BESA: BERT-based Simulated Annealing for Adversarial Text Attacks

Xinghao Yang[1] , **Weifeng Liu**[2] , **Dacheng Tao**[3] and **Wei Liu**[1,*]

[1]School of Computer Science, University of Technology Sydney, Australia
[2]College of Control Science and Engineering, China University of Petroleum (East China), China
[3]JD Explore Academy, JD.com, China
xinghao.yang@student.uts.edu.au, liuwf@upc.edu.cn, dacheng.tao@gmail.com, wei.liu@uts.edu.au

Abstract

Modern Natural Language Processing (NLP) models are known immensely brittle towards text adversarial examples. Recent attack algorithms usually adopt word-level substitution strategies following a pre-computed word replacement mechanism. However, their resultant adversarial examples are still imperfect in achieving grammar correctness and semantic similarities, which is largely because of their unsuitable candidate word selections and static optimization methods. In this research, we propose BESA, a BERT-based Simulated Annealing algorithm, to address these two problems. Firstly, we leverage the BERT Masked Language Model (MLM) to generate contextual-aware candidate words to produce fluent adversarial text and avoid grammar errors. Secondly, we employ Simulated Annealing (SA) to adaptively determine the word substitution order. The SA provides sufficient word replacement options via internal simulations, with an objective to obtain both a high attack success rate and a low word substitution rate. Besides, our algorithm is able to jump out of local optima with a controlled probability, making it closer to achieve the best possible attack (i.e., the global optima). Experiments on five popular datasets manifest the superiority of BESA compared with existing methods, including TextFooler, BAE, BERT-Attack, PWWS, and PSO.

1 Introduction

Deep Neural Networks (DNNs) have shown the vulnerability to adversarial examples in Natural Language Processing (NLP) field for text classification [Papernot *et al.*, 2016]. Adversarial text example is the original input with some malicious perturbations. These modifications are usually imperceptible to human readers but can easily subvert DNNs correct prediction. Therefore, exploring potential textual adversarial attacks is crucial to understand DNNs behaviors and ensure their robust performance, especially on those security-critical applications, such as sentiment analysis and toxic comment detection [Zhang *et al.*, 2019].

Existing text attack algorithms can be roughly divided into three groups: character-level attacks, sentence-level attacks, and word-level attacks. The character-level attacks (e.g., $noise \rightarrow nosie$) result in misspelled words, and the sentence-level attacks (e.g. inserting a sentence into the original text) usually lead to dramatic semantic changes. Therefore, most recent works have focused on the word-level attacks with two main steps: (1) determining the substitution words, and (2) determining which word(s) to be substituted. In the first step, the substitution words should be semantically close to the original word so that the modifications are imperceptible to humans. Typical strategies search candidate words from the Glove embedding space [Alzantot *et al.*, 2018], the WordNet synonym space [Ren *et al.*, 2019], and the HowNet sememe space [Zang *et al.*, 2020]. However, these algorithms create the candidates subset for every *single* word and ignore their contextual environments, which easily produce out-of-context modifications [Li *et al.*, 2020]. In the second step, most prior works perform word replacement by descending order of word importance score (WIS). There are various definitions for WIS, such as probability weighted word saliency (PWWS) [Ren *et al.*, 2019] and the changes of DNNs' confidence after deleting a word [Jin *et al.*, 2020; Garg and Ramakrishnan, 2020], etc. However, altering words by a fixed (or static) WIS order usually leads to local optimal and word over-substitution. For example, we empirically found that changing the top-3 WIS words $\{top1, top2, top3\}$ altogether can not mislead a classifier, but sometimes forming a two-word combination $\{top1, top3\}$ can make it.

In this paper, we propose BESA, a BERT-based Simulated Annealing algorithm, to address these problems and generate fluent adversarial text examples. Specifically, we employ the BERT Masked Language Model (BERT-MLM) [Devlin *et al.*, 2018] to generate candidates in the first step. The BERT-MLM is trained from both left-to-right and right-to-left (i.e., bi-directorial text sequences) on extremely large English text data with general-purpose. Therefore, it is capable of predicting contextual fitting words [Devlin *et al.*, 2018]. In the second step, we propose to determine the word replacement priority via the Simulated Annealing (SA) optimization method [Kirkpatrick *et al.*, 1983]. The SA breaks the fixed (static) replacement order and provides more options of word replacements, which is significantly effective in finding high-quality

*Corresponding Author

adversarial examples. Besides, it allows our algorithm to escape from local optima by a controlled probability, making it closer to the best possible word substitution in the attack. Extensive experiments demonstrate that the proposed BESA achieves the highest attack success rate via a low word substitution rate. We emphasize our main contributions as below.

- We propose to generate natural substitution tokens via a pre-trained BERT-MLM. These tokens can well fit the text contextual and be read smoothly by humans.
- We design an effective Simulated Annealing (SA) method to determine the word replacement priority. The SA objective function is designed for accomplishing a high attack success rate (ASR) and low word substitution rate (WSR).
- We evaluate the effectiveness of our overall model, the BESA, on five public datasets. Experimental results manifest that our BESA not only improves the ASR and reduces the WSR, but also shows superiorities in keeping grammar correctness and semantic similarities.

2 Related Works

We categorize textual attack methods into character-level attacks, sentence-level attacks, and word-level attacks. The character-level attacks handle text adversarial samples by inserting, deleting, or swapping characters [Liang *et al.*, 2018]. However, these operations produce misspelled words, which can be easily defended by a spelling checker. Sentence-level attack concatenates a distracting sentence with the input text to confuse the reading comprehension system [Jia and Liang, 2017]. Nevertheless, the generated sentences are usually human unreadable and semantically uninterpretable.

Word-level attack replaces the original words with their synonyms by solving two main challenges (1) how to select synonym candidates and (2) how to determine the word replacement priority. Early works employ Glove embedding vectors to represent words and identify the nearest neighbors as synonym candidates [Alzantot *et al.*, 2018; Jin *et al.*, 2020]. However, the Glove embedding usually fails to distinguish antonym from synonym. For example, the word pair $\{east, west\}$ and $\{expensive, cheaper\}$ are close in the Glove embedding space, but they are semantically opposite. Therefore, a counter-fitting post process is essential to ensure the semantic similarity. Compared with Glove embedding, searching synonyms from human-annotated linguistic thesauri is a more straightforward approach. [Ren *et al.*, 2019] selected synonyms from WordNet, which groups words based on their meanings, and designed the probability weighted word saliency (PWWS) for greedy word perturbation. [Zang *et al.*, 2020] presented a sememe-based word substitution, which searched replacement tokens from HowNet and found adversarial example via particle swarm optimization (PSO). However, Glove-based and thesaurus-based strategies collect substitution words for every single word, ignoring how well they fit into the context.

Several recent works take advantage of language models to predict contextual friendly tokens, intending to generate more fluent text adversarial samples [Garg and Ramakrishnan, 2020; Li *et al.*, 2020]. However, both of them adopt a static word replacement order guided by the word importance score (WIS), leading to redundancy word substitution. The difference lies in that [Garg and Ramakrishnan, 2020] defined the WIS as probability decrease of the correct label after deleting a word, while [Li *et al.*, 2020] replaced each of the original words by a dummy symbol [MASK].

3 The BESA Method

This section details our BESA algorithm. Before delving into details, we present the attack settings and expectations.

3.1 Black-box Untargeted Attack

The BESA algorithm is designed for black-box attack, where no DNNs architecture, internal parameter, and gradient information are accessible, a setting close to real-life attack scenarios. The only capacity of a black-box adversary is inputting an example to the victim model and querying its prediction results, acting as a regular user.

Suppose we are given a text dataset with N samples $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_N\}$, belonging to L labels $\mathcal{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \cdots, \mathbf{Y}_L\}$, a DNNs classifier F needs to learn a input-output mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$ so that any input $\mathbf{X} \in \mathcal{X}$ can be classified to the true label $\mathbf{Y}_{true}$, by optimizing the posterior probability:

$$\arg\max_{\mathbf{Y}_i \in \mathcal{Y}} P(\mathbf{Y}_i|\mathbf{X}) = \mathbf{Y}_{true} \tag{1}$$

An adversarial text example $\mathbf{X}^*$ is defined as the original input $\mathbf{X}$ that is modified with slight perturbation $\Delta\mathbf{X}$, i.e., $\mathbf{X}^* = \mathbf{X} + \Delta\mathbf{X}$. A text adversarial attacker expects the perturbation $\Delta\mathbf{X}$ is imperceptible to human but can mislead the classifier F into any label other than the true label (untargeted attack) or a user-specified label (targeted attack). In this work, we design BESA as an untargeted attack. In linguistic perspectives, the adversarial example should be lexical correct, grammar correct, and semantic similar to the original texts. These constraints aim to ensure humans cannot spot the difference before and after an attack. To this end, we generate contextual friendly synonyms via BERT-MLM (Sec 3.2) and reduce the substitution rate by SA optimization (Sec 3.3).

3.2 BERT-based Candidate Selection

Since BERT Masked Language Model (BERT-MLM) is trained by masking a proportion of words, it is capable of generating supplementary words to reconstruct the sentence. Suppose the input sentence $\mathbf{X} = \{w_1, w_2, \cdots, w_n\}$ contains n words, for each word w_i, we replace it with the [MASK] token and then predict the substitution words $\mathbb{S}_i$ with BERT-MLM. The substitution words are contextual-aware and well-fitted with the surrounding text, but can not guarantee semantic similarity. For example, in the sentence "I [MASK] this move", predicting the [MSAK] as "like" or "hate" are equally fluent but delivering different sentiment. Therefore, we employ the Universal Sentence Encoder (USE) [Cer *et al.*, 2018] to calculate the semantic similarity score and filter out low-semantic-consistency words by a threshold. To avoid grammar errors, we further filter out candidates that have different Part-of-Speech (POS) tags with the original word. These two filter steps are uniformly denoted in line 5 of Algorithm 1.

Algorithm 1: The proposed BESA algorithm

Input: Original sentence with n words $\mathbf{X} = (w_1, \cdots, w_n)$
Input: DNNs classifier F
Output: Adversarial example $\mathbf{X}_{adv}$

1 Initialization: USE threshold $= 0.5$, the highest temperature $T_{max} = 1000$, the lowest temperature $T_{min} = 50$, internal simulation steps $K = 20$, the balance parameter $\delta = 0.01$, attack radius parameter $\sigma = 3$, the initial adversarial example $\mathbf{X}_{adv} = \mathbf{X}$, and the initial time $t = 0$;
/* BERT-based Candidate Selection */
2 **for** $i = 1$ **to** n **do**
3 Replace the w_i with [MASK];
4 Generate the candidate set $\mathbb{B}_i$ with BERT-MLM;
5 $\mathbb{B}_i = Filter(\mathbb{B}_i)$;
6 Find the best candidate w_i^* from $\mathbb{B}_i$ via Eq. (5);
/* Definition of the Objective Function */
7 **Function** $J(F, \mathbf{X}, \mathbf{X}_{adv}, \delta)$**:**
8 Calculate the true label score: $P_{true} = F(\mathbf{X}_{adv})$;
9 Count the perturbation cost: $cost = len(\mathbf{X} \neq \mathbf{X}_{adv})$;
10 Objective value: $y = P_{true} + \delta \times cost$;
11 **return** y;
/* The Simulated Annealing Optimization */
12 **while** $T_{max} > T_{min}$ **do**
13 **for** $k = 1$ **to** K **do**
14 $y = J(F, \mathbf{X}, \mathbf{X}_{adv}, \delta)$;
15 $index = t + randint(0, \sigma t)$; ▷ Random index
16 **if** $index > n$ **then**
17 continue;
18 Replace the $index$ word in Eq. (7) to craft $\mathbf{X}_{adv}^k$;
19 $y_{new} = J(F, \mathbf{X}, \mathbf{X}_{adv}^k, \delta)$;
20 **if** $y_{new} - y < 0$ **then**
21 $\mathbf{X}_{adv} = \mathbf{X}_{adv}^k$; ▷ Good modification
22 **else**
23 $p = e^{-(y_{new}-y)/T_{max}}$; ▷ Metropolis
24 $r = random(0, 1)$;
25 **if** $r < p$ **then**
26 $\mathbf{X}_{adv} = \mathbf{X}_{adv}^k$; ▷ Bad modification
27 **if** $F(\mathbf{X}_{adv}) \neq \mathbf{Y}_{true}$ **then**
28 **return** $\mathbf{X}_{adv}$; ▷ Successful attack
29 $t = t + 1$;
30 $T_{max} = T_{max}/(1 + t)$; ▷ Quick cooling
31 **return** Adversarial example $\mathbf{X}_{adv}$

After filtering, every $w_i' \in \mathbb{S}_i$ is a potential candidate for replacing the original word w_i. To select the best candidate from $\mathbb{S}_i$, we define the candidate importance score $I_{w_i'}$ as the true score probability reduction:

$$I_{w_i'} = P(\mathbf{Y}_{true}|\mathbf{X}) - P(\mathbf{Y}_{true}|\mathbf{X}_i'), \forall w_i' \in \mathbb{S}_i \tag{2}$$

where

$$\mathbf{X} = \{w_1, w_2, \cdots, w_i, \cdots, w_n\} \tag{3}$$

$$\mathbf{X}_i' = \{w_1, w_2, \cdots, w_i', \cdots, w_n\} \tag{4}$$

Then we identify the candidate w_i' that attains the highest $I_{w_i'}$ as the best substitution word. The best candidate selection method $R(w_i, \mathbb{S}_i)$ can be denoted as below:

$$w_i^* = R(w_i, \mathbb{S}_i) = \arg\max_{w_i' \in \mathbb{S}_i} I_{w_i'} \tag{5}$$

Repeating these steps on every word completes the first stage of our algorithm, which is summarized in Algorithm 1 from line 2 to line 6. Based on the best substitution words, we obtain n adversarial examples $\{\mathbf{X}_1^*, \mathbf{X}_2^*, \cdots, \mathbf{X}_n^*\}$ with each $\mathbf{X}_i^* = \{w_1, \cdots, w_i^*, \cdots, w_n\}$. The change of true label probability between $\mathbf{X}$ and $\mathbf{X}_i^*$ indicates the strongest attack effect that can be achieved by altering w_i.

$$\Delta P_i^* = P(\mathbf{Y}_{true}|\mathbf{X}) - P(\mathbf{Y}_{true}|\mathbf{X}_i^*) \tag{6}$$

A straightforward way to determine the word substitution priority is by the descending order of the ΔP_i^*, where the resorted sequence order is given in Eq. (7).

$$H = \{\Delta P_1^* > \Delta P_2^* > \cdots > \Delta P_n^*\} \tag{7}$$

However, sequentially modifying words by this static order easily leads to word over-substitution, as there is no theoretical guarantee that the top-k words combination contributes most in misleading a classifier. For example, replacing the two word $\{top1, top3\}$ in H sometimes even better than replacing three words $\{top1, top2, top3\}$. Therefore, it is critical to optimize word substitution priority.

3.3 Simulated Annealing (SA) Optimization

Before elaborating on our algorithm, we first explain the concepts of the original Simulated Annealing (SA). The SA is a combinatorial optimization algorithm inspired by the physical annealing procedure of metals. It targets to find the minimal value of an objective function from many independent molecules (i.e., variables). Specifically, the algorithm starts from a "melting" state with high temperature and gradually reduces the temperature until the system "freezes" to a steady-state. At each temperature, the interval simulation must proceed long enough for the system to reach equilibrium. During simulating, any better molecular movement that decreases the objective function is directly accepted. Nevertheless, modifications that increase the objective function are also accepted with a transition probability. Therefore, the SA is possible to transition from the local optimum, while a static greedy search and particle swarm optimization (PSO) are incapable of avoiding such a risk.

In this paper, we treat the best word substitution order determination as a multivariates combinatorial optimization problem. Specifically, a word in the sentence corresponds to a molecule in search space, and each molecule position movement corresponds to a word substitution. To optimize the word substitution priority by SA, we first initialize essential variables in line 1 of Algorithm 1, including the highest temperature T_{max}, the lowest temperature T_{min}, the internal simulation steps at each temperature K, etc. When designing the objective function (Algorithm 1 from line 7 to line 11), we expect that a good adversarial example $\mathbf{X}_{adv}$ can deliver a low true label probability

$$P_{true} = F(\mathbf{X}_{adv}) \tag{8}$$

and simultaneously preserve a low perturbation cost

$$cost = len(\mathbf{X} \neq \mathbf{X}_{adv}) \tag{9}$$

where $len(\mathbf{X} \neq \mathbf{X}_{adv})$ count the number of different words between $\mathbf{X}$ and $\mathbf{X}_{adv}$. The final objective value balances Eq.

(8) and Eq. (9) with an parameter δ

$$y = P_{true} + \delta \times cost \tag{10}$$

The SA optimization starts from line 12 of Algorithm 1 with the highest temperature T_{max}. At each temperature, enough simulations (K) are attempted to reach the equilibrium. Each simulation first queries the objective function J to obtain the original objective value of y in line 14. Then the SA algorithm randomly selects a molecular from the whole molecular space to move. In this work, we restrict the word selection interval with an attack radius parameter σ, and the target word index is determined by Eq. (11)

$$index = t + randint(0, \sigma t) \tag{11}$$

where $randint(0, \sigma t)$ randomly selects a integer from $[0, \sigma t]$. This enables us to locate the most important word in the initial time ($t = 0$) and enlarges the attack radius as time goes by. Then we replace the $index$-th word of H to craft a new adversarial example in line 18 and query the new objective value y_{new} in line 19. To decide whether the modification is accepted or not, we employ the typical Metropolis principle

$$p = \begin{cases} 1 & , \quad y_{new} - y < 0 \\ e^{-(y_{new}-y)/T_{max}} & , \quad y_{new} - y > 0 \end{cases} \tag{12}$$

If the word replacement decreases the objective function, we accept it as a good modification in line 21. Otherwise, we accept a bad modification by a probability as listed from line 23 to line 26. After modification, if the classifier F is misled, we return the successful adversarial example in line 28; otherwise, the internal simulation continues to iterate. If no successful adversarial example can be found through all internal iterations, the time t will pass and the temperature will decrease. For efficiency, we adopt the quick cooling function

$$T_{max} = \frac{T_{max}}{1 + t} \tag{13}$$

This procedure continues until we achieve a successful attack or the temperature reduced to T_{min}. Intuitively, the SA optimization provides sufficient combination options with internal simulations and can jump out of local optima by Metropolis criteria. These properties are significant in achieving a high success attack rate with a low perturbation cost.

4 Experiments and Analysis

We provide the source code in the supplementary material to ensure the results in this section are reproducible.

4.1 Datasets and Victim Models

We conduct experiments on five widely used text datasets, including IMDB [Maas *et al.*, 2011] , Question Natural Language Inference (QNLI) [Rajpurkar *et al.*, 2016], Movie Review (MR) [Pang and Lee, 2005], Stanford Natural Language Inference (SNLI) [Bowman *et al.*, 2015], and SST-2 [Socher *et al.*, 2013]. The IMDB, MR, and SST-2 are used for binary sentiment classification tasks, where QNLI and SNLI are used for natural language inference. Statistical details of these datasets are summarized in Table 1.

Dataset	# Train	# Test	# Valid.	# Classes	# Words
IMDB	25,000	25,000	—	2	235.73
QNLI	104,743	5,463	5,463	2	36.68
MR	8,530	1,066	1,066	2	18.49
SNLI	550,152	10,000	10,000	3	20.27
SST-2	67,349	1,821	872	2	8.67

Table 1: Statistic information of the five datasets. "# Words" denotes the average number of words (i.e., average text length).

We apply our BESA to six victim models, such as CNN, LSTM, BERT [Devlin *et al.*, 2018], DistilBERT [Sanh *et al.*, 2019], RoBERTa [Liu *et al.*, 2019], and XLNet [Yang *et al.*, 2019]. The CNN consists of a 200-dimensional embedding layer and a 1-D convolutional layer containing 150 filters with filter sizes of 3×4×5. The LSTM consists of a 200-dimensional embedding layer and a bidirectional LSTM layer with 150 hidden states. Both CNN and LSTM have a dropout rate of 0.3. We download BERT (bert-base-uncased), RoBERTa (roberta-base), XLNet (xlnet-base-cased), and DistilBERT (distilbert-base-uncased, distilbert-base-cased) from the Transformers model hub HuggingFace[1]. Different versions of DistilBERT are implemented on different datasets, i.e., (uncased: IMDB, QNLI) and (cased: SNLI, SST-2). The original test results are listed in Table 2's "ACC" column.

4.2 Baselines

To evaluate the effectiveness of our BESA, we compare it with typical black-box word-level attack algorithms, such as TextFooler (**TEFO**) [Jin *et al.*, 2020], BERT-based Adversarial Examples (**BAE**) [Garg and Ramakrishnan, 2020], BERT-Attack (**BEAT**) [Li *et al.*, 2020], **PWWS** [Ren *et al.*, 2019], and **PSO** [Zang *et al.*, 2020]. As mentioned in Section 2, most existing works (TEFO, BAE, BEAT, and PWWS) rephrase words by a static WIS descending order, which is inflexible in finding the optimal solution. [Zang *et al.*, 2020] treated every sentence as a particle and optimize its location in the search space via PSO. However, the PSO needs abundant of query to find the final modification, leading to high computational cost and low efficiency (Table 3).

4.3 Evaluation Metrics and Experiment Settings

Evaluation Metrics. We evaluate the attack performance mainly by the Attack Success Rate (ASR) and Word Substitution Rate (WSR). The ASR is defined as the proportion of successful adversarial examples to the total number of correctly classified samples. The WSR refers to the percentage of replaced words occupied by the total number of words in the sentence. Intuitively, a rational attacker expects a high ASR with a low WSR. Besides, we assess the quality of the adversarial text examples by calculating the grammar errors and semantic consistency.

Experiment Settings. The parameter settings for our BESA are given in line 1 of Algorithm 1. Parameter tuning details are listed in supplementary. For all baselines, we use their author recommended parameter value. To achieve efficiency, we randomly select 1000 samples from the test set to

[1]https://huggingface.co/models

Data	Victim Model	ACC	Attack Success Rate (ASR)						Word Substitution Rate (WSR)					
			TEFO	BAE	BEAT	PWWS	PSO	BESA	TEFO	BAE	BEAT	PWWS	PSO	BESA
IMDB	CNN	81%	**100%**	**100%**	99.88%	**100%**	**100%**	**100%**	2.83%	1.86%	3.32%	1.83%	1.43%	**1.37%**
	LSTM	82%	**100%**	**100%**	99.88%	**100%**	**100%**	**100%**	3.14%	1.93%	3.47%	2.09%	1.46%	**1.31%**
	DisBERT	91.2%	99.12%	96.93%	95.29%	**99.67%**	—	**99.67%**	6.54%	3.28%	5.82%	3.73%	—	**1.79%**
	BERT	91.3%	93.87%	89.92%	84.67%	97.15%	—	**99.78%**	10.12%	4.76%	7.84%	5.21%	—	**2.70%**
	XLNet	94.8%	98.21%	99.16%	96.62%	99.26%	—	**100%**	8.64%	3.51%	6.32%	4.85%	—	**2.11%**
	RoBERTa	93.7%	97.55%	97.76%	94.88%	98.29%	—	**99.36%**	10.13%	3.82%	6.67%	5.74%	—	**2.06%**
QNLI	DisBERT	79.8%	92.70%	94.87%	90.99%	89.17%	93.27%	**100%**	12.28%	6.80%	9.40%	9.22%	7.34%	**5.77%**
	BERT	91.4%	92.56%	92.67%	89.17%	90.15%	94.42%	**98.80%**	11.98%	6.64%	9.11%	8.43%	6.82%	**5.73%**
	RoBERTa	91.8%	84.64%	92.48%	86.82%	85.40%	92.70%	**99.89%**	12.04%	7.10%	9.31%	9.52%	7.28%	**6.84%**
MR	CNN	76.7%	93.35%	95.44%	83.31%	94.26%	96.22%	**99.87%**	17.04%	12.98%	15.06%	13.22%	**11.53%**	12.05%
	LSTM	77.8%	91.77%	95.37%	84.06%	93.70%	95.37%	**99.49%**	15.61%	11.71%	13.59%	13.07%	10.91%	**10.88%**
	BERT	84.3%	73.90%	83.16%	63.23%	81.02%	92.41%	**97.86%**	20.91%	14.44%	15.32%	14.67%	11.93%	**11.02%**
	XLNet	84.3%	77.35%	89.02%	70.94%	83.64%	92.68%	**98.97%**	20.87%	13.52%	15.38%	14.88%	**11.79%**	11.93%
	RoBERTa	88.8%	77.14%	90.32%	71.85%	82.66%	91.78%	**95.95%**	21.47%	13.27%	15.71%	15.95%	12.30%	**11.12%**
SNLI	DisBERT	85.5%	91.58%	98.36%	98.48%	98.25%	94.27%	**100%**	8.40%	6.17%	7.80%	7.60%	6.36%	**5.87%**
	BERT	89.8%	89.87%	98.89%	98.66%	98.44%	92.54%	**99.33%**	7.97%	6.31%	8.19%	7.72%	6.30%	**5.81%**
SST-2	CNN	82.7%	92.37%	95.48%	86.41%	93.90%	96.67%	**99.45%**	17.09%	12.53%	15.40%	13.10%	**11.47%**	11.49%
	LSTM	84.5%	93.22%	96.20%	86.43%	93.89%	96.47%	**99.73%**	17.55%	12.83%	15.31%	13.53%	**11.45%**	11.53%
	BERT	92.4%	80.52%	90.57%	72.08%	86.60%	91.32%	**98.76%**	20.61%	13.30%	15.65%	16.01%	12.80%	**12.21%**
	DisBERT	90%	85.22%	93.50%	77.71%	87.52%	92.48%	**98.85%**	19.36%	12.72%	15.38%	14.65%	12.28%	**11.46%**
	RoBERTa	94%	75.49%	91.95%	72.93%	84.63%	91.71%	**99.39%**	21.95%	13.17%	16.29%	16.62%	13.34%	**11.77%**

Table 2: The attack success rate (ASR) and average word substitution rate (WSR) of various attack algorithms on five text datasets. The **best results** are highlighted in **bold**. The "ACC" column shows the original test accuracy without attacks, and DisBERT is short for DistilBERT.

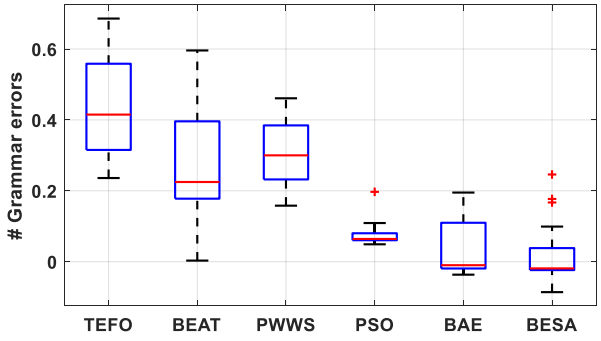

Figure 1: The number of grammar errors increased after attacks.

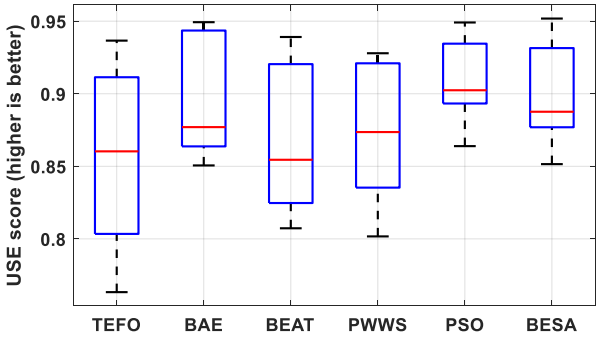

Figure 2: The semantic similarity before and after attacks.

craft adversarial samples. All experiments are implemented on the TextAttack framework [Morris *et al.*, 2020].

4.4 Experimental Results

The experimental results of both ASR and WSR are listed in Table 2. For the ASR, our BESA achieves nearly perfect results on all scenarios and outperforms the baselines, often by a large margin. The WSR results show that our BESA obtains such attack effects by making the lowest word modifications in most cases. We attribute this superiority to the SA optimization, as this is the major improvement of our BESA compared with baselines (BAE and BEAT). Additionally, Table 2 results show that the advanced models (BERT, DistilBERT, RoBERTa, and XLNet) are generally more robust than traditional classifiers (CNN and LSTM), as attacking them need more word substitutions but obtaining less success rate.

Grammar Errors. Grammatical errors in a sentence largely affect its fluency, so fewer grammar errors are naturally preferred. We employ the LanguageTool[2] to detect the spell and grammar errors for the original samples and the corresponding adversarial samples on all datasets. Figure 1 plots the number of grammar errors induced by various attack algorithms. Clearly, our BESA brings the minimum number of syntax errors and sometimes even fixes the original errors.

Semantic Consistency. The semantic similarity between the original sample and the adversarial sample plays an important role in ensuring that readers do not change their initial decisions. We adopt the Universal Sentence Encoder (USE) score to indicate the semantic consistency before and after attacks. Figure 2 shows the USE score on the five datasets and

[2]https://languagetool.org/

Dataset	TEFO	BAE	BEAT	PWWS	PSO	BESA
IMDB	17562	7580	19154	26760	—	56251
QNLI	2610	2458	3055	6073	169089	23448
MR	1845	1945	3381	2598	53699	6373
SNLI	835	1022	1446	2461	6458	3864
SST-2	952	1116	1604	1237	23341	2820

Table 3: Time (in seconds) needed in attacking the BERT model.

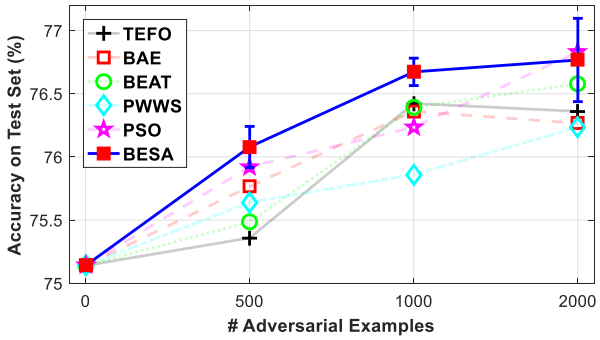

Figure 3: Adversarial retraining results on MR dataset. The higher the accuracy, the more robust of the model after retraining.

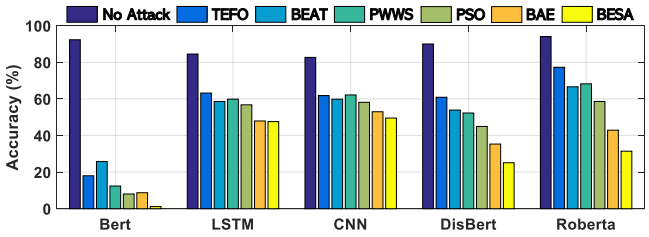

Figure 4: Transfer attack on SST-2 dataset. Lower accuracy indicates higher transferability (the lower the better).

SST-2 (Label change): Negative (51%) → Positive (98%)
Very special effects, brilliantly bold ~~colors~~ visuals and heightened reality can't hide the giant achilles' heel in "stuart little 2": there 's just no story, folks.
MR (Label change): Negative (56%) → Positive (81%)
The first mistake, I suspect, is casting shatner as a legendary professor and kunis as a brilliant ~~college~~ arts student–where's pauly shore as the rocket scientist?

Table 4: Adversarial examples crafted by BESA. The blue and red color denotes the original and substitution words, respectively.

all victim models. Results illustrate that our BESA achieves comparable (second highest) semantic consistency.

Efficiency Analysis. We carry out all experiments on Enterprise Linux Workstation 7.7 with Intel(R) Xeon(R) Gold 6150 2.7GHz CPU, NVIDIA Quadro P5000 16G GPU, and 176GB RAM. Table 3 lists the time consuming of various methods by attacking BERT. Results show that our BESA costs more time than static counterparts (TEFO, BAE, BEAT, PWWS) but more efficient than the PSO. Besides, the PSO is not suitable for attacking very deep models (DisBERT, BERT, XLNet, RoBERTa) with long text input. For example, it only completed 73 IMDB samples in one week to cheat BERT. For this reason, results of PSO are shown as infeasible to obtain (i.e., dash "—") in Tables 2 and 3.

4.5 Adversarial Training

Adversarial training is a prevalent technique to improve the DNNs robustness by incorporating adversarial examples into the training set. To validate this, we train the LSTM model on the MR dataset and obtains 75.14% test accuracy. Then we randomly generate and join $\{500, 1000, 2000\}$ MR adversarial samples to its training data and retrain the LSTM. The 3-run main accuracy and standard deviation for our BESA are shown in Figure 3. We only show the average accuracy for baselines to make the results clear. Figure 3 demonstrates that the model robustness gradually improved by appending more adversarial examples, and our BESA brings greater robustness improvement than the five baselines.

4.6 Transferability

The transferability of adversarial attack implies whether its adversarial samples designed for misleading the classifier F can also fool an unknown classifier F'. We evaluate the transferability on SST-2 dataset. Specifically, we choose the adversarial examples crafted for attacking BERT to perform the transfer attack on four unknown models (LSTM, CNN, DistilBERT, and RoBERTa). Their classification accuracy on both clean data (no attack) and adversarial data are shown in Figure 4. Figure 4 illustrates that our BESA generates adversarial samples with overall higher transferability than baselines.

4.7 Qualitative Examples

We list two adversarial examples crafted by our BESA in Table 4. Both examples successfully mislead the LSTM model from Negative to Positive with high confidence (i.e., $> 80\%$). Owing to the BERT-MLM, the substitution words (visuals and arts) are well-fit the surrounding text and do not change the sentiment label from human perception. More examples can be found in the supplementary material.

5 Conclusion and Future Work

In this paper, we proposed an innovative word-level text attack algorithm named BESA (BERT-based Simulated Annealing). The BESA employs the BERT Mask Language Model to generate substitution candidates and optimizes the word substitution order via Simulated Annealing (SA). We evaluate the effectiveness of BESA by comparing it with representative baselines on widely used text datasets. Experimental results illustrate that our BESA achieves the highest attack success rate by modifying the least words. Due to the contextual-awareness and the low modification rate of our method, BESA makes fewer grammar errors and keeps a high semantic similarity. Additionally, the adversarial examples generated by our algorithm show good properties in transfer attack and adversarial retrain. In the future, improving the language model to further enhance the semantic similarity is a promising work direction.

References

[Alzantot *et al.*, 2018] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, 2018.

[Bowman *et al.*, 2015] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, 2015.

[Cer *et al.*, 2018] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Garg and Ramakrishnan, 2020] Siddhant Garg and Goutham Ramakrishnan. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, 2020.

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

[Jin *et al.*, 2020] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 8018–8025, 2020.

[Kirkpatrick *et al.*, 1983] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[Li *et al.*, 2020] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, 2020.

[Liang *et al.*, 2018] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4208–4215, 2018.

[Liu *et al.*, 2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[Maas *et al.*, 2011] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 142–150, 2011.

[Morris *et al.*, 2020] John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP. In *Proceedings of the 2020 Conference on EMNLP: System Demonstrations*, pages 119–126, 2020.

[Pang and Lee, 2005] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics*, page 115–124, 2005.

[Papernot *et al.*, 2016] Nicolas Papernot, Patrick D. McDaniel, Ananthram Swami, and Richard E. Harang. Crafting adversarial input sequences for recurrent neural networks. *CoRR*, abs/1604.08275, 2016.

[Rajpurkar *et al.*, 2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

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

[Sanh *et al.*, 2019] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[Socher *et al.*, 2013] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[Yang *et al.*, 2019] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763, 2019.

[Zang *et al.*, 2020] Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080, 2020.

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