DISCRETE ATTACKS AND SUBMODULAR OPTIMIZATION WITH APPLICATIONS TO TEXT CLASSIFICATION

Anonymous Authors[1]

ABSTRACT

Adversarial examples are carefully constructed modifications to an input that completely change the output of a classifier but are imperceptible to humans. Despite these successful attacks for continuous data (such as image and audio samples), generating adversarial examples for discrete structures such as text has proven significantly more challenging. In this paper we formulate the attacks with discrete input on a set function as an optimization task. We prove that this set function is submodular for some popular neural network text classifiers. This finding guarantees a $1 - 1/e$ approximation factor for attacks that use the greedy algorithm. Meanwhile, we show how to use the gradient of the attacked classifier to guide the greedy search. Empirical studies with our proposed optimization scheme show significantly improved attack ability and efficiency, on three different text classification tasks over various baselines. We also use a joint sentence and word paraphrasing technique to maintain the original semantics and syntax. This is validated by a human subject evaluation in subjective metrics on the quality and semantic coherence of our generated adversarial text.

1 INTRODUCTION

Adversarial examples are carefully constructed modifications to an input that completely change the output of a classifier but are imperceptible to humans. Spam filtering and the carefully-crafted emails designed to fool these early classifiers are the first examples of adversarial machine learning going back to 2004 (Dalvi et al., 2004; Lowd & Meek, 2005); see also the comprehensive survey by Biggio et al. (Biggio & Roli, 2017). Szegedy et al. (Szegedy et al., 2013) discovered that deep neural network image classifiers can be fooled with tiny pixel perturbations; exploration of this failure of robustness has received significant attention recently, see e.g. (Goodfellow et al., 2014; Moosavi Dezfooli et al., 2016; Papernot et al., 2016b; Evtimov et al., 2017; Su et al., 2018). Adversarial training (Goodfellow et al., 2014; Madry et al., 2017) seems to be the state of the art in defense against adversarial attacks, but creating robust classifiers remains challenging, especially for large image classifiers, see e.g. Athalye at al. (Athalye et al., 2018).

Despite these successful attacks for continuous data (such as image and audio samples), generating adversarial examples for discrete structures such as text and code has proven significantly more challenging in two aspects:

One challenge is how to develop a fast yet (provably) effective attacking scheme. Gradient-based adversarial attacks for continuous data no longer directly apply to discrete structures. Although some variants are proposed when the model is differentiable to the embedding layer (Papernot et al., 2016a; Li et al., 2016; Ebrahimi et al., 2017; Gong et al., 2018), this line of methods achieve efficiency but suffer from poor success rate.

Meanwhile, another natural idea is to find feasible replacement for individual features like words or characters. However, since the space of possible combinations of substitutions grows exponentially with the length of input data, finding the optimal combination of substitutions is intractable. Recent heuristic attacks on NLP classifiers operate by greedy character-level or word-level replacements (Ebrahimi et al., 2017; Kuleshov et al., 2018; Yang et al., 2018). However, greedy methods are usually slow, and it's theoretically not understood when they achieve good performance.

The other issue is how to maintain the original functionality of the input. Specifically for text, it remains challenging to preserve semantic and syntactic properties of the original input from the point of view of a human. Existing methods either require to change too many features, or change the original meaning. For instance, (Kuleshov et al., 2018) alters up to 50% of words in each input document to achieve a 30% success rate. (Gong et al., 2018) attacks the document by replacing with completely different words. (Jia & Liang, 2017) inserts irrelevant sentences to the original text. Such changes can be easily detected by humans.

In this paper we argue that these limitations can be be resolved with the framework we propose. We highlight our main contributions as follows:

We propose a general framework for discrete attacks. We apply our framework to designing adversarial attacks for text classifiers but our techniques apply more broadly. For instance, the attacks include but are not limited to malware detection, spam filtering, or even discrete attacks defined on continuous data, e.g., segmentations from an image.

We formulate the attacks with discrete input on a set function as an optimization task. This problem, however, is provably NP-hard even for convex classifiers. We unify existing gradient-based as well as greedy methods using a general combinatorial optimization via further assumptions. We note that gradient methods solve a relaxed problem in polynomial time; while greedy algorithm for creating attacks has a provable $1 - 1/e$ approximation factor assuming the set function is submodular. We theoretically show that for two natural classes of neural network text classifiers, the set functions defined by the attacks are submodular. We specifically analyze two classes of classifiers: The first is word-level CNN without dropout or softmax layers. The second is a recurrent neural network (RNN) with one-dimensional hidden units and arbitrary time steps.

Nevertheless, greedy methods can be very time consuming when the space of attacks is large. We show how to use the gradient of the attacked classifier to guide the combinatorial search. Our proposed gradient-guided greedy method is inspired by the greedy coordinate descent Gauss-Southwell rule from continuous optimization theory. The key idea is that we use the magnitude of the gradient to decide which features to attack in a greedy fashion.

We extensively validate the proposed attacks empirically. With the proposed optimization scheme, we show significantly improved attack performance over most recent baselines. Meanwhile we propose a joint sentence and word paraphrasing technique to simultaneously ensure retention of the semantics and syntax of the text.

2 Related Work

Broadly speaking, adversarial examples refer to minimally modified natural examples that are spurious but perceptually similar and that lead to inconsistent decision making between humans and machine learning models. An example is automatically classifying an adversarial stop sign image (according to humans) as a speed limit sign. For continuous data such as images or audio, generating adversarial examples is often accomplished by crafting additive perturbations of natural examples, resulting in visually imperceptible or inaudible noise that misleads a target machine learning model. These small yet effective perturbations are difficult for humans to detect, but will cause an apparently well-trained machine learning model to misbehave; in particular, neural networks have been shown to be susceptible to such attacks (Szegedy et al., 2013), giving rise to substantial concern about safety-critical and security-centric machine learning applications.

For classifiers with discrete input structures, a simple approach for generating adversarial examples is to replace each feature with similar alternatives. Such features for text classification tasks are usually individual words or characters. Such attacks can be achieved using continuous word embeddings or with respect to some designed score function; this approach has been applied to attack NLP classifiers (Papernot et al., 2016a; Li et al., 2016; Miyato et al., 2016; Samanta & Mehta, 2017; Liang et al., 2017; Yao et al., 2017; Gong et al., 2018; Kuleshov et al., 2018; Gao et al., 2018; Alzantot et al., 2018; Yang et al., 2018) and sequence-to-sequence models (Ebrahimi et al., 2017; Wong, 2017; Zhao et al., 2018; Cheng et al., 2018). The work in (Ribeiro et al., 2018) considers semantically equivalent rules for debugging NLP models, but under the same input structure. This is a natural but limited practice to only consider attacks within one input structure, namely word or characters, but no joint attacks, nor the effect incurred from sentences. Unlike prior work, we conduct a joint sentence and word paraphrasing technique. It considers sentence-level factors and allows more degrees of freedom in generating text adversarial examples, by exploring the rich set of semantically similar paraphrased sentences.

Jia and Liang studied adversarial examples in reading comprehension systems by inserting additional sentences (Jia & Liang, 2017), which is beyond the concept of this paper since the approach changes the original meanings. Another related line of research, although not cast as adversarial examples, focuses on improving model robustness against out-of-vocabulary terms (Belinkov & Bisk, 2017) or obscured embedding space representations (Mrkšić et al., 2016).

3 Preliminary

In this paper, we propose a general framework for generating adversarial examples with discrete input data. A collection of such data and corresponding attacks are presented in Table 1.

To present our mathematical formulation, we start by introducing some notation.

Input Structure. Let the input $\mathbf{x} = [x_1, x_2, \cdots, x_n] \in \mathcal{X}^n$ be a list of n features (might be padded). For text environment, the feature space $\mathcal{X}$ could be the character, word, phrase, or sentence space. For the problem of malware detection, $\mathbf{x}$ could be a concatenation of code pieces.

Remark 1. *For concrete usage, we use $w \in \mathcal{W}$ to denote word space, and $s \in \mathcal{S}$ to denote sentences to distinguish the differences.*

Task: Sentiment Analysis. Classifier: LSTM. Original: 100% Positive. ADV label: 100% Negative.

I suppose I should write a review here since my little Noodle-oo is currently serving as their spokes dog in the photos. We both love Scooby Do's. They treat my little butt-faced dog like a prince and are receptive to correcting anything about the cut that I perceive as being weird. Like that funny poofy pompadour. Mohawk it out, yo. Done. In like five seconds my little man was looking fabulous and bad ass. Not something easily accomplished with a prancing pup that literally chases butterflies through tall grasses. (He ended up looking like a little lamb as the cut grew out too. So adorable.) The shampoo they use here is also amazing. Noodles usually smells like tacos (a combination of beef stank and corn chips) but after getting back from the Do's, he smelled like Christmas morning! Sugar and spice and everything nice instead of frogs and snails and puppy dog tails. He's got some gender identity issues to deal with. ~~The pricing is also cheaper than some of the big name conglomerates out there~~ The price is cheaper than some of the big names below. I'm talking to you Petsmart! I've taken my other pup to Smelly Dog before, but unless I need dog sitting play time after the cut, I'll go with Scooby's. They genuinely seem to like my little Noodle monster.

Task: Fake-News Detection. Classifier: LSTM. Original label: 100% Fake. ADV label: 77% Real

~~Man~~ Guy punctuates high-speed chase with stop at In-N-Out Burger drive-thru Print [Ed.~~—Well, that's~~ Okay, that 's a new one.] ~~A~~ One man is in custody after leading police on a bizarre chase into the east Valley on Wednesday night. Phoenix police ~~began~~ has begun following the suspect in Phoenix and the pursuit continue~~d~~ into the east Valley, but it took a bizarre turn when the suspect stopped at an In-N-Out Burger restaurant's ~~drive-thru~~ drive-through near Priest and Ray Roads in Chandler. The suspect appeared to order food, but then drove away and got out of his pickup truck near Rock Wren Way and Ray Road. He ~~then ran into a backyard~~ ran to the backyard and tried to ~~get into a house through the back door~~ get in the home.

Task: Spam Filtering. Classifier: WCNN. Original label: 100% None-spam. ADV label: 100% Spam

> > Hi All,
> > I'm new to R from a C and Octave/Matlab background. ~~I am trying to > > construct~~ I 'm trying to build some classes in R to which I want to attach ~~pieces of~~ data.
> > First, is attr(obj, 'member name') > > this? > > No, it isn't. You seem to be trying to deduce new-style classes from a > > representation used before R 2.4, (actually, still used) > > but in any case it would not be » sensible. ~~Please consult~~ Contact John M. Chambers. Programming with Data. > > Springer, New York, 1998, and/or William N. Venables and Brian D. Ripley. > > S Programming. Springer, New York, 2000, or for a shorter online resource: > > http://www.stat.auckland.ac.nz/S-Workshop/Gentleman/Methods.pdf > > Unfortunately, all of those references are at least 4 years out of > date when it comes to S4 methods. Is there any comprehensive > reference of the current implementation of the S4 OO system apart from > the source code? Not that I know of, and ~~it is~~ it's a moving target. (E.g. I asked recently about some anomalies in the S4 bit introduced for 2.4.0 and what the intended semantics are.) I've said before that I believe we can only help solve some of the efficiency issues with S4 if we have a technical manual. It is unfair to pick out S4 here, but the 'R Internals' manual is an attempt to document important implementa~~tion~~ ing details (mainly by studying the code), and that has only got most of the way through src/main/*.c.

Figure 1. Examples of generated adversarial examples. The color red denotes sentence-level paraphrasing, and blue denotes word-level paraphrasing.

input data	task
document	text classification
code	malware detection
url address	malicious website check

Table 1. Applications to the framework.

Embedding V. The embedding layer is a key transition from discrete input data into continuous space, which could then be fed into the classifier. For text domain, we typically use the bag-of-words embedding or word-to-vector embedding.

For a bag-of-words embedding, $V : \mathcal{X}^n \to \mathbb{R}^D$ represents a document as the statistics of word counts, i.e., the summation of each word's one-hot representation. Meanwhile, word-to-vector embeddings characterize different words as D-dimensional vectors, i.e., $V(x) \in \mathbb{R}^D, \forall x \in \mathcal{X}$. When there's no ambiguity, we also use $V : \mathcal{X}^n \to \mathbb{R}^{n\times D}$ to denote the concatenation of word vectors of the input document as a list of words.

Transformation Indexing. Suppose each feature $x \in \mathcal{X}$ has (at most) $k-1$ possible replacements, denoted by $x^{(i)}, i \in [k-1](\equiv \{1, 2, \cdots, k-1\})$. For future use, we also define $x^{(0)} = x, \forall x \in \mathcal{X}$.
A valid transformation T is the combined replacement of each individual feature $x_i, i \in [n]$. Therefore we index T by a vector $\mathbf{l} \in \{0, 1, \cdots, k-1\}^n$, and l_i indicates the index of each replacement i. Namely, $T_{\mathbf{l}}(\mathbf{x} = [x_1, x_2, \cdots, x_n]) = [x_1^{(l_1)}, x_2^{(l_2)}, \cdots, x_n^{(l_n)}]$. An example with word replacement in the text classification environment could be found in Figure 2.

Classifier output C_y. We consider a targeted attack, i.e., we want to maximize the output probability C over a specific target label y.

In this paper, we use a regular lower-case symbol to denote a

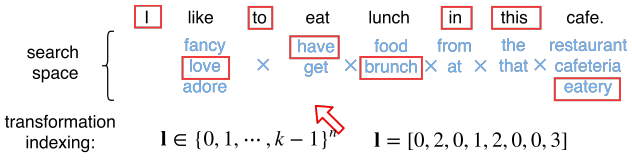

Figure 2. An illustration of the transformation indexing when applying to a text sentence. In this example, the transformation denoted as $\mathbf{l}$ modifies the original sentence to the new one shown in the red boxes.

scalar or a single feature, and use a bold lower-case symbol for a vector or a list of features.

3.1 Problem Setup

In most scenarios, we only allow transformations on at most m features, then the constraint is $\|\mathbf{l}\|_0 \leq m$. Therefore we present the adversarial attack problem formally:

Problem 1. *For some input data $\mathbf{x} \in \mathcal{X}^n$, we try to find a feasible transformation $T_{\mathbf{l}^*}$, where $\mathbf{l}^* \in \{0, 1, \cdots k-1\}^n$ is the index, such that:*

$$\mathbf{l}^* = \underset{\|\mathbf{l}\|_0 \leq m}{\arg\max}\, C_y\left(V\left(T_{\mathbf{l}}(\mathbf{x})\right)\right). \tag{1}$$

Or similarly, we want to find the set of features to attack, i.e.,

$$S^* = \underset{|S| \leq m}{\arg\max}\, f(S), \tag{2}$$

where we defined the set function $f : 2^{[n]} \to \mathbb{R}$, $f(S) = \max_{supp(\mathbf{l}) \subset S} C_y(V(T_{\mathbf{l}}(\mathbf{x})))$.

The set function $f(S)$ represents the classifier output for the target label y if we apply a set of transformations S. We are therefore searching over all possible sets of up to m replacements to maximize the probability of the target label output of a classifier.

Remark 2. *In this paper, we focus on replacements via word and sentence paraphrasing for empirical studies. However, our formulation is general enough to represent any set of discrete transformations. Possible transformations include replacement with the nearest neighbor of the gradient direction (Gong et al., 2018) and word vectors (Kuleshov et al., 2018), or flipping characters within each word (Ebrahimi et al., 2017). We will also conduct a thorough experimental comparisons among different choices.*

4 Theoretical Analysis

First, notice that the original problem is computationally intractable in general:

Proposition 1. *For a general classifier C_y, the problem 1 is NP-hard. Specifically, even for some convex C_y, the problem 1 can be polynomially reduced to subset sum and hence is NP-hard.*

All proofs in this paper are presented in the Appendix.

4.1 Unifying Related Methodology via Further Assumptions

Fortunately, with further assumptions it becomes possible to solve problem 1, above, in polynomial time. Some existing heuristics are proposed to generate adversarial examples for the text classification problem. Though usually not specifically proposed in the relevant literature, we unify the underlying assumptions for these heuristics to succeed in polynomial time in this section.

One possible assumption is that the original function C_y is smooth, which could afterwards be approximated by its first-order Taylor expansion:

$$\begin{aligned} C_y(V(T_{\mathbf{l}}(\mathbf{x}))) \quad &= \quad C_y(\mathbf{v}) + \langle \nabla C_y(\mathbf{v}), V(T_{\mathbf{l}}(\mathbf{x})) - \mathbf{v} \rangle \\ &\quad + \mathcal{O}\left(\|V(T_{\mathbf{l}}(\mathbf{x})) - \mathbf{v}\|_2^2\right) \end{aligned}$$

where $\mathbf{v} = V(\mathbf{x})$. Therefore, Problem 1 can be relaxed as follows:

Problem 2. *Given gradient $\nabla C_y(\mathbf{v})$, where $\mathbf{v} = V(\mathbf{x})$, maximize function C_y by its first-order Taylor expansion:*

$$\mathbf{l}^* = \underset{\|\mathbf{l}\|_0 \leq m}{\arg\max}\, V(T_{\mathbf{l}}(\mathbf{x}))^\top \nabla C_y(\mathbf{v}). \tag{3}$$

Problem 2 is similar to the Frank-Wolfe method (Frank & Wolfe, 1956) in continuous optimization and is easy to solve:

Proposition 2. *Problem 2 can be solved in polynomial time for both bag-of-words and word to vector embeddings. Specifically, $f(S) = \arg\max_{supp(\mathbf{l}) \subset S} V(T_{\mathbf{l}}(\mathbf{x}))^\top \Delta C_y(\mathbf{v})$ could be written as $\sum_{i \in S} w_i$ for some w irrelevant to S, where $\mathbf{v} = V(\mathbf{x})$.*

Related methods like (Gong et al., 2018) are attempts to solve problem 2. They propose to conduct transformations via replacement by synonyms chosen by (3). However, activations like ReLU break the smoothness of the function, and first order Taylor expansion only cares about very local information, while embeddings for word synonyms could be actually not that close to each other. Consequently, this unnatural assumption prevents related gradient-based attacks to achieve good performance.

Besides smoothness, another more natural assumption is that $f(S)$ in the original problem 1 is submodular (Narayanan, 1997; Fujishige, 2005). Submodular is a property that is defined for set functions, which characterizes the diminishing returns of the function value change as the size of the input set increases.

Definition 1. *(Schrijver, 2003) If Ω is a finite set, a submodular function is a set function $f : 2^{\Omega} \rightarrow \mathbb{R}$, where 2^{Ω} denotes the power set of Ω , which satisfies one of the following equivalent conditions.*

1. *For every $X, Y \subseteq \Omega$ with $X \subseteq Y$ and every $x \in \Omega \setminus Y$ we have that $f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y)$.*

2. *For every $S, T \subseteq \Omega$ we have that $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$.*

3. *For every $X \subseteq \Omega$ and $x_1, x_2 \in \Omega \setminus X$ we have that $f(X \cup \{x_1\}) + f(X \cup \{x_2\}) \geq f(X \cup \{x_1, x_2\}) + f(X)$.*

With the design of $f(S)$ in Problem 1 to be monotone non-decreasing and if we further assume f to be submodular, our task becomes to maximize a monotone submodular function subject to a cardinality constraint (Nemhauser et al., 1978). Therefore, greedy method guarantees a good approximation of the optimal value of Problem 1:

Claim 1. *In problem 1, f is monotone non-decreasing. Furthermore, if the function f is submodular, greedy methods achieve a $(1 - 1/e)$-approximation of the optimal solution in polynomial time.*

Both our work and the optimization scheme from (Kuleshov et al., 2018) propose some variants of greedy methods with the underlying submodular assumption.

The greedy method proposed in (Kuleshov et al., 2018) selects candidate replacements directly by function value, one word at a time, which we will refer as the objective-guided greedy method. We will propose a more efficient yet comparable effective greedy method that is guided by the gradient magnitude in Section 5.2, and compare with the above two methods in Section 6.3. As an extension from the continuous optimization, our method uses the well-studied Gauss-Southwell rule (Nutini et al., 2015) that is provably better than random selection. In each iteration, we determine and select the most important words by the gradient norm of words' embeddings, and then find the greediest transformation within the search space of the selected words. The advantage is that we are able to conduct multiple replacements in one iteration and thus take into consideration the joint effect of multiple words replacements. We will introduce our method, which we call Gradient-Guided Greedy Word Paraphrasing in Algorithm 3, and will show empirical performance comparison with the (objective-guided) greedy method (Kuleshov et al., 2018) and the gradient method used in (Gong et al., 2018) in Section 6.3.

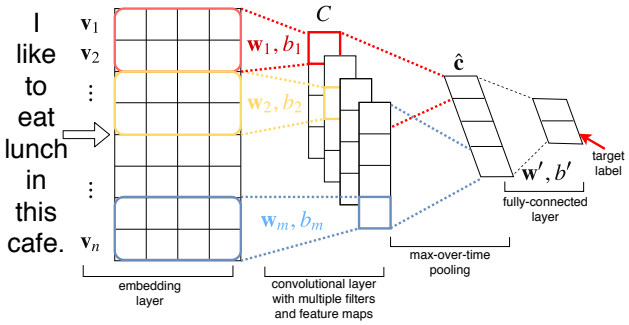

Figure 3. Model architecture of simplified W-CNN for an example sentence.

4.2 Submodular Neural Networks on the Set of Attacks

To argue that submodular is a natural assumption, we study and summarize the neural networks are submodular on the set of attacks.

In (Bilmes & Bai, 2017), it provides a class of submodular functions used in the deep learning community called deep submodular functions. Nevertheless the deep submodular functions are not necessarily applicable to our set function. We hereby formally prove the following two kinds of neural networks, that are ubiquitously used for text classification, indeed satisfying submodular property on the set of attacks under some conditions.

Simplified W-CNN (Kim, 2014)
Denote the stride as s, the number of grams (window size) h, and the word vector of the i-th word in a document as $\mathbf{v}_i (\equiv V(x_i))$. Then the output for the convolutional layer is a matrix $C = [c_{ij}]_{i \in [n/s], j \in [m]}$ from n words and m filters:

$$c_{ij} = \phi(\mathbf{w}_j^{\top} \mathbf{v}_{s(i-1)+1:s(i-1)+h} + b_j), \ i = 1, 2, \cdots n/s,$$

where $\mathbf{w}_j \in \mathbb{R}^{Dh}$ is the j-th filter, b_j is the corresponding bias term and ϕ is the non-linear, and non-decreasing activation such as ReLU, tanh and sigmoid function. $\mathbf{v}_{i:j}$ denotes the concatenation of word vectors in the window of words i through j, namely $[\mathbf{v}_i^{\top}, \mathbf{v}_{i+1}^{\top}, \cdots \mathbf{v}_j^{\top}]^{\top} \in \mathbb{R}^{D(j-i+1)}$. Each filter $\mathbf{w}_j$ is applied to individual windows of words to produce a feature map $\mathbf{c}^j = [c_{1j}, c_{2j}, \cdots c_{n/s,j}]^{\top}$.

Afterwards, a max-over-time pooling is applied to each feature map to form the penultimate layer $\hat{\mathbf{c}} = [\hat{c}_1, \hat{c}_2, \cdots \hat{c}_m]$, where $\hat{c}_i$ is the largest value in $\mathbf{c}^j$:

$$\hat{c}_j = \max_i c_{ij}.$$

Compared to the original (Kim, 2014) paper, we only omit the dropout and softmax layer, and instead consider the following WCNN classifier output for a target label:

$$C^{\text{WCNN}}(\mathbf{v}_{1:n}) = \mathbf{w}' \cdot \hat{\mathbf{c}} + b' \tag{4}$$

Theorem 1. *We consider the simple version of W-CNN classifier described in (4), and suppose there's no overlapping between each window, i.e., $s \geq h$, and $\mathbf{w}'$ has all non-negative values. If further we only look at transformations that will increase the output, i.e., $\mathbf{w}_j^\top V(x_i^{(t)}) \geq \mathbf{w}_j^\top V(x_i), \forall i \in [n], j \in [m], t \in [k-1]$, then $f^{WCNN}(S) = \max_{supp(\mathbf{1}) \in S} C^{WCNN}(V(T_{\mathbf{1}}(x)))$ is submodular.*

The proof sketch is as follows. Every coordinate in $\hat{\mathbf{c}}$ is a combination of max pooling over a modular function and is therefore submodular. And finally sums of submodular functions is still submodular.

Besides word-level CNN, another network that is popular in the NLP community is the recurrent neural network (RNN) or its variants. We will show that under some conditions, RNN satisfies submodular property.

Recurrent Neural Network with One-dimensional Hidden Units

Consider a RNN with T time steps and each hidden layer is a single node. Then for all $t \leq T$, given the value of a previous hidden state $h_{t-1} \in \mathbb{R}$ and an input word vector $\mathbf{v}_{t-1} \in \mathbb{R}^D$ ($\mathbf{v}_t \equiv V(x_t)$), RNN computes the next hidden state h_t and output vector $\mathbf{o}_t \in \mathbb{R}$ as:

$$h_t = \phi(w h_{t-1} + \mathbf{m}^\top \mathbf{v}_{t-1} + b) \tag{5}$$

The classifier output is $C^{\text{RNN}}(\mathbf{v}_{1:T}) = y h_T$.

Theorem 2. *For a recurrent neural network with T time steps and one-dimensional hidden nodes described in (5), if w and y are positive, and the activation is a non-decreasing concave function, then $f^{RNN}(S) = \max_{supp(\mathbf{1}) \in S} C^{RNN}(V(T_{\mathbf{1}}(\mathbf{x})))$ is submodular.*

This result is quite surprising, since the word vectors influence the network's output on different time steps and are by no means separable. In the proof, we first show that a same amount of change induced on an intermediate layer has a diminishing effect when the network is attacked on more features. Then together with the concavity and non-decreasing property of the network, we are able to finish the proof.

5 Adversarial Text Examples via Paraphrasing

In order to conduct adversarial attacks on models with discrete input data like text, one essential challenge is how to select suitable candidate replacements such that the generated text is both semantic meaning preserving and syntactically valid. Another key issue is how to develop an efficient yet effective optimization scheme to find good transformations. To solve the above two issues, we now propose our methodology for generating adversarial examples for text.

5.1 Joint Sentence and Word Paraphrasing

To coincide with our definition of adversarial examples for text, we need to determine appropriate word and sentence paraphrasing methods that maintain the semantic meaning of original text. Our scheme is to generate an initial set for word and sentence replacements with a well-studied paraphrasing corpus and then filter out discrepant choices based on their semantic and syntactic similarities to the original text. Similar mechanism was also conducted by (Kuleshov et al., 2018) to generate word replacement candidates.

Paraphrasing Corpus.

For word paraphrasing, we use the Paragram-SL999 (Wieting et al., 2015) of 300 dimensional paragram embeddings to generate neighboring paraphrasing for words. For sentences, we use the pretrained model from Wieting and Gimpel's Para-nmt-50m project (Wieting & Gimpel, 2017) to generate sentence paraphrases.

We further specify semantic and syntactic constraints to ensure good quality in adversarial texts:

Semantic similarity.

We use the Word Mover Distance (WMD) (Kusner et al., 2015) to measure semantic dissimilarity. For sentence pairs, WMD captures the minimum total semantic distance that the embedded words of one sentence need to "travel" to the embedded words of another sentence. While for words, WMD directly measures the distance between their embeddings.

Syntactic similarity.

Alongside the semantic constraint, one should also ensure that the generated sentence is fluent and natural. We make use of a language model as in (Kuleshov et al., 2018), $P : \mathcal{X}^n \to [0, 1]$ to calculate the probability of the adversarial sentence, and require:

$$|\ln(P(\mathbf{x})) - \ln(P(\mathbf{x}'))| \leq \delta,$$

where $\mathbf{x}'$ is the adversarial sentence paraphrased from $\mathbf{x}$.

In Algorithm 1, we present the whole procedure of finding the neighboring sets to conduct our proposal joint sentence and word paraphrasing attack. While with more details, we show how to use the objective value as well as gradient information to guide the search in Algorithm 2 (for sentences) and 3 (for words).

5.2 Gradient-Guided Greedy Method

In Section 3.1 we have demonstrated the difficulty of finding the best transformation from combinatorially many choices. Here we specify our proposal, gradient-guided greedy word paraphrasing, as shown in Algorithm 3. We can see that we first use gradient values to determine the index set of N words $(w_{i_1}, w_{i_2}, \cdots w_{i_N})$ that we want to replace (step 4-5). Then in step 7-15 we create a candidate set of all possible transformations in $W_{i_1} \times \cdots \times W_{i_N}$. Finally, we choose

Algorithm 1 *Joint Sentence And Word Paraphrasing*$(C_y, \mathbf{x}^{(0)}, P, \delta, \lambda_s, \lambda_w, \delta_s, \delta_w, \tau, k)$

Input: Classifier C associated with target label y, input document $\mathbf{x}^{(0)}$, language model P trained on the training set, syntactic threshold δ, sentence and word paraphrasing ratio λ_s, λ_w, termination threshold τ, WMD threshold δ_s, δ_w, limit number of paraphrases k.
Conduct sentence separation $\mathbf{x}^{(0)} \rightarrow [s_1, s_2, \cdots s_l], s_i \in \mathcal{S}, 1 \leq i \leq l$. (See Remark 1).
Create sentence neighboring set $\mathbf{S} = \{S_1, S_2, \cdots S_l\}$, where each $S_i \subset \mathcal{S}$ satisfies that $|S_i| \leq k$ and $WMD(s_i, s) \leq \delta_s, \forall s \in S_i$.
$\mathbf{x}^{(1)} \leftarrow$ *Greedy Sentence Paraphrasing*$(C_y, \mathbf{x}^{(0)}, \mathbf{S}, \lambda_s, \tau)$ in Alg. 2.
If $C_y(V(\mathbf{x})) \geq \tau$ **Return** $\mathbf{x}^{(1)}$
Conduct word separation $\mathbf{x}^{(1)} \rightarrow [w_1, w_2, \cdots w_n], w_i \in \mathcal{W}, 1 \leq i \leq n$.
Create word neighboring set $\mathbf{W} = \{W_1, W_2, \cdots W_n\}$, where each $W_i \subset \mathcal{W}$ satisfies that $|W_i| \leq k$ and $WMD(w_i, w) \leq \delta_w, |P(\mathbf{x}^{(1)}) - P(\mathbf{x}'(w))| \leq \delta, \forall w \in W_i$, where $\mathbf{x}'(w)$ is text $\mathbf{x}^{(1)}$ in which w_i is substituted by w.
$\mathbf{x}^{(2)} \leftarrow$ *Gradient Guided Greedy Word Paraphrasing*$(C_y, \mathbf{x}^{(1)}, \mathbf{W}, \lambda_w, \tau)$ in Alg. 3.
Return $\mathbf{x}^{(2)}$

Algorithm 2 *Greedy Sentence Paraphrasing*$(C_y, \mathbf{x}, \mathbf{S}, \lambda_s, \tau)$

Input: Document $\mathbf{x}$ as list of sentences $[s_1, s_2, \cdots, s_l]$, sentence neighboring sets $\mathbf{S} = \{S_1, S_2 \cdots S_n\}$, model C_y and parameters λ_s, τ.
while $C_y(V(\mathbf{x})) \leq \tau$ and number of sentence paraphrased $\leq \lambda_s l$ **do**
  Create candidate set $M = \emptyset$
  for $j = 1, 2, \cdots, l$ **do**
    for $s \in S_j$ **do**
      Substitute s_j by s to get x' and add it to the candidate set $M \leftarrow M \cup \{\mathbf{x}'\}$.
    end for
    $\mathbf{x} \leftarrow \arg\max_{\mathbf{x}' \in M} C_y(V(x'))$
  end for
end while

the best paraphrase combinations within the candidate set. In this way, we are able to conduct multiple replacements in one iteration and thus take into consideration the joint effect of multiple words replacements.

This method is based on an intuition derived from coordinate descent with the Gauss-Southwell rule in the continuous optimization theory; normally, updating the coordinates with the highest absolute gradient values is provably faster than optimizing over random coordinates (Nutini et al., 2015). We only conduct this method in word paraphrasing, since the gradient information of sentence embedding is less trustworthy. Usually sentence paraphrasing changes the number of words. The calculated gradient before paraphrasing step might not even correspond to the right position of the new sentence. Therefore it makes more sense to use the objective value only and goes back to our Algorithm 2.

6 Experiments

In this section, we provide empirical evidence of the advantages of our attack scheme via joint sentence and word paraphrasing on both two WCNN and LSTM models and various classification tasks.

6.1 Tasks and Models.

We focus on attacking the following state-of-the-art models which also echo our theoretical analysis:

Word-level Convolutional Network (WCNN).
We implement a convolutional neural network (Kim, 2014) with a temporal convolutional layer of kernel size 3 and a max-pooling layer, followed by a fully connected layer for the classification output.

Long Short Term Memory classifier (LSTM).
The LSTM Classifier (Hochreiter & Schmidhuber, 1997) is well-suited to classifying text sequences of various lengths. We construct a one-layered LSTM with 512 hidden nodes, following the architecture used in (Kuleshov et al., 2018; Zhang et al., 2015).

We carried out experiments on three different text classification tasks: fake-news detection, spam filtering and sentiment analysis; these tasks are also considered in (Kuleshov et al., 2018). The corresponding datasets include:

Fake/Real News.
The fake news repository (McIntire, 2017) contains 6336 clean articles of both fake and real news in a 1:1 ratio (5336 training and 1000 testing), with both left- and right-wing sites as sources.

Trec07p (emails).
The TREC 2007 Public Spam Corpus (Trec07p) contains 75,419 messages of ham (non-spam) and spam in a 1:2 ratio. We preprocess the data and retain only the main content in each email. We randomly hold out 10% as testing data.

Algorithm 3 *Gradient Guided Greedy Word Paraphrasing*$(C_y, \mathbf{x}, \mathbf{W}, \lambda_w, \tau)$

Input: Document x as a list of words $[w_1, w_2, \cdots, w_n]$, word neighboring sets $\mathbf{W} = \{W_1, W_2 \cdots W_n\}$, model C_y and parameters λ_w, τ.
Let N (that we set as 5) be the number of words to replace at most in each iteration
while $C_y(\mathbf{x}) \leq \tau$ and number of words paraphrased $\leq \lambda_w n$ **do**
  Compute score for each word $\mathbf{p}$: $p_i = \|\nabla_i C_y(\mathbf{v})\|_2$, where $\mathbf{v} = V(\mathbf{x})$ and ∇_i denotes the gradient with respect to the embedding of the i-th word in x.
  Get the indices $I = \{i_1, i_2, \cdots i_N\}$: the N largest indices in $\mathbf{p}$.
  Create candidate set $M = \{\mathbf{x}\}$
  for $j \in I$ **do**
    Let the new candidate set $\bar{M} \leftarrow \emptyset$
    for $\bar{\mathbf{x}} \in M$ **do**
      for $w \in W_j$ **do**
        Substitute the j-th word in $\bar{\mathbf{x}}$ by w to get $\mathbf{x}'$ and add it to the candidate set $\bar{M} \leftarrow \bar{M} \cup \{\mathbf{x}'\}$.
      end for
    end for
    $M \leftarrow M \cup \bar{M}$
  end for
  $\mathbf{x} \leftarrow \arg\max_{\mathbf{x}' \in M} C_y(\mathbf{x}')$
end while

Dataset	WCNN				LSTM			
	Origin	ADV (ours)	ADV (Kuleshov et al., 2018)		Origin	ADV (ours)	ADV (Kuleshov et al., 2018)	
News	93.1%	**35.4%**	71.0%	70.5%*	93.3%	**16.5%**	37.0%	22.8%*
Trec07p	99.1%	**48.6%**	64.5%	63.5%*	99.7%	**31.1%**	39.8%	37.6%*
Yelp	93.6%	**23.1%**	39.0%	41.2%*	96.4%	30.0%	**24.0%**	29.2%*

Table 2. Classifier accuracy on each dataset. Origin and ADV respectively stand for the clean and adversarial testing results. For all datasets, we set word paraphrasing ratio to be $\lambda_w = 20\%$ for our method (ADV(ours)). We include results from (Kuleshov et al., 2018) for comparison. The first column indicates reported value in their paper; while the consequent column marked by asterisk is our implementation using greedy method in (Kuleshov et al., 2018) and the same word neighboring set as our method. Both results use large $\lambda_w = 50\%$ and allow many more word replacements.

Yelp review polarity.
The Yelp reviews dataset was obtained from the Yelp Dataset Challenge in 2015. The Polarity dataset was constructed for a binary classification task that labels 1 and 2 stars as negative and 3 or 4 stars as positive (Zhang et al., 2015). The dataset contains 560,000 training and 38,000 testing documents.

6.2 General Settings

For the training procedure, we use similar settings for the WCNN and LSTM classifier. We extracted the top 100,000 most frequent words to form the vocabulary. The first layer of both WCNN and LSTM is the embedding that transforms individual word into a 300-dimensional vector using the pretrained $word2vec$ embeddings (Mikolov et al., 2013). We randomly hold out 10% training data as validation set to choose the number of epochs and use a constant mini-batch size of 16.
We manually selected the hyperparameters for each dataset. We set the termination threshold $\tau = 0.7$, and set a neighbor size k for possible paraphrases to be 15. We set the semantic similarity $\delta_w = \delta_s = 0.75$[1] for all datasets and syntactic bound $\delta_2 = 2$ for news and yelp datasets, and $\delta = \infty$ for Trec07p; the email dataset contains many corrupted words rendering the language model ineffective. For all datasets, we only allow $\lambda_w = 20\%$ word paraphrasing. We set the sentence paraphrasing ratio $\lambda_s = 20\%$ for yelp and news dataset, and for spam $\lambda_s = 60\%$.

6.3 Accuracy comparisons.

After setting up the experimental environment, we now present the empirical studies in several aspects. In Table 2 we present the original and adversarial test accuracy on the three datasets with the two chosen models, where we allow 20% word replacements. We also include the presented adversarial accuracy from (Kuleshov et al., 2018) for reference. Since the word neighboring sets for the two methods are different and the values are not directly comparable, one might argue that we have broaden the search space of words to make the problem easier.

[1] We use the WMD similarity in python's spacy package. The similarity is in [0,1] basis where 1 means identical and 0 means complete irrelevant.

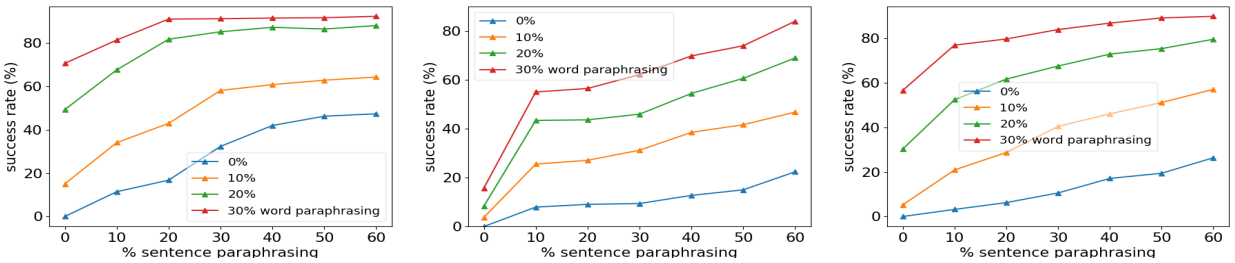

Figure 4. Success rate of attacking the LSTM classifier with different ratios of allowed paraphrasing.

Method		objective-guided greedy (Kuleshov et al., 2018)		gradient method (Gong et al., 2018)		ours (Alg. 3)	
		$\lambda_w = 5\%$	$\lambda_w = 20\%$	$\lambda_w = 5\%$	$\lambda_w = 20\%$	$\lambda_w = 5\%$	$\lambda_w = 20\%$
News	*SR:*	26.2%	28.4%	9.93%	12.8%	**39.7%**	**45.4%**
	time:	0.79	1.46	**0.13**	**0.21**	0.26	0.31
Trec07p	*SR:*	5.1%	24.9%	0.86%	3.4%	**12.9 %**	**45.3 %**
	time:	0.19	0.33	**0.03**	**0.05**	0.07	0.09
Yelp	*SR:*	12.7%	45.0%	4.2%	9.1%	**20.7%**	**55.9%**
	time:	0.15	0.21	**0.02**	**0.03**	**0.02**	0.05

Table 3. Attack success rate (denoted by SR) and time comparisons of each optimization mechanism. The performance is reported on the WCNN classifier. Here objective-guided greedy indicates the greedy method used in (Kuleshov et al., 2018), and the gradient method is the one suggested in (Gong et al., 2018). We can see that even when only applying Algorithm 3, our optimization method is more effective among others.

Therefore we also implemented the greedy mechanism in (Kuleshov et al., 2018) using the same word replacement set as our method has chosen (marked by $*$). Both the reported values from (Kuleshov et al., 2018) and our implementation allow 50% word replacements. From Table 2 we can see that in both settings, we are able to successfully flip more prediction classes with fewer word paraphrases. We hereby conclude that joint sentence and word level paraphrasing is much more effective than mere word replacements. Meanwhile, since sentence-level attacks almost perfectly preserve the original meaning, our method could be less susceptible to humans. In the Appendix we use some concrete examples to show the significantly improved quality of our generated adversarial texts compared to (Kuleshov et al., 2018; Gong et al., 2018).[2] In the examples, we could see that sometimes by simplifying or changing the language, or even by making the slightest changes like adding or erasing space, the sentence paraphrase could make a tremendous difference to the classifier output. Consequently, our method does far fewer word level alterations than other methods and greatly reduces the possibility of syntactic or grammar errors.

[2]Since the former code is not available online, we implemented their algorithms. We use their chosen parameters to generate the adversarial examples to compare the quality of sentences in the appendix. While in Section 5.2 we use the same word neighboring sets for all algorithms to make a fair comparison of the optimization schemes.

To further investigate the joint effect from combining sentence and word level attacks, we also study how each model is susceptible to different degrees of change permitted for both attack levels. Therefore we tested and presented the joint influence in Figure 4 for ratios of sentence paraphrasing λ_s ranging from 0% to 60%, as well as for allowed word paraphrasing percentages λ_w: 0%, 10%, 20% and 30%. In all datasets, sentence paraphrasing is especially effective when we allow only a few word paraphrases. For instance, in the sentiment analysis task, we could only successfully attack $\sim$5% reviews by paraphrasing 10% of words. But after conducting 60% sentence paraphrasing beforehand, the success rate increases to almost 60%.

6.4 Optimization Method Comparisons for Word-level Attacks.

To investigate the effectiveness of our proposed gradient-guided greedy method, we implement and compare the time consumption and success rate with Algorithm 3 and the other two techniques: the gradient method (Gong et al., 2018) and the objective-guided greedy method (Kuleshov et al., 2018). To make a fair comparisons of the optimization schemes, we do not conduct sentence level paraphrasing in any of the methods, and we use the same hyperparameters and settings as suggested in Section 6.1. We observe that our

Dataset	Task I			Task II		
	News	Trec07p	Yelp	News	Trec07p	Yelp
Original	70.0%	80.0%	100.0%	3.06 ± 0.67	3.23 ± 0.31	1.93 ± 0.55
Adversarial	50.0%	80.0%	100.0%	3.13 ± 0.50	3.10 ± 0.40	2.10 ± 1.05

Table 4. Human-subject validation. Task I measures classification accuracy while Task II the subjective likelihood that each example was crafted by a human (scale from 1 to 5). We used five participants, each shown $n = 60$ text examples, half original and half generated using our algorithm. The quality of the generated adversarial text (Task II) is near equal to the original and in fact, slightly higher for the Yelp dataset, but this finding is not necessarily statistically significant.

Dataset	LSTM			WCNN		
	News	Trec07p	Yelp	News	Trec07p	Yelp
Test (before)	93.3%	99.7%	96.4%	93.1%	99.1%	93.6%
Test (after)	94.5%	99.5%	97.3%	93.8%	99.2%	94.9%
ADV (before)	16.5%	31.1%	30.0%	35.4%	48.6%	23.1%
ADV (after)	32.7%	50.1%	46.7%	40.0%	54.2%	44.4%

Table 5. Performance of adversarial training.

scheme is especially more appealing to WCNN, partially because of the dropout layer. The small alteration of one word replacement at a time (Kuleshov et al., 2018) is not significant enough to be considered as true gains or the noise from the dropout. While our method replaces 5 words per iteration to capture more difference, thus it is easier to distinguish the change from the dropout randomness. From Table 3 we can see that our method requires only $1/5$ to $1/3$ time cost relative to the objective-guided greedy method and also achieves better success rate. While gradient method fails to produce good performance when we allow a small set of word replacements.

6.5 Human Evaluation Validation

Despite the significantly higher attack proportion of our text examples, our aim is to deliver a message that is faithful to and coherent with the original text. To evaluate the quality of these generated text examples, we presented a number of original and adversarial text pairs (randomly shuffled before the test) to five human evaluators. The evaluators were asked to complete two tasks: I) Assign the correct label to each text sample; II) Rate each text sample with respect the the likelihood that was crafted by a human (scale from 1 to 5). We adopted a majority vote for task I, and averaged the results from five evaluators for task II. As shown in Table 4, we found that human evaluators tend to achieve similar performance for each kind of text in both tasks, indicating that text examples generated via joint sentence and word paraphrasing are indeed coherent and faithful to the original texts in the relevant respects.

6.6 Adversarial Training.

Finally, we investigated whether our adversarial examples could help improve model robustness. For each dataset, we randomly selected 20% of the training data and generated adversarial examples from them using Algorithm 1. We then merged these adversarial examples with corrected labels into the training set and retrained the model. We present the testing and adversarial accuracy before and after this adversarial training process in Table 5. Under almost all circumstances, adversarial training improved the generalization of the model and made it less susceptible to attack.

7 Conclusion

In this paper, we propose a general framework for discrete attacks. Mathematically, we formulate the adversarial attack as an optimization task on a set of attacks. We then theoretically prove that greedy method guarantees a $1 - 1/e$ approximation factor for two classes of neural network for text classification task. Empirically, we propose a gradient-guided greedy method that inherits the efficiency of gradient method and ability to attack of greedy method. Specifically, we investigate joint sentence and word paraphrasing to generate attacking space that maintain the original semantics and syntax for text adversarial examples.

References

Alzantot, M., Sharma, Y., Elgohary, A., Ho, B.-J., Srivastava, M., and Chang, K.-W. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*, 2018.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumvent-

ing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Belinkov, Y. and Bisk, Y. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*, 2017.

Biggio, B. and Roli, F. Wild patterns: Ten years after the rise of adversarial machine learning. *arXiv preprint arXiv:1712.03141*, 2017.

Bilmes, J. and Bai, W. Deep submodular functions. *arXiv preprint arXiv:1701.08939*, 2017.

Cheng, M., Yi, J., Zhang, H., Chen, P.-Y., and Hsieh, C.-J. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *arXiv preprint arXiv:1803.01128*, 2018.

Dalvi, N., Domingos, P., Sanghai, S., Verma, D., et al. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 99–108. ACM, 2004.

Ebrahimi, J., Rao, A., Lowd, D., and Dou, D. Hotflip: White-box adversarial examples for nlp. *arXiv preprint arXiv:1712.06751*, 2017.

Evtimov, I., Eykholt, K., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati, A., and Song, D. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945*, 1, 2017.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Naval Research Logistics (NRL)*, 3(1-2):95–110, 1956.

Fujishige, S. *Submodular functions and optimization*, volume 58. Elsevier, 2005.

Gao, J., Lanchantin, J., Soffa, M. L., and Qi, Y. Black-box generation of adversarial text sequences to evade deep learning classifiers. *arXiv preprint arXiv:1801.04354*, 2018.

Gong, Z., Wang, W., Li, B., Song, D., and Ku, W.-S. Adversarial texts with gradient methods. *arXiv preprint arXiv:1801.07175*, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jia, R. and Liang, P. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

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

Kuleshov, V., Thakoor, S., Lau, T., and Ermon, S. Adversarial examples for natural language classification problems, 2018. URL https://openreview.net/forum?id=r1QZ3zbAZ.

Kusner, M., Sun, Y., Kolkin, N., and Weinberger, K. From word embeddings to document distances. In *International Conference on Machine Learning*, pp. 957–966, 2015.

Li, J., Monroe, W., and Jurafsky, D. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

Liang, B., Li, H., Su, M., Bian, P., Li, X., and Shi, W. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*, 2017.

Lowd, D. and Meek, C. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 641–647. ACM, 2005.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

McIntire, G. Fake news dataset. https://github.com/GeorgeMcIntire/fake_real_news_dataset, 2017.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Miyato, T., Dai, A. M., and Goodfellow, I. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*, 2016.

Moosavi Dezfooli, S. M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number EPFL-CONF-218057, 2016.

Mrkšić, N., Séaghdha, D. O., Thomson, B., Gašić, M., Rojas-Barahona, L., Su, P.-H., Vandyke, D., Wen, T.-H., and Young, S. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*, 2016.

Narayanan, H. *Submodular functions and electrical networks*, volume 54. Elsevier, 1997.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

Nutini, J., Schmidt, M., Laradji, I., Friedlander, M., and Koepke, H. Coordinate descent converges faster with the gauss-southwell rule than random selection. In *International Conference on Machine Learning*, pp. 1632–1641, 2015.

Papernot, N., McDaniel, P., Swami, A., and Harang, R. Crafting adversarial input sequences for recurrent neural networks. In *Military Communications Conference, MILCOM 2016-2016 IEEE*, pp. 49–54. IEEE, 2016a.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pp. 582–597. IEEE, 2016b.

Ribeiro, M. T., Singh, S., and Guestrin, C. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pp. 856–865, 2018.

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

Schrijver, A. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

Su, D., Zhang, H., Chen, H., Yi, J., Chen, P.-Y., and Gao, Y. Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models. *ECCV*, 2018.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Wieting, J. and Gimpel, K. Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *arXiv preprint arXiv:1711.05732*, 2017.

Wieting, J., Bansal, M., Gimpel, K., Livescu, K., and Roth, D. From paraphrase database to compositional paraphrase model and back. *arXiv preprint arXiv:1506.03487*, 2015.

Wong, C. Dancin seq2seq: Fooling text classifiers with adversarial text example generation. *arXiv preprint arXiv:1712.05419*, 2017.

Yang, P., Chen, J., Hsieh, C.-J., Wang, J.-L., and Jordan, M. I. Greedy attack and gumbel attack: Generating adversarial examples for discrete data. *arXiv preprint arXiv:1805.12316*, 2018.

Yao, Y., Viswanath, B., Cryan, J., Zheng, H., and Zhao, B. Y. Automated crowdturfing attacks and defenses in online review systems. In *ACM Conference on Computer and Communications Security*, pp. 1143–1158, 2017.

Zhang, X., Zhao, J. J., and LeCun, Y. Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626, 2015. URL http://arxiv.org/abs/1509.01626.

Zhao, Z., Dua, D., and Singh, S. Generating natural adversarial examples. *ICLR; arXiv preprint arXiv:1710.11342*, 2018.