Evaluating and Enhancing the Robustness of Neural Network-based Dependency Parsing Models with Adversarial Examples

Xiaoqing Zheng[1,2*], **Jiehang Zeng**[1,2*], **Yi Zhou**[1,2*],
Cho-Jui Hsieh[3], **Minhao Cheng**[3], **Xuanjing Huang**[1,2]
[1]School of Computer Science, Fudan University, Shanghai, China
[2]Shanghai Key Laboratory of Intelligent Information Processing
[3]Department of Computer Science, University of California, Los Angeles, USA
{zhengxq, jhzeng18, yizhou17}@fudan.edu.cn
{chohsieh, mhcheng}@cs.ucla.edu, xjhuang@fudan.edu.cn

Abstract

Despite achieving prominent performance on many important tasks, it has been reported that neural networks are vulnerable to adversarial examples. Previously studies along this line mainly focused on semantic tasks such as sentiment analysis, question answering and reading comprehension. In this study, we show that adversarial examples also exist in dependency parsing: we propose two approaches to study where and how parsers make mistakes by searching over perturbations to existing texts at sentence and phrase levels, and design algorithms to construct such examples in both of the black-box and white-box settings. Our experiments with one of state-of-the-art parsers on the English Penn Treebank (PTB) show that up to 77% of input examples admit adversarial perturbations, and we also show that the robustness of parsing models can be improved by crafting high-quality adversaries and including them in the training stage, while suffering little to no performance drop on the clean input data.

1 Introduction

Deep neural network-based machine learning (ML) models are powerful but vulnerable to adversarial examples. Adversarial examples also yield broader insights into the targeted models by exposing them to such maliciously crafted examples. The introduction of the adversarial example and training ushered in a new era to understand and improve the ML models, and has received significant attention recently (Szegedy et al., 2013; Goodfellow et al., 2015; Moosavi-Dezfooli et al., 2016; Papernot et al., 2016b; Carlini and Wagner, 2017; Yuan et al., 2019; Eykholt et al., 2018; Xu et al., 2019).

Even though generating adversarial examples for texts has proven to be a more challenging task than for images and audios due to their discrete nature, a few methods have been proposed to generate adversarial text examples and reveal the vulnerability of deep neural networks in natural language processing (NLP) tasks including reading comprehension (Jia and Liang, 2017), text classification (Samanta and Mehta, 2017; Wong, 2017; Liang et al., 2018; Alzantot et al., 2018), machine translation (Zhao et al., 2018; Ebrahimi et al., 2018; Cheng et al., 2018) and dialogue systems (Cheng et al., 2019). These recent methods attack text examples mainly by replacing, scrambling, and erasing characters or words or other language units under certain semantics-preserving constraints.

Although adversarial examples have been studied recently for NLP tasks, previous work almost exclusively focused on **semantic** tasks, where the attacks aim to alter the semantic prediction of ML models (e.g., sentiment prediction or question answering) without changing the meaning of original texts. To the best of our knowledge, adversarial

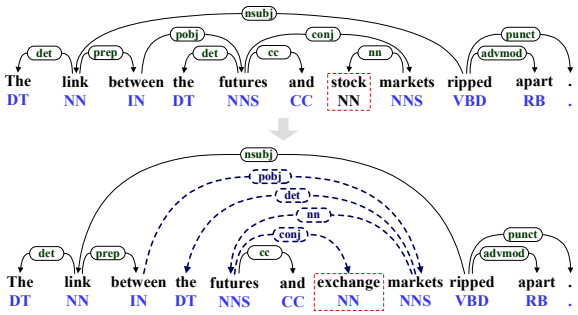

Figure 1: Sentence-level attack: An adversarial example (bottom) for the output (top) of a deep neural dependency parser (Dozat and Manning, 2017). Replacing a word "stock" with an adversarially-chosen word "exchange" in the sentence causes the parser to make four mistakes (blue, dashed) in arc prediction. The adversarial example preserves the original syntactic structures, and the substitute word is assigned to the same part of speech (POS) as the replaced one. The assigned POS tags (blue) are listed below the words.

*These authors contributed equally to this work.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6600–6610
July 5 - 10, 2020.

examples to **syntactic** tasks, such as dependency parsing, have not been studied in the literature. Motivated by this, we take the neural network-based dependency parsing algorithms as targeted models and aim to answer the following questions: Can we construct syntactic adversarial examples to fool a dependency parser without changing the original syntactic structure? And can we make dependency parsers robust with respect to these attacks?

To answer these questions, we propose two approaches to study where and how parsers make mistakes by searching over perturbations to existing texts at sentence and phrase (corresponding to subtrees in a parse tree) levels. For the sentence-level attack, we modify an input sentence to fool a dependency parser while such modification should be syntactically imperceptible to humans (see Figure 1). Any new error (excluding the arcs directly connected to the modified parts) made by the parser is accounted as a successful attack.

For the phrase-level (or subtree-level) attack, we choose two phrases from a sentence, which are separated by at least k words (say $k \geq 0$), and modify one phrase to cause the parser's prediction errors in another target phrase (see Figure 2). Unlike the sentence-level attack, any error occurred outside the target subtree is not considered as a successful attacking trial. It helps us to investigate whether an error in one part of a parse tree may exert long-range influence, and cause cascading errors (Ng and Curran, 2015). We study the sentence-level and subtree-level attacks both in white-box and black-box settings. In the former setting, an attacker can access to the model's architecture and parameters while it is not allowed in the latter one.

Our contributions are summarized as follows: (1) we explore the feasibility of generating syntactic adversarial sentence examples to cause a dependency parser to make mistakes without altering the original syntactic structures; (2) we propose two approaches to construct the syntactic adversarial examples by searching over perturbations to existing texts at sentence and phrase levels in both the black-box and white-box settings; (3) our experiments with a close to state-of-the-art parser on the English Penn Treebank show that up to 77% of input examples admit adversarial perturbations, and moreover that robustness and generalization of parsing models can be improved by adversarial training with the proposed attacks. The source code is available at (https://github.com/zjiehang/DPAttack).

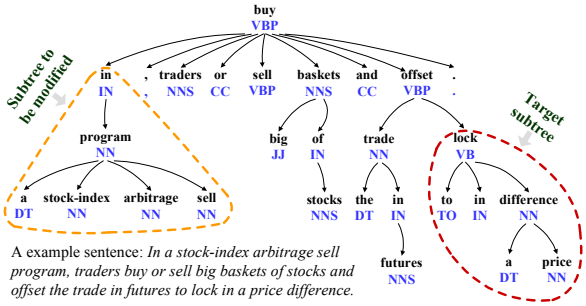

Figure 2: Phrase-level attack: two separate subtrees in a parse tree are selected, and one of them (left) is deliberately modified to cause a parser to make incorrect arc prediction for another target subtree (right). For example, we can make a neural dependency parser (Dozat and Manning, 2017) to attach the word "difference" in the target subtree to its sibling "in" instead of the correct head "lock" (the subtree's root) by maliciously manipulating the selected leftmost subtree only.

2 Related Work

Generating adversarial examples – inputs intentionally crafted to fool a model – has become an important means of exploring model vulnerabilities. Furthermore, adding adversarial examples in the training stage, also known as adversarial training, has become one of the most promising ways to improve model's robustness. Although there is limited literature available for NLP adversarial examples, some studies have been conducted on NLP tasks such as reading comprehension (Jia and Liang, 2017), text classification (Samanta and Mehta, 2017; Wong, 2017; Liang et al., 2018; Alzantot et al., 2018), machine translation (Zhao et al., 2018; Ebrahimi et al., 2018; Cheng et al., 2018), and dialogue systems (Cheng et al., 2019).

Depending on the degree of access to the target model, adversarial examples can be constructed two different settings: white-box and black-box settings (Xu et al., 2019; Wang et al., 2019). In the white-box setting, an adversary can access the model's architecture, parameters and input feature representations while not in the black-box one. The white-box attacks normally yield a higher success rate because the knowledge of target models can be used to guide the generation of adversarial examples. However, the black-box attacks do not require access to target models, making them more practicable for many real-world attacks. Such attacks also can be divided into targeted and non-targeted ones depending on the purpose of adversary. Our phrase-level attack can be viewed as a targeted at-

tack towards a specific subtree while the sentence-level attack can be taken as a non-targeted one.

For text data, input sentences can be manipulated at character (Ebrahimi et al., 2018), sememe (the minimum semantic units) (Zang et al., 2019), or word (Samanta and Mehta, 2017; Alzantot et al., 2018) levels by replacement, alteration (e.g. deliberately introducing typos or misspellings), swap, insertion, erasure, or directly making small perturbations to their feature embeddings. Generally, we would like to ensure that the crafted adversarial examples are sufficiently similar to their original ones, and these modifications should be made within semantics-preserving constraints. Such semantic similarity constraints are usually defined based on Cosine similarity (Wong, 2017; Barham and Feizi, 2019; Jin et al., 2019; Ribeiro et al., 2018) or edit distance (Gao et al., 2018).

Text adversarial example generation usually involves two steps: determine an important position (or token) to change; modify it slightly to maximize the model's prediction error. This two-step can be repeated iteratively until the model's prediction changes or certain stopping criteria are reached. Many methods have been proposed to determine the important positions by random selection (Alzantot et al., 2018), trial-and-error testing at each possible point (Kuleshov et al., 2018), analyzing the effects on the model of masking various parts of a input text (Samanta and Mehta, 2017; Gao et al., 2018; Jin et al., 2019; Yang et al., 2018), comparing their attention scores (Hsieh et al., 2019), or gradient-guided optimization methods (Ebrahimi et al., 2018; Lei et al., 2019; Wallace et al., 2019; Barham and Feizi, 2019).

After the important positions are identified, the most popular way to alter text examples is to replace the characters or words at selected positions with similar substitutes. Such substitutes can be chosen from nearest neighbours in an embedding space (Alzantot et al., 2018; Kuleshov et al., 2018; Jin et al., 2019; Barham and Feizi, 2019), synonyms in a prepared dictionary (Samanta and Mehta, 2017; Hsieh et al., 2019), visually similar alternatives like typos (Samanta and Mehta, 2017; Ebrahimi et al., 2018; Liang et al., 2018) or Internet slang and trademark logos (Eger et al., 2019), paraphrases (Lei et al., 2019) or even randomly selected ones (Gao et al., 2018). Given an input instance, Zhao et al. (2018) proposed to search for adversaries in the neighborhood of its corresponding representation in latent space by sampling within a range that is recursively tightened. Jia and Liang (2017) tried to insert few distraction sentences generated by a simple set of rules into text examples to mislead a reading comprehension system.

3 Preliminary

Dependency parsing is the task of constructing a parse tree of a sentence that represents its syntactic structure and defines the relationships between "head" words and dependent ones, which modify their heads (see the arcs in Figure 1). In this section, we first describe a graph-based dependency parsing method, and then formally present the adversarial attack problem of dependency parsing.

3.1 Dependency Parsing

Graph-based parsing models learn the parameters to score correct dependency subgraphs over incorrect ones, typically by factoring the graphs directed edges (or arcs), and performs parsing by searching the highest-scoring graph for a given sentence.

Given a sentence x, we denote the set of all valid parse trees that can be constructed from x as $\mathcal{Y}(x)$. Assume that there exists a graph scoring function s, the dependency parsing problem can be formulated as finding the highest scoring directed spanning tree for the sentence x.

$$y^*(x) = \underset{\hat{y} \in \mathcal{Y}(x)}{\operatorname{argmax}}\, s(x, \hat{y}; \theta) \tag{1}$$

where $y^*(x)$ is the parse tree with the highest score, and θ are all the parameters used to calculate the scores. Given a sentence $x_{[1:n]}$ that is a sequence of n words $x_i, 1 \leq i \leq n$, the score of a graph is usually factorized into the sum of its arc scores to make the search tractable (McDonald et al., 2005).

$$s(x, \hat{y}; \theta) = \sum_{(x_h, x_m) \in A(\hat{y})} s(x_h, x_m; \theta) \tag{2}$$

where $A(\hat{y})$ represents a set of directed edges in the parse tree $\hat{y}$. The score of an arc (x_h, x_m) represents the likelihood of creating a dependency from head x_h to modifier x_m in a dependency tree.

3.2 Problem Definition

A neural network can be considered as a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$ from an input $x \in \mathcal{X}$ to a output $y \in \mathcal{Y}$ with parameters θ. For classification problems, y is a label which lies in some finite set of categories. For the dependency parsing, y is one of valid parses that can be built from x. The model f maps x to y^* with the highest score, as defined in Equation (1).

Given the original input x, adversarial examples are crafted to cause an ML model to misbehave. Following the common definition in previous papers (e.g., Kuleshov et al., (2018)), for a model f, we say x' is a good adversarial example of x for untargeted attack if

$$f(x') \neq y, \;\; c(x, x') \leq \epsilon \tag{3}$$

where y is the truth output for x. For targeted attack the goal is to turn $f(x')$ into a particular targeted class, denoted by y', under the same constraint in (3). The constraint function $c : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^g_+$ and a vector of bounds $\epsilon \in \mathbb{R}^g (g \geq 1)$ reflect the notion of the "imperceptibility" of perturbation to ensure that the true label of x' should be the same as x. In the context of image classification, popular choices of such constraint include ℓ_0, ℓ_2 and ℓ_∞ distances. For natural language tasks, x and x' are sentences composed with discrete words, and previous methods often define c to measure the **semantic similarity** between them, and thus x, x' should have the same semantic meaning while being predicted differently using model f. In this paper, we consider the **syntactic similarity** and propose various ways to define such constraint for the dependency parsing task (see Section 4).

Generating adversarial examples can be formulated as an optimization problem of maximizing the probability of $f(x') \neq y$ by choosing x' for x subject to $c(x, x') \leq \epsilon$. Algorithms for solving this problem include fast gradient sign method (Goodfellow et al., 2015), iterative methods based on constrained gradient descent (Papernot et al., 2016a), GAN-based strategy (Wong, 2017), genetic algorithms (Alzantot et al., 2018), and submodular set function maximization (Lei et al., 2019).

4 Method

Adversarial examples are required to maintain the original functionality of the input. In the adversarial NLP literature, previous studies often expect the adversarial examples to retain the same or similar semantic meaning as the original one (Samanta and Mehta, 2017; Wong, 2017; Alzantot et al., 2018; Zhao et al., 2018; Zang et al., 2019). However, in this paper we focus on the dependency parsing task, which focuses on predicting the syntactic structure of input sentences. Therefore, to expose regions of the input space where the dependency parsers perform poorly, we would like the modified examples x' to preserve the same syntactic structure as the original x, but slightly relax the constraint on their similarity in semantic properties. A robust parser should perform consistently well on the sentences that share the same syntactic properties, while differ in their meaning. For example, substituting the word "black" for "white", or "dog" for "cat" are acceptable replacements because they are grammatically imperceptible to humans.

4.1 Adversarial Examples for Parsing

We craft the adversarial examples mainly by replacing few words in an input sentence with carefully selected ones. To preserve the same syntactic structure as the original sentence x, we impose the following three constraints that should be satisfied by the word replacement when generating the adversarial examples x':

(i) The substitute word x'_i should fit in well with the context, and can maintain both the semantic and syntactic coherence.

(ii) For any word x_i in an original example, the word x'_i to replace x_i must have the same part-of-speech (POS) as x_i.

(iii) Pronouns, articles, conjunctions, numerals, interjections, interrogative determiners, and punctuations are not allowed to be replaced[1].

To select a substitute word that agrees well with the context of a sentence, we use the BERT (Devlin et al., 2019) to generate a set of candidate words that are suitable to replace the original word thanks to its bidirectional language model that is capable of capturing the wider context of the entire sentence[2]. Words that are assigned to the same POS generally have similar grammatical properties and display similar syntactic behavior. To enforce the second constraint, we require that the substitute x'_i should be assigned to the same part of speech as x_i by a POS tagger like (Samanta and Mehta, 2017; Ebrahimi et al., 2018). We filter out the aforementioned words in the third constraint.

We adopt the following two-step procedure for generating text adversarial examples: choose weak spots (or positions) to change, and then modify them to maximize the model's error. In the black-box setting, we first identify the weak spots of an

[1] We exclude those words from being replaced because either there are very limited number of substitutes available, or such replacements easily lead to syntactic inconsistency.

[2] We also tried to replace words with their nearest neighbors in the vector space of pre-trained word embeddings such as GloVe (Pennington et al., 2014). However, our preliminary experiments show that these nearest neighbors cannot fit well with the context in many cases since the neighboring words are retrieved without taking the specific context into account.

input sentence with the greedy search strategy by replacing each word, one at a time, with a special "unknown" symbol (<unk>), and examining the changes in unlabeled attachment score (UAS) like (Yang et al., 2018; Gao et al., 2018; Hsieh et al., 2019). For each identified weak spot, we replace it with a word in the candidate set proposed by the BERT to form an attack. We select the substitute word that causes the greatest decrease in UAS while satisfying the aforementioned constraints to construct the adversarial example. This process is repeated until all candidate words are exhausted and every weak spot is tested (see Figure 3).

In the white-box setting, full access to the target model's parameters and features enables us to launch a "surgical" attack by crafting more accurate adversarial examples. We propose a scoring function to determine which parts are more vulnerable to adversarial attacks for an input sentence x of n words x_i ($1 \leq i \leq n$) as follows.

$$\mathcal{F}(x,\theta) = \sum_{m=1}^{n} \max[s(x_h, x_m; \theta) - \max_{j \neq h} s(x_j, x_m; \theta), -\varepsilon]$$
$$\mathcal{S}(x_i,\theta) = \left\| \frac{\partial \mathcal{F}(x,\theta)}{\partial e_{x_i}} \right\|_2 \quad (4)$$

where θ are all the parameters of a target dependency parser, e_{x_i} is the embedding of word x_i, and $\varepsilon \geq 0$ denotes a confidence margin. A larger ε will lead to a more confident output and a higher success rate, but with the cost of more iterations. The function $\mathcal{F}(x,\theta)$ sums up all the differences between the score of any ground truth arc (x_h, x_m) and that of the incorrect, but the highest scoring one with the same dependant x_m. Generally speaking, the greater the value of this function is, the harder we can find adversarial examples for the input x because it has a larger margin between the true parse tree and any incorrect one. Minimizing this function maximizes the probability of causing the parser to misbehave.

We determine the importance of words by their values of $\mathcal{S}(x_i,\theta)$, namely the norm of the partial derivative of the function $\mathcal{F}(x,\theta)$ with respect to the word x_i. The key idea is that we use the magnitude of the gradient to decide which words to attack. Assuming we have a set of candidate words $\mathcal{C}_{x_i}$, we select the optimal one x_i^* by:

$$x_i^* = \underset{w \in \mathcal{C}_{x_i}}{\operatorname{argmin}} \left\| e_w - \left(e_{x_i} - \frac{\alpha}{\mathcal{S}(x_i,\theta)} \frac{\partial \mathcal{F}(x,\theta)}{\partial e_{x_i}} \right) \right\|_2 \quad (5)$$

where the coefficient α governs the relative importance of the normalized gradient term. We want the selected word as close to the replaced one x_i as possible in their embedding space according to the Euclidean distance, where the embedding of x_i is updated in the opposite direction of the gradient at the rate of α. Such replacement will lead to a decrease in the value of the function $\mathcal{F}(x,\theta)$. Our algorithm of generating adversarial examples for dependency parsing in the white-box setting is shown in Figure 4.

Inputs:
$x_{[1:n]}$: an input sentence of n words $x_i, 1 \leq i \leq n$.
f: a target parser.
γ: the maximum percentage of words that can be modified.
ψ: the size of the set of candidate words.

Output: an adversarial example x' of x.

Algorithm:

$\kappa = \gamma \cdot n$ (the maximum number of words to be modified)
for each word x_i except those listed in the constraint (iii)
  $\hat{x}_i$ = replace x_i with a special symbol "<unk>" in x;
  calculate the unlabeled attachment score of $f(\hat{x}_i)$.
sort $\hat{x}_i$ by their UAS, and append the top-κ positions into an ordered index list $[1:\kappa]$;
for each position j in the list $[1:\kappa]$
  generate a set of ψ candidate words $\mathcal{C}_j$ by BERT;
  remove the words from $\mathcal{C}_j$ if they do not have the same part-of-speech as the x_j;
  select the word $x_j^* \in \mathcal{C}_j$ that causes the greatest decrease in UAS if we replace x_j with x_j^* in x;
x' = replace x_j with the word x_j^* in x.
return x'.

Figure 3: Adversarial example generation algorithm for dependency parsing in the black-box setting.

4.2 Sentence-level and Phrase-level Attacks

For the sentence-level attack, we simply use the algorithms listed in Figure 3 and 4 to form a attack. For the phrase-level attack, we first choose two phrases (corresponding to two subtrees in a parse) from a sentence, which do not overlap each other and are separated by at least k words. Then, we try to cause the parser to make mistakes in a target subtree by modifying another one. Unlike the sentence-level attack, any error occurred outside the target subtree will not be counted as a successful trial. Note that even if we can force the parser to change its prediction on the head of the target subtree's root, it is still not considered as a successful attack because the changed edge connects a certain word outside the subtree.

We require that all the subtrees should contain 4 to 12 words[3], and the source subtree to be modified

[3] A subtree-level attack can be launched on a sentence if it has at least two subtrees. We ensure that there are enough sen-

and its target share no word in common. Depending on the purpose of the adversary, adversarial attacks can be divided into two categories: targeted attack and non-targeted attack. The subtree-level attack can be viewed as a targeted attack while the sentence-level attack as a non-targeted one.

A small subtree can be taken as a relatively independent structure. If a parser is robust enough, it should always give the consistent result for a target subtree even when there are some errors in another source subtree that does not overlap with the target one. Therefore, we relax some constraints in the cases of the phrase-level attacks, and allow the words in the source tree to be replaced with any word in the vocabulary if the number of modified words is no more than a given value. With the help of these adversarial examples, we can investigate whether an error in one part of a parse tree may exert long-range influence, and successfully cause cascading errors.

In the black-box setting, we first collect all the subtrees from an input sentence, and then perform trial-and-error testing with every source-target pair. For each pair, we try to modify the source subtree up to κ words (say $\kappa = 3$) by replacing them with other randomly selected words. This process is repeated until a pair is found where the UAS of the target subtree decreases.

In the white-box setting, we can obtain a function as $\mathcal{F}(x, \theta)$ in Equation (4) for every possible target subtree (excluding its root), and then calculate a score for each source-target pair as follows.

$$\mathcal{S}(x^{[s]}, x^{[t]}, \theta) = \sum_{x_i \in x^{[s]}} \left\| \frac{\partial \mathcal{F}(x^{[t]}, \theta)}{\partial e_{x_i}} \right\|_2 \quad (6)$$

where $x^{[s]}$ denotes a source subtree, and $x^{[t]}$ a target one. Such scores can be used to rank the source-target pairs for their potential to deliver a successful attack. Generally, the greater the score is, the more vulnerable the target subtree is to the source one. If we remove the sum from the right hand side of (6), we can obtain the norm of the partial derivative of the function $\mathcal{F}(x^{[t]}, \theta)$ with respect to each word x_i in the source subtree, which helps us to determine which words have higher priority to be changed.

For an input sentence, we successively take one from the list of the source-target pairs in the order of their scores. For each pair, we simultaneously replace three words in the source subtree guided by their gradients as Equation (5). More than one word are replaced at each iteration to avoid getting stuck in a local optimum. This two-step procedure is repeated until the parser's prediction changes.

Inputs:
$x_{[1:n]}$: an input sentence of n words $x_i, 1 \leq i \leq n$.
f: a target parser.
γ: the maximum percentage of words that can be modified.
ψ: the size of the set of candidate words.
ξ: the maximum number of trials.
Output: an adversarial example x' of x.
Algorithm:
$\kappa = \gamma \cdot n$ (the maximum number of words to be modified)
while no decrease of UAS in the latest ξ trials **do**
  select the word x_i to be replaced as Equation (4);
  if the number of words to replace is greater than κ **then break**;
  generate a set of ψ candidate words $\mathcal{C}_i$ by BERT;
  remove the words from $\mathcal{C}_i$ if they do not have the same part-of-speech as the x_i;
  choose the word $x_i^* \in \mathcal{C}_i$ to replace x_i as Equation (5);
  x' = replace x_i with the word x_i^* in x.
return x'.

Figure 4: Adversarial example generation algorithm for dependency parsing in the white-box setting.

5 Experiments

We first describe the target parser as well as its three variants, evaluation dataset, and hyper-parameter settings. We then report the empirical results of the proposed adversarial attacking and training. We also list some adversarial examples generated by our attacking algorithms in Table 5.

5.1 Target Parser and Its Variants

We choose the graph-based dependency parser proposed by Dozat and Manning (2017) as our target model. This well-known parser achieved 95.7% unlabeled attachment scores (UAS) and 94.1% labeled attachment scores (LAS) on English PTB dataset and close to state-of-the-art performance on standard treebanks for five other different natural languages (Buchholz and Marsi, 2006).

Specifically, Dozat and Manning (2017) extends bidirectional LSTM-based approach (Kiperwasser and Goldberg, 2016) with biaffine classifiers to predict arcs and labels. They presented two variants of their model: one takes only words as input, and the other takes both the words and their POS tags. Moreover, we use the Stanford POS tagger (Toutanova et al., 2003) to generate the POS tag for each word. In addition to these two, we add a new

tence examples for the experiment. According to our statistics on the English PTB test set, 35.14% sentences have two such subtrees, 17.18% have three, and 8.98% have four or more.

Model	Max%	Word-based			Word + POS			Character-based		
		UAS	#Word	Succ%	UAS	#Word	Succ%	UAS	#Word	Succ%
Clean	−−	95.52	−−	−−	95.58	−−	−−	95.73	−−	−−
Black-box	5%	90.91	0.99	42%	90.87	1.00	41%	91.18	1.09	37%
	10%	89.38	1.52	49%	90.20	1.54	43%	88.49	1.99	51%
	15%	88.69	2.23	51%	89.86	2.24	44%	85.89	3.08	60%
White-box	5%	87.80	0.60	55%	89.76	0.50	46%	90.37	0.40	37%
	10%	83.73	1.50	68%	86.36	1.40	61%	86.58	1.20	54%
	15%	80.35	2.40	77%	83.75	2.10	69%	83.25	1.90	64%

Table 1: Results of sentence-level adversarial attacks on a state-of-the-art parser with the English Penn Treebank in both the black-box and white-box settings. "Word-based", "Word + POS", and "Character-based" denote three variants of the model (Dozat and Manning, 2017) with differences in their input forms. "Max%" denotes the maximum percentage of words that are allowed to be modified, "UAS" unlabeled attachment scores, "#Word" the average number of words actually modified, and "Succ%" the success rate in terms of the number of sentences.

Model	Word-based			Word + POS			Character-based		
	Original	Adv [b]	Adv [w]	Original	Adv [b]	Adv [w]	Original	Adv [b]	Adv [w]
Clean	95.52	95.59	95.16	95.58	95.53	95.05	95.73	95.55	95.34
Attack [b]	88.69	90.03	89.98	89.86	91.86	91.60	85.89	92.93	89.89
Attack [w]	80.35	80.82	88.87	83.75	84.89	90.32	83.25	84.10	86.56

Table 2: Performance of adversarial training. "Clean" stands for the testing results on the clean data, and "Attack [b]" and "Attack [w]" respectively denote the accuracy under test-time attacks in the black-box ([b]) and white-box ([w]) settings. "Original" and "Adv" denotes the testing and adversarial accuracy of the models without and with the adversarial training.

variant that takes characters as inputs, and uses a bidirectional LSTM to generate word representations from the character embeddings.

Model	POS	Word-based		Word + POS	
		ΔUAS	Succ%	ΔUAS	Succ%
Black-box	**JJ**	−1.89	23%	−1.13	17%
	NN	−2.00	24%	−1.25	20%
	RB	−3.13	37%	−2.43	31%
	VB	−7.42	48%	−6.17	41%
	IN	−11.10	67%	−9.22	62%
White-box	**JJ**	−4.48	37%	−2.23	25%
	NN	−10.53	65%	−8.33	57%
	RB	−4.09	40%	−3.14	35%
	VB	−13.36	73%	−10.51	63%
	IN	−15.58	87%	−13.24	85%

Table 3: The attack success rate and the corresponding changes in UAS by modifying the words with different part-of-speech. "JJ" denotes adjective, "NN" noun, "RB" adverb, "VB" verb, and "IN" preposition.

5.2 Datasets and Hyper-parameter Settings

We evaluate our methods on the English Penn Treebank (PTB), converted into Stanford dependencies using version 3.3.0 of the Stanford dependency converter (de Marneffe et al., 2006)[4]. We follow the standard PTB split, using section 2-21 for training, section 22 for development and 23 for testing.

[4]We ask for the copula (linking verbs) to remain the head when its complement is an adjective or noun.

For the target parsing models, we use the same choice of hyperparameters as (Dozat and Manning, 2017): 100-dimensional uncased word embeddings and POS tag vectors; three bi-directional LSTM layers (400 dimensions in each direction); and 500- and 100-dimensional ReLU MLP layers for arc and label predictions respectively. For the character-based variant, we use 100-dimensional character vectors, and 200-dimensional LSTM. The other hyper-parameters were tuned with the PTB 3.3.0 development set by trying only a few different settings. In the following experiments, the maximum size of candidate words ψ was set to 50, the coefficient α in Equation (5) to 15, and the maximum number of trials to 40. For each example, we terminate the trials immediately if the drop in UAS is more than 30% in the white-box setting.

5.3 Results of the Sentence-level Attacks

We now report the empirical studies of adversarial attacks for sentence-level methods. In Table 1, we present both clean accuracy and accuracy under attacks on PTB with the three variants of the parsing model (Dozat and Manning, 2017), where we allow three different, 5%, 10% and 15% word replacements. A success rate is defined as the number of sentences successfully modified (causing the model to make errors) divided by all the number of sen-

tences to be attempted. The results show that the proposed attacks are effective. With less than two words perturbed on average, our white-box attack can consistently achieve $> 60\%$ success rate.

We also observe that the word-based model is most vulnerable to the adversarial examples among the three variants. Its performance drops 15.17% in UAS, and 77% sentence examples admit the adversarial perturbations under the white-box attack with 15% word replacement. The model taking the words and their POS tags as input ("Word + POS") seems to be more robust against adversarial examples in both settings. One reasonable explanation is that we require the substitute words to have the same part-of-speech as the original ones, and the model can produce more consistent results with the help of the POS tags. The white-box attacks are clearly much more effective than the black-box ones across the three variants of the parsing model and different word replacement rates.

Despite having high success rates, we want to know whether the generated examples are syntactically faithful to and coherent with the original sentences. To evaluate the quality of these adversarial examples, we randomly collect 100 sentences and their adversarial examples each generated in the black-box and white-box settings, and presented them to three human evaluators. The evaluators were asked to examine whether each generated example still preserve the original syntactic structure. We adopted a majority vote for the results, and found that 80% examples generated in the white-box setting and 75% in the black-box setting are considered unchanged in their syntactic structures.

The three human evaluators are postgraduate students with at least three years of research experience in syntactic parsing. Those three annotators' pairwise-agreement percentages are 90%, 82%, and 82% for the adversarial examples generated in the white-box setting, and 93%, 85%, 84% for those generated in the black-box setting. Their average Kappa coefficients are 53.8% (white-box), and 67.3% (black-box) respectively. In Table 5, we listed five sentences and their adversarial examples generated by our algorithms each in the black-box and white-box settings, which were randomly extracted from the PTB test set.

We would like to know which type of words to modify is most likely to form a successful attack like (Hashemi and Hwa, 2016). In this experiment, we only allowed to replace the words belonging to one part of speech, and also tried to generate adversarial examples by replacing prepositions, which is forbidden in the above experiments. It can be seen from Table 3 that the following dependencies especially suffer: prepositional, verbal and adverbial phrases. Not surprisingly most of the errors occur with structures which are inherently hard to attach in the dependency parsing.

Model	Word-based		Word + POS	
	$k \geq 0$	$k \geq 1$	$k \geq 0$	$k \geq 1$
Black-box	34.73%	21.72%	19.61%	10.06%
White-box	40.06%	28.66%	25.35%	15.82%

Table 4: The success rate of the phrase-level attacks.

5.4 Results of the Phrase-level Attacks

For the phrase-level attacks, we aim to study whether changes in a source subtree can alter the prediction on another target subtree (see an illustration in Figure 2). We tried two different settings: one asks for the source and target subtrees to be separated by at least one word ($k \geq 1$), and another only requires those two subtrees do not overlap with each other ($k \geq 0$). In the case of $k \geq 0$, we can find 1420 sentence examples from the test set, while for $k \geq 1$, there are 1340 valid examples that can be used to deliver phrase-level attacks (there are 2416 sentences in total in the PTB test set). Note that all the subtrees should contain 4 to 12 words. For each source-target pair, we allow to modify the source subtree up to 3 words. For some sentences, their adversarial examples can be generated by replacing just one or two words.

The success rate for the phrase-level attacks is defined as the ratio between the number of the sentences where there is at least one source-target subtree pair, such that a modification in the source subtree causes the model to make errors in the target subtree, and the number of the sentences that contain at least one source-target subtree pair, regardless of whether the model is caused to make an error or not. It can be seen from Table 4 that with only three words perturbed, the proposed white-box attack can achieve 27.47% success rate on average for all the settings. The white-box attacks are again much more effective, and spend less than 50% of the time to find the most vulnerable pairs than the black-box ones. Like the sentence-level attacks, verbal and prepositional phrases have been shown to be more susceptible to such attacks.

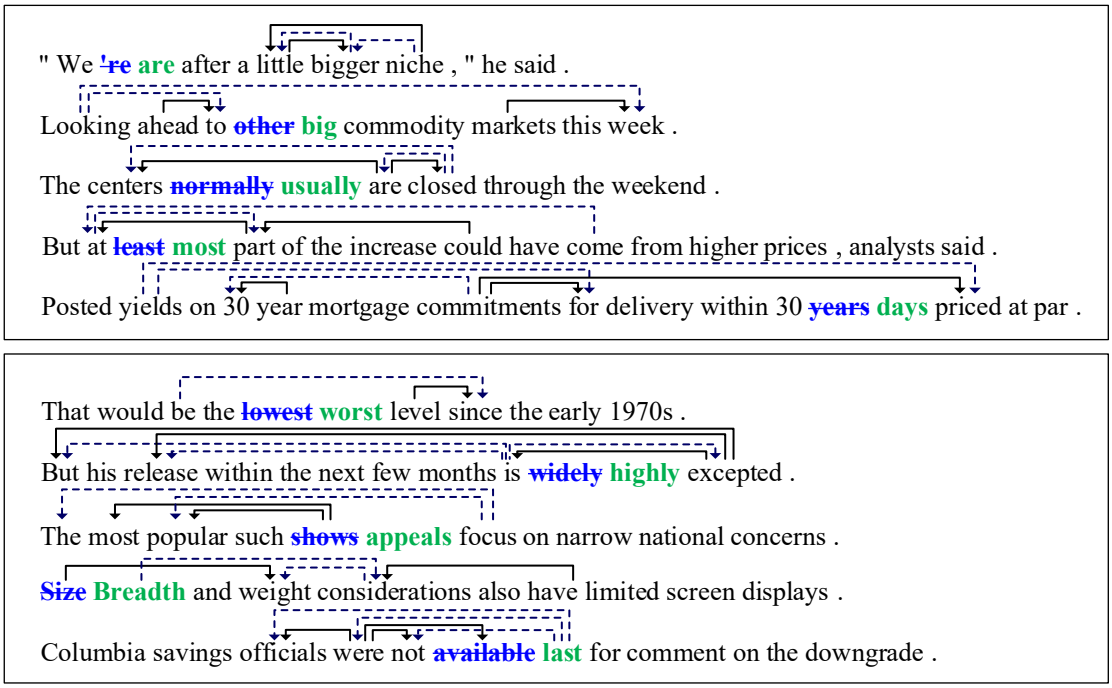

Figure 5: Five adversarial examples each generated by our algorithms in the black-box (top) and white-box (bottom) settings. These adversarial examples were randomly extracted from the test set of the English Penn Treebank (PTB). The original words are highlighted in bold blue font while the substitute words are highlighted in bold green ones. The incorrect arcs (i.e. head-modifier pairs) predicted by the target parser are indicated by dash arrows while the ground truth arcs are indicated by solid arrows.

5.5 Adversarial Training

We also investigated whether our adversarial examples can aid in improving model robustness. We randomly selected 50% of the training data and generated adversarial examples from them using the algorithms listed in Figure 3 and 4. We merged these adversarial examples with the original training set. Some previous studies show that the models tend to overfit the adversarial examples, and their performance on the clean data will drop if too many adversarial examples are used. Therefore, we used a similar training strategy.

The testing and adversarial performance with and without adversarial training are listed in Table 2. Under all circumstances, adversarial training improved the generalization of the models and made them less vulnerable to the attacks, while suffering little to no loss in on the clean data. For example, 88.69 (column 1, row 2) is the accuracy achieved by the original model on the adversarial examples generated in the black-box setting, 90.03 (column 2, row 2) and 89.98 (column 3, row 2) are the accuracy achieved on the perturbed test data with the test-time adversarial attacks by the models with the adversarial training. It is clear that the robustness of parsing models was improved by the adversarial training. Furthermore, from the first row of Table 2 these robust models suffer from little to no performance drop on the clean testing data.

6 Conclusion

In this paper, we study the robustness of neural network-based dependency parsing models. To the best of our knowledge, adversarial examples to syntactic tasks, such as dependency parsing, have not been explored in the literature. We develop the first adversarial attack algorithms for this task to successfully find the blind spots of parsers with high success rates. Furthermore, by applying adversarial training using the proposed attacks, we are able to significantly improve the robustness of dependency parsers without sacrificing their performance on clean data.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported by National Key R&D Program of China (No. 2018YFC0830902), Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01) and Zhangjiang Lab.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Samuel Barham and Soheil Feizi. 2019. Interpretable adversarial training for text. *Computing Research Repository*, arXiv: 1905.12864.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the International Conference on Computational Natural Language Learning*.

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

Minhao Cheng, Wei Wei, and Cho-Jui Hsieh. 2019. Evaluating and enhancing the robustness of dialogue systems: A case study on a negotiation agent. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

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. *Computing Research Repository*, 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 *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the International Conference on Learning Representations*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Steffen Eger, Gozde Gul Sahin, Andreas Rucklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text processing like humans do: Visually attacking and shielding NLP systems. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. *Computing Research Repository*, arXiv: 1801.04354.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*.

Homa B. Hashemi and Rebecca Hwa. 2016. An evaluation of parser robustness for ungrammatical sentences. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yu-Lun Hsieh, Minhao Cheng, Da-Cheng Juan, Wei Wei, Wen-Lian Hsu, and Cho-Jui Hsieh. 2019. On the robustness of self-attentive models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

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. *Computing Research Repository*, arXiv: 1907.11932.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Volodymyr Kuleshov, Shantanu Thakoor, Tingfung Lau, and Stefano Ermon. 2018. Adversarial examples for natural language classification problems. *OpenReview Submission*, id: r1QZ3zbAZ.

Qi Lei, Lingfei Wu, Pin-Yu Chen, Alexandros G. Dimakis, Inderjit S. Dhillon, and Michael Witbrock. 2019. Discrete adversarial attacks and submodular optimization with applications to text classification. In *Proceedings of the Conference on Systems and Machine Learning*.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep text classification can be fooled. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the International Conference on Language Resources and Evaluation*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Dominick Ng and James R. Curran. 2015. Identifying cascading errors using constraints in dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2016a. The limitations of deep learning in adversarial settings. In *Proceedings of the IEEE European Symposium on Security and Privacy*.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016b. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

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

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *Computing Research Repository*, arXiv: 1312.6199.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*.

Wenqi Wang, Lina Wang, Benxiao Tang, Run Wang, and Aoshuang Ye. 2019. A survey: Towards a robust deep neural network in text domain. *Computing Research Repository*, arXiv: 1902.07285.

Catherine Wong. 2017. DANCin SEQ2SEQ: Fooling text classifiers with adversarial text example generation. *Computing Research Repository*, arXiv: 1712.05419.

Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. 2019. Adversarial attacks and defenses in images, graphs and text: A review. *Computing Research Repository*, arXiv: 1909.08072.

Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I. Jordan. 2018. Greedy attack and gumbel attack: Generating adversarial examples for discrete data. *Computing Research Repository*, arXiv: 1805.12316.

Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824.

Yuan Zang, Chenghao Yang, Fanchao Qi, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2019. Textual adversarial attack as combinatorial optimization. *Computing Research Repository*, arXiv: 1910.12196.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *Proceedings of the International Conference on Learning Representations*.