.2 Training with Reinforcement learning

We fine-tune our model to fool a target classifier by learning a policy that maximizes a specific discrete metric formulated based on the constraints required to generate adversarial examples. In our work, we use the self-critical approach of Rennie et al. [34] as our policy gradient training algorithm.

Self-critical sequence training (SCST) In SCST approach, the model learns to gather more rewards from its sampled sequences that bring higher rewards than its best greedy counterparts. First, we compute two sequences: (a) y' sampled from the model's distribution $p(y'_j|y'_{<j}, h)$ and (b) $\hat{y}$ obtained by greedily decoding ($argmax$ predictions) from the distribution $p(\hat{y}_j|\hat{y}_{<j}, h)$ Next, rewards $r(y'_j), r(\hat{y}_j)$ are computed for both the sequences using a reward function $r(\cdot)$, explained in Section 5.2. We train the model by minimizing:

$$J_{rl} = -\sum_j (r(y') - r(\hat{y}))logp(\hat{y}_j|\hat{y}_{<j}, h) \tag{7}$$

Here $r(\hat{y})$ can be viewed as the baseline reward. This approach, therefore, explores different sequences that produce higher reward compared to the current best policy.

Rewards The reward $r(\hat{y})$ for the sequence generated is a weighted sum of different constraints required for generating adversarial examples. Since our model operates at word and character levels, we therefore compute three rewards: adversarial reward, semantic similarity and lexical similarity reward. The reward should be high when: (a) the generated sequence causes the target model to produce a low classification prediction probability for its ground truth category, (b) semantic similarity is preserved and (c) the changes made to the original text are minimal.

Adversarial Reward Given a target model T, it takes a text sequence y and outputs prediction probabilities P across various categories of the target model. Given an input sample (x, l), we compute a perturbation using our AEG model and produce a sequence y. We compute the adversarial reward as $R_A = (1 - P_l)$, where the ground truth l is an index to the list of categories and P_l is the probability that the perturbed generated sequence y belongs to target ground truth l. Since we want the target classifier to make mistakes, we promote it by rewarding higher when the sequences produce low target probabilities.

Semantic Similarity Inspired by the work of Li et al. [24], we train a deep matching model that can represent the degree of match between two texts. We use character based biLSTM models with attention [25] to handle word and character level perturbations. The matching model will help us compute the the semantic similarity R_S between the text generated and the original input text.

Lexical Similarity Since our model functions at both character and word level, we compute the lexical similarity. The purpose of this reward is to keep the changes as minimal as possible to just fool the target classifier. Motivated by the recent work of Moon et al. [28], we pretrain a deep neural network to compute approximate Levenshtein distance R_L composed of character based bi-LSTM model. We replicate that model by generating a large number of text with perturbations in the form of insertions, deletions or replacements. We also include words which are prominent nicknames, abbreviations or inconsistent notations to have more lexical similarity. This is generally not possible using direct Levenshtein distance computation. Once trained, it can produce a purely lexical embedding of the text without semantic allusion. This can be used to compute the lexical similarity between the generated text y and the original input text x for our purpose.

Finally, we combine all these three rewards using:

$$r(y) = \gamma_A R_A + \gamma_S R_S + \gamma_L R_L \tag{8}$$

where $\gamma_A, \gamma_S, \gamma_L$ are hyperparameters that can be modified depending upon the kind of textual generations expected from the model. The changes inflicted by different reward coefficients can be seen in Section 6.5.

5.3 Training Details

We trained our models on 4 GPUs. The parameters of our hybrid encoder-decoder were uniformly initialized to $[-0.1, 0.1]$. The optimization algorithm used is Adam [21]. The encoder word embedding matrices were initialized with 300-dimensional Glove vectors [33]. During reinforcement training, we used plain stochastic gradient descent with a learning rate of 0.01. Using a held-out validation set, the hyper-parameters for our experiments are set as follows: $\gamma_A = 1, \gamma_S = 0.5, \gamma_L = 0.25$.

6 Experiments

In this section, we describe the evaluation setup used to measure the effectiveness of our model in generating adversarial examples. The success of our model lies in its ability to fool the target classifier. We pretrain our models with dataset that generates a number of character and word perturbations. We elaborate on the experimental setup and the results below.

Datasets	Details	Model	Accuracy
IMDB Review	Classes: 2; #Train: 25k;	CNN-Word [20]	89.95%
AG's News	Classes: 4; #Train: 120k;	CNN-Char [42]	89.11%

Table 1. Summary of data and models used in our experiments.

6.1 Setup

We conduct experiments on different datasets to verify if the accuracy of the deep learning models decrease when fed with the adversarial examples generated by our model. We use benchmark sentiment classification and news categorization datasets and the details are as follows:

- Sentiment classification: We trained a word-based convolutional model (CNN-Word) [20] on IMDB sentiment dataset [2]. The dataset contains 50k movie reviews in total which are labeled as positive or negative. The trained model achieves a test accuracy of 89.95% which is relatively close to the state-of-the-art results on this dataset.
- News categorization: We perform our experiments on AG's news corpus [3] with a character-based convolutional model (CNN-Char) [42]. The news corpus contains titles and descriptions of various news articles along with their respective categories. There are four categories: World, Sports, Business and Sci/Tech. The trained CNN-Char model achieves a test accuracy of 89.11%.

Table 1 summarizes the data and models used in our experiments. We compare our proposed model with the following black-box non-targeted attacks:

- **Random**: We randomly select a word in the text and introduce some perturbation to that word in the form of a character replacement or synonymous word replacement. No specific strategy to identify importance of words.
- **NMT-BT**: We generate paraphrases of the sentences of the text using a back-translation approach [16]. We used pretrained English↔German translation models to obtain back-translations of input examples.
- **DeepWordBug** [10]: A scoring function is used to determine the important tokens to change. The tokens are then modified to evade a target model.
- **No-RL**: We use our pretrained model without the reinforcement learning objective.

The performance of these methods are measured by the percentage fall in accuracy of these models on the generated adversarial texts. Higher the percentage dip in the accuracy of the target classifier, more effective is our model.

[2] http://ai.stanford.edu/ amaas/data/sentiment/
[3] https://github.com/mhjabreel/CharCNN/tree/master/data/ag_news_csv

6.2 Quantitative Analysis

We analyze the effectiveness of our approach by comparing the results from using two different baselines against character and word-based models trained on different datasets. Table 2 demonstrates the capability of our model. Without the reinforcement learning objective, the No-RL model performs better than the back-translation approach(NMT-BT). The improvement can be attributed to the word and character perturbations introduced by our hybrid encoder-decoder model as opposed to only paraphrases in the former model. Our complete *AEG* model outperforms all the other models with significant drop in accuracy. For the CNN-Word, DeepWordBug decreases the accuracy from 89.95% to 28.13% while *AEG* model further reduces it to 18.5%.

Models	**IMDB** (CNN-Word)	**AG's News** (CNN-Char)
Random	2.46%	9.64%
NMT-BT	25.38%	22.45%
DeepWordBug	68.73%	65.80%
No-RL (Ours)	38.05%	33.58%
AEG (Ours)	**79.43%**	**72.16%**

Model Variants	IMDB	News Corpus
Char-dec	73.5	68.64%
No pert	71.45%	65.91%

Table 2. Left: Performance of our AEG model on IMDB and AG's News dataset using word and character based CNN models respectively. Results indicate the percentage dip in the accuracy by using the corresponding attacking model over the original accuracy. **Right:** Performance of different variants of our model.

It is important to note that our model is able to expose the weaknesses of the target model irrespective of the nature of the model (either word or character level). It is interesting that even simple lexical substitutions and paraphrases can break such models on both datasets we tested. Across different models, the character-based models are less susceptible to adversarial attacks compared to word-based models as they are able to handle misspellings and provide better generalizations.

6.3 Human Evaluation

We also evaluated our model based on human judgments. We conducted an experiment where the workers were presented with randomly sampled 100 adversarial examples generated by our model which were successful in fooling the target classifier. The examples were shuffled to mitigate ordering bias, and every example was annotated by three workers. The workers were asked to label the sentiment of the sampled adversarial example. For every adversarial example shown, we also showed the original text and asked them to rate their similarity on a scale from 0 (Very Different) to 3 (Very Similar). We found that the perturbations produced by our model do not affect the human judgments significantly as 94.6%

of the human annotations matched with the ground-truth label of the original text. The average similarity rating of 1.916 also indicated that the generated adversarial sequences are semantics-preserving.

6.4 Ablation Studies

In this section, we make different modifications to our encoder and decoder to weigh the importance of these techniques: (a) No perturbation vector (No Pert) and finally (b) a simple character based decoder (Char-dec) but involves perturbation vector. Table 2 shows that the absence of hybrid decoder leads to a significant drop in the performance of our model. The main reason we believe is that hybrid decoder is able to make targeted attacks on specific words which otherwise is lost while generating text using a pure-character based decoder. In the second case case, the most important words associated with the prediction of the target model are identified by the summary vector. When the perturbation vector is used, it carries forward this knowledge and decides if a perturbation should be performed at this step or not. This can be verified even in Figure 3, where the regions of high attention get perturbed in the text generated.

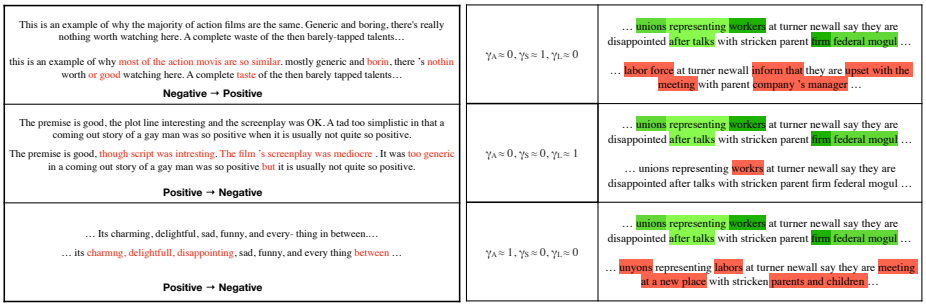

Fig. 3. Left: Examples from IMDB reviews dataset, where the model introduces misspellings or paraphrases that are sufficient to fool the target classifier. **Right**: Effect of coefficients of the reward function. The first line is the text from the AG's news corpus. The second line is the generated by the model given specific constraints on the reward coefficients. The examples do not necessarily lead to misclassification. The text in green are attention scores indicating relevance of classification. The text in red are the perturbations introduced by our model.

6.5 Qualitative Analysis

We qualitatively analyze the results by visualizing the attention scores and the perturbations introduces by our model. We further evaluate the importance of hyperparameters $\gamma_{(.)}$ in the reward function. We set only one of the hyperparameters closer to 1 and set the remaining closer to zero to see how it affects

the text generation. The results can be seen in Figure 3. Based on a subjective qualitative evaluation, we make the following observations:

- Promisingly, it identifies the most important words that contribute to particular categorization. The model introduces misspellings or word replacements without significant change in semantics of the text.
- When the coefficient associated only with adversarial reward goes to 1, it begins to slowly deviate though not completely. This is motivated by the initial pretraining step on paraphrases and perturbations.

7 Conclusion

In this work, we have introduced a *AEG*, a model capable of generating adversarial text examples to fool the black-box text classification models. Since we do not have access to gradients or parameters of the target model, we modelled our problem using a reinforcement learning based approach. In order to effectively baseline the REINFORCE algorithm for policy-gradients, we implemented a self-critical approach that normalizes the rewards obtained by sampled sentences with the rewards obtained by the model under test-time inference algorithm. By generating adversarial examples for target word and character-based models trained on IMDB reviews and AG's news dataset, we find that our model is capable of generating semantics-preserving perturbations that leads to steep decrease in accuracy of those target models. We conducted ablation studies to find the importance of individual components of our system. Extremely low values of the certain reward coefficient constricts the quantitative performance of the model can also lead to semantic divergence. Therefore, the choice of a particular value for this model should be motivated by the demands of the context in which it is applied. One of the main challenges of such approaches lies in the ability to produce more synthetic data to train the generator model in the distribution of the target model's training data. This can significantly improve the performance of our model. We hope that our method motivates a more nuanced exploration into generating adversarial examples and adversarial training for building robust classification models.

References

1. Alzantot, M., Sharma, Y., Elgohary, A., Ho, B.J., Srivastava, M., Chang, K.W.: Generating natural language adversarial examples. arXiv preprint arXiv:1804.07998 (2018)
2. Anderson, E.J., Ferris, M.C.: Genetic algorithms for combinatorial optimization: the assemble line balancing problem. ORSA Journal on Computing **6**(2), 161–173 (1994)
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
4. Belinkov, Y., Bisk, Y.: Synthetic and natural noise both break neural machine translation. arXiv preprint arXiv:1711.02173 (2017)

5. Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., Roli, F.: Evasion attacks against machine learning at test time. In: Joint European conference on machine learning and knowledge discovery in databases. pp. 387–402. Springer (2013)
6. Biggio, B., Roli, F.: Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition **84**, 317–331 (2018)
7. Chen, H., Huang, S., Chiang, D., Dai, X., Chen, J.: Combining character and word information in neural machine translation using a multi-level attention. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). vol. 1, pp. 1284–1293 (2018)
8. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
9. Ebrahimi, J., Rao, A., Lowd, D., Dou, D.: Hotflip: White-box adversarial examples for nlp. arXiv preprint arXiv:1712.06751 (2017)
10. Gao, J., Lanchantin, J., Soffa, M.L., Qi, Y.: Black-box generation of adversarial text sequences to evade deep learning classifiers. arXiv preprint arXiv:1801.04354 (2018)
11. Gilmer, J., Adams, R.P., Goodfellow, I., Andersen, D., Dahl, G.E.: Motivating the rules of the game for adversarial example research. arXiv preprint arXiv:1807.06732 (2018)
12. Gong, Z., Wang, W., Li, B., Song, D., Ku, W.S.: Adversarial texts with gradient methods. arXiv preprint arXiv:1801.07175 (2018)
13. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. stat **1050**, 20 (2015)
14. Guo, C., Rana, M., Cisse, M., van der Maaten, L.: Countering adversarial images using input transformations. arXiv preprint arXiv:1711.00117 (2017)
15. Iter, D., Huang, J., Jermann, M.: Generating adversarial examples for speech recognition. Stanford Technical Report (2017)
16. Iyyer, M., Wieting, J., Gimpel, K., Zettlemoyer, L.: Adversarial example generation with syntactically controlled paraphrase networks. arXiv preprint arXiv:1804.06059 (2018)
17. Jia, R., Liang, P.: Adversarial examples for evaluating reading comprehension systems. arXiv preprint arXiv:1707.07328 (2017)
18. Kalchbrenner, N., Blunsom, P.: Recurrent continuous translation models. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. pp. 1700–1709 (2013)
19. Kereliuk, C., Sturm, B.L., Larsen, J.: Deep learning and music adversaries. IEEE Transactions on Multimedia **17**(11), 2059–2071 (2015)
20. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
22. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 (2016)
23. Lan, W., Qiu, S., He, H., Xu, W.: A continuously growing dataset of sentential paraphrases. In: Proceedings of The 2017 Conference on Empirical Methods on Natural Language Processing (EMNLP). pp. 1235–1245. Association for Computational Linguistics (2017), `http://aclweb.org/anthology/D17-1127`
24. Li, Z., Jiang, X., Shang, L., Li, H.: Paraphrase generation with deep reinforcement learning. arXiv preprint arXiv:1711.00279 (2017)

25. Lin, Z., Feng, M., Santos, C.N.d., Yu, M., Xiang, B., Zhou, B., Bengio, Y.: A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130 (2017)
26. Luong, M.T., Manning, C.D.: Achieving open vocabulary neural machine translation with hybrid word-character models. arXiv preprint arXiv:1604.00788 (2016)
27. Luong, M.T., Sutskever, I., Le, Q.V., Vinyals, O., Zaremba, W.: Addressing the rare word problem in neural machine translation. arXiv preprint arXiv:1410.8206 (2014)
28. Moon, S., Neves, L., Carvalho, V.: Multimodal named entity disambiguation for noisy social media posts. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 2000–2008 (2018)
29. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2574–2582 (2016)
30. Mopuri, K.R., Babu, R.V., et al.: Gray-box adversarial training. arXiv preprint arXiv:1808.01753 (2018)
31. Papernot, N., McDaniel, P., Goodfellow, I.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint arXiv:1605.07277 (2016)
32. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against deep learning systems using adversarial examples. arXiv preprint (2016)
33. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
34. Rennie, S.J., Marcheret, E., Mroueh, Y., Ross, J., Goel, V.: Self-critical sequence training for image captioning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7008–7024 (2017)
35. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in neural information processing systems. pp. 3104–3112 (2014)
36. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
37. Wieting, J., Gimpel, K.: Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. arXiv preprint arXiv:1711.05732 (2017)
38. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural computation **1**(2), 270–280 (1989)
39. Wong, C.: Dancin seq2seq: Fooling text classifiers with adversarial text example generation. arXiv preprint arXiv:1712.05419 (2017)
40. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al.: Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016)
41. Xiao, C., Zhu, J.Y., Li, B., He, W., Liu, M., Song, D.: Spatially transformed adversarial examples. arXiv preprint arXiv:1801.02612 (2018)
42. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances in neural information processing systems. pp. 649–657 (2015)