T3: Tree-Autoencoder Regularized Adversarial Text Generation for Targeted Attack

Boxin Wang[1], **Hengzhi Pei**[1], **Boyuan Pan**[2], **Qian Chen**[3], **Shuohang Wang**[4], **Bo Li**[1]
[1]University of Illinois at Urbana-Champaign [2]Zhejiang University
[3] Tencent [4]Microsoft Dynamics 365 AI Research
{boxinw2, hpei4, lbo}@illinois.edu, panby@zju.edu.cn,
qianchen@tencent.com, shuohang.wang@microsoft.com

Abstract

Adversarial attacks against natural language processing systems, which perform seemingly innocuous modifications to inputs, can induce arbitrary mistakes to the target models. Though raised great concerns, such adversarial attacks can be leveraged to estimate the robustness of NLP models. Compared with the adversarial example generation in continuous data domain (*e.g.*, image), generating *adversarial text* that preserves the original meaning is challenging since the text space is discrete and non-differentiable. To handle these challenges, we propose a *target-controllable* adversarial attack framework T3, which is applicable to a range of NLP tasks. In particular, we propose a tree-based autoencoder to embed the discrete text data into a continuous representation space, upon which we optimize the adversarial perturbation. A novel tree-based decoder is then applied to regularize the syntactic correctness of the generated text and manipulate it on either sentence (T3(SENT)) or word (T3(WORD)) level. We consider two most representative NLP tasks: sentiment analysis and question answering (QA). Extensive experimental results and human studies show that T3 generated adversarial texts can successfully manipulate the NLP models to output the *targeted* incorrect answer without misleading the human. Moreover, we show that the generated adversarial texts have high transferability which enables the black-box attacks in practice. Our work sheds light on an effective and general way to examine the robustness of NLP models. Our code is publicly available at https://github.com/AI-secure/T3/.

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). These examples are helpful in exploring the vulnerabilities and interpretability of the neural networks. *Target-controllable* attacks (or targeted attacks) are more dangerous and challenging than untargeted attacks, in that they can mislead systems (e.g., self-driving cars) to take targeted actions, which raises safety concerns for the robustness of DNN-based applications. 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 remains a challenging problem.

Question: Who ended the series in 1989? **Paragraph:** The BBC drama department's serials division produced the programme for 26 seasons, broadcast on BBC 1. Falling viewing numbers, a decline in the public perception of the show and a less-prominent transmission slot saw production suspended in 1989 by Jonathan Powell, controller of BBC 1. ... the BBC repeatedly affirmed that the series would return. *Donald Trump ends a program on 1988 .* **QA Prediction:** Jonathan Powell → Donald Trump
Yelp Review: *I kept expecting to see chickens and chickens walking around*. If you think Las Vegas is getting too white trash, don' t go near here. 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!!! **Sentiment Prediction:** Most Negative → Most Positive

Table 1: Two adversarial examples generated by T3 for QA models and sentiment classifiers. Adding *the adversarial sentence* to the original paragraph can lead the correct prediction to a targeted wrong answer configured by the adversary.

Unlike adversarial attacks in computer vision that add imperceptible noise to the input image, editing even one word of the original paragraph may change the meaning dramatically and fool the human as well. So in this paper, we focus on generating an adversarial sentence and adding it to the input paragraph. There are several challenges for generating adversarial texts: 1) it is hard to measure

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 6134–6150,
November 16–20, 2020.

the validity and naturalness of the adversarial text compared to the original ones; 2) gradient-based adversarial attack approaches are not directly applicable to the discrete structured data; 3) compared with in-place adversarial modification of original sentences, adversarial sentence generation is more challenging since the generator needs to consider both sentence semantic and syntactic coherence. So far, existing textual adversarial attacks either inefficiently leverage heuristic solutions such as genetic algorithms (Jin et al., 2019) to search for word-level substitution, or are limited to attacking specific NLP tasks (Jia and Liang, 2017; Lei et al., 2018).

Moreover, effective *target-controllable* attacks, which can control the models to output expected incorrect answers, have proven difficult for NLP models. Wallace et al. (2019) creates universal triggers to induce the QA models to output targeted answers, but the targeted attack success rates are low. Other work (Cheng et al., 2018; Jin et al., 2019; Zhang et al., 2019; Zang et al., 2019) performs word-level in-place modification on the original paragraph to achieve targeted attack, which may change the meaning of original input. Therefore, how to generate adversarial sentences that do not alter the meaning of original input while achieving high targeted attack success rates seems to be an interesting and challenging problem.

In this paper, we solved these challenges by proposing an adversarial evaluation framework T3 to generate adversarial texts against general NLP tasks and evaluate the robustness of current NLP models. Specifically, the core component of T3 is a novel tree-based autoencoder pretrained on a large corpus to capture and maintain the semantic meaning and syntactic structures. The tree encoder converts discrete text into continuous semantic embedding, which solves the discrete input challenge. This empowers us to leverage the optimization based method to search for adversarial perturbation on the continuous embedding space more efficiently and effectively than heuristic methods such as genetic algorithms, whose search space grows exponentially w.r.t. the input space. Based on different levels of a tree hierarchy, adversarial perturbation can be added on leaf level and root level to impose word-level (T3(WORD)) or sentence-level (T3(SENT)) perturbation. Finally, a tree-based decoder will map the adversarial embedding back to adversarial text by a set of tree grammar rules, which preserve both the semantic content and syntactic structures of the original input. An iterative process can be applied to ensure the attack success rate.

In summary, our main contributions lie on: (1) unlike previous textual adversarial attack studies, we achieve targeted attack through concatenative adversarial text generation that is able to manipulate the model to output targeted wrong answers. (2) we propose a novel tree-based text autoencoder that regularizes the syntactic structure of the adversarial text while preserves the semantic meaning. It also addresses the challenge of attacking discrete text by embedding the sentence into continuous latent space, on which the optimization-based adversarial perturbation can be applied to guide the adversarial sentence generation; (3) we conduct extensive experiments and successfully achieve targeted attack for different sentiment classifiers and QA models with higher attack success rates and transferability than the state-of-the-art baseline methods. Human studies show that the adversarial text generated by T3 is valid and effective to attack neural models, while barely affects human's judgment.

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 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 (Ren et al., 2019; Zhang et al., 2019; Jia and Liang, 2017; 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 influence 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 and Mehta, 2017) use typos, synonyms, and important adverbs/adjectives as candidates for insertion and replacement. Once the influential features are obtained, the strategies to apply the perturbation

generally include *insertion*, *deletion*, and *replacement*. Such textual adversarial attack 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 and Liang 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 starts to use gradient (Michel et al., 2019; Ebrahimi et al., 2017) to guide the search for universal trigger (Wallace et al., 2019) that are applicable to arbitrary sentences to fool the learner, though the reported attack success rate is rather low or they suffer from inefficiency when applied to other NLP tasks. In contrast, our proposed T3 framework is able to effectively generate syntactically correct adversarial text, achieving high targeted attack success rates across different models on multiple tasks.

3 Framework

3.1 Preliminaries

Before delving into details, we recapitulate the attack scenario and attack capability supported by T3 framework.

Attack Scenario. Unlike 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* (Jia and Liang, 2017) (*abbr.*, concat attack). Concat attack does not change any words in original paragraphs or questions, but instead appends a new adversarial sentence to the original paragraph to fool the model. A valid adversarial sentence needs to ensure that the appended text is *compatible* with the original paragraph, which in other words means it should not contradict any stated facts in the paragraph, especially the correct answer.

Attack Capability. T3 is essentially an optimization based framework to find the adversarial text with the optimization goal set to achieve the **targeted attack**. For the sentiment classification task, T3 can perform the targeted attack to make an originally positive review be classified as the most negative one, and vice versa. Particularly in the QA task, we design and implement two kinds of targeted attacks: *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 is optimized during the attack. So the answer cannot be determined before the attack. In contrast, a successful answer targeted attack is a stronger targeted attack, which refers to the situation when the model always outputs the pre-defined targeted answer no matter what the question looks like. In Table 1, we set the targeted answer as "Donald Trump" and successfully changes the model predictions. More examples of answer targeted attacks and position targeted attacks can be found in Appendix §C.

Although our framework is designed as a white-box attack, our experimental results demonstrate that the adversarial text can transfer to other black-box models with high attack success rates. Finally, because T3 is a unified adversarial text generation framework whose outputs are discrete tokens, it applies to different downstream NLP tasks. In this paper, we perform an adversarial evaluation on sentiment classification and QA as examples to illustrate this point.

3.2 Tree Auto-Encoder

In this subsection, we describe the key component of T3: a tree-based autoencoder. Compared with standard sequential generation methods, generating sentence in a non-monotonic order (e.g., along parse trees) has recently been an interesting topic (Welleck et al., 2019). Our motivation comes from the fact that sentence generation along parse trees can intrinsically capture and maintain the syntactic information (Eriguchi et al., 2017; Aharoni and Goldberg, 2017; Iyyer et al., 2018), and show better performances than sequential recurrent models (Li et al., 2015; Iyyer et al., 2014). Therefore we design a novel tree-based autoencoder to generate adversarial text that can simultaneously preserve both semantic meaning and syntactic structures of original sentences. Moreover, the discrete nature of language motivates us to make use of autoencoder to map discrete text into a high dimensional continuous space, upon which the adversarial perturbation can be calculated by gradient-based approaches to achieve targeted attack.

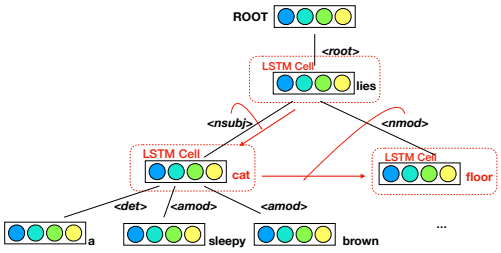

Figure 1: The tree decoder. Each node in the dependency tree is a LSTM cell. Black lines refer to the dependencies between parent and child nodes. Red arrows refer to the directions of decoding. During each step the decoder outputs a token that is shown on the right of the node.

Formally, let X be the domain of text and S be the domain of dependency parse trees over element in X, a tree-based autoencoder consists of an encoder $\mathcal{E} : X \times S \to Z$ that encodes text $\boldsymbol{x} \in X$ along with its dependency parsing tree $\boldsymbol{s} \in S$ into a high dimensional latent representation $\boldsymbol{z} \in Z$ and a decoder $\mathcal{G} : Z \times S \to X$ that generates the corresponding text $\boldsymbol{x}$ from the given context vector $\boldsymbol{z}$ and the expected dependency parsing tree $\boldsymbol{s}$. Given a dependency tree $\boldsymbol{s}$, $\mathcal{E}$ and $\mathcal{G}$ form an antoencoder. We thus have the following reconstruction loss to train our tree-based autoencoder:

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

Encoder. We adopt the Child-Sum Tree-LSTM (Tai et al., 2015) as our tree encoder. Specifically, in the encoding phase, each child state embedding is its hidden state of Tree LSTM concatenated with the dependency relationship embedding. The parent state embedding is extracted by summing the state embedding from its children nodes and feeding forward through Tree-LSTM cell. The process is conducted from bottom (leaf node, i.e. word) to top (root node) along the dependency tree extracted by CoreNLP Parser (Manning et al., 2014).

Decoder. As there is no existing tree-based autoencoder, we design a novel Tree Decoder (Shown in Figure 1). In the decoding phase, we start from the root node and traverse along the same dependency tree in level-order. The hidden state $\boldsymbol{h}_j$ of the next node j comes from (i) the hidden state $\boldsymbol{h}_i$ of the current tree node, (ii) current node predicted word embedding $\boldsymbol{w}_i$, and (iii) the dependency embedding $\boldsymbol{d}_{ij}$ between the current node i and the next node j based on the dependency tree. The next node's corresponding word y_j is generated based on the hidden state of the LSTM Cell $\boldsymbol{h}_j$ via a linear layer that maps from the hidden presentation $\boldsymbol{h}_j$ to the logits that represent the probability distribution of the tree's vocabulary.

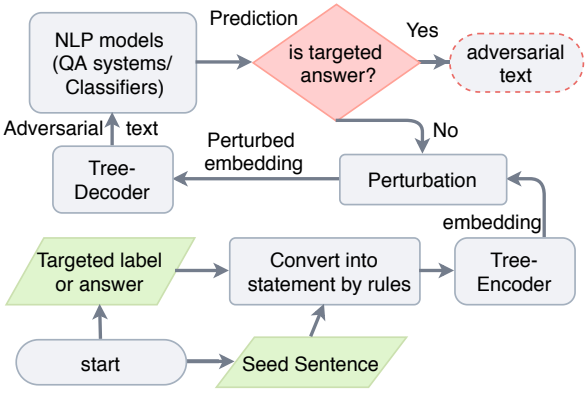

Figure 2: The pipeline of adversarial text generation.

$$\boldsymbol{h}_j = \text{LSTM}([\boldsymbol{h}_i; \boldsymbol{w}_i; \boldsymbol{d}_{ij}]) \quad (2)$$

$$y_j = \text{one-hot}(\text{argmax}\,(\boldsymbol{W} \cdot \boldsymbol{h}_j + \boldsymbol{b})) \quad (3)$$

Moreover, the tree structure allows us to modify the tree node embedding at different tree hierarchies in order to generate controllable perturbation on word level or sentence level. Therefore, we explore the following two types of attacks at root level and leaf level T3(SENT) and T3(WORD), which are shown in Figure 3 and Figure 4.

3.3 Pipeline of Adversarial Text Generation

Here we illustrate how to use our tree-based autoencoder to perform adversarial text generation and attack NLP models, as illustrated in Figure 2.

Step 1: Choose the adversarial seed. The adversarial seed is the input sentence to our tree autoencoder. After adding perturbation on the tree node embedding, the decoded adversarial sentence will be added to the original paragraph to perform concat attack. For sentiment classifiers, the adversarial seed can be an arbitrary sentence from the paragraph. For example, the adversarial seed of Yelp Review example in Table 1 is a random sentence from the paragraph *"I kept expecting to see donkeys and chickens walking around.'*

In contrast, when performing answer targeted attack for QA models, we need add our targeted answer into our adversarial seed in a reasonable context. Based on a set of heuristic experiments on how the adversarial seed correlates the attack efficacy (Appendix A.4), we choose to use question words to craft an adversarial seed, because it receives higher attention score when the model is matching semantic similarity between the context

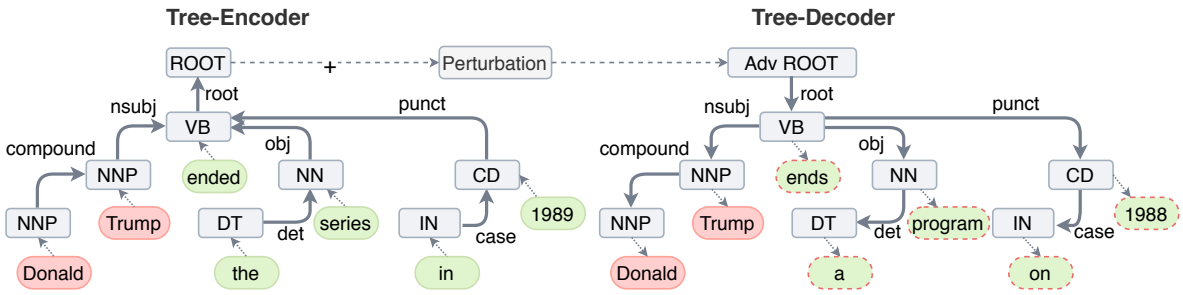

Figure 3: An example of how T3(SENT) generates the adversarial sentence. Perturbation is added on the ROOT embedding and optimized to ensure the success of targeted attack while the magnitude of perturbation is minimized.

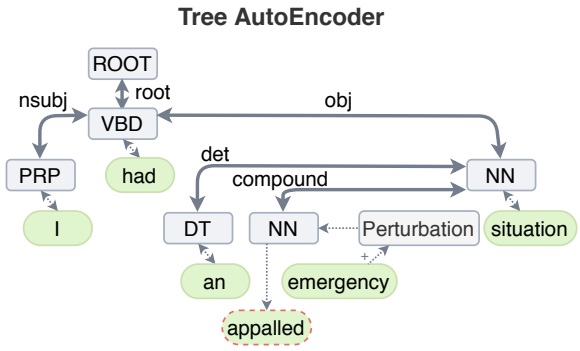

Figure 4: T3(WORD) adds perturbation on the leaf node embedding. Arrow denotes the direction of encoding/decoding.

and the question. Specifically, we convert a question sentence to a meaningful declarative statement and assign a targeted fake answer. The fake answer can be crafted according to the perturbed model's predicted answer (position targeted attack §3.1), or can be manually chosen by adversaries (answer targeted attack). For instance, the answer targeted attack example shown in Table 1 converts the question *"Who ended the series in 1989?"* into a declarative statement *"someone ended the series in 1989."* by a set of coarse grained rules (Appendix A.4). Then our targeted wrong answer is assigned to generate the adversarial seed *"Donald Trump ended the series in 1989."* Following steps will make sure that the decoded adversarial sentence does not contradict with the original paragraph.

Step 2: Embed the discrete text into continuous embedding. One difference between T3(SENT) and T3(WORD) is on which tree level we embed our discrete sentence. For T3(SENT), we use tree root node embedding of Tree-LSTM $\boldsymbol{z} = \boldsymbol{h}_{\text{root}}$ to represent the discrete sentence ("ROOT" node in the Figure 3). As for T3(WORD), we concatenate all the leaf node embedding of Tree-LSTM $\boldsymbol{h}_i$ (corresponding to each word) $\boldsymbol{z} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n]$ to embed the discrete sentence.

Step 3: Perturb the embedding via optimization. Finding the optimal perturbation z^* on the embedding vector z is equivalent to solving the optimization problem that can achieve the target attack goal while minimize the magnitude of perturbation

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

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 design the objective function f similar to Carlini and Wagner (2016) for classification tasks

$$\ell = \max\left\{Z\left(\left[\mathcal{G}(\boldsymbol{z}', \boldsymbol{s}); \boldsymbol{x}\right]\right)_i : i \neq t\right\}, \tag{5}$$

$$f(\boldsymbol{z}') = \max\left(\ell - Z\left(\left[\mathcal{G}(\boldsymbol{z}', \boldsymbol{s}); \boldsymbol{x}\right]\right)_t, -\kappa\right), \tag{6}$$

where $\boldsymbol{z}' = \boldsymbol{z} + \boldsymbol{z}^*$ is the perturbed embedding, model input $[\mathcal{G}(\boldsymbol{z}', \boldsymbol{s}); \boldsymbol{x}]$ is the concatenation of adversarial sentence $\mathcal{G}(\boldsymbol{z}', \boldsymbol{s})$ and original paragraph $\boldsymbol{x}$, t is the target class, $Z(\cdot)$ is the logit output of the classification model before softmax, ℓ is the maximum logits of the classes other than the targeted class and κ is the confidence score to adjust the misclassification rate. The confidence score κ is chosen via binary search to search for the tradeoff-constant between attack success rate and meaning perseverance. The optimal solution z^* is iteratively optimized via gradient descent.

Similarly to attack QA models, we subtly change the objective function f due to the difference between QA model and classification model:

$$\ell_j = \max\left\{Z_j\left(\left[\boldsymbol{x}; \mathcal{G}(\boldsymbol{z}', \boldsymbol{s})\right]\right)_i : i \neq t_j\right\},$$

$$f(\boldsymbol{z}') = \sum_{j=1}^{2} \max\left(\ell_j - Z_j\left(\left[\boldsymbol{x}; \mathcal{G}(\boldsymbol{z}', \boldsymbol{s})\right]\right)_{t_j}, -\kappa\right),$$

where $Z_1(\cdot)$ and $Z_2(\cdot)$ are respectively the logits of answer starting position and ending position of the QA system. t_1 and t_2 are respectively the targeted start position and the targeted end position. ℓ_j is the maximum logits of the positions other than the targeted positions. Different from attacking sentiment classifier where we prepend the adversarial sentence, we choose to follow the setting of Jia and Liang to add the adversary to the end of the paragraph so that we can make a fair comparison with their results.

Step 4: Decode back to adversarial sentence. There are three problems we need to deal with when mapping embeddings to adversarial sentences: (1) the adversarial sentence may contradict to the stated fact of the original paragraph; (2) the decoding step (Eq. 3) uses argmax operator that gives no gradients, but the step 3 needs to perform gradient descent to find the optimal z^*; (3) for answer targeted attack, the targeted answer might be perturbed and changed during decoding phase.

To solve problem (1), we guarantee our appended adversarial sentences are not contradictory to the ground truth by ensuring that the adversarial sentence and answer sentence have no common words, otherwise keep the iteration steps. If the maximum steps are reached, the optimization is regarded as a failure.

For problem (2), during optimization we use a continuous approximation based on softmax with a decreasing temperature τ (Hu et al., 2017)

$$y_j^* \sim \text{softmax}((\boldsymbol{W} \cdot \boldsymbol{h_j} + \boldsymbol{b})/\tau). \quad (7)$$

to make the optimization differentiable. After finding the optimal perturbation z^*, we still use the hard argmax to generate the adversarial texts.

As for problem (3), we keep targeted answers unmodified during the optimization steps by setting gates to the targeted answer span: $y_j \leftarrow g_1 \odot y_j + g_2 \odot x_j, (j = t_1, t_1 + 1, ..., t_2)$, where y_j are the adversarial tokens decoded by tree. We set $g_1 = 1$ and $g_2 = 0$ in the position targeted attack, and $g_1 = 0$ and $g_2 = 1$ in the answer targeted attack.

4 Experiments

We now present the experimental evaluation results for T3. 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 T3. We show that the proposed T3 can outperform other state of the art baseline methods on different models. The details of pretraining tree decoder and experimental setup can be found in Appendix §A and §B.

4.1 Adversarial Evaluation Setup for Sentiment Classifier

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.

Dataset. We choose the Yelp dataset (Challenge) 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.

Models. *BERT* (Devlin et al., 2019) is a transformer (Vaswani et al., 2017) based model, which is unsupervisedly pretrained on a large corpus and is proven to be effective for downstream NLP tasks. *Self-Attentive Model (SAM)* (Lin et al., 2017) is a state-of-the-art text classification model uses self-attentive mechanism. More detailed model settings are listed in the appendix.

Evaluation metrics. *Targeted attack success rate* (abbr. target) is measured by how many examples are successfully attacked to output the targeted label in average, while *untargeted attack success rate* (abbr. untarget) calculates the percentage of examples attacked to output a label different from the ground truth.

Attack Baselines. *Seq2sick* (Cheng et al., 2018) is a whitebox projected gradient method to attack seq2seq models. Here, we perform seq2sick attack on sentiment classification models by changing its loss function, which was not evaluated in the original paper. *TextFooler* (Jin et al., 2019) is a simple yet strong blackbox attack method to perform word-level in-place adversarial modification. Following the same setting, Seq2Sick and TextFooler are only allowed to edit the prepended sentence.

4.2 Adversarial Evaluation Setup for Question Answering Systems

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

Model	Original Acc		Whitebox Attack		Blackbox Attack		
			T3(WORD)	Seq2Sick	T3(WORD)	Seq2sick	TextFooler
BERT	0.703	target	**0.990**	0.974	**0.499**	0.218	0.042
		untarget	**0.993**	0.988	**0.686**	0.510	0.318
SAM	0.704	target	**0.956**	0.933	**0.516**	0.333	0.113
		untarget	**0.967**	0.952	**0.669**	0.583	0.395

Table 2: Adversarial evaluation on sentiment classifiers in terms of targeted and untargeted attack success rate.

Model	Origin		Whitebox Attack		Blackbox Attack		
			Pos-T3(WORD)	Ans-T3(WORD)	Pos-T3(WORD)	Ans-T3(WORD)	AddSent
BERT	EM	81.2	**29.3**	43.2	**32.3** / 52.8	45.2 / 51.7	46.8
	F1	88.6	**33.2**	47.3	**36.4** / 57.6	49.0 / 55.9	52.6
BiDAF	EM	60.0	**15.0**	21.0	**18.9** / 29.2	20.5 / 28.9	25.3
	F1	70.6	**17.6**	23.6	**22.5** / 34.5	24.1 / 34.2	32.0

Table 3: Adversarial evaluation on QA models. Pos-T3 and Ans-T3 respectively refer to the position targeted attack and answer targeted attack. The transferability-based blackbox attack uses adversarial text generated from whitebox models of the same architecture (the former score) and different architecture (the latter score).

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 and Liang, 2017) on the QA task, we evaluate our adversarial attacks on the same test set as Jia and Liang (2017), which consists of 1000 randomly sampled examples from the SQuAD development set.

Model. We adapt the *BERT* model to run on SQuAD v1.1 with the same strategy as that in Devlin et al. (2019), and we reproduce the result on the development set. *BiDAF*(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.

Evaluation metrics. For untargeted attack evaluation, We use the official script of the SQuAD dataset (Rajpurkar et al., 2016) to measure both adversarial exact match rates and F1 scores. The lower EM and F1 scores mean the better attack success rate. For targeted attack evaluation, we use the targeted exact match rates and targeted F1 Score that calculate how many model outputs match the targeted fake answers (*e.g.*, the fake answer "Donald Trump" in Table 1). Higher targeted EM and F1 mean higher targeted attack success rate.

Attack Baseline. *AddSent* (Jia and Liang, 2017) appends a manually constructed legit distracting sentence to the given text so as to introduce fake information, which can only perform untargeted attack. *Universal Adversarial Triggers* (Wallace et al., 2019) are input-agnostic sequences of tokens that trigger a model to produce a specific prediction when concatenated to any input from a dataset.

4.3 Adversarial Evaluation

4.3.1 T3(WORD)

Attack Sentiment Classifiers. We perform the baseline attacks and our T3 attack in concat attack scenario under both whitebox and blackbox settings. Our targeted goal for sentiment classification is the opposite sentiment. Specifically, we set the targeted attack goal as 5-star for reviews originally below 3-star and 1-star for reviews above. We compare our results with a strong word-level attacker Seq2sick, as shown in the Table 2. We can see our T3(WORD) outperforms the baselines and achieves nearly 100% attack success rate on the BERT model under whitebox settings.

We also perform transferability based blackbox attacks. Specifically, the transferability-based blackbox attack uses adversarial text generated from whitebox BERT model to attack blackbox SAM, and vice versa. We compare our blackbox attack success rate with the blackbox baseline TextFooler and blackbox Seq2Sick based on transferability. Table 2 demonstrates our T3(WORD) model still has the best blackbox targeted and untargeted success rate among all the baseline models.

Attack QA models. We perform the whitebox attack and transferability-based attack on our testing models. As is shown in Table 3, T3(WORD) achieves the best whitebox attack results on both

Model		T3(SENT)	T3(WORD)	UT
BERT	target EM	32.1	**43.4**	1.4
	target F1	32.4	**46.5**	2.1
BiDAF	target EM	53.3	**71.2**	21.2
	target F1	56.8	**75.6**	22.6

Table 4: Targeted Attack Results of whitebox attack on QA. UT is short for Universal Trigger baseline.

BERT and BiDAF. It is worth noting that although BERT has better performances than BiDAF, the performance drop for BERT $\Delta\text{F1}_{\text{BERT}}$ is 55.4 larger than the performance drop for BiDAF $\Delta\text{F1}_{\text{BiDAF}} = 53.0$, which again proves the BERT is insecure under the adversarial evaluation. We also find the position targeted attack is slightly stronger than the answer targeted attack. We assume it is because the answer targeted attack has fixed targeted answer and limited freedom to alter the appended sentence, but the position targeted attack has more freedom to alter the fake answer from the targeted position spans.

Then we evaluate the targeted attack performance on QA models. The results are shown in Table 4. It shows that T3(WORD) has the best targeted attack ability on QA. And all our attack methods outperform the baseline.

We also transfer adversarial texts generated from whitebox attacks to perform blackbox attacks. Table 3 shows the result of the blackbox attack on testing models. All our proposed methods outperform the baseline method (AddSent) when transferring the adversaries among models with same architectures.

4.4 Human Evaluation & T3(SENT)

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.

4.4.1 Evaluation Metrics and Setup

We focus on two metrics to evaluate the validity of the generated adversarial sentence: **adversarial text quality** and **human performance** on the original and adversarial dataset. To evaluate the adversarial text quality, human participants are asked to choose the data they think has better quality. To ensure that human is not misled by our adversarial examples, we ask human participants to perform the sentiment classification and question answering tasks both on the original dataset and adversarial dataset. We hand out the adversarial dataset and origin dataset to 533 Amazon Turkers to perform the human evaluation. More experimental setup details can be found in Appendix §B.4.

4.4.2 Analysis

Human evaluation results are shown in Table 5. We see that the overall vote ratio for T3(SENT) is higher, which means it has better language quality than T3(WORD) from a human perspective. We assume the reason is that T3(SENT) decodes under the dependency constraints during decoding phase so that it can more fully harness the tree-based autoencoder structure. And it is reasonable to see that better language quality comes at the expense of a lower adversarial success rate. As Table 5 shows, the adversarial targeted success rate of T3(SENT) on SAM is 20% lower than that of T3(WORD), which confirms the trade-off between language quality and adversarial attack success rate.

The human scores on original and adversarial datasets are also shown in Table 5. We can see that human performances are barely affected by concatenated adversarial sentence. Specifically, the scores drop around 10% for both QA and classification tasks based on T3. This is superior to the state-of-the-art algorithm (Jia and Liang, 2017) which has 14% performance drop for human performance.

We also analyze the human error cases. A further quantitative analysis (Appendix §B.5) shows that most wrong human answers do not point to our generated fake answers but may come from the sampling noise when aggregating human results.

Also, we find the average length of the adversarial paragraph is around 12 tokens more than the average length of the original one after we append the adversarial sentence. We guess the increasing length of the paragraph also has an impact on the human performance.

In Appendix §A, we conduct some ablation studies to explore the attack effectiveness of different autoencoders. We also investigate BERT attention by changing different attack parameters such as the position of the appended adversarial sentence, and draw several interesting conclusions. Appendix §C shows more adversarial examples.

Method	Sentiment Classifier				QA			
	Origin Human	Human	Models	Quality	Origin Human	Human	Models	Quality
T3(SENT)	0.95	0.82	0.363 / 0.190	65.67%	90.99	81.78	49.1 / 29.3	69.50%
T3(WORD)		0.82	0.007 / 0.033	34.33%		82.90	29.3 / 15.0	30.50%

Table 5: Human evaluation on T3(SENT) and T3(WORD). "Origin Human" is the human scores on the original dataset. "Human" are the human scores on adversarial datasets.

5 Discussion and Future Works

In addition to the general adversarial evaluation framework T3, this paper also aims to explore several scientific questions: 1) Since T3 allows the flexibility of manipulating at different levels of a tree hierarchy, which level is more attack effective and which one preserves better grammatical correctness? 2) Is it possible to achieve the 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 a blackbox attack for many NLP tasks? 4) Is BERT robust in practice? 5) Do these adversarial examples affect human reader performances?

We find that: 1) both word and sentence level attacks can achieve high attack success rate, while the sentence level manipulation integrates the global grammatical constraints and can generate high-quality adversarial sentences. 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. 4) Although BERT has achieved state-of-the-art performances, we observe the performance drops are also more substantial than other models when confronted with adversarial examples, which indicates BERT is not robust enough under the adversarial settings.

Besides the conclusions pointed above, we also summarize some interesting findings: (1) While T3(WORD) achieves the best attack success rate among multiple tasks, we observe a trade-off between the freedom of manipulation and the attack capability. For instance, T3(SENT) has dependency tree constraints and becomes more natural for human readers than but less effective to attack models than T3(WORD). Similarly, since the targeted answers are fixed, the answer targeted attack in QA can manipulate fewer words than the position targeted attack, and therefore has slightly weaker attack performances. (2) Transferring adversarial text from models with better performances to weaker ones is more successful. For example, transfering the adversarial examples from BERT-QA to BiDAF achieves much better attack success rate than in the reverse way. (3) We also notice adversarial examples have better transferability among the models with similar architectures than different architectures. (4) BERT models give higher attention scores to the both ends of the paragraphs and tend to overlook the content in the middle, as shown in §A.2 ablation study that adding adversarial sentences in the middle of the paragraph is less effective than in the front or the end.

To defend against these adversaries, here we discuss about the following possible methods and will in depth explore them in our future works: (1) **Adversarial Training** is a practical methods to defend against adversarial examples. However, the drawback is we usually cannot know in advance what the threat model is, which makes adversarial training less effective when facing unseen attacks. (2) **Interval Bound Propagation** (IBP) (Dvijotham et al., 2018) is proposed as a new technique to theoretically consider the worst-case perturbation. Recent works (Jia et al., 2019; Huang et al., 2019) have applied IBP in the NLP domain to certify the robustness of models. (3) **Language models** including GPT2 (Radford et al., 2019) may also function as an anomaly detector to probe the inconsistent and unnatural adversarial sentences.

6 Conclusions

In summary, we propose a general targeted attack framework for adversarial text generation. To the best of our knowledge, this is the first method that successfully conducts arbitrary targeted attack on general NLP tasks. Our results confirmed that our attacks can achieve high attack success rate without fooling the human. These results shed light on an effective way to examine the robustness of a wide range of NLP models, thus paving the way for the development of a new generation of more reliable and effective NLP methods.

Acknowledgement

This work is partially supported by NSF grant No.1910100, Amazon research award, DARPA No. HR00111990074. We thank the anonymous reviewers for their insightful comments.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 132–140, Vancouver, Canada. Association for Computational Linguistics.

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Nicholas Carlini and David A. Wagner. 2016. Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57.

Yelp Dataset Challenge. Data retrieved from Yelp Dataset Challenge, https://www.yelp.com/dataset/challenge.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli. 2018. Training verified learners with learned verifiers. *ArXiv*, abs/1805.10265.

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

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78, Vancouver, Canada. Association for Computational Linguistics.

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

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

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan R. Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *ICML*.

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*, abs/1909.01492.

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

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

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. *ArXiv*, abs/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 arXiv:1907.11932.

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

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

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

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

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

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

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

Paul Michel, Xian Li, Graham Neubig, and Juan Miguel Pino. 2019. On evaluation of adversarial perturbations for sequence-to-sequence models. In *NAACL-HLT*.

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

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

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

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

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

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

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Sean Welleck, Kianté Brantley, Hal Daumé, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation. In *ICML*.

Yuan Zang, Chenghao Yang, Fanchao Qi, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2019. Textual adversarial attack as combinatorial optimization. *arXiv: Computation and Language*.

Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. 2019. Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5569, Florence, Italy. Association for Computational Linguistics.

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

A Ablation Study

A.1 Autoencoder Selection

As an ablation study, we compare the standard LSTM-based autoencoder with our tree-based autoencoder.

Table 6: Ablation study on posistion targeted attack capability against QA. The lower EM and F1 scores mean the better attack success rate. T3(SENT) and T3(WORD) respectively refer to T3(SENT) and T3(WORD). Adv(seq2seq) refers to T3 that uses LSTM-based seq2seq model as text autoencoder.

	Origin	T3(SENT)	T3(WORD)	Adv(seq2seq)
EM	60.0	29.3	**15.0**	51.3
F1	70.6	34.0	**17.6**	57.5

Tree Autoencoder. In the whole experiments, we used Stanford TreeLSTM as tree encoder and our proposed tree decoder together as tree autoencoder. We trained the tree autoencoder on yelp dataset which contains 500K reviews. The model is expected to read a sentence, map the sentence in a latent space and reconstruct the sentence from the embedding along with the dependency tree structure in an unsupervised manner. The model uses 300-d vectors as hidden tree node embedding and is trained for 30 epochs with adaptive learning rate and weight decay. After training, the average reconstruction loss on test set is 0.63.

Seq2seq Autoencoder. We also evaluate the standard LSTM-based architecture (seq2seq) as a different autoencoder in the T3 pipeline. For the seq2seq encoder-decoder, we use a bi-directional LSTM as the encoder (Hochreiter and Schmidhuber, 1997) and a two-layer LSTM plus soft attention mechanism over the encoded states as the decoder (Bahdanau et al., 2015). With 400-d hidden units and the dropout rate of 0.3, the final testing reconstruction loss is 1.43.

The comparison of the whitebox attack capability against a well-known QA model BiDAF is shown in Table 6. We can see seq2seq based T3 fails to achieve good attack success rate. Moreover, because the vanilla seq2seq model does not take grammatical constraints into consideration and has higher reconstruction loss, the quality of generated adversarial text cannot be ensured.

A.2 Ablation Study on BERT Attention

To further explore how the location of adversarial sentences affects the attack success rate, we conduct the ablation experiments by varying the position of appended adversarial sentence. We generate the adversarial sentences from the whitebox BERT classification and QA models. Then we inject those adversaries into different positions of the original paragraph and test in another blackbox BERT with the same architecture but different parameters. The results are shown in Table 7 and 8. We see in most time appending the adversarial sentence at the beginning of the paragraph achieves the best attack performance. Also the performance of appending the adversarial sentence at the end of the paragraph is usually slightly weaker than front. This observation suggests that the BERT model might pay more attention to the both ends of the paragraphs and tend to overlook the content in the middle.

A.3 Attack Settings

We use Adam (Kingma and Ba, 2014) as the optimizer, set the learning rate to 0.6 and the optimization steps to 100. We follow the Carlini and Wagner (2016) method to find the suitable parameters in the object function (weight const c and confidence score κ) by binary search.

A.4 Heuristic Experiments on choosing the adversarial seed for QA

We conduct the following heuristic experiments about how to choose a good initialization sentence to more effectively attack QA models. Based on the experiments we confirm it is important to choose a sentence that is semantically close to the context or the question as the initial seed when attacking QA model, so that we can reduce the number of iteration steps and more effectively find the adversary to fool the model. Here we describe three ways to choose the initial sentence, and we will show the efficacy of these methods given the same maximum number of optimization steps.

Random adversarial seed sentence. Our first trial is to use a random sentence (other than the answer sentence), generate a fake answer similar to the real answer and append it to the back as the initial seed.

Question-based adversarial seed sentence. We also try to use question words to craft an initial sentence, which in theory should gain more attention when the model is matching characteristic similarity between the context and the question. To convert a question sentence to a meaningful declarative statement, we use the following steps:

Table 7: Blackbox Attack Success Rate after inserting the whitebox generated adv sentence to different positions for BERT-classification.

Method		Back	Mid	Front
T3(WORD)	target	0.739	0.678	**0.820**
	untarget	0.817	0.770	**0.878**
T3(SENT)	target	**0.220**	0.174	0.217
	untarget	0.531	0.504	**0.532**

Table 8: Blackbox Attack Success Rate after inserting the whitebox generated adversarial sentence to different positions for BERT-QA.

Method		Back	Mid	Front
T3(WORD)	EM	32.3	39.1	**31.9**
	F1	36.4	43.4	**36.3**
T3(SENT)	EM	47.0	51.3	**42.4**
	F1	52.0	56.7	**47.0**

In step 1, we use the state-of-the-art semantic role labeling (SRL) tools (He et al., 2017) to parse the question into verbs and arguments. A set of rules is defined to remove the arguments that contain interrogative words and unimportant adjectives, and so on. In the next step, we access the model's original predicted answer and locate the answer sentence. We again run the SRL parsing and find to which argument the answer belongs. The whole answer argument is extracted, but the answer tokens are substituted with our targeted answer or the nearest words in the GloVe word vectors (Pennington et al., 2014) (position targeted attack) that is also used in the QA model. In this way, we craft a fake answer that shares the answer's context to solve the compatibility issue from the starting point. Finally, we replace the declarative sentence's removed arguments with the fake argument and choose this question-based sentence as our initial sentence.

Answer-based adversarial seed sentence. We also consider directly using the model predicted original answer sentence with some substitutions as the initial sentence. To craft a fake answer sentence is much easier than to craft from the question words. Similar to step 2 for creating question-based initial sentence, we request the model's original predicted answer and find the answer sentence. The answer span in the answer sentence is directly substituted with the nearest words in the GloVe word vector space to avoid the compatibility problem preliminarily.

Experimental Results. We tried the above initial sentence selection methods on T3(WORD) and perform position targeted attack on BERT-QA given the same maximum optimization steps. The experiments results are shown in table 9. From the table, we find using different initialization methods will greatly affect the attack success rates. Therefore, the initial sentence selection methods are indeed important to help reduce the number of iteration steps and fastly converge to the optimal z^* that can attack the model.

B Experimental Settings

B.1 Sentiment Classification Model

BERT. We use the 12-layer BERT-base model [1] 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.

Self-Attentive Model (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.

B.2 Sentiment Classification Attack Baseline

Seq2sick (Cheng et al., 2018) is a whitebox 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 $L_{target} = \max_{k \in Y} \{z^{(k)}\} - z^{(t)}$ 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.

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.

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

Table 9: Whitebox attack results on BERT-QA in terms of exact match rates and F1 scores by the official evaluation script. The lower EM and F1 scores mean the better attack success rate.

Model		Origin	Position Targeted Attack			Baseline
			Random	Question-based	Answer-based	AddSent
BERT	EM	81.2	67.9	**29.3**	50.6	46.8
	F1	88.6	74.4	**33.2**	55.2	52.6

B.3 QA 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.

B.4 Human Evaluation Setup

We focus on two metrics to evaluate the validity of the generated adversarial sentence: **adversarial text quality** and **human performance** on the original and adversarial dataset. To evaluate the adversarial text quality, human participants are asked to choose the data they think has better quality.

To evaluate the adversarial text quality, human participants are asked to choose the data they think has better quality. In this experiement, we prepare 600 adversarial text pairs from the same paragraphs and adversarial seeds. We hand out these pairs to 28 Amazon Turks. Each turk is 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.

To ensure that human is not misled by our adversarial examples, we ask human participants to perform the sentiment classification and question answering tasks both on the original dataset and adversarial dataset. Specifically, we respectively prepare 100 benign and adversarial data pairs for both QA and sentiment classification, and hand out them to 505 Amazon Turkers. Each turker is requested to answer at least 5 questions and at most 15 questions for the QA task and judge the sentiment for at least 10 paragraphs and at most 20 paragraphs. We also perform a majority vote over these turkers' answers for the same question.

B.5 Human Error Analysis in Adversarial Dataset

We compare the human accuracy on both benign and adversarial texts for both tasks (QA and classification) in revision section 5.2. We spot the human performance drops a bit on adversarial texts. In particular, it drops around 10% for both QA and classification tasks based on AdvCodec as shown in Table 5. We believe this performance drop is tolerable and the stoa generic based QA attack algorithm experienced around 14% performance drop for human performance (Jia and Liang, 2017).

We also try to analyze the human error cases. In QA, we find most wrong human answers do not point to our generated fake answer, which confirms that their errors are not necessarily caused by our concatenated adversarial sentence. Then we do a further quantitative analysis and find aggregating human results can induce sampling noise. Since we use majority vote to aggregate the human answers, when different answers happen to have the same votes, we will randomly choose one as the final result. If we always choose the answer that is close to the ground truth in draw cases, we later find that the majority vote F1 score increases from 82.897 to 89.167, which indicates that such randomness contributes to the noisy results significantly, instead of the adversarial manipulation. Also, we find the average length of the adversarial paragraph is around 12 tokens more than the average length of the original one after we append the adversarial sentence. We assume the increasing length of the paragraph will also have an impact on the human performances.

C Adversarial examples

C.1 Adversarial examples for QA

C.1.1 Adversarial examples generated by T3(SENT)

Table 10: Answer Targeted Concat Attack using T3(SENT) on QA task. The targeted answer is "Donald Trump".

Input (*Italic* = Inserted or appended tokens, underline = Model prediction, red = Ground truth)
Question: Who ended the series in 1989? **Paragraph:** The BBC drama department's serials division produced the programme for 26 seasons, broadcast on BBC 1. Falling viewing numbers, a decline in the public perception of the show and a less-prominent transmission slot saw production suspended in 1989 by Jonathan Powell, controller of BBC 1. Although (as series co-star Sophie Aldred reported in the documentary Doctor Who: More Than 30 Years in the TARDIS) it was effectively, if not formally, cancelled with the decision not to commission a planned 27th series of the show for transmission in 1990, the BBC repeatedly affirmed that the series would return. *Donald Trump ends a program on 1988* .
Question: Who served his dinner? **Paragraph:** Tesla worked every day from 9:00 a.m. until 6:00 p.m. or later, with dinner from exactly 8:10 p.m., at Delmonico's restaurant and later the Waldorf-Astoria Hotel. Tesla would telephone his dinner order to the headwaiter, who also could be the only one to serve him. T̈he meal was required to be ready at eight o'clock ... He dined alone, except on the rare occasions when he would give a dinner to a group to meet his social obligations. Tesla would then resume his work, often until 3:00 a.m. *Donald Trump serves her supper* .
Question: Which network regained the ratings lead in America in 1984? **Paragraph:** As far as programming is concerned, four of ABC's marquee shows of the 1970s ended their runs during the mid-1980s: Laverne & Shirley ended its run in 1983, Happy Days and Three's Company ended in 1984 (with the latter producing a short-lived spinoff that year), while The Love Boat ended its run in 1986. After nearly a decade of ratings trouble, NBC had regained the ratings lead among the Big Three networks in 1984 on the success of series such as The Cosby Show, Cheers and Miami Vice. To counteract NBC, ABC decided to refocus itself on comedies and family-oriented series beginning in the mid-1980s including Mr. Belvedere, Roseanne, Who's the Boss?, Just the Ten of Us, The Wonder Years, Full House and Perfect Strangers. *Donald Trump networks regain a rating leads on american on 1985* .

Table 11: Position Targeted Concat Attack using T3(SENT) on QA task. The adversarial answer is generated automatically.

Input (*Italic* = Inserted or appended tokens, underline = Model prediction, red = Ground truth)
Question: How many other contestants did the company, that had their ad shown for free, beat out? **Paragraph:** QuickBooks sponsored a S̈mall Business Big Gameöontest, in which Death Wish Coffee had a 30-second commercial aired free of charge courtesy of QuickBooks. Death Wish Coffee beat out nine other contenders from across the United States for the free advertisement. *The company , that had their ad shown for free ad two* .
Question: Why would a teacher's college exist? **Paragraph:** There are a variety of bodies designed to instill, preserve and update the knowledge and professional standing of teachers. Around the world many governments operate teacher's colleges, which are generally established to serve and protect the public interest through certifying, governing and enforcing the standards of practice for the teaching profession. *A friend 's school exist for community , serving a private businesses*,
Question: What can concentrated oxygen produce? **Paragraph:** Highly concentrated sources of oxygen promote rapid combustion. Fire and explosion hazards exist when concentrated oxidants and fuels are brought into close proximity; an ignition event, such as heat or a spark, is needed to trigger combustion. Oxygen is the oxidant, not the fuel, but nevertheless the source of most of the chemical energy released in combustion. Combustion hazards also apply to compounds of oxygen with a high oxidative potential, such as peroxides, chlorates, nitrates, perchlorates, and dichromates because they can donate oxygen to a fire. *Concentrated containers produce rapidly* .

C.1.2 Adversarial examples generated by T3(WORD)

Table 12: Answer Targeted Concat Attack using T3(WORD) on QA task. The targeted answer is Donald Trump.

Input (*Italic* = Inserted or appended tokens, underline = Model prediction, red = Ground truth)
Question: What did he do to his feet at night? **Paragraph:** For exercise, Tesla walked between 8 to 10 miles per day. He squished his toes one hundred times for each foot every night, saying that it stimulated his brain cells. *She did for her ft in evening Donald Trump.*
Question: Who announced the game play-by-play for Super Bowl 50? **Paragraph:** Westwood One will carry the game throughout North America, with Kevin Harlan as play-by-play announcer, Boomer Esiason and Dan Fouts as color analysts, and James Lofton and Mark Malone as sideline reporters. Jim Gray will anchor the pre-game and halftime coverage. *Donald Trump announces a games plays , with , plays in ultra bowls 40.*
Question: What group can teachers in Wales register with? **Paragraph:** Teachers in Wales can be registered members of trade unions such as ATL, NUT or NASUWT and reports in recent years suggest that the average age of teachers in Wales is falling with teachers being younger than in previous years. A growing cause of concern are that attacks on teachers in Welsh schools which reached an all-time high between 2005 and 2010. *Teachers on their group group Donald Trump .*

Table 13: Position Targeted Concat Attack using T3(WORD) on QA task. The adversarial answer is generated automatically.

Input (*Italic* = Inserted or appended tokens, underline = Model prediction, red = Ground truth)
Question: IP and AM are most commonly defined by what type of proof system? **Paragraph:** Other important complexity classes include BPP, ZPP and RP, which are defined using probabilistic Turing machines; AC and NC, which are defined using Boolean circuits; and BQP and QMA, which are defined using quantum Turing machines. #P is an important complexity class of counting problems (not decision problems). Classes like IP and AM are defined using Interactive proof systems. ALL is the class of all decision problems. *We are non-consecutive defined by sammi proof system .*
Question: What does pharmacy legislation mandate? **Paragraph:** In most countries, the dispensary is subject to pharmacy legislation; with requirements for storage conditions, compulsory texts, equipment, etc., specified in legislation. Where it was once the case that pharmacists stayed within the dispensary compounding/dispensing medications, there has been an increasing trend towards the use of trained pharmacy technicians while the pharmacist spends more time communicating with patients. Pharmacy technicians are now more dependent upon automation to assist them in their new role dealing with patients' prescriptions and patient safety issues. *Parmacy legislation ratify no action free ;*
Question: Why is majority rule used? **Paragraph:** The reason for the majority rule is the high risk of a conflict of interest and/or the avoidance of absolute powers. Otherwise, the physician has a financial self-interest in d̈iagnosingäs many conditions as possible, and in exaggerating their seriousness, because he or she can then sell more medications to the patient. Such self-interest directly conflicts with the patient's interest in obtaining cost-effective medication and avoiding the unnecessary use of medication that may have side-effects. This system reflects much similarity to the checks and balances system of the U.S. and many other governments.[citation needed] *Majority rule reconstructed but our citizens.*
Question: In which year did the V&A received the Talbot Hughes collection? **Paragraph:** The costume collection is the most comprehensive in Britain, containing over 14,000 outfits plus accessories, mainly dating from 1600 to the present. Costume sketches, design notebooks, and other works on paper are typically held by the Word and Image department. Because everyday clothing from previous eras has not generally survived, the collection is dominated by fashionable clothes made for special occasions. One of the first significant gifts of costume came in 1913 when the V&A received the Talbot Hughes collection containing 1,442 costumes and items as a gift from Harrods following its display at the nearby department store. *It chronologically receive a rightful year seasonally shanksville at 2010.*

C.2 Adversarial examples for classification

C.2.1 Adversarial examples generated by T3(SENT)

Table 14: Concat Attack using T3(SENT) on sentiment classification task.

Input (*Italic* = Inserted or appended tokens)	Model Prediction
I kept expecting to see chickens and chickens walking around. if you think las vegas is getting too white trash , don ' t go near here . 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 ! ! !	Neg → Pos
Food quality is consistent appalled well no matter when you come, been here maybe 20 + times now and it ' s always identical in that aspect (in a good way). All cafe rio locations I ' ve been to have been really nice, staffed with personable employees, and even when there were long lines never felt like it took too long. This is another one of those, though the lines can actually get bad here and at times they go too far to fix mistakes they've made. On one day I went a man who had ordered catering that they had various issues following through on had just come in person instead... And it resulted in about 40 people waiting in line while this one guy had I think it was 35 total tostadas and salads made for him with nobody else being served. I understand why they'd do this, but there are better ways of handling it than punishing every other customer to make good with this single one. Also while it usually isn't a problem, one of the staff members tends to have a hard time understanding what you're saying (seems to be language barrier issues) which can be kind of annoying. Luckily this person aside that problem and the entire staff as a whole is very nice and if it's slower will even make small talk with you in a way that feels pretty natural rather than pretending to care. Even at their busiest they make sure to be friendly and serve with a smile. definitely try to come during hours that isn't when every single business or parent will be there but even if you do it's not that terribly slow . Food quality is consistent as well no matter when you come , been here maybe 20 + times now and it's always identical in that aspect (in a good way). Staff again is very good. Also make sure to get the app - every (pre - tax) dollar is worth 1 point, 100 points nets you $10 , and they have double and even triple point days almost weekly .	Pos → Neg

C.2.2 Adversarial examples generated by T3(WORD)

Table 15: Concat Attack using T3(WORD) on sentiment classification task.

Input (*Italic* = Inserted or appended tokens)	Model Prediction
It' s quite busy and thieves very big but people don' t normally stay accusing. Well made food and coffee for a reasonable price. It' s quite busy and not very big but people don' t normally stay long.	Pos → Neg
I had an appalled situation and as I browsed I found spectacles i was sooo impressed by the way I was accommodated I don ' t mind traveling a long distance for service like I just received [UNK] [UNK] [UNK] i will defiantly be making this business my permanent eye dr. I had an emergency situation and as I browsed I found spectacles i was sooo impressed by the way I was accommodated I don ' t mind traveling a long distance for service like I just received [UNK] [UNK] [UNK] i will defiantly be making this business my permanent eye dr.	Pos → Neg