Grey-box Adversarial Attack And Defence For Sentiment Classification

Ying Xu *
IBM Research
Australia

Xu Zhong *
IBM Research
Australia

Antonio Jimeno Yepes
IBM Research
Australia

Jey Han Lau
University of Melbourne
Australia

Abstract

We introduce a grey-box adversarial attack and defence framework for sentiment classification. We address the issues of differentiability, label preservation and input reconstruction for adversarial attack *and* defence in one unified framework. Our results show that once trained, the attacking model is capable of generating high-quality adversarial examples substantially faster (one order of magnitude less in time) than state-of-the-art attacking methods. These examples also preserve the original sentiment according to human evaluation. Additionally, our framework produces an improved classifier that is robust in defending against multiple adversarial attacking methods. Code is available at: https://github.com/ibm-aur-nlp/adv-def-text-dist

1 Introduction

Recent advances in deep neural networks have created applications for a range of different domains. In spite of the promising performance achieved by neural models, there are concerns around their robustness, as evidence shows that even a slight perturbation to the input data can fool these models into producing wrong predictions (Goodfellow et al., 2014; Kurakin et al., 2016). Research in this area is broadly categorised as *adversarial machine learning*, and it has two sub-fields: *adversarial attack*, which seeks to generate adversarial examples that fool target models; and *adversarial defence*, whose goal is to build models that are less susceptible to adversarial attacks.

A number of adversarial attacking methods have been proposed for image recognition (Goodfellow et al., 2014), NLP (Zhang et al., 2020) and speech recognition (Alzantot et al., 2018a). These methods are generally categorised into three types: white-box, black-box and grey-box attacks. White-box attacks assume full access to the target models and often use the gradients from the target models to guide the craft of adversarial examples. Black-box attacks, on the other hand, assume no knowledge on the architecture of the target model and perform attacks by repetitively querying the target model. Different from the previous two, grey-box attacks train a *generative model* to generate adversarial examples and only assume access to the target model during the training phrase. The advantages of grey-box attacking methods include higher time efficiency; no assumption of access to target model during attacking phase; and easier integration into adversarial defending algorithms. However, due to the discrete nature of texts, designing grey-box attacks on text data remains a challenge.

In this paper, we propose a grey-box framework that generates high quality textual adversarial examples while simultaneously trains an improved sentiment classifier for adversarial defending. Our contributions are summarised as follows:

- We propose to use Gumbel-softmax (Jang et al., 2016) to address the differentiability issue to combine the adversarial example generator and target model into one unified trainable network.
- We propose multiple competing objectives for adversarial attack training so that the generated adversarial examples can fool the target classifier while maintaining similarity with the input examples. We considered a number of similarity measures to define a successful attacking example for texts, such as lexical and semantic similarity and label preservation.[1]
- To help the generative model to reconstruct input sentences as faithfully as possible, we introduce a novel but simple copy mechanism to

[*] This work was completed during the employment of the authors in IBM Research Australia.

[1] Without constraint on label preservation, simply flipping the ground-truth sentiment (e.g. *the movie is great* → *the movie is awful*) can successfully change the output of a sentiment classifier even though it is not a useful adversarial example.

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4078–4087
June 6–11, 2021. ©2021 Association for Computational Linguistics

the decoder to selectively copy words directly from the input.

- We assess the adversarial examples beyond just attacking performance, but also content similarity, fluency and label preservation using both automatic and human evaluations.
- We simultaneously build an improved sentiment classifier while training the generative (attacking) model. We show that a classifier built this way is more robust than adversarial defending based on adversarial examples augmentation.

2 Related Work

Most white-box methods are gradient-based, where some form of the gradients (e.g. the sign) with respect to the target model is calculated and added to the input representation. In image processing, the fast gradient sign method (FGSM; Goodfellow et al. (2014)) is one of the first studies in attacking image classifiers. Some of its variations include Kurakin et al. (2016); Dong et al. (2018). These gradient-based methods could not be applied to texts directly because perturbed word embeddings do not necessarily map to valid words. Methods such as DeepFool (Moosavi-Dezfooli et al., 2016) that rely on perturbing the word embedding space face similar roadblocks.

To address the issue of embedding-to-word mapping, Gong et al. (2018) propose to use nearest-neighbour search to find the closest words to the perturbed embeddings. However, this method treats all tokens as equally vulnerable and replace all tokens with their nearest neighbours, which leads to non-sensical, word-salad outputs. A solution to this is to replace tokens one-by-one in order of their vulnerability while monitoring the change of the output of the target models. The replacement process stops once the target prediction has changed, minimising the number of changes. Examples of white-box attacks that utilise this approach include TYC (Tsai et al., 2019) and HOTFLIP (Ebrahimi et al., 2017).

Different to white-box attacks, black-box attacks do not require full access to the architecture of the target model. Chen et al. (2017) propose to estimate the loss function of the target model by querying its *label probability distributions*, while Papernot et al. (2017) propose to construct a substitute of the target model by querying its *output labels*. The latter approach is arguably more realistic because in most cases attackers only have access to output labels rather than their probability distributions. There is relatively fewer studies on black-box attacks for text. An example is TEXTFOOLER, proposed by Jin et al. (2019), that generates adversarial examples by querying the label probability distribution of the target model. Another is proposed by Alzantot et al. (2018b) where genetic algorithm is used to select the word for substitution.

Grey-box attacks require an additional training process during which full access to the target model is assumed. However, post-training, the model can be used to generate adversarial examples without querying the target model. Xiao et al. (2018) introduce a generative adversarial network to generate the image perturbation from a noise map. It is, however, not trivial to adapt the method for text directly. It is because text generation involves discrete decoding steps and as such the joint generator and target model architecture is non-differentiable.

In terms of adversarial defending, the most straightforward method is to train a robust model on data augmented by adversarial examples. Recently, more methods are proposed for texts, such as those based on interval bound propagation (Jia et al., 2019; Huang et al., 2019), and dirichlet neighborhood ensemble (Zhou et al., 2020).

3 Methodology

The purpose of adversarial attack is to slightly perturb an input example x for a pre-trained target model (e.g. a sentiment classifier) f so that $f(x) \neq y$, where y is the ground truth of x. The perturbed example x' should *look* similar to x, which can be measured differently depending on the domain of the input examples.

3.1 General Architecture

We propose a grey-box attack and defence framework which consists of a generator $\mathcal{G}$ (updated), and two copies of a pre-trained target classifier: a static classifier $\mathcal{C}$ and an updated/augmented classifier $\mathcal{C}^*$.[2] During the training phase, the output of $\mathcal{G}$ is directly fed to $\mathcal{C}$ and $\mathcal{C}^*$ to form a joint architecture. Post-training, the generator $\mathcal{G}$ is used independently to generate adversarial examples (adversarial attack); while the augmented classifier $\mathcal{C}^*$ is an improved classifier with increased robustness (adversarial defence).

[2] $\mathcal{C}$ and $\mathcal{C}^*$ start with the same pre-trained weights, although only $\mathcal{C}^*$ is updated during training.

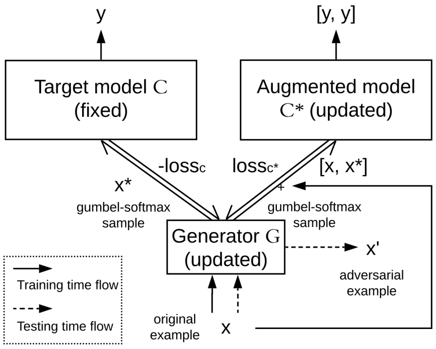


Figure 1: Grey-box adversarial attack and defence framework for sentiment classification.

The training phase is divided into attacking steps and defending steps, where the former updates only the generator $\mathcal{G}$ and learns to introduce slight perturbation to the input by maximising the objective function of the target model $\mathcal{C}$. The latter updates $\mathcal{C}^*$ and $\mathcal{G}$ by feeding both original examples and adversarial examples generated by $\mathcal{G}$. Here, the adversarial examples are assumed to share the same label with their original examples. Effectively, the defending steps are training an improved classifier with data augmented by adversarial examples.

Generating text with discrete decoding steps (e.g. *argmax*) makes the joint architecture not differentiable. Therefore we propose to use Gumbel-softmax (Jang et al., 2016) to approximate the categorical distribution of the discrete output. For each generation step i, instead of sampling a word from the vocabulary, we draw a Gumbel-softmax sample x_i^* which has the full probability distribution over words in the vocabulary: the probability of the generated word is close to 1.0 and other words close to zero. We obtain the input embedding for $\mathcal{C}$ and $\mathcal{C}^*$ by multiplying the sample x_i^* with the word embedding matrix, $M_{\mathcal{C}}$, of the target model $\mathcal{C}$: $x_i^* \cdot M_{\mathcal{C}}$. Figure 1 illustrates our grey-box adversarial attack and defence framework for text.

The generator $\mathcal{G}$ can be implemented as an auto-encoder or a paraphrase generator, essentially differentiated by their data conditions: the former uses the input sentences as the target, while the latter uses paraphrases (e.g. PARANMT-50M (Wieting and Gimpel, 2017)). In this paper, we implement $\mathcal{G}$ as an auto-encoder, as our preliminary experiments found that a pre-trained paraphrase generator performs poorly when adapted to our test domain, e.g. Yelp reviews.

3.2 Objective Functions

Our auto-encoder $\mathcal{G}$ generates an adversarial example given an input example. It tries to reconstruct the input example but is also regulated by an adversarial loss term that 'discourages' it from doing so. The objectives for the attacking step are given as follows:

$$L_{adv} = \log p_{\mathcal{C}}(y|x, \theta_{\mathcal{C}}, \theta_{\mathcal{G}}) \tag{1}$$

$$L_{s2s} = -\log p_{\mathcal{G}}(x|x, \theta_{\mathcal{G}}) \tag{2}$$

$$L_{sem} = cos\left(\frac{1}{n}\sum_{i=0}^{n} emb(x_i), \frac{1}{n}\sum_{i=0}^{n} emb(x_i^*)\right) \tag{3}$$

where L_{adv} is essentially the negative cross-entropy loss of $\mathcal{C}$; L_{s2s} is the sequence-to-sequence loss for input reconstruction; and L_{sem} is the cosine similarity between the averaged embeddings of x and x^* (n = number of words). Here, L_{s2s} encourages x' (produced at test time) to be lexically similar to x and helps produce coherent sentences, and L_{sem} promotes semantic similarity. We weigh the three objective functions with two scaling hyper-parameters and the total loss is: $L = \lambda_1(\lambda_2 L_{s2s} + (1 - \lambda_2)L_{sem}) + (1 - \lambda_1)L_{adv}$ We denote the auto-encoder based generator trained with these objectives as **AE**.

An observation from our preliminary experiments is that the generator tends to perform imbalanced attacking among different classes. (e.g. AE learns to completely focus on one direction attacking, e.g. positive-to-negative or negative-to-positive attack). We found a similar issue in white-box attack methods such as FGSM Goodfellow et al. (2014) and DeepFool (Moosavi-Dezfooli et al., 2016). To address this issue, we propose to modify L_{adv} to be the maximum loss of a particular class in each batch, i.e.

$$L_{adv} = max_{t=1}^{|C|}(L_{adv}^t) \tag{4}$$

where L_{adv}^t refers to the adversarial loss of examples in the t-th class and $|C|$ the total number of classes. We denote the generator trained with this alternative loss as **AE+BAL**.

For adversarial defence, we use the same objective functions, with the following exception: we replace L_{adv} in Equation (1) with the objective function of the classifier $\mathcal{C}^*$, i.e.

$$L_{def} = -\log p_{\mathcal{C}^*}([y, y]|[x, x^*], \theta_{\mathcal{C}^*}, \theta_{\mathcal{G}}) \tag{5}$$

We train the model $\mathcal{C}^*$ using both original and adversarial examples (x and x^*) with their original label (y) to prevent $\mathcal{C}^*$ from overfitting to the adversarial examples.

3.3 Label Preservation

One of the main challenges of generating a textual adversarial example is to preserve its original ground truth label, which we refer to as *label preservation*. It is less of an issue in computer vision because slight noises added to an image is unlikely to change how we perceive the image. In text, however, slight perturbation to a sentence could completely change its ground truth.

We use sentiment classification as context to explain our approach for label preservation. The goal of adversarial attack is to generate an adversarial sentence whose sentiment is flipped according to the target model prediction but preserves the original ground truth sentiment from the perspective of a human reader. We propose two ways to help label preservation. The first approach is task-agnostic, i.e. it can work for any classification problem, while the second is tailored for sentiment classification.

Label smoothing (+LS). We observe the generator has a tendency to produce adversarial examples with high confidence, opposite sentiment scores from the static classifier $\mathcal{C}$. We explore the use of label smoothing (Müller et al., 2019) to force the generator generate examples that are closer to the decision boundary, to discourage the generator from completely changing the sentiment. We incorporate label smoothing in Eq. 1 by redistributing the probability mass of true label uniformly to all other labels. Formally, the smoothed label $y_{ls} = (1 - \alpha) * y + \alpha/K$ where α is a hyperparameter and K is the number of classes. For example, when performing negative-to-positive attack, instead of optimising $\mathcal{G}$ to produce adversarial examples with label distribution {pos: 1.0, neg: 0.0} (from $\mathcal{C}$), label distribution {pos: 0.6, neg: 0.4} is targeted. Generator trained with this additional constraint is denoted with the **+LS** suffix.

Counter-fitted embeddings (+CF). Mrkšić et al. (2016) found that unsupervised word embeddings such as GloVe (Pennington et al., 2014) often do not capture synonymy and antonymy relations (e.g. *cheap* and *pricey* have high similarity). The authors propose to post-process pre-trained word embeddings with lexical resources (e.g. WordNet) to produce counter-fitted embeddings that better capture these lexical relations. To discourage the generator $\mathcal{G}$ from generating words with opposite sentiments, we experiment with training $\mathcal{G}$ with counter-fitted embeddings. Models using counter-fitted embeddings is denoted with **+CF** suffix.

3.4 Generator with Copy Mechanism (+CPY)

White-box or black-box attacking methods are based on adding, removing, or replacing tokens in input examples. Therefore maintaining similarity with original examples is easier than grey-box methods that generate adversarial examples word-by-word from scratch. We introduce a simple copy mechanism that helps grey-box attack to produce faithful reconstruction of the original sentences.

We incorporate a static copy mask to the decoder where it only generates for word positions that have not been masked. E.g., given the input sentence $x = [w_0, w_1, w_2]$, target $x^* = [w_0, w_1, w_2]$, and mask $m = [1, 0, 1]$, at test time the decoder will "copy" from the target for the first input (w_0) and third input token (w_2) to produce w_0 and w_2, but for the second input token (w_1) it will decode from the vocabulary. During training, we compute cross-entropy only for the unmasked input words.

The static copy mask is obtained from one of the pre-trained target classifiers, C-LSTM (Section 4.2). C-LSTM is a classifier with a bidirectional LSTM followed by a self-attention layer to weigh the LSTM hidden states. We rank the input words based on the self-attention weights and create a copy mask such that only the positions corresponding to the top-N words with the highest weights are generated from the decoder. Generally sentiment-heavy words such as *awesome* and *bad* are more likely to have higher weights in the self-attention layer. This self attention layer can be seen as an importance ranking function (Morris et al., 2020b) that determines which tokens should be replaced or replaced first. Models with copy mechanism are denoted with the **+CPY** suffix.

4 Experiments and Results

4.1 Dataset

We conduct our experiments using the Yelp review dataset.[3] We binarise the ratings,[4] use spaCy for tokenisation,[5] and keep only reviews with ≤ 50 tokens (hence the dataset is denoted as `yelp50`).

[3] https://www.yelp.com/dataset

[4] Ratings$\geq$4 is set as positive and ratings$\leq$2 as negative.

[5] https://spacy.io

We split the data in a 90/5/5 ratio and downsample the positive class in each set to be equivalent to the negative class, resulting in 407,298, 22,536 and 22,608 examples in train/dev/test set respectively.

4.2 Implementation Details

For the target classifiers ($\mathcal{C}$ and $\mathcal{C}^*$), we pre-train three sentiment classification models using `yelp50`: C-LSTM (Wang et al., 2016), C-CNN (Kim, 2014) and C-BERT. C-LSTM is composed of an embedding layer, a 2-layer bidirectional LSTMs, a self-attention layer, and an output layer. C-CNN has a number of convolutional filters of varying sizes, and their outputs are concatenated, pooled and fed to a fully-connected layer followed by an output layer. Finally, C-BERT is obtained by fine-tuning the BERT-Base model (Devlin et al., 2018) for sentiment classification. We tune learning rate, batch size, number of layers and number of hidden units for all classifiers; the number of attention units for C-LSTM and convolutional filter sizes and dropout rates for C-CNN specifically.

For the auto-encoder, we pre-train it to reconstruct sentences in `yelp50`.[6] During pre-training, we tune learning rate, batch size, number of layers and number of hidden units. During the training of adversarial attacking, we tune λ_1 and λ_2, and learning rate lr. We also test different temperature τ for Gumbel-softmax sampling and found that $\tau = 0.1$ performs the best. All word embeddings are fixed.

More hyper-parameter and training configurations are detailed in the supplementary material.

4.3 Attacking Performance

Most of the existing adversarial attacking methods have been focusing on improving the attack success rate. Recent study show that with constraints adjusted to better preserve semantics and grammaticality, the attack success rate drops by over 70 percentage points (Morris et al., 2020a). In this paper, we want to understand — given a particular success rate — the quality (e.g. fluency, content/label preservation) of the generated adversarial samples. Therefore, we tuned all attacking methods to achieve the same levels of attack success rates; and compare the quality of generated adversarial examples. [7] Note that results for adversarial attack are obtained by using the $\mathcal{G} + \mathcal{C}$ joint architecture, while results for adversarial defence are achieved by the $\mathcal{G} + \mathcal{C} + \mathcal{C}^*$ joint architecture.

4.3.1 Evaluation Metrics

In addition to measuring how well the adversarial examples fool the sentiment classifier, we also use a number of automatic metrics to assess other aspects of adversarial examples, following Xu et al. (2020):

Attacking performance. We use the standard classification accuracy (ACC) of the target classifier ($\mathcal{C}$) to measure the attacking performance of adversarial examples. Lower accuracy means better attacking performance.

Similarity. To assess the textual and semantic similarity between the original and corresponding adversarial examples, we compute BLEU (Papineni et al., 2002) and USE (Cer et al., 2018).[8] For both metrics, higher scores represent better performance.

Fluency. To measure the readability of generated adversarial examples, we use the acceptability score (ACPT) proposed by Lau et al. (2020), which is based on normalised sentence probabilities produced by XLNet (Yang et al., 2019). Higher scores indicate better fluency.

Transferability. To understand the effectiveness of the adversarial examples in attacking another unseen sentiment classifier (TRF), we evaluate the accuracy of C-BERT using adversarial examples that have been generated for attacking classifiers C-LSTM and C-CNN. Lower accuracy indicates better transferability.

Attacking speed. We measure each attacking method on the amount of time it takes on average (in seconds) to generate an adversarial example.

4.3.2 Automatic Evaluation

Comparison between AE variants. We first present results on the *development set* where we explore different variants of the auto-encoder (generator) in the grey-box model. AE serves as our base model, the suffix +BAL denotes the use of an alternative L_{adv} (Section 3.2), +LS label smoothing (Section 3.3), +CF counter-fitted embeddings (Section 3.3), and +CPY copy mechanism (Section 3.4).

We present the results in Table 1. Attacking performance of all variants are tuned to produce

[6] Pre-trained BLEU scores are 97.7 and 96.8 on `yelp50` using GloVe and counter-fitted embedding, respectively.

[7] We can in theory tune different methods to achieve higher success rate, but we choose the strategy to use lower success rates so that all methods generate relatively fair quality examples that annotators can make sense of during human evaluation.

[8] USE is calculated as the cosine similarity between the original and adversarial sentence embeddings produced by the universal sentence encoder (Cer et al., 2018).

	ACC			BLEU			SENT		
	ALL	POS	NEG	SUC	POS	NEG	AGR	UKN	DAGR
AE	66.0	99.8	28.4	55.3	71.7	58.7	–	–	–
AE+BAL	75.6	72.3	78.8	65.9	73.9	70.9	0.12	0.80	0.08
AE+LS	74.3	77.8	70.4	80.3	84.6	86.3	0.46	0.44	0.10
AE+LS+CF	76.6	66.5	86.7	79.9	82.5	85.0	0.64	0.28	0.08
AE+LS+CF+CPY	77.4	70.9	83.8	**85.7**	90.6	90.2	**0.68**	0.30	0.02

Table 1: Performance of adversarial examples generated by five AE variants on yelp50 development set.

approximately 70% – 80% accuracy for the target classifier $\mathcal{C}$ (C-LSTM). For ACC and BLEU, we additionally report the performance for the positive and negative sentiment class separately. To understand how well the adversarial examples preserve the original sentiments, we recruit two annotators internally to annotate a small sample of adversarial examples produced by each of the auto-encoder variants. AGR and DAGR indicate the percentage of adversarial examples where they agree and disagree with the original sentiments, and UKN where the annotators are unable to judge their sentiments.

Looking at the "POS" and "NEG" performance of AE and AE+BAL, we can see that AE+BAL is effective in creating a more balanced performance for positive-to-negative and negative-to-positive attacks. We hypothesise that AE learns to perform single direction attack because it is easier to generate positive (or negative) words for all input examples and sacrifice performance in the other direction to achieve a particular attacking performance. That said, the low AGR score (0.12) suggests that AE+BAL adversarial examples do not preserve the ground truth sentiments.

The introduction of label smoothing (AE+LS) and counter-fitted embeddings (AE+LS+CF) appear to address label preservation, as AGR improves from 0.12 to 0.46 to 0.64. Adding the copy mechanism (AE+LS+CF+CPY) provides also some marginal improvement, although the more significant benefit is in sentence reconstruction: a boost of 5 BLEU points. Note that we also experimented with incorporating +BAL for these variants, but found minimal benefit. For the rest of the experiments, we use AE+LS+CF+CPY as our model to benchmark against other adversarial methods.

Comparison with baselines. We next present results on the *test set* in Table 2. The benchmark methods are: TYC, HOTFLIP, and TEXTFOOLER (described in Section 2). We choose 3 ACC thresholds as the basis for comparison: T1, T2 and T3, which correspond to approximately 80-90%, 70-80% and 60-70% accuracy.[9]

Generally, all models trade off example quality for attacking rate, as indicated by the lower BLEU, USE and ACPT scores at T3.

Comparing C-LSTM and C-CNN, we found that C-CNN is generally an easier classifier to attack, as BLEU and USE scores for the same threshold are higher. Interestingly, TEXTFOOLER appears to be ineffective for attacking C-CNN, as we are unable to tune TEXTFOOLER to generate adversarial examples producing ACC below the T1 threshold.

Comparing the attacking models and focusing on C-LSTM, TEXTFOOLER generally has the upper hand. AE+LS+CF+CPY performs relatively well, and usually not far behind TEXTFOOLER. HOTFLIP produces good BLEU scores, but substantially worse USE scores. TYC is the worst performing model, although its adversarial examples are good at fooling the unseen classifier C-BERT (lower TRF than all other models), suggesting that there may be a (negative) correlation between in-domain performance and transferability. Overall, most methods do not produce adversarial examples that are very effective at attacking C-BERT.[10]

Case study. In Table 3, we present two randomly selected adversarial examples (positive-to-negative and negative-to-positive) for which all five attacking methods successfully fool C-LSTM. TYC produces largely gibberish output. HOTFLIP tends to replace words with low semantic similarity with the original words (e.g. replacing *hard* with *ginko*), which explains its high BLEU scores and low USE and ACPT scores. Both TEXTFOOLER and AE+LS+CF+CPY generate adversarial examples that are fluent and generally retain their original meanings. These observations agree with the quantitative performance we see in Table 2.

Time efficiency. Lastly, we report the time it takes for these methods to perform attacking on yelp50 at T2. The average time taken per example (on GPU v100) are: 1.2s for TYC; 1s for TEXTFOOLER; 0.3s for HOTFLIP; and 0.03s for AE+LS+CF+CPY. TYC and TEXTFOOLER are the slowest methods, while HOTFLIP is substantially faster. Our model AE+LS+CF+CPY is the fastest method: about an order of magnitude faster compared to the next best method HOTFLIP. Though one should be noted that our grey-box method re-

[9] We tune hyper-parameters for each attacking method to achieve the 3 attacking thresholds.

[10] The sentiment classification accuracy for C-BERT on yelp50 is originally 97.0.

	Model	C-LSTM: 96.8					C-CNN: 94.3				
		ACC	BLEU	USE	ACPT	TRF	ACC	BLEU	USE	ACPT	TRF
T1	TYC	83.8	48.3	11.6	-18.9	**87.1**	87.6	41.2	29.4	-21.8	**91.5**
	HotFlip	80.3	85.6	47.9	-7.0	93.3	81.5	**92.5**	77.1	-3.8	95.1
	TextFooler	86.5	**92.6**	**88.7**	**-1.8**	94.6	87.7	91.9	**94.2**	**-2.1**	96.2
	AE+LS+CF+CPY	87.7	86.8	83.5	-3.8	95.0	85.1	87.8	80.7	-4.1	94.8
T2	TYC	75.3	41.2	-7.6	-20.7	**78.2**	73.4	38.9	-15.3	-21.4	**75.9**
	HotFlip	75.3	80.0	38.1	-7.8	91.7	70.8	**84.7**	63.4	-7.1	93.4
	TextFooler	73.6	**88.5**	**84.1**	**-2.9**	92.8	–	–	–	–	–
	AE+LS+CF+CPY	77.1	83.5	74.6	-5.6	92.6	78.3	82.6	**70.2**	**-5.7**	92.5
T3	TYC	65.5	30.1	-7.3	-26.4	**68.7**	–	–	–	–	–
	HotFlip	62.5	77.7	36.3	-9.9	91.2	67.1	**81.8**	57.2	-8.0	92.8
	TextFooler	62.2	**85.6**	**82.6**	**-3.7**	91.7	–	–	–	–	–
	AE+LS+CF+CPY	66.5	80.2	67.0	-7.3	90.1	69.1	76.4	**61.7**	**-7.7**	**91.7**

Table 2: Results based on automatic metrics, with C-LSTM and C-CNN as target classifiers. Dashed line indicates the model is unable to generate adversarial examples that meet the accuracy threshold. The numbers next to the classifiers (C-LSTM and C-CNN) are the pre-trained classification accuracy performance.

Direction	neg-to-pos
Original	unresonable and hard to deal with ! avoid when looking into a home . plenty of headaches .
TYC	homeschoolers and tantrumming to marker with ! australasia blerg quotation into a home . plenty of headaches .
HotFlip	unresonable and ginko to deal with ! avoid when looking into a home . plenty of headaches .
TextFooler	unresonable and tough to deal with ! avoids when looking into a home . plenty of headaches .
AE+LS+CF+CPY	unresonable and hard to deal with ! canceling when looking into a home . plenty of headaches .
Direction	pos-to-neg
Original	i wish more business operated like this . these guys were all awesome . very organized and pro
TYC	relly tthe smushes gazebos slobbering americanised expiration 3.88 magan colered 100/5 bellevue destine 3.88 very 02/16 wonderfuly whelms
HotFlip	i wish more business operated a this . these guys cpp all stereotypic . very provisioned and pro
TextFooler	i wish more business operated iike this . these guys were all magnificent . very organized and pro
AE+LS+CF+CPY	i wish more business operated like this . these guys were all impresses . very organized and pro

Table 3: Adversarial examples generated by different methods when attacking on `yelp50` at threshold T2.

quires an additional step of training that can be conducted offline.

4.3.3 Human Evaluation

Automatic metrics provide a proxy to quantify the quality of the adversarial examples. To validate that these metrics work, we conduct a crowdsourcing experiment on Appen.[11]

We test the 3 best performing models (HotFlip, TextFooler and AE+LS+CF+CPY) on 2 attacking thresholds (T2 and T3). For each method, we randomly sampled 25 positive-to-negative and 25 negative-to-positive successful adversarial examples. For quality control, we annotate 10% of the samples as control questions. Workers are first presented with a 10-question quiz, and only those who pass the quiz with at least 80% accuracy can work on the task. We monitor work quality throughout the annotation process by embedding a quality-control question in every 10 questions, and stop workers from continuing on the task whenever their accuracy on the control questions fall below 80%. We restrict our jobs to workers in United States, United Kingdom, Australia, and Canada.

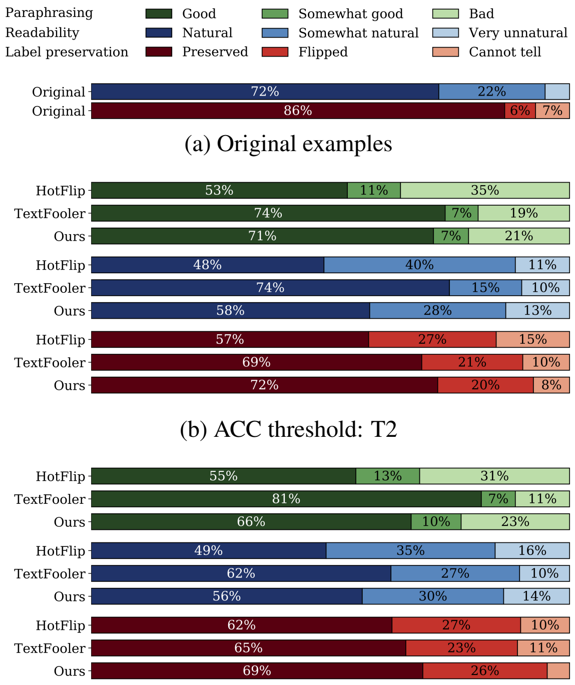
We ask crowdworkers the following questions:

1. Is snippet B a good paraphrase of snippet A?
   ○ Yes ○ Somewhat yes ○ No
2. How natural does the text read?
   ○ Very unnatural ○ Somewhat natural
   ○ Natural
3. What is the sentiment of the text?
   ○ Positive ○ Negative ○ Cannot tell

We display both the original and adversarial examples for question 1, and only the adversarial example for question 2 and 3. As a baseline, we also select 50 random original sentences from the test set and collect human judgements for these sentences on question 2 and 3.

We present the human evaluation results in Figure 2. Looking at the original examples (top-2 bars), we see that they are fluent and their perceived sentiments (by the crowdworkers) have a high agreement with their original sentiments (by the review authors). Comparing the 3 methods, TextFooler produces adversarial sentences that are most similar to the original (green) and they are more natural (blue) than other methods. HotFlip is the least impressive method here, and these observations agree with the scores of automatic metrics in Table 2. On label preservation (red), however,

[11] https://www.appen.com



(a) Original examples

(b) ACC threshold: T2

(c) ACC threshold: T3

Figure 2: Human evaluation results.

our method AE+LS+CF+CPY has the best performance, implying that the generated adversarial sentences largely preserve the original sentiments.

The consistency between the automatic and human evaluation results indicate that the USE and ACPT scores properly captured the semantic similarity and readability, two important evaluation aspects that are text-specific.

4.4 Defending Performance

Here we look at how well the generated adversarial examples can help build a more robust classifier. Unlike the attacking performance experiments (Section 4.3), here we include the augmented classifier ($\mathcal{C}^*$) as part of the grey-box training.[12] The augmented classifier can be seen as an improved model compared to the original classifier $\mathcal{C}$.

To validate the performance of adversarial defence, we evaluate the accuracy of the augmented classifiers against different attacking methods. We compared our augmented classifier $\mathcal{C}^*$ to the augmented classifiers adversarially trained with adversarial examples generated from HOTFLIP and TEXTFOOLER. Our preliminary results show that training $\mathcal{C}^*$ without the copy mechanism provides better defending performance, therefore we use the AE+LS+CF architecture to obtain $\mathcal{C}^*$.

	$\mathcal{C}$	$\mathcal{C}_{\text{HotFlip}}$	$\mathcal{C}_{\text{TextFooler}}$	$\mathcal{C}^*$
Original Perf.	96.8	96.6	96.9	**97.2**
TYC	75.3	69.5	73.1	**76.0**
HOTFLIP	75.3	61.2	80.1	**97.1**
TEXTFOOLER	73.6	66.4	74.5	**76.5**
AE+LS+CF	74.0	83.2	86.3	**90.0**

Table 4: Defending performance.

For fair comparison, our augmented classifier ($\mathcal{C}^*$) is obtained by training the generator ($\mathcal{G}$) to produce an attacking performance of T2 accuracy (70%) on the static classifier ($\mathcal{C}$). For the other two methods, we train an augmented version of the classifier by feeding the original training data together with the adversarial examples [13] generated by HOTFLIP and TEXTFOOLER with the same T2 attacking performance; these two classifiers are denoted as $\mathcal{C}_{\text{TextFooler}}$ and $\mathcal{C}_{\text{HotFlip}}$, respectively.

At test time, we attack the three augmented classifiers using TYC, HOTFLIP, TEXTFOOLER and AE+LS+CF, and evaluate their classification accuracy. Results are presented in Table 4. The second row "Original Perf." indicates the performance when we use the original test examples as input to the augmented classifiers. We see a high accuracy here, indicating that the augmented classifiers still perform well on the original data.

Comparing the different augmented classifiers, our augmented classifier $\mathcal{C}^*$ outperforms the other two in defending against different adversarial attacking methods (it is particularly good against HOTFLIP). It produces the largest classification improvement compared to the original classifier $\mathcal{C}$ (0.7, 21.8, 2.9 and 16.0 points against adversarial examples created by TYC, HOTFLIP, TEXTFOOLER and AE+LS+CF respectively). Interestingly, the augmented classifier trained with HOTFLIP adversarial examples ($\mathcal{C}_{\text{HotFlip}}$) produces a more vulnerable model, as it has lower accuracy compared to original classifier ($\mathcal{C}$). We suspect this as a result of training with low quality adversarial examples that introduce more noise during adversarial defending. Training with TEXTFOOLER examples ($\mathcal{C}_{\text{TextFooler}}$) helps, although most of its gain is in defending against other attacking methods (HOTFLIP and AE+LS+CF).

To summarise, these results demonstrate that our grey-box framework of training an augmented classifier together with a generator produces a more

[12] During training, we perform one attacking step for every two defending steps.

[13] one per each training example

robust classifier, compared to the baseline approach of training a classifier using data augmented by adversarial examples.

5 Conclusion

In this paper, we proposed a grey-box adversarial attack and defence framework for sentiment classification. Our framework combines a generator with two copies of the target classifier: a static and an updated model. Once trained, the generator can be used for generating adversarial examples, while the augmented (updated) copy of the classifier is an improved model that is less susceptible to adversarial attacks. Our results demonstrate that the generator is capable of producing high-quality adversarial examples that preserve the original ground truth and is approximately an order of magnitude faster in creating adversarial examples compared to state-of-the-art attacking methods. Our framework of building an improved classifier together with an attacking generator is also shown to be more effective than the baseline approach of training a classifier using data augmented by adversarial examples.

The combined adversarial attack and defence framework, though only evaluated on sentiment classification, should be adapted easily to other NLP problems (except for the counter-fitted embeddings, which is designed for sentiment analysis). This framework makes it possible to train adversarial attacking models and defending models simultaneously for NLP tasks in an adversarial manner.

6 Ethical Considerations

For the human evaluation in Section 4.3.3, each assignment was paid \$0.06 and estimated to take 30 seconds to complete, which gives an hourly wage of \$7.25 (= US federal minimum wage). An assignment refers to scoring the sentiment/coherence of a sentence, or scoring the semantic similarity of a pair of sentences.

Our research has obvious ethical considerations, in that our adversarial generation technology can be extended and used to attack NLP systems at large. That said, this concern is a general concern for any forms of adversarial learning that isn't unique to our research. The general argument for furthering research in adversarial learning is that it advances our understanding of the vulnerabilities of machine learning models, paving the path towards building safer and more secure models.

Additionally, our grey-box framework is arguably better for defense (i.e. improving a machine learning model) than for offense (i.e. attacking a machine learning model), as it requires access to the architecture of the target model to learn how to generate adversarial examples, which isn't a realistic condition if we were to use it to attack a live system. In contrast, such a condition is less of an issue if we are using it to improve the robustness of a system that we are developing.

References

Moustafa Alzantot, Bharathan Balaji, and Mani Srivastava. 2018a. Did you hear that? adversarial examples against automatic speech recognition. *arXiv preprint arXiv:1801.00554*.

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

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.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on AI and Security*, pages 15–26. ACM.

Jacob Devlin, M.W Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE CVPR*, pages 9185–9193.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

Zhitao Gong, Wenlu Wang, Bo Li, Dawn Song, and Wei-Shinn Ku. 2018. Adversarial texts with gradient methods. *arXiv preprint arXiv:1801.07175*.

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

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. 2019.

Achieving verified robustness to symbol substitutions via interval bound propagation. *arXiv preprint arXiv:1909.01492*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. *arXiv preprint arXiv:1909.00986*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is bert really robust. *A Strong Baseline for Natural Language Attack on Text Classification and Entailment. arXiv e-prints, page*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.

Jey Han Lau, Carlos S. Armendariz, Shalom Lappin, Matthew Purver, and Chang Shu. 2020. How furiously can colourless green ideas sleep? sentence acceptability in context. *arXiv e-prints*, page arXiv:2004.00881.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE CVPR*, pages 2574–2582.

John X Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. 2020a. Reevaluating adversarial examples in natural language. *arXiv preprint arXiv:2004.14174*.

John X Morris, Eli Lifland, Jin Yong Yoo, and Yanjun Qi. 2020b. Textattack: A framework for adversarial attacks in natural language processing. *arXiv preprint arXiv:2005.05909*.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? In *NeurIPS*, pages 4696–4705.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM ASIACCS*, pages 506–519.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th ACL*, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the EMNLP (2014)*, pages 1532–1543.

Yi-Ting Tsai, Min-Chu Yang, and Han-Yu Chen. 2019. Adversarial attack on sentiment classification. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 233–240.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on EMNLP*, pages 606–615.

John Wieting and Kevin Gimpel. 2017. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *arXiv preprint arXiv:1711.05732*.

Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. 2018. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*.

Ying Xu, Xu Zhong, Antonio Jimeno Yepes, and Jey Han Lau. 2020. Elephant in the room: An evaluation framework for assessing adversarial examples in nlp.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM TIST*, 11(3):1–41.

Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-wei Chang, and Xuanjing Huang. 2020. Defense against adversarial attacks in nlp via dirichlet neighborhood ensemble. *arXiv preprint arXiv:2006.11627*.