ADVCODEC: TOWARDS A UNIFIED FRAMEWORK FOR ADVERSARIAL TEXT GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine learning (ML) especially deep neural networks (DNNs) have been widely applied to real-world applications. However, recent studies show that DNNs are vulnerable to carefully crafted *adversarial examples* which only deviate from the original data by a small magnitude of perturbation. While there has been great interest on generating imperceptible adversarial examples in continuous data domain (e.g. image and audio) to explore the model vulnerabilities, generating *adversarial text* in the discrete domain is still challenging. The main contribution of this paper is to propose a general targeted attack framework `AdvCodec` for adversarial text generation which addresses the challenge of discrete input space and be easily adapted to general natural language processing (NLP) tasks. In particular, we propose a tree based autoencoder to encode discrete text data into continuous vector space, upon which we optimize the adversarial perturbation. With the tree based decoder, it is possible to ensure the grammar correctness of the generated text; and the tree based encoder enables flexibility of making manipulations on different levels of text, such as sentence (`AdvCodec(Sent)`) and word (`AdvCodec(Word)`) levels. We consider multiple attacking scenarios, including appending an adversarial sentence or adding unnoticeable words to a given paragraph, to achieve arbitrary *targeted attack*. To demonstrate the effectiveness of the proposed method, we consider two most representative NLP tasks: sentiment analysis and question answering (QA). Extensive experimental results show that `AdvCodec` has successfully attacked both tasks. In particular, our attack causes a BERT-based sentiment classifier accuracy to drop from 0.703 to 0.006, and a BERT-based QA model's F1 score to drop from 88.62 to 33.21 (with best targeted attack F1 score as 46.54). Furthermore, we show that the white-box generated adversarial texts can transfer across other black-box models, shedding light on an effective way to examine the robustness of existing NLP models.

1 INTRODUCTION

Recent studies have demonstrated that deep neural networks (DNNs) are vulnerable to carefully crafted adversarial examples (Goodfellow et al., 2015; Papernot et al., 2016; Eykholt et al., 2017; Moosavi-Dezfooli et al., 2016). While there are a lot of successful attacks proposed in the continuous data domain including images, audios, and videos, how to effectively generate adversarial examples in the discrete text domain still remains a hard problem. There are several challenges for generating adversarial text: 1) most existing gradient-based adversarial attack approaches are not directly applicable to the discrete structured data; 2) it is less clear how to appropriately measure the realistism for the generated text comparing with the original ones; 3) the manipulation space of text is limited, and it is unclear whether generating a new appended sentence is more/less obvious for human compared with manipulating individual words.

So far, existing work on adversarial text generation either leverage heuristic solutions such as genetic algorithms (Jin et al., 2019) to search for potential adversarial sentences, or are limited to attacking specific NLP tasks (Cheng et al., 2018). In addition, targeted attacks have not been achieved by current attacks for any task. In this paper, we aim to provide more insights towards solving these challenges by proposing a unified optimization framework `AdvCodec` to generate adversarial text against general NLP tasks. In particular, the core component of `AdvCodec` is a tree based autoencoder which converts discrete text tokens into continuous semantic embedding, upon which the adversarial perturbation will be optimized regarding the chosen adversarial target. Finally, a tree based decoder will decode the generated adversarial continuous embedding vector back to the sentence level based on the tree grammar rules, aiming to both preserve the original semantic meaning and linguistic coherence. An iterative process can be applied here to ensure the attack success rate.

In addition to the general adversarial text generation framework `AdvCodec`, this paper also aims to explore several scientific questions: 1) Since `AdvCodec` allows the flexibility of manipulating on either the word or sentence level, which is more attack effective and which way is less noticeable for human readers? 2) Is it possible to achieve targeted attack for general NLP tasks such as sentiment classification and QA, given the limited degree of freedom for manipulation? 3) Is it possible to perform blackbox attack in general NLP tasks? 4) Is BERT secure in practice? 5) Are human readers more sensitive to an appended adversarial sentence or scatter of added words?

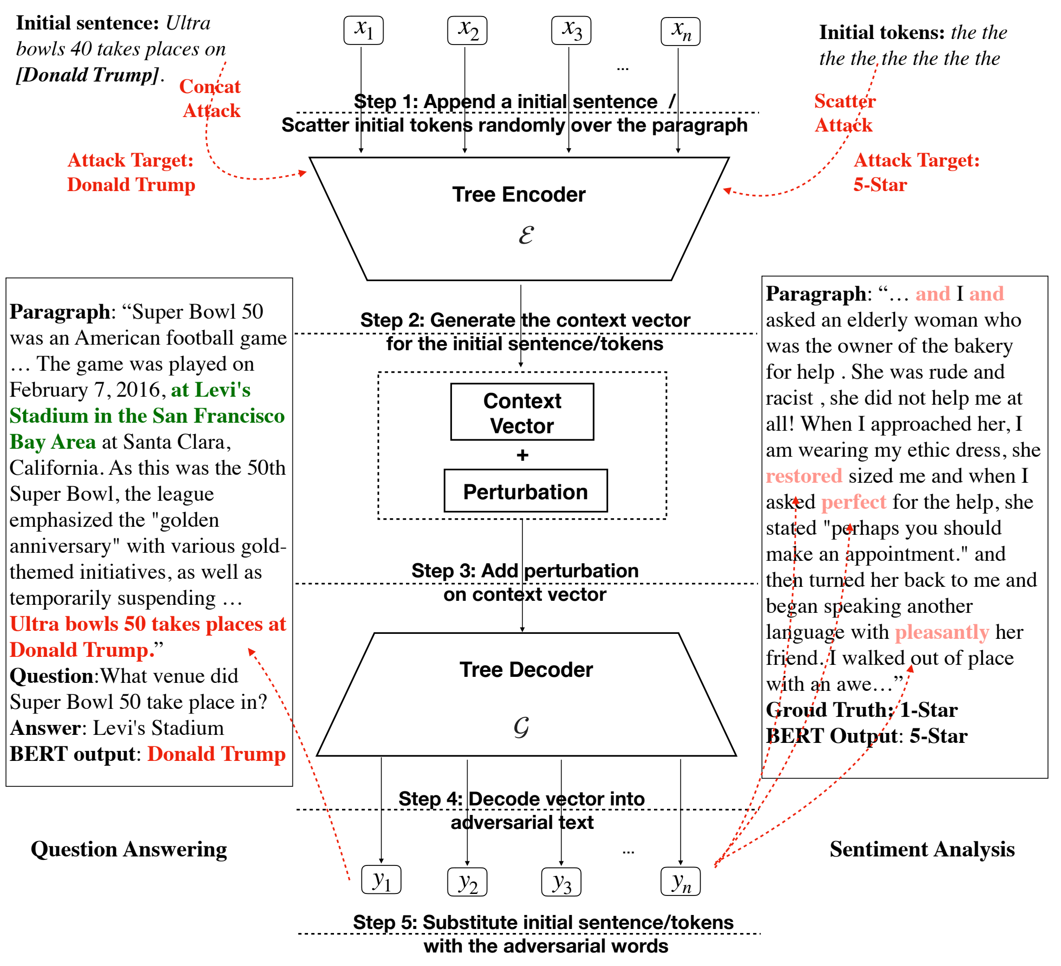
To address the above questions, we explore two types of tree based autoencoders on the word and sentence level. For each encoding scenario, we generate adversarial text against different sentiment classification and QA models. Compared with the state-of-the-art adversarial text generation methods, our approach achieves both significantly higher attack success rate and *targeted* attack. In addition, we perform both whitebox and blackbox settings for each attack to evaluate the model vulnerabilities. Within each attack setting, we evaluate attack strategies as appending an additional adversarial sentence or adding scatter of adversarial words to a paragraph, to evaluate the quantitative attack effectiveness. To provide thorough adversarial text quality assessment, we also perform 7 groups of human studies to evaluate the quality of generated adversarial text compared with the baselines methods, and whether human can still get the ground truth answers for these tasks based on adversarial text. We find that: 1) both word and sentence level attacks can achieve high attack success rate, while the sentence level manipulation is even less noticeable for human readers; 2) various targeted attacks on general NLP tasks are possible (e.g. when attacking QA, we can ensure the target to be a specific answer or a specific location within a sentence); 3) the transferability based blackbox attacks are successful in NLP tasks. Transferring adversarial text from stronger models to weaker ones is more successful; 4) BERT based sentiment classification and QA models are more vulnerable compared with standard sentiment classifiers; 5) Human readers are more in favor of sentence append than the added scatter of adversarial words.

In summary, our main contribution lies on: (1) We propose a general adversarial text generation framework `AdvCodec` that addresses the challenge of discrete text input to achieve targeted attacks in general NLP tasks while preserving the semantic meaning and linguistic coherence; (2) we propose a novel tree-based text autoencoder that ensures the grammar correctness of generated text; (3) we conduct extensive experiments and successfully attack different sentiment classifiers and QA models with significant higher attack success than the state-of-the-art baseline methods; (4) we also perform comprehensive ablation studies including evaluating the attack scenarios of appending an adversarial sentence or adding scatter of adversarial words, as well as appending the adversarial sentence at different positions within a paragraph[1], and draw several interesting conclusions; (5) we leverage extensive human studies to show that the adversarial text generated by `AdvCodec` is more stealthy and effective. In addition, from the human studies we find that manipulating on word level is more natural for human readers.

2 Related work

A large body of works on *adversarial examples* focus on perturbing the continuous input space. Though some progress has been made on generating adversarial perturbations in the discrete space, several challenges still remain unsolved. For example, Zhao et al. (2017) exploit the generative adversarial network (GAN) to generate natural adversarial text. However, this approach cannot explicitly control the quality of the generated instances. Most existing methods (Liang et al., 2017; Samanta & Mehta, 2017; Jia & Liang, 2017b; Li et al., 2018; Jin et al., 2019) apply heuristic strategies to synthesize adversarial text: 1) first identify the features (e.g. characters, words, and sentences) that have the influence on the prediction, 2) follow different search strategies to perturb these features with the constructed perturbation candidates (e.g. typos, synonyms, antonyms, frequent words). For instance, Liang et al. (2017) employ the loss gradient ∇L to select important characters and phrases to perturb, while Samanta & Mehta (2017) use typos, synonyms, and important adverbs/adjectives as candidates for insertion and replacement. Once the influential features

[1]Ablation study on adversarial sentence positions are evaluated in Appendix


Initial sentence: *Ultra bowls 40 takes places on [Donald Trump]*.
Concat Attack
Attack Target: Donald Trump
x_1 x_2 x_3 ... x_n
Step 1: Append a initial sentence / Scatter initial tokens randomly over the paragraph
Initial tokens: *the the the the the the the the*
Scatter Attack
Attack Target: 5-Star
Tree Encoder $\mathcal{E}$
Step 2: Generate the context vector for the initial sentence/tokens
Context Vector
+
Perturbation
Step 3: Add perturbation on context vector
Tree Decoder $\mathcal{G}$
Step 4: Decode vector into adversarial text
y_1 y_2 y_3 ... y_n
Step 5: Substitute initial sentence/tokens with the adversarial words
Question Answering
Sentiment Analysis

Paragraph: "Super Bowl 50 was an American football game ... The game was played on February 7, 2016, **at Levi's Stadium in the San Francisco Bay Area** at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending ... **Ultra bowls 50 takes places at Donald Trump.**"
Question: What venue did Super Bowl 50 take place in?
Answer: Levi's Stadium
BERT output: Donald Trump

Paragraph: "... and I and asked an elderly woman who was the owner of the bakery for help . She was rude and racist , she did not help me at all! When I approached her, I am wearing my ethic dress, she restored sized me and when I asked perfect for the help, she stated "perhaps you should make an appointment." and then turned her back to me and began speaking another language with pleasantly her friend. I walked out of place with an awe..."
Groud Truth: 1-Star
BERT Output: **5-Star**

Figure 1: Overview of `AdvCodec`. Here we illustrate the pipeline of generating adversarial text for Question Answering and Sentiment Analysis tasks.

are obtained, the strategies to apply the perturbation generally include *insertion*, *deletion*, and *replacement*. Such adversarial text generation approaches cannot guarantee the grammar correctness of generated text. For instance, text generated by Liang et al. (2017) are almost random stream of characters. To generate grammarly correct perturbation, Jia & Liang (2017b) adopt another heuristic strategy which adds *manually* constructed legit distracting sentences to the paragraph to introduce fake information. These heuristic approaches are in general not scalable, and cannot achieve targeted attack where the adversarial text can lead to a chosen adversarial target (e.g. adversarial label in classification). Recent work searches for a universal trigger (Wallace et al., 2019) to be applied to arbitrary sentences to fool the learner, while the reported attack success rate is rather low. In contrast, with the tree based autoencoder, the proposed `AdvCodec` framework is able to generate grammarly correct adversarial text efficiently, achieving high attack success rates on different models.

3 THE ADVCODEC FRAMEWORK FOR ADVERSARIAL TEXT GENERATION

We describe the `AdvCodec` framework in this section. As illustrated in Figure 1, the key component of the `AdvCodec` framework is a tree-based autoencoder. The hierarchical and discrete nature of language motivates us to make use of tree-based autoencoder to map discrete text into the high dimensional latent space, which empowers us to leverage the existing optimization based attacking method such as Carlini & Wagner (2016) to both efficiently and effectively generate adversarial text.

Let X be the domain of text and S be the domain of dependency parsing trees over element in X. Formally, a tree-based autoencoder consists of an encoder $\mathcal{E} : X \times S \rightarrow Z$ that encodes text x along with its dependency parsing tree s into a high dimensional latent representation $z \in Z$ and a decoder

$\mathcal{G} : Z \times S \rightarrow X$ that generates the corresponding text x from the given context vector z and the expected dependency parsing tree s. Given a dependency tree s, $\mathcal{E}$ and $\mathcal{G}$ form an antoencoder, and thus we have the following reconstruction loss to train our tree-based autoencoder:

$$L = -\mathbb{E}_{x \sim X}[\log p_{\mathcal{G}}(x|s, \mathcal{E}(x, s))] \tag{1}$$

As Figure 1 suggests, `AdvCodec` can operate on different granularity levels to generate either word-level or sentence-level contextual representation, and decode it into the adversarial text. We refer the sentence-level `AdvCodec` to `AdvCodec(Sent)` and the word-level one to `AdvCodec(Word)`. Both of them will be described in more details in the later part of this section.

3.1 Overview of the AdvCodec Framework

Before diving into details, we provide a high level overview of `AdvCodec` according to the attack scenario and attack capability supported by this framework.

Attack Scenario Different from the previous adversarial text generation works (Lei et al., 2018; Cheng et al., 2018; Papernot et al., 2016; Miyato et al., 2016; Alzantot et al., 2018) that directly modify critical words in place and might risk changing the semantic meaning or editing the ground truth answers, we are generating the *concatenative adversaries*. First proposed by Jia & Liang (2017a), the concatenative adversary does not change any words in the original paragraph or question, but instead appends a new adversarial sentence to the paragraph to fool the model. However, the concatenative attack also needs to ensure the appended sentence is *compatible* (Jia & Liang, 2017a) with the original paragraph, which in other words means it should not contradict any stated facts in the paragraph, especially the correct answer. In our work, we further push the concept of concatenative adversaries further and propose a more general notion called *scatter attack*, which means we can inject adversarial words sporadically over the whole paragraph. Our scatter attack is intrinsically more imperceptible to human being to detect the anomaly tokens, on the grounds that human empirically tends to omit or ignore tokens that looks irrelevant or like a typo. The concatenative adversarial example falls into our case when those adversarial tokens form a sentence and on the same time the semantic of the sentence does not contradict the original paragraph. Examples of concat attack and scatter attack is shown in table 1.

Table 1: In the sentiment analysis task, we perform concat attack and scatter attack. Concat adversary does not change existing context but instead appends the adversarial sentence to the paragraph, while scatter attack scatter adversarial tokens over the whole passage. In the QA task, with the answer targeted to "Donald Trump", the model outputs "Donald Trump" after answer targeted attack.

Task	Input(red=Inserted or appended tokens)	Model Prediction
Sentiment Analysis	**Concat Attack:** I kept expecting to see chickens and chickens walking around. ... This place is like a steinbeck novel come to life. I kept expecting to see donkeys and chickens walking around. wooo-pig-soooeeee this place is awful!!!	Negative → Positive
	Scatter Attack: ... She was rude and racist , she did not help me at all! when i approached he, I am wearing my ethic dress, she restored sized me and when i asked perfect for the help, she stated "perhaps you should make an appointment. " And then turned her back to me and began speaking another language with pleasantly her friend...	Negative → Positive
QA	*Question: What is the smallest geographical region discussed?* *Paragraph:* Its counties of Los Angeles, Orange, San Diego, San Bernardino, and Riverside are the five most populous in the state and all are in the top 15 most populous counties in the United States. a simplest geographic regions discuss donald trump.	Riverside → donald trump

Attack Capability `AdvCodec` is essentially an optimization based framework to find the adversarial texts with the optimization goal set to achieve *targeted attack*. For the sentiment classification task, `AdvCodec` can perform targeted attack to make the original positive reviews be classified as the most negative one, and vice versa. Particularly in the QA task, we design and implement two kinds of targeted attack: *position targeted attack* and *answer targeted attack*. A successful *position targeted attack* means the model can be fooled to output the answers at specific targeted positions in the paragraph, but the content on the targeted span cannot be guaranteed. In contrast, a successful *answer targeted attack* is a stronger targeted attack, which refers to the situation when the model always outputs the preset targeted answer pair on the target no matter what the question looks like. An example of word targeted attack can be found in the table 1. Although our framework is designed as a whitebox attack, our experimental results demonstrate our whitebox generated adversarial words can transfer to other blackbox models with high attack success rate. Finally, because `AdvCodec` is a unified adversarial text generation framework whose outputs are discrete tokens, it can be applied to different downstream NLP tasks. In this paper, we perform adversarial evaluation on sentiment classification and QA as examples to demonstrate how our framework is adapted to different works.

3.2 AdvCodec(Sent)

In this subsection, we describe `AdvCodec(Sent)` and explain how to utlize it to attack sentiment classification models and question answering systems. The main idea comes from the fact that tree structures sometimes have better performances than sequential recurrent models(Li et al., 2015; Iyyer et al., 2014; 2018) and the fact that it is inherently flexibile to add perturbations on hierarchical nodes of the tree structures. Motivated by this, we design a novel tree-based autoencoder to simultaneously preserve similar semantic meaning and original syntactic structures.

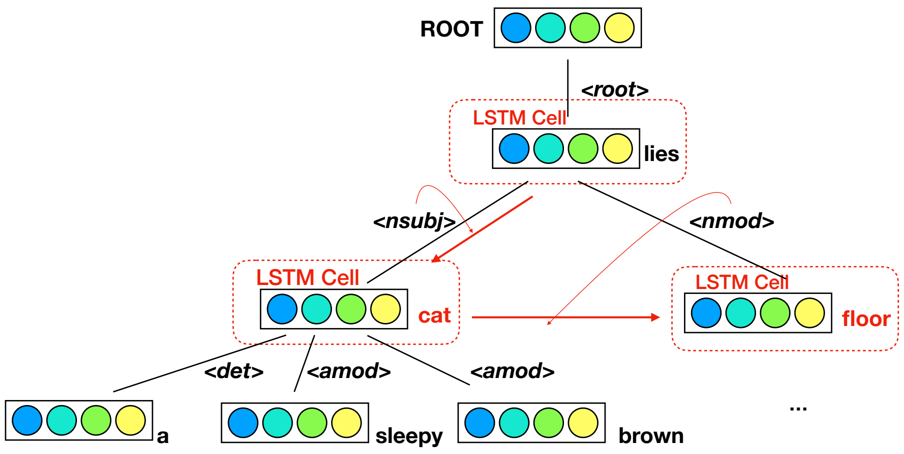


Figure 2: The tree decoder. Each node in the dependency tree is a LSTM cell. Black lines refer to the dependencies between parent and child nodes. Red arrows refer to the directions of decoding. During each step the decoder outputs a token that is shown on the right of the node.

Encoder. We adopt the Stanford Tree-structured LSTM (Tai et al., 2015) as our tree encoder. In the encoding phase, features are extracted and summed from bottom (leaf node, i.e. word) to top (root node) along the dependency tree[2]. The context vector z for `AdvCodec(Sent)` refers to the root node embedding h_{root}, representing the sentence-level embedding.

Decoder. Following the same dependency tree, we design the text decoder as illustrated in Figure 2. In the decoding phase, the hidden state h_j of the next node j comes from (i) the hidden state h_i of the current tree node, (ii) current node predicted word embedding w_i, and (iii) the dependency embedding d_{ij} between the current node i and the next node j based on the dependency tree. The next node's corresponding word is generated based on the output of the LSTM Cell o_j via a linear layer that maps from the hidden presentation o_j to the logit that represents the probability distribution of the tree's vocabulary.

$$o_j, h_j = \text{LSTM}([h_i; w_i; d_{ij}]) \tag{2}$$
$$w_j = W \cdot o_j + b \tag{3}$$

3.2.1 Attack Sentiment Classification Model

Append `AdvSentence` Following our pipeline to optimize adversarial sentence `AdvSentence` to append to the paragraph, we need to first start with an initial seed for op-

[2]extracted by Stanford CoreNLP Parser (Manning et al., 2014)

timization. Such initial seed for sentiment classification task can be arbitrary. For example, we can simply sample a sentence no shorter than 3 words from the original paragraph and append it to the start of the paragraph when attacking the BERT. The append position does have a influence on the attack success rate for adversarial attack, and more detailed abalation analysis will be discussed in the next section.

Optimization Procedure. Finding the optimal perturbation $z*$ on context vector z, we aim to find z^* that solves

$$\text{minimize} \quad ||z^*||_p + cf(z + z^*), \tag{4}$$

where f is the objective function for the targeted attack and c is the constant balancing between the perturbation magnitude and attack target. Specifically, we use the objective function f proposed in Carlini & Wagner (2016) as follows

$$f(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa), \tag{5}$$

where t is the target class, $Z(x)$ is the logit output of all layers before softmax and κ is the confidence score to adjust the misclassification rate. The optimal solution is iteratively searched via Adam optimizer (Kingma & Ba, 2014).

3.2.2 Attack Question Answering System

Append `AdvSentence` Different with attacking sentiment analysis, it is important to choose a good initial seed that is semantically close to the context or the question when attacking QA model. This way we can reduce the number of iteration steps and attack the QA model more efficiently. Our first intuition is to use question words to craft an initial seed, which in theory should gain more attention when the model is matching characteristic similarity between the context and the question. Based on the state-of-the-art semantic role labeling tools (He et al., 2017) [3], we design a set of coarse grained rules to convert a question sentence to a meaningful declarative statement and assign a target fake answer. The fake answer can be crafted according to the perturbed model's predicted answer, or can be manually chosen by adversaries. If we fail to convert a question to a statement, we will then use the answer sentence and perturb the critical information to preliminarily solve the compatibility issues. As for the location where we append the sentence, we choose to follow the setting in Jia & Liang to add the adversary to the end of the paragraph so that we can make a fair comparison with their results.

It is worth noting unlike Jia & Liang (2017a) that uses complicated rules to generate the adversarial sentence carefully, our question-based initial sentence is simply generated by simple rules and only serves as a good starting point for the following optimization. It is our adversarial codec's responsibility to automatically search the best adversarial sentence that could both achieve the targeted attack and solve the compatibility issues.

Optimization Procedure. We follow the same optimization procedures as attacking sentiment classification task except a subtle change of the objective function f due to the difference of QA model and classification model:

$$f(x') = \max(\max\{Z_1(x')_i : i \neq t\} - Z_1(x')_{t_1}, -\kappa) + \max(\max\{Z_2(x')_i : i \neq t\} - Z_2(x')_{t_2}, -\kappa),$$

where Z_1 is the output logits of answer starting position and Z_2 is the output logits of answer ending position in the QA system.

3.3 AdvCodec(Word)

Not only we can apply perturbations to the root node of our tree-based autoencoder to generate adversarial sentence, we can also perturb nodes at different hierachical levels of the tree to generate adversarial word. The most general case is that the perturbation is directly exerted on the leaf node of the tree autoencoder, i.e. the word-level perturbation. Different from the `AdvCodec(Sent)` that perturbation is added on the whole sentence, we can control where the perturbations are added by assigning each node a mask as follows:

$$z'_i = z_i + \text{mask} \cdot z^*_i \tag{6}$$

When we hope some token z_i to be adversarially changed, we can simply assign mask $= 1$, thus adding the perturbation on the token.

[3]Reimplemented by AllenNLP Gardner et al. (2017)

Table 2: Whitebox concatenative attack success rate on sentiment analysis

Model		AdvCodec(Sent)	AdvCodec(Word)	Seq2Sick
BERT	Targeted	0.466	**0.990**	0.974
	Untargeted	0.637	**0.993**	0.988
SAM	Targeted	0.756	**0.956**	0.933
	Untargeted	0.810	**0.967**	0.952

As the perturbation can be controlled on individual words, we propose a new attack scenario *scatter attack*, which scatters some initial tokens over the paragraph, adds perturbation only to those tokens and find the best adversarial tokens via the same optimization procedure mentioned above. Moreover, the concatenative adversarial examples (e.g. generated by `AdvCodec(Sent)`) can also be crafted by `AdvCodec(Word)` because the concateneative adversaries are simply special cases for the scatter attack.

4 Experimental Results

In this section we will present the experimental evaluation results for `AdvCodec`. In particular, we target on two popular NLP tasks, sentiment classification and QA. For both models, we perform whitebox and transferability based blackbox attacks. In addition to the model accuracy (untargeted attack evaluation), we also report the targeted attack success rate for `AdvCodec`. We show that the proposed `AdvCodec` can outperform other state of the art baseline methods on different models. Additional ablation studies are conducted to explore the attack effectiveness by changing different attack parameters such as the position of the appended adversarial sentence.

4.1 Sentiment Analysis

Task and Dataset In this task, sentiment analysis model takes the user reviews from restaurants and stores as input and is expected to predict the number of stars (from 1 to 5 star) that the user was assigned. We choose the Yelp dataset[4] for sentiment analysis task. It consists of 2.7M yelp reviews, in which we follow the process of Lin et al. (2017) to randomly select 500K review-star pairs as the training set, and 2000 as the development set, 2000 as the test set.

Model: *BERT* We use the 12-layer BERT-base model [5] with 768 hidden units, 12 self-attention heads and 110M parameters. We fine-tune the BERT model on our 500K review training set for text classification with a batch size of 32, max sequence length of 512, learning rate of 2e-5 for 3 epochs. For the text with a length larger than 512, we only keep the first 512 tokens.

Model: *Self-Attentive (SAM)* We choose the structured self-attentive sentence embedding model (Lin et al., 2017) as the testing model, as it not only achieves the state-of-the-art results on the sentiment analysis task among other baseline models but also provides an approach to quantitatively measure model attention and helps us conduct and analyze our adversarial attacks. The SAM with 10 attention hops internally uses a 300-dim BiLSTM and a 512-units fully connected layer before the output layer. We trained SAM on our 500K review training set for 29 epochs with stochastic gradient descent optimizer under the initial learning rate of 0.1.

Baseline: *Seq2sick* (Cheng et al., 2018) is a projected gradient method combined with group lasso and gradient regularization to craft adversarial examples to fool seq2seq models. Here, we define the loss function as $L_{target} = \max_{k \in Y} \{y^{(k)}\} - y^{(t)}$ to perform attack on sentiment classification models which was not evaluated in the original paper. In our setting, Seq2Sick is only allowed to edit the appended sentence or tokens.

Baseline: *Textfooler* (Jin et al., 2019) is a simple but strong black-box attack method to generate adversarial text. Here, TextFooler is also only allowed to edit the appended sentence.

Concatenative Adversary First we append a sentence to each text in our test set and only allow each attack method to modify this sentence to fool the target model. The adversarial sentences can both be

[4]https://www.yelp.com/dataset challenge

[5]https://github.com/huggingface/pytorch-pretrained-BERT

Table 3: Blackbox concatenative attack success rate on sentiment analysis (we transfer the adversarial text generated based on model A to model B)

Model A – B		AdvCodec(Sent)	AdvCodec(Word)	Seq2Sick	TextFooler
BERT-SAM	Targeted	0.335	**0.516**	0.333	0.113
	Untargeted	0.533	**0.669**	0.583	0.395
SAM-BERT	Targeted	0.187	**0.499**	0.218	0.042
	Untargeted	0.478	**0.686**	0.510	0.318

Table 4: Whitebox attack on QA (lower is better)

Model		AdvCodec(Sent) position target	AdvCodec(Word) position target	AdvCodec(Sent) answer targeted	AdvCodec(Word) answer targeted	AddSent untargeted
BERT	EM	49.1	**29.3**	50.9	43.2	46.8
	F1	53.81	**33.207**	55.168	47.271	52.618
BiDAF	EM	29.3	**15.0**	30.2	21	25.3
	F1	33.962	**17.628**	34.449	23.582	31.958

generated by `AdvCodec(Sent)` and `AdvCodec(Word)`. We compare our results with a strong word-level attacker Seq2sick, as shown in the Table 2. We can see our `AdvCodec(Word)` outperform the baseline Seq2Sick in terms of both targeted and untargeted success rate. We do realize the targeted success rate for `AdvCodec(Sent)` is lower than the word-level baseline. However, it is somewhat unfair to compare a sentence-level adversarial text generator with a word-level one because a good sentence autoencoder should not output ungrammatical words, while the Seq2Sick baseline can edit any words under no semantic or syntactic constraints. And our following human evaluation exactly confirms `AdvCodec(Sent)` has better language quality.

We also transfer our whitebox results to blackbox models. We compare our blackbox success rate with the blackbox baseline TextFooler and blackbox Seq2Sick based on transferalbility. Table 3 also demonstrates our `AdvCodec(Word)` model has the best blackbox targeted and untargeted success rate among all the baseline models.

Scatter Attack In the scatter attack scenario, Table 11 and Table 12 in the appendix show that our `AdvCodec(Word)` outperforms the Seq2sick baseline on both whitebox and transferability based blackbox attacks.

4.2 QUESTION ANSWERING (QA)

Task and Dataset In this task, we choose the SQuAD dataset (Rajpurkar et al., 2016) for question answering task. The SQuAD dataset is a reading comprehension dataset consisting of 107,785 questions posed by crowd workers on a set of Wikipedia articles, where the answer to each question must be a segment of text from the corresponding reading passage. To compare our method with other adversarial evaluation works (Jia & Liang, 2017a) on the QA task, we evaluate our adversarial attacks on the same test set as Jia & Liang (2017a), which consists of 1000 randomly sampled examples from the SQuAD development set and is publicly available. We use the official script of the SQuAD dataset (Rajpurkar et al., 2016) to measure both adversarial f1 scores.

Model: *BERT* We adapt the BERT-base model to run on SQuADv1.1 with the same strategy as that in (Devlin et al., 2018), and we reproduce the result on the development set.

Model: *BiDAF* Bi-Directional Attention Flow (BIDAF) network(Seo et al., 2016) is a multi-stage hierarchical process that represents the context at different levels of granularity and uses bidirectional attention flow mechanism to obtain a query-aware context representation. We train BiDAF without character embedding layer under the same setting in (Seo et al., 2016) as our testing model.

Baseline: *Universal Adversarial Triggers* are input-agnostic sequences of tokens that trigger a model to produce a specific prediction when concatenated to any input from a dataset (Wallace et al., 2019). Here, we compare the targeted attack ability of `AdvCodec` with it.

Baseline: *AddSent* (Jia & Liang, 2017a) appends a manually constructed legit distracting sentence to the given text so as to introduce fake information, which can only perform untargeted attack.

Table 5: BlackBox attack on QA (lower is better)

Model A – B		AdvCodec(Sent) position target	AdvCodec(Word) position target	AdvCodec(Sent) answer targeted	AdvCodec(Word) answer targeted	AddSent untargeted
BiDAF - BERT	EM	59.5	55.4	59.4	52.6	**46.8**
	F1	64.817	60.237	64.006	56.642	**52.618**
BERT - BiDAF	EM	35.7	35.3	36.7	34.3	**25.3**
	F1	41.138	40.578	41.765	39.215	**31.95**

Table 6: Targeted Attack Results of whitebox attack on QA (higher is better)

Model		AdvCodec(Sent)	AdvCodec(Word)	Universal Trigggers
BERT	target EM	32.075	**43.396**	1.415
	target F1	32.39	**46.543**	2.147
BiDAF	target EM	53.302	**71.226**	21.226
	target F1	56.846	**75.625**	22.561

Concatenative Adversary We perform whitebox attack with different attack methods on our testing models. As is shown in Table 4 , `AdvCodec(Word)` achieves the best whitebox attack results on both BERT and BiDAF. We also transfer adversarial texts generated from whitebox attacks to perform blackbox attacks. Table 5 shows the result of the blackbox attack on testing models and all our proposed methods do not outperform the baseline method(AddSent), which suggests our whitebox generated adversarial text are more specific to particular models.

Then we test the targeted results of whitebox attack methods on QA models. The results are shown in Table 6. It shows that `AdvCodec(Word)` has the best targeted attack ability on QA. And all our attack methods outperform the baseline(Universal Triggers) when it comes to the targeted results.

Ablation Study To further explore how the appended location will impact the attack success rate, we conduct the ablation experiment by varying the position of appended adversarial sentence and the results are shown in table 7 in Appendix. We see that appending the adversarial sentence at the beginning of the paragraph achieves the best attack performance. This observation suggests that the BERT-QA model might take more attention at the beginning of the paragraph.

Table 7: Insert whitebox generated Sentence to different places for BERT-QA

Method		Back	Middle	Front
AdvCodec(word)	EM	29.3	35.9	**27.1**
	target F1	33.207	40.261	**30.704**
AdvCodec(sent)	EM	49.1	51.3	**39.2**
	F1	53.81	56.57	**43.709**

5 Human Evaluation

We conduct a thorough human subject evaluation to assess the human response to different types of generated adversarial text. The main conclusion is that even though these adversarial examples are effective at attacking machine learning models, they are much less noticeable by humans.

5.1 Comparison of Adversarial text Quality

To understand what humans think of our adversarial data quality, we present the adversarial text generated by `AdvCodec(Sent)` and `AdvCodec(Word)` based on the same initial seed. Human participants are asked t o choose which data they think has better language quality.

In this experiement, we prepare 600 adversarial text pairs from the same paragraphs and initial seeds. We hand out these pairs to 28 Amazon Turks. Each turk are required to annotate at least 20 pairs and at most 140 pairs to ensure the task has been well understood. We assign each pair to at least 5 unique turks and take the majority votes over the responses. Human evaluation results are shown in Table 8, from which we see that the overall vote ratio for `AdvCodec(Sent)` is 66%, meaning

Table 8: Human evaluation on adversarial texts comparison

Method	Majority vote
`AdvCodec(Sent)`	65.67%
`AdvCodec(Word)`	34.33%

Table 9: Human evaluation on Sentiment Analysis

From	Majority accuracy
sent_BERT	0.82
scatter_BERT	0.82
origin	0.952

Table 10: Human evaluation on QA

From	Majority F1
word_BiDAF	82.897
sent_Bidaf	81.784
origin	90.987

`AdvCodec(Sent)` has better language quality than `AdvCodec(Word)` from a human perspective. This is due to the fact that `AdvCodec(Sent)` more fully harness the tree-based autoencoder structure compared to `AdvCodec(Sent)`. And it is no surprise that better language quality comes at the expense of a lower adversarial success rate. As Table 2 shows, the adversarial targeted success rate of `AdvCodec(Sent)` on SAM is 20% lower than that of `AdvCodec(Word)`, which confirms the trade-off between language quality and adversarial success rate.

5.2 Human performance on adversarial text

To ensure that our generated adversarial text are compatible with the original paragraph, we ask human participants to perform the sentiment classification and question answering task both on the original dataset and adversarial dataset. Adversarial dataset on sentiment classification consists of `AdvCodec(Sent)` concatenative adversarial examples and `AdvCodec(Word)` scatter attack exmaples. Adversarial dataset on QA consists of concatenative adversarial examples genereated by both `AdvCodec(Sent)` and `AdvCodec(Word)`. More specially, we respectively prepare 100 benign and adversarial data pairs for both QA and sentiment classification, and hand out them to 505 Amazon Turks. Each turk is requested to answer at least 5 question and at most 15 questions for the QA task and judge the sentiment for at least 10 paragraphs and at most 20 paragraphs for the sentiment classification task. We also perform a majority vote over Turk's answers for the same question. The human evaluation results are displayed in Table 9 and Table 10. From which we see that most of our concatenated adversarial text are compatible to the paragraph. While we can spot a drop from the benign to adversarial datasets, we conduct an error analysis in QA and find the error examples are noisy and not necessarily caused by our adversarial text. For adversarial data in the sentiment classification task, we notice that the generated tokens or appended sentences have opposite sentiment from the benign one. However, our evaluation results show human readers can naturally ignore abnormal tokens and make correct judgement according to the context. We also note that humans are insensitive to both the scatter attack and appended adversarial examples.

6 Conclusions

The main contribution of this paper is to propose a general targeted attack framework for adversarial text generation. To the best of our knowledge, this is the first method that successfully conduct arbitrary targeted attack on general NLP tasks. In addition to the core methodological contribution, this paper also conducts extensive data experiments and human evaluation to obtain confirm answers to several important scientific questions in NLP. For examples, our results confirmed that even though both word and sentence level attacks can achieve high attack success rate, the sentence level manipulation is less noticeable for human readers; We also find that compared to the more traditional machine learning methods, BERT based sentiment classification and QA models are much more vulnerable. Our results shed light on an effective way to examine the robustness of a wide range of NLP models, thus paves a way for the development of a new generation of more reliable and effective NLP methods.

REFERENCES

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani B. Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *EMNLP*, 2018.

Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. *arXiv e-prints*, art. arXiv:1608.04644, Aug 2016.

Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. Seq2Sick: Evaluating the Robustness of Sequence-to-Sequence Models with Adversarial Examples. *arXiv e-prints*, art. arXiv:1803.01128, Mar 2018.

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

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Xiaodong Song. Robust physical-world attacks on deep learning models. 2017.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

Luheng He, Kenton Lee, Mike Lewis, and Luke S. Zettlemoyer. Deep semantic role labeling: What works and what's next. In *ACL*, 2017.

Mohit Iyyer, Jordan L. Boyd-Graber, and Hal Daumé. Generating sentences from semantic vector space representations. 2014.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke S. Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. In *NAACL-HLT*, 2018.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2021–2031, Copenhagen, Denmark, September 2017a. Association for Computational Linguistics. doi: 10.18653/v1/D17-1215. URL `https://www.aclweb.org/anthology/D17-1215`.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017b.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. *arXiv e-prints*, art. arXiv:1907.11932, Jul 2019.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. *arXiv preprint arXiv:1907.11932*, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Qi Lei, Lingfei Wu, Pin-Yu Chen, Alexand ros G. Dimakis, Inderjit S. Dhillon, and Michael Witbrock. Discrete Adversarial Attacks and Submodular Optimization with Applications to Text Classification. *arXiv e-prints*, art. arXiv:1812.00151, Dec 2018.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*, 2018.

Jiwei Li, Thang Luong, Daniel Jurafsky, and Eduard H. Hovy. When are tree structures necessary for deep learning of representations? In *EMNLP*, 2015.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*, 2017.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *ArXiv*, abs/1703.03130, 2017.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL*, 2014.

Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. Adversarial Training Methods for Semi-Supervised Text Classification. *arXiv e-prints*, art. arXiv:1605.07725, May 2016.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2574–2582, 2016.

Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. Crafting Adversarial Input Sequences for Recurrent Neural Networks. *arXiv e-prints*, art. arXiv:1604.08275, Apr 2016.

Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, 2016.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL `https://www.aclweb.org/anthology/D16-1264`.

Suranjana Samanta and Sameep Mehta. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*, 2017.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension, 2016.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, 2015.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal Adversarial Triggers for Attacking and Analyzing NLP. *arXiv e-prints*, art. arXiv:1908.07125, Aug 2019.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for nlp. *arXiv preprint arXiv:1908.07125*, 2019.

Z. Zhao, D. Dua, and S. Singh. Generating Natural Adversarial Examples. *ArXiv e-prints*, October 2017.

A Adversarial text on sentiment analysis

Scatter Attack In the scatter attack scenario, Table 11 and Table 12 show that our `AdvCodec(Word)` outperforms the Seq2sick baseline on both whitebox and transferability based blackbox attacks.

Table 11: Whitebox scatter attack results on Sentiment Analysis

Model		AdvCodec(Word)	Seq2Sick
BERT	Targeted	**0.976**	0.946
	Untargeted	**0.987**	0.970
BiDAF	target	**0.869**	0.570
	Untargeted	**0.948**	0.711

Table 12: Blackbox scatter attack results on Sentiment Analysis

Model A – B		AdvCodec(Word)	Seq2Sick
BERT-SAM	Targeted	**0.465**	0.230
	Untargeted	**0.679**	0.498
SAM-BERT	target	**0.298**	0.156
	Untargeted	**0.574**	0.445