TextHoaxer: Budgeted Hard-Label Adversarial Attacks on Text

Muchao Ye,[1] Chenglin Miao,[2] Ting Wang,[1] Fenglong Ma[1*]

[1]The Pennsylvania State University, University Park, Pennsylvania 16802
[2]University of Georgia, Athens, Georgia 30602
muchao@psu.edu, cmiao@uga.edu, ting@psu.edu, fenglong@psu.edu

Abstract

This paper focuses on a newly challenging setting in hard-label adversarial attacks on text data by taking the budget information into account. Although existing approaches can successfully generate adversarial examples in the hard-label setting, they follow an ideal assumption that the victim model does not restrict the number of queries. However, in real-world applications the query budget is usually tight or limited. Moreover, existing hard-label adversarial attack techniques use the genetic algorithm to optimize discrete text data by maintaining a number of adversarial candidates during optimization, which can lead to the problem of generating low-quality adversarial examples in the tight-budget setting. To solve this problem, in this paper, we propose a new method named TextHoaxer by formulating the budgeted hard-label adversarial attack task on text data as a gradient-based optimization problem of perturbation matrix in the continuous word embedding space. Compared with the genetic algorithm-based optimization, our solution only uses a single initialized adversarial example as the adversarial candidate for optimization, which significantly reduces the number of queries. The optimization is guided by a new objective function consisting of three terms, i.e., semantic similarity term, pair-wise perturbation constraint, and sparsity constraint. Semantic similarity term and pair-wise perturbation constraint can ensure the high semantic similarity of adversarial examples from both comprehensive text-level and individual word-level, while the sparsity constraint explicitly restricts the number of perturbed words, which is also helpful for enhancing the quality of generated text. We conduct extensive experiments on eight text datasets against three representative natural language models, and experimental results show that TextHoaxer can generate high-quality adversarial examples with higher semantic similarity and lower perturbation rate under the tight-budget setting.

Introduction

Deep neural networks (DNNs) have gone through rapid development in recent years, and they have been successfully utilized in various natural language processing (NLP) tasks such as text classification (Zhang, Zhao, and LeCun 2015) and natural language inference (Bowman et al. 2015). Although researchers are thrilled to witness the employment of DNN technologies in making machines better understand human language, they still doubt the robustness and security of DNNs despite their superior prediction performance. Thus, to improve the reliability of implementing DNNs in real-world NLP applications, it is necessary to improve the techniques for understanding the robustness of text DNNs.

To achieve this goal, the primary task is to reveal the sensitivity of DNNs by enhancing the adversarial attack technique, which is to generate adversarial examples on text to fool DNNs and change their correct predictions to incorrect ones by adding small perturbation to the input text data. Existing researches have gradually made the setting of generating adversaries more and more realistic from conducting white-box adversarial attacks (Ebrahimi et al. 2018) to **black-box** ones (Li et al. 2019; Ren et al. 2019; Maheshwary, Maheshwary, and Pudi 2021). In the early stage, black-box methods are usually proposed in the *soft-label* setting, which requires the victim model to provide the *probability scores of all categories* (Li et al. 2019; Ren et al. 2019). However, it is still unrealistic because in the real-world scenarios, DNNs are deployed through application programming interfaces (APIs), and users have no access to both parameter gradients and probability distributions of all categories. Thus, researchers further develop black-box text adversarial attack methods in the **hard-label (decision-based)** setting, which only utilizes the *predicted labels* with the maximum probability output by the victim models (Maheshwary, Maheshwary, and Pudi 2021). Although we shall be delightful to see such a progress in text adversarial attacks, there still exists a gap between the existing methodologies and the real-world adversarial attack setting.

To elaborate, in real-world applications, DNN systems usually restrict the number of queries from users to defend against malicious attacks. Therefore, it is impossible for attackers to constantly query the text DNN systems. However, such an important constraint, i.e., *hard-label adversarial attacks with a **tight budget***, is ignored by existing work (Maheshwary, Maheshwary, and Pudi 2021). In fact, this drawback stems from the optimization approach used by existing work. Since text data are discrete and gradient-based optimization approaches cannot directly work on such a discrete space, existing work uses the genetic algorithm (GA) for optimization. To move the adversarial examples close to the decision boundary, the GA-based optimization approaches

[*]Corresponding author.

must generate a large population of adversarial candidates first because a single one will not give GA enough search space to generate good adversarial examples. This step unavoidably cost many queries from text DNN system, which can lead to the failure of GA-based optimization when the query budget is tight. Thus, how to design an effective optimization approach for the tight-budget hard-label adversarial attacks is a new challenge that needs our attention.

To address this challenge, in this paper we propose a new method named TextHoaxer for conducting adversarial attacks in the tight-budget hard-label setting, which formulates the hard-label adversarial attack as a **gradient-based optimization** problem in the *word embedding space*. The intuition behind such a formulation for the tight-budget hard-label setting is that searching optimal adversarial example in the high-dimensional continuous word embedding space allows us to represent word perturbations in a perturbation matrix with an identical shape for different adversarial examples. Consequently, unlike the GA-based optimization that needs a large population of adversarial candidates for optimization, we only use one adversarial candidate and employ the gradients of the objective function w.r.t. the perturbation matrix to generate high-quality adversarial examples.

The objective function consists of three terms, i.e., semantic similarity term, pair-wise perturbation constraint, and sparsity constraint. Since the goal of text adversarial attacks is to generate adversarial examples with higher semantic similarity, we use an auxiliary function to measure the comprehensive text-level semantic similarity between the adversarial example x' and the original input x and then to guide the optimization. Besides, we use two constraints to further enhance the quality of generated adversarial examples. Intuitively, in the word embedding space, the smaller vector difference (i.e., the perturbation matrix $\mathbf{P}$) between x' and x means the higher similarity of these two sentences in the semantic space. Thus, we use the L_2 distance between each pair of words in the embedding space as the pair-wise perturbation constraint. Similarly, a smaller number of changed words also indicates higher semantic similarity. Towards this end, we design a sparsity constraint to explicitly restrict the number of perturbed words. Using this objective function w.r.t. $\mathbf{P}$ formulated in the word embedding space, we then optimize the generated adversarial example given a single initialized candidate via the alternating optimization method. To sum up, our contributions are as follows:

- To the best of our knowledge, we are the first to utilize the blessing of dimensionality (Gorban and Tyukin 2018) of word embedding space and formulate the hard-label text adversarial attack problem as a gradient-based optimization problem in a high-dimensional space. The implementation code is available at https://github.com/machinelearning4health/TextHoaxer.
- By utilizing the semantic representation encoded in the embedding space, we propose a new method named TextHoaxer, which is able to generate adversarial examples with higher semantic similarity under the tight-budget setting without consuming unnecessary queries, e.g., on repeated operations for constructing an adversarial example group like the GA-based optimization.
- We propose an optimization formulation w.r.t. the perturbation matrix by taking both semantic deviations and sparsity into consideration. The proposed framework can continuously search better substitutes for original words while penalizing the substitutions in insignificant positions. As a result, the semantic similarity of generated text adversarial examples can further improve during the optimization.
- The experimental results show that the proposed method is able to generate adversarial examples with higher semantic similarity and lower perturbation rate in eight commonly used text datasets compared to existing methods in the tight-budget setting. Furthermore, additional quantitative and qualitative experiments also verify the effectiveness of the proposed TextHoaxer.

Methodology

Problem Formulation

Suppose we have a text sample x with n words $x = [w_1, w_2, \cdots, w_n]$ whose ground truth label is y. We call x' an adversarial example when x' can make the victim model f that previously could correctly classify x into the class of y ($f(x) = y$) change its prediction, i.e.,

$$f(x') \neq f(x),$$

which is constructed by replacing the original words w_i's with a synonym s in the synonym set $\mathbf{S}_{w_i}$. The goal of the text adversarial attack task is to generate an optimal adversarial example x^* among all x'. Note that in this paper, we focus on the more realistic and practical hard-label setting, and the black-box victim model f only outputs the discrete predicted label $\hat{y} = f(x')$.

Since text data consists of discrete words whose change can be perceived by humans, we always want the optimized adversarial example x^* to be semantically closest to the original text sample x. Thus, the objective function of this task can be defined as follows:

$$x^* = \min_{x'} -\mathcal{G}(x, x'),\ s.t.\ f(x') \neq f(x), \tag{1}$$

where $\mathcal{G}(x, x')$ denotes the semantic similarity between x and x', and the minus sign is for minimization formulation.

The Proposed TextHoaxer

Overview Existing worksuse heuristic algorithms such as GA (Maheshwary, Maheshwary, and Pudi 2021) to search a solution to the objective function Eq. (1). However, as discussed in the previous section, the GA-based optimization needs a large number of queries to generate adversarial examples with high semantic similarity, which is not suitable for the tight-budget hard-label setting. To address this problem, we propose TextHoaxer as shown in Figure 1, which maintains only one adversarial candidate x'_0 that walks around in a high-dimensional word embedding space to find the optimal adversarial candidate x^* based on the perturbation matrix $\mathbf{P}$.

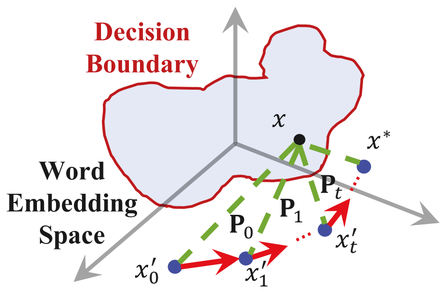


Figure 1: Overview of TextHoaxer for hard-label text adversarial attack. With an initialized adversarial example x'_0, the optimization in TextHoaxer is guided by the perturbation matrix calculated in the word embedding space.

Initialization For each input x, we first get an initialized adversarial candidate x'_0 by random initialization (Maheshwary, Maheshwary, and Pudi 2021), if there exists an x'_0 that satisfies $f(x'_0) \neq f(x)$. For each word w_i in x, we can utilize a pretrained word embedding space $\mathcal{H}$ such as Counter-Fitted Word Vectors (Mrksic et al. 2016) to obtain its word embedding $\mathbf{e}_i = [e_{(i,1)}, e_{(i,2)}, \cdots, e_{(i,m)}] \in \mathbb{R}^{1\times m}$, where m is the dimension of word embedding. As a result, we can get an $n \times m$ embedding matrix $\mathbf{E}$ for x and similarly $\mathbf{E}_0$ for x'_0 in $\mathcal{H}$. Based on $\mathbf{E}$ and $\mathbf{E}_0$, we can get an initial perturbation matrix $\mathbf{P}_0 = \mathbf{E}_0 - \mathbf{E}$. By minimizing the perturbation matrix under the objective function, TextHoaxer is able to efficiently optimize x'_0 and improve its semantic similarity.

Objective Function An ideal adversarial example x^* should have high semantic similarity with the original text x. Otherwise, it is easy for humans to realize it is adversarial. Thus, the goal of this task is to optimize the initialized x'_0 to make it have higher semantic similarity and close to the decision boundary of the victim model f. To this end, we incorporate an auxiliary semantic similarity calculation function $\mathrm{sim}(\cdot,\cdot)$ to guide the optimization, where $\mathrm{sim}(\cdot,\cdot)$ is a model that can output the semantic similarity between two text samples, e.g., Universal Sequence Encoder (Cer et al. 2018). Recall that the perturbation matrix $\mathbf{P}$ is a combination of difference vectors denoting how far the substitution words deviate the original words in the embedding space. Given the perturbation matrix $\mathbf{P}$ at any step during optimization, we can utilize the embedding matrix $\mathbf{E}' = \mathbf{E} + \mathbf{P}$ to get a new adversarial text example x' first. More specifically, for each embedding $\mathbf{e}'_i = \mathbf{e}_i + \mathbf{p}_i$ of the i-th word in $\mathbf{E}'$, we map it to a word s in the synonym set $\mathbf{S}_{w_i}$ (including w_i itself) by calculating the L_2 norm between $\mathbf{e}'_i$ and the word embedding of every $s \in \mathbf{S}_{w_i}$. Then the synonym with the smallest distance will be treated as the i-th word w'_i in x'.

After getting the new adversarial example x' in the process described above, we explicitly define our objective function in Eq. (1) by measuring the semantic similarity with $\mathrm{sim}(\cdot,\cdot)$ based on the perturbation matrix $\mathbf{P}$ as follows:

$$\ell_{\mathrm{sim}}(\mathbf{P}) = -\mathrm{sim}(x, x'),\ s.t.\ f(x') \neq f(x), \tag{2}$$

where the minus sign is for minimization formulation. Note that our objective function is defined in the continuous word embedding space. To increase the semantic similarity between x and x' and try to achieve the optimal, Eq. (2) may update each element in $\mathbf{P}$, which leads to the dramatic change of the perturbation matrix $\mathbf{P}$. In such a case, when we map $\mathbf{E}'$ to x', most words in x' will be different from x, even for those unimportant words that are the same in both x and x'_0. Thus, only optimizing Eq. (2) cannot completely guarantee low perturbation rate and may hurt the quality of the generated adversarial example. To mitigate this problem, we propose two effective constraints as follows.

Pair-wise Perturbation Constraint. It is known that the word embedding space contains semantic information of different words. If we want to generate an adversarial example x^* with high semantic similarity, it is beneficial that each pair of the original word w_i and its substitute is close to each other in the space $\mathcal{H}$. Since the matrix $\mathbf{P}$ can be written as $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n]^{\mathrm{T}}$ where $\mathbf{p}_i$ is the vector difference of w_i and its substitute in the embedding space, we use the following constraint to restrict the deviation of semantics for each pair to achieve the goal of optimizing the semantic similarity of adversarial example, which can be measured by the squared L_2 norm of $\mathbf{p}_i$, i.e.,

$$\ell_{\mathrm{pwp}}(\mathbf{P}) = \sum_{i=1}^{n} ||\mathbf{p}_i||_2^2. \tag{3}$$

Sparsity Constraint. Another aspect that we need to consider for optimizing $\mathbf{P}$ is the sparsity of word substitution because retaining original words is helpful for maintaining original semantics. Towards this end, we divide each $\mathbf{p}_i$ into two components, i.e., magnitude γ_i and direction $\boldsymbol{\rho}_i$, and we then have $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n]^{\mathrm{T}} = [\gamma_1\boldsymbol{\rho}_1, \gamma_2\boldsymbol{\rho}_2, \cdots, \gamma_n\boldsymbol{\rho}_n]^{\mathrm{T}}$.

Now we can employ the magnitude component to constrain the sparsity, and ideally, all magnitudes are close to 0. The following sparsity constraints is used for guiding the optimization:

$$\ell_{\mathrm{spa}}(\mathbf{P}) = \sum_{i=1}^{n} |\gamma_i|. \tag{4}$$

Final Loss. Based on Eq. (2), Eq. (3), and Eq. (4), we have the following overall objective function to help us generate adversarial examples with high semantic similarity and small perturbation rate, i.e.,

$$\begin{aligned} \min_{\mathbf{P}} \mathcal{L} = \min_{\mathbf{P}}\ & \lambda_1\ell_{\mathrm{sim}} + \lambda_2\ell_{\mathrm{pwp}} + \lambda_3\ell_{\mathrm{spa}}, \\ & s.t.\ f(x') \neq f(x), \end{aligned} \tag{5}$$

where λ_1, λ_2 and λ_3 are positive scalar hyperparameters, and the constraint $f(x') \neq f(x)$ can be easily tested by putting each generated sample into the victim model .

Optimization Procedure In this subsection, we discuss how to optimize $\mathbf{P}$ in Eq. (5). As mentioned before, for each perturbation in the embedding space, $\mathbf{P}$ can be written as $\mathbf{P} = [\gamma_1\boldsymbol{\rho}_1, \gamma_2\boldsymbol{\rho}_2, \cdots, \gamma_n\boldsymbol{\rho}_n]^{\mathrm{T}}$. Thus, given the objective function in Eq (5), we need to optimize both $[\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \cdots, \boldsymbol{\rho}_n]$ and $\boldsymbol{\gamma} = [\gamma_1, \cdots, \gamma_n]$, respectively. Since they do not have closed form solutions, we need to optimize them in an alternating optimization fashion. In other words, we need to estimate their solutions by firstly optimizing $\boldsymbol{\rho}_i$'s with fixed γ_i's, and in turn, optimizing γ_i's with fixed $\boldsymbol{\rho}_i$'s. Suppose that we optimize them by an iterative process in T steps. The optimization process in each step t ($0 \leq t \leq T-1$) is described as below, where we denote x'_t as the adversarial example we have during step t.

Optimizing $\boldsymbol{\rho}_i$. In step t, we fix all the γ_i's and start to optimize $[\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \cdots, \boldsymbol{\rho}_n]$, which is solved in a gradient descent fashion similar to (Cheng et al. 2019). To specify, we first randomly sample a perturbation direction $\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_n]^{\mathrm{T}} \in \mathbb{R}^{n \times m}$ from a zero-mean Gaussian distribution, and we can get a neighboring perturbation

$$\mathbf{V} = \mathbf{P} + \beta \mathbf{U} = [\mathbf{v}_1, \cdots, \mathbf{v}_n]^{\mathrm{T}}, \tag{6}$$

where β is a hyperparameter for controlling the perturbation strength. After that, we generate a new adversarial example v'_t that is a neighbor of previous adversarial example x'_t according to the perturbation $\mathbf{V}$ in a similar manner how we construct x': we replace the original words of x in descending order of L_2 norm of the perturbation $\mathbf{V}$, and the i-th word in v'_t is the synonym $s \in \mathbf{S}_{w_i}$ whose embedding is closest to the embedding $\mathbf{e}_i + \mathbf{v}_i$ measured by L_2 norm. The replacement process stops as soon as the corresponding neighboring sample v'_t becomes adversarial, i.e., $f(v'_t) \neq f(x)$.

Now given v'_t and x'_t, using their corresponding semantic similarity scores with original sample x, i.e., $\mathrm{sim}(x, v'_t)$ and $\mathrm{sim}(x, x'_t)$, we get the gradient direction w.r.t. Eq (2) for the i-th perturbation $\mathbf{p}_i$ in step t,

$$\mathbf{g}_i^{(t)} = -\frac{\mathrm{sim}(x, v'_t) - \mathrm{sim}(x, x'_t)}{\beta} \cdot \mathbf{u}_i, \tag{7}$$

and then we update the perturbation $\mathbf{p}_i$ for the i-th word in step t by Eq. (5) as

$$\mathbf{p}_i \leftarrow \mathbf{p}_i - \eta_1(\mathbf{g}_i^{(t)} + 2\lambda_2 \mathbf{p}_i), \tag{8}$$

where η_1 is a hyperparameter, and $\boldsymbol{\rho}_i$ at time step t is updated as $\boldsymbol{\rho}_i = \mathbf{p}_i / \gamma_i$ with fixed γ_i.

Optimizing γ_i. After we optimize $[\boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \cdots, \boldsymbol{\rho}_n]$, we then begin to optimize $\boldsymbol{\gamma} = [\gamma_1, \cdots, \gamma_n]$. According to Eq. (5), we can optimize each γ_i by its gradient,

$$\gamma_i \leftarrow \gamma_i - \eta_2 \nabla_{\gamma_i} \mathcal{L}, \tag{9}$$

where η_2 is a hyperparameter, and $\nabla_{\gamma_i} \mathcal{L}$ is calculated by

$$\nabla_{\gamma_i} \mathcal{L} = \lambda_1 \partial \ell_{\mathrm{sim}} / \partial \gamma_i + \theta(\lambda_3, \gamma_i), \tag{10}$$

where $\theta(\lambda_3, \gamma_i)$ is the soft threshold function as follows:

$$\theta(\lambda_3, \gamma_i) = \begin{cases} \gamma_i + \lambda_3 & if\ \gamma_i \leq -\lambda_3, \\ 0 & if\ |\gamma_i| \leq \lambda_3, \\ \gamma_i - \lambda_3 & if\ \lambda_3 \leq \gamma_i, \end{cases} \tag{11}$$

and $\partial \ell_{\mathrm{sim}} / \partial \gamma_i$ is estimated by removing the i-th word of the adversarial example.

During the T iterations of alternatively optimizing $\boldsymbol{\rho}_i$'s and γ_i's, we can obtain new adversarial example x' after getting new perturbation matrix, and we maintain the adversarial example that has the highest semantic similarity. The best one attained after T iterations is used as the optimized solution x^* for solving Eq. (5).

Experiments

In this section, we evaluate the performance of our proposed attack method. Specifically, we first introduce the experimental settings and then analyze the experimental results.

Experimental Settings

Datasets We adopt the following text datasets that are collected against different text classification and natural language inference: (1) MR (Pang and Lee 2005), a movie review dataset used in the task of sentiment classification with two categories (2) AG (Zhang, Zhao, and LeCun 2015), a news classification dataset with 4 classes including "world", "sports", "business", and "science"; (3) Yahoo (Zhang, Zhao, and LeCun 2015), a topic classification dataset collected from the questions and answers of Yahoo with 10 clasess; (4) Yelp (Zhang, Zhao, and LeCun 2015), a widely used text dataset collected for the binary sentiment classification task; (5) IMDB (Maas et al. 2011), another binary sentiment classification dataset collected from movie reviews; (6) SNLI (Bowman et al. 2015) and (7) MNLI (Williams, Nangia, and Bowman 2018), two datasets collected for the natural language inference task; and (8) mMNLI, a variant of the MNLI dataset with mismatched premise and hypotheses pairs. In our experiments, we follow the settings of (Maheshwary, Maheshwary, and Pudi 2021) and (Jin et al. 2020), taking the same 1,000 test samples of each dataset to conduct adversarial attack.

Victim Models For fair comparison with existing hard-label adversarial attack methods, we adopt the following victim models: BERT (Devlin et al. 2019), WordCNN (Kim 2014), and WordLSTM (Hochreiter and Schmidhuber 1997), which are widely used in natural language processing tasks. For WordCNN and WordLSTM, the input words are embedded into 200 dimensional Glove (Pennington, Socher, and Manning 2014) embeddings.

Baselines The adopted baseline methods are all proposed for the black-box text adversarial attack. For the methods that are originally proposed in the soft-label setting, we take the hard-label scores as their inputs. The baselines include (1) TextFooler (Jin et al. 2020), a soft-label text adversarial attack method which replaces the original words in the order of the position importance scores, and the replacement of original words are determined by the semantic similarity change it brings; (2) PWWS (Ren et al. 2019), an improved saliency score-based method that takes the soft-label score changes into consideration; (3) TextBugger (Li et al. 2019), another soft-label black-box method that uses the saliency score as the guide of adversarial examples construction; (4) DeepWordBug (Gao et al. 2018), one of the earliest works on soft-label black-box adversarial attack, effectively employing the saliency score of each word to determine the order of words being attacked; and (5) HLBB (Maheshwary, Maheshwary, and Pudi 2021), a recent method proposed in the setting of hard-label text adversarial attack, which employs the genetic algorithm to optimize the semantic similarity between the generated text sample and original samples.

The first step of baselines and TextHoaxer is to find an initialized adversarial example. For the hard-label adversar-

Dataset	Method	BERT			WordCNN			WordLSTM		
		Acc	Sim	Pert	Acc	Sim	Pert	Acc	Sim	Pert
MR	TextFooler	1.0%	56.7%	16.294%	0.7%	58.2%	15.961%	0.7%	58.6%	16.026%
	PWWS		59.3%	16.082%		61.6%	15.695%		60.2%	16.438%
	TextBugger		61.7%	15.056%		64.5%	14.615%		62.5%	15.184 %
	DeepWordBug		61.5%	15.120%		62.8%	14.882%		63.6%	15.183%
	HLBB		62.5%	14.532%		64.4%	14.028%		63.5%	14.462%
	TextHoaxer		**67.3%**	**11.905%**		**68.6%**	**12.056%**		**67.3%**	**12.324%**
AG	TextFooler	2.8%	57.6%	18.954%	1.4%	68.8%	14.845%	5.7%	60.0%	18.902%
	PWWS		58.4%	18.538%		70.0%	15.353%		61.1%	18.689%
	TextBugger		60.6%	17.871%		72.0%	14.167%		62.9%	17.592%
	DeepWordBug		60.2%	17.845%		71.7%	14.315%		63.4%	17.579%
	HLBB		60.5%	17.769%		71.9%	13.855 %		61.8%	17.890%
	TextHoaxer		**63.2%**	**15.766%**		**73.9%**	**12.716%**		**63.8%**	**16.520%**
Yahoo	TextFooler	0.5%	64.0%	8.594%	0.8%	69.4%	9.197%	1.9%	61.5%	10.140%
	PWWS		65.7%	8.555%		70.2%	9.158%		62.2%	10.298%
	TextBugger		66.6%	8.266%		71.6%	8.775%		63.8%	9.504%
	DeepWordBug		65.8%	8.228%		71.6%	8.787%		63.3%	9.895%
	HLBB		68.7%	7.453%		71.9%	8.564%		63.8 %	9.531%
	TextHoaxer		**70.2%**	**6.841%**		**74.8%**	**7.740%**		**67.0%**	**8.502%**
Yelp	TextFooler	5.2%	69.5%	10.874%	0.6%	76.8%	10.300%	3.2%	76.9%	9.120%
	PWWS		70.5%	10.884%		77.8%	10.311%		77.0%	9.201%
	TextBugger		71.1%	10.814%		78.2%	9.898%		78.2%	8.920%
	DeepWordBug		71.7%	10.518%		78.3%	9.823%		78.6%	8.832%
	HLBB		71.9%	10.411%		79.7%	9.102%		78.8%	8.654%
	TextHoaxer		**73.8%**	**9.585%**		**81.3%**	**8.545%**		**80.4%**	**8.108%**
IMDB	TextFooler	0.1%	82.2%	5.804%	0.0%	89.4%	4.433%	0.3%	86.7%	4.626%
	PWWS		82.1%	5.822%		89.3%	4.519%		87.5%	4.665%
	TextBugger		82.9%	5.662%		89.7%	4.349%		87.3%	4.606%
	DeepWordBug		82.7%	5.612%		89.6%	4.412%		87.3%	4.516%
	HLBB		83.2%	5.571%		89.2%	4.529 %		87.6%	4.464%
	TextHoaxer		**84.7%**	**5.202%**		**90.1%**	**4.266%**		**88.8%**	**4.197%**

Table 1: Comparison of semantic similarity (Sim) and perturbation rate (Pert) with budget limit of 1,000 when attacking against text classification models. Acc stands for model prediction accuracy after adversarial attack, which is determined by the random initialization step and the same for different adversarial attack methods.

ial attack methods in the image domain, they (Chen, Jordan, and Wainwright 2020) usually attain initialized adversarial examples by gradually increasing of uniform noise weight to the original image until it is misclassified due to the unavailability of soft-label scores. Similarly, a random word initialization method is adopted by HLBB to find initialized adversarial examples and can achieve relatively high attack success rate. Thus, in experiments for fair comparison all methods use the same random word initialization to initialize the adversarial examples, which leads to *the same attack success rate after adversarial attacks for all methods*.

Evaluation Metrics The evaluation metrics we use to quantify the quality of the generated adversarial examples include **semantic similarity** and **perturbation rate**. The *semantic similarity* $\mathrm{sim}(\cdot, \cdot)$ is calculated by putting the original text sample and generated adversarial example into the Universal Sequence Encoder (Cer et al. 2018), which is in the range of [0, 1] and the higher the better. As for the *perturbation rate*, it is defined as the ratio of the changed words in the generated sample compared to the original text sample, and the lower the better.

Setting of TextHoaxer TextHoaxer is implemented with an NVIDIA V100 GPU. The used word embedding space $\mathcal{H}$ is from Counter-Fitted Word Vectors (Mrksic et al. 2016). The synonym $\mathbf{S}_{w_i}$ for each word w_i is the same with the baselines. We set $\lambda_1 = 1$, $\lambda_2 = \lambda_3 = 0.1$, $\beta = 0.5$, $\eta_1 = 0.3$ and $\eta_2 = 0.05$, and all γ_i's are initialized as 0.3. We keep λ_2 and λ_3 relatively smaller than λ_1 to make them as regularizers, β and η_1 relatively large to have large step size, and η_2 and γ_i's relatively small to achieve high sparsity.

Experimental Results

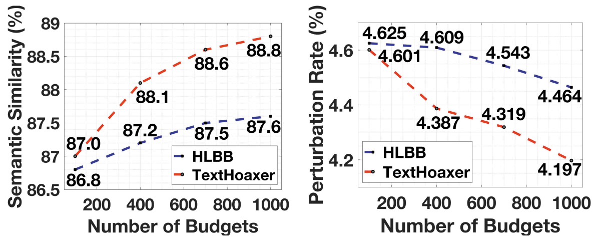
Performance Comparison In this paper, the budget is defined as the number of allowed queries from the attacker. Based on the initialized adversarial examples generated by the random initialization method, our goal is to further find out the adversarial examples with higher semantic similarity and lower perturbation rate under the tight budget. Due to the space limit, we only show the comparison results when budget is 1,000, which are relatively tight. As shown in Table 1

Model	SNLI			MNLI			mMNLI		
	Acc	Sim	Pert	Acc	Sim	Pert	Acc	Sim	Pert
TextFooler	1.3%	29.7%	20.043%	2.9%	42.6%	16.455%	1.7%	43.7%	16.035%
PWWS		31.9%	20.016%		44.5%	16.426%		45.7%	16.056%
TextBugger		33.2%	19.322%		47.1%	15.208%		48.0%	15.055%
DeepWordBug		33.4%	19.200%		46.8%	15.270%		47.4%	15.248%
HLBB		35.9%	18.510%		49.6%	14.498%		50.7%	14.349%
TextHoaxer		**38.7%**	**16.615%**		**52.9%**	**12.730%**		**54.4%**	**12.453%**

Table 2: Comparison of semantic similarity and perturbation rate when attacking against natural language inference model (BERT) with budget limit of 1,000.

Method	Average Rank of Adversarial Example w.r.t. Similarity
TextHoaxer	**3.39 (SD=1.44)**
HLBB	**3.39** (SD=1.51)
TextFooler	3.43 (SD=1.79)
DeepWordBug	3.52 (SD=1.72)
PWWS	3.58 (SD=1.55)
TextBugger	3.67 (SD=1.55)

Table 3: Human evaluation results. "SD" denotes the standard deviation, and the lower SD is the better one.



(a) IMDB (Sim) (b) IMDB (Pert)

Figure 2: Performance comparison w.r.t. different budget limits between HLBB and TextHoaxer against WordLSTM.

and Table 2, when the query limit is 1,000, our method can generate optimized adversarial examples with higher semantic similarity and lower perturbation rate in all cases for text classification and natural language inference models. Particularly, for the MR dataset, which contains relatively short text, the average semantic similarity of the adversarial examples generated by our method is 4.8%, 4.1%, and 3.7% higher than that of the second best method when the victim models are BERT, WordCNN, and WordLSTM, respectively, and the average perturbation rate is 2.627%, 1.972%, and 2.138% lower than that of the second best method when the victim models are BERT, WordCNN, and WordLSTM, respectively. Furthermore, since HLBB is a strong hard-label baseline, we compare the semantic similarity and perturbation rate w.r.t. different budget limits between HLBB and TextHoaxer with budget sampling at 100, 400, 700 and 1,000. Without the loss of generality, we show the comparison results of attacking WordLSTM on IMDB dataset. As shown in Figure 2, the average semantic similarity of TextHoaxer keeps on increasing as the budget increases, and it is always higher than that of HLBB. We also observe that the average perturbation rate of TextHoaxer has the trend of decreasing as the budget increases, and it is always lower than that of HLBB. These results further verify that TextHoaxer has the capacity of generating adversarial examples with high semantic similarity and low perturbation rate in the tight-budget setting.

The reasons why TextHoaxer outperforms existing black-box methods in the tight-budget hard-label setting are as follows. First, for the soft-label methods, they need to replace all the synonyms for the attacked word one by one to find out the best substitute, which will cost lots of query budgets, and in the hard-label setting it is hard for them to distinguish the semantic difference between synonyms when they do not know the prediction probability distribution for different categories. As for hard-label methods such as HLBB, they need to maintain an adversarial example group and select the best one among them as the output. Thus, they have to spend some query budgets repeatedly on the same operation for different individuals, which is disadvantageous in the tight-budget setting. As for TextHoaxer, its optimization direction is guided by the gradients w.r.t perturbation matrix in the word embedding space, which avoids the trouble of trying all synonyms and maintaining an candidate group.

Human Evaluation and Case Study Additionally, we conduct a human evaluation experiment and ask participants to evaluate the semantic similarity of adversarial examples generated by the different methods. In this experiment, from MR, AG, Yelp and IMDB we randomly select 5 original samples in each of them to set up the evaluation samples. In each set question we randomize the order of adversarial examples output by different methods and ask the human evaluators to rank the semantic similarity of 6 corresponding adversarial examples generated by baselines and TextHoaxer after reading the original text sample, and they are allowed to have tied ranks for different samples. Given their results, we then calculate the mean rank and standard deviation (SD) of all samples by each method. As Table 3 shows, the adversarial examples generated by the proposed method and HLBB are ranked the top w.r.t. semantic similarity evaluated by humans. Thus, the human evaluation results further validate that the adversarial examples crafted by TextHoaxer retain important semantic information of original text samples. In Table 4, we also show some adversarial examples

Adversarial Example	Change of Prediction
MR: An intense (ponderous) and effective film about loneliness and the chilly anonymity of the environments where so many of us spend so much of our time	Positive $\rightarrow$ Negative
AG: Arctic team (grouping) finds ship remains. A team retracing the route of a group of victorian arctic explorers (colonist) have found parts of their 172 year old ship.	Science $\rightarrow$ World
Yahoo: What is 13 as a fraction (percentage)? 13	Science & Mathematics $\rightarrow$ Business & Finance
Yelp: Very (rather) nice staff, I'm going to miss it as I move a little further away however I use the online ordering and it's great (noteworthy) to wish other stores did this	Positive $\rightarrow$ Negative

Table 4: Adversarial examples generated by TextHoaxer in different datasets against the WordLSTM model. The substitute words for each sample are ndicated by parentheses.

generated by TextHoaxer in different datasets. We can observe that TextHoaxer has the ability of substituting original words with semantically similar synonyms and fooling the victim models successfully. For instance, when people read the adversarial example generated in the Yelp dataset which replaces "very" to "rather" and "great" to "noteworthy", they usually will not think they have much semantic difference.

Related Work

Text adversarial attack begins to receive attention after DNNs such as BERT (Devlin et al. 2019) have achieved state-of-the-art performance in NLP tasks, and researchers also find that text DNNs can also be easily fooled by text adversarial examples. It is inspired by the observation in images that DNNs are so sensitive that their predictions can be easily changed to incorrect labels when the adversarial image with the human-imperceptible perturbation is put into the DNNs (Carlini and Wagner 2017; Kurakin, Goodfellow, and Bengio 2017; Chen et al. 2017; Brendel, Rauber, and Bethge 2018), and adversarial examples are usually used to improve the robustness of DNNs (Goodfellow, Shlens, and Szegedy 2015; Zhu et al. 2020; Zhou et al. 2021c). For instance, (Jia and Liang 2017) proposes the ADDSENT method which successfully fools the reading comprehension systems in a four-step procedure. Text adversarial attacks can be categorized into **white-box** (Ebrahimi et al. 2018; Zhou et al. 2021a) and **black-box** (Li et al. 2019; Jin et al. 2020; Ren et al. 2019; Maheshwary, Maheshwary, and Pudi 2021; Gao et al. 2018; Zhou et al. 2021b) settings, where the latter cannot use parameter gradients.

This paper focuses on a more realistic **hard-label black-box adversarial attack** setting. To illustrate, early-stage black-box adversarial attack methods are proposed in the *soft-label* setting, which rely on the probability distribution of all categories to craft adversarial examples. For instance, to generate adversarial examples, (Li et al. 2019) uses the predicted probability changes after the word is removed and after the word is substituted to determine the order of words being attacked and the substitute of attacked word, respectively. The latter *hard-label* setting is more challenging because the attackers could only know the predicted label in adversarial example generation, and it has not received great attention yet. To the best of our knowledge, (Maheshwary, Maheshwary, and Pudi 2021) is one of the few successful initial explorations in this setting, which generates an initial adversarial example by random initialization first and later moving the adversarial example close to the decision boundary by the genetic algorithm. One problem of existing work is that they are not proposed for a realistic setting where query budget are limited. In this tight-budget setting, HLBB (Maheshwary, Maheshwary, and Pudi 2021) can have an inefficiency problem because the used genetic algorithm spends numerous queries on an adversarial example group to move adversarial examples near to the decision boundary. As for TextHoaxer, it alleviates the inefficiency problem in the tight-budget hard-label setting by formulating this problem as a gradient-based optimization problem for perturbation matrix in the word embedding space, which can reduce the consumption of queries by optimizing merely a single adversarial candidate.

There have been research works (Chen, Jordan, and Wainwright 2020; Cheng et al. 2019) in using gradient-based methods to conduct hard-label adversarial attacks on images. Compared to the methods on images, our work further considers the mapping from gradients to discrete words and includes two constraints for finding replacements.

Conclusion

Hard-label text adversarial attack with a tight budget is a more realistic and challenging problem for revealing the robustness of text DNNs. Previous soft-label-based methods cannot be directly applied in this scenario, and existing techniques proposed for the hard-label setting suffer from the inefficiency problem in a tight-budget setting for repeated query consumption. To bridge such a methodological gap, we propose a perturbation matrix optimization framework named TextHoaxer based on a given word embedding space, and harness the gradients of objective function w.r.t. perturbation matrix to improve the efficiency of adversarial attacks in the tight-budget hard-label setting. Additionally, the objective function consists of semantic similarity term, pairwise perturbation and sparsity constraints constructed from the perturbation matrix, which are able to improve the semantic similarity and reduce the perturbation rate of initialized adversarial examples, and it is solved in an alternating optimization fashion. As for its performance, quantitative and qualitative experiment results demonstrate that TextHoaxer is capable of generating high-quality adversarial examples with higher semantic similarity and lower perturbation rate in the tight-budget setting.

Acknowledgements

Wang is partially supported by the National Science Foundation under Grant No. 1951729, 1953813 and 1953893.

References

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, 632–642. The Association for Computational Linguistics.

Brendel, W.; Rauber, J.; and Bethge, M. 2018. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. In *ICLR*. OpenReview.net.

Carlini, N.; and Wagner, D. A. 2017. Towards Evaluating the Robustness of Neural Networks. In *S&P*, 39–57. IEEE Computer Society.

Cer, D.; Yang, Y.; Kong, S.-y.; Hua, N.; Limtiaco, N.; John, R. S.; Constant, N.; Guajardo-Céspedes, M.; Yuan, S.; Tar, C.; et al. 2018. Universal sentence encoder.

Chen, J.; Jordan, M. I.; and Wainwright, M. J. 2020. HopSkipJumpAttack: A Query-Efficient Decision-Based Attack. In *S&P*, 1277–1294. IEEE.

Chen, P.; Zhang, H.; Sharma, Y.; Yi, J.; and Hsieh, C. 2017. ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models. In *AISec@CCS*, 15–26. ACM.

Cheng, M.; Le, T.; Chen, P.; Zhang, H.; Yi, J.; and Hsieh, C. 2019. Query-Efficient Hard-label Black-box Attack: An Optimization-based Approach. In *ICLR*. OpenReview.net.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, 4171–4186. Association for Computational Linguistics.

Ebrahimi, J.; Rao, A.; Lowd, D.; and Dou, D. 2018. HotFlip: White-Box Adversarial Examples for Text Classification. In *ACL*, 31–36. Association for Computational Linguistics.

Gao, J.; Lanchantin, J.; Soffa, M. L.; and Qi, Y. 2018. Black-Box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers. In *S&P Workshops*, 50–56. IEEE Computer Society.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In *ICLR*.

Gorban, A. N.; and Tyukin, I. Y. 2018. Blessing of dimensionality: mathematical foundations of the statistical physics of data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2118): 20170237.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8): 1735–1780.

Jia, R.; and Liang, P. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *EMNLP*, 2021–2031. Association for Computational Linguistics.

Jin, D.; Jin, Z.; Zhou, J. T.; and Szolovits, P. 2020. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. In *AAAI*, 8018–8025. AAAI Press.

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*, 1746–1751. ACL.

Kurakin, A.; Goodfellow, I. J.; and Bengio, S. 2017. Adversarial examples in the physical world. In *ICLR*. OpenReview.net.

Li, J.; Ji, S.; Du, T.; Li, B.; and Wang, T. 2019. TextBugger: Generating Adversarial Text Against Real-world Applications. In *NDSS*. The Internet Society.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning Word Vectors for Sentiment Analysis. In *ACL*, 142–150. The Association for Computer Linguistics.

Maheshwary, R.; Maheshwary, S.; and Pudi, V. 2021. Generating Natural Language Attacks in a Hard Label Black Box Setting. In *AAAI*, 13525–13533. AAAI Press.

Mrksic, N.; Séaghdha, D. Ó.; Thomson, B.; Gasic, M.; Rojas-Barahona, L. M.; Su, P.; Vandyke, D.; Wen, T.; and Young, S. J. 2016. Counter-fitting Word Vectors to Linguistic Constraints. In Knight, K.; Nenkova, A.; and Rambow, O., eds., *NAACL-HLT*, 142–148. The Association for Computational Linguistics.

Pang, B.; and Lee, L. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *ACL*, 115–124. The Association for Computer Linguistics.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*, 1532–1543. ACL.

Ren, S.; Deng, Y.; He, K.; and Che, W. 2019. Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency. In *ACL*, 1085–1097. Association for Computational Linguistics.

Williams, A.; Nangia, N.; and Bowman, S. R. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *NAACL-HLT*, 1112–1122. Association for Computational Linguistics.

Zhang, X.; Zhao, J. J.; and LeCun, Y. 2015. Character-level Convolutional Networks for Text Classification. In *NeurIPS*, 649–657.

Zhou, Y.; Wang, H.; He, J.; and Wang, H. 2021a. From Intrinsic to Counterfactual: On the Explainability of Contextualized Recommender Systems. arXiv:2110.14844.

Zhou, Y.; Wu, J.; Wang, H.; and He, J. 2021b. Adversarial Robustness through Bias Variance Decomposition: A New Perspective for Federated Learning. arXiv:2009.09026.

Zhou, Y.; Xu, J.; Wu, J.; Nasrabadi, Z. T.; Körpeoglu, E.; Achan, K.; and He, J. 2021c. PURE: Positive-Unlabeled Recommendation with Generative Adversarial Network. In *KDD*, 2409–2419. ACM.

Zhu, C.; Cheng, Y.; Gan, Z.; Sun, S.; Goldstein, T.; and Liu, J. 2020. FreeLB: Enhanced Adversarial Training for Natural Language Understanding. In *ICLR*. OpenReview.net.