Expert Systems With Applications 213 (2023) 119170

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Efficiently generating sentence-level textual adversarial examples with Seq2seq Stacked Auto-Encoder

Ang Li [a], Fangyuan Zhang [a], Shuangjiao Li [a], Tianhua Chen [c], Pan Su [a,b], Hongtao Wang [a,b,*]

[a] School of Control and Computer Engineering, North China Electric Power University, Baoding, China
[b] Hebei Key Laboratory of Knowledge Computing for Energy and Power, Baoding, China
[c] Department of Computer Science, School of Computing and Engineering, University of Huddersfield, Huddersfield, UK

ARTICLE INFO

Keywords:
Sentence-level attack
Textual adversarial examples
Deep neural network
Stacked auto-encoder

ABSTRACT

In spite deep learning has advanced numerous successes, recent research has shown increasing concern on its vulnerability over adversarial attacks. In Natural Language Processing, crafting high-quality adversarial text examples is much more challenging due to the discrete nature of texts. Recent studies perform transformations on characters or words, which are generally formulated as combinatorial optimization problems. However, these approaches suffer from inefficiency due to the high dimensional search space. To address this issue, in this paper, we propose an end-to-end Seq2seq Stacked Auto-Encoder (SSAE) neural network, which generates adversarial text examples efficiently via direct network inference. SSAE has two salient features. The outer auto-encoder preserves syntactic and semantic information to the original examples. The inner auto-encoder projects sentence embedding into a high-level semantic representation, on which constrained perturbations are superimposed to increase adversarial ability. Experimental results suggest that SSAE has a higher attack success rate than existing word-level attack methods, and is 100x to 700x faster at attack speed on IMDB dataset. We further find out that the adversarial examples generated by SSAE have strong transferability to attack different victim models.

1. Introduction

Despite recent advancement of Deep Neural Network (DNN) has spawned numerous successes across various domains, many researches have identified that DNNs are vulnerable to adversarial attacks. By introducing small perturbations, adversarial examples are able to deliberately fool the target model leading to incorrect results. Although generally considered more difficult in the natural language processing (NLP) domain, adversarial examples have been exploited in a number of NLP tasks, such as text classification (Garg & Ramakrishnan, 2020; Zang et al., 2020), machine translation (Yu et al., 2021), question answering (Wallace et al., 2019), and textual entailment (Zang et al., 2020; Zhao et al., 2018).

Unlike attacking computer vision tasks where imperceptible perturbations can be easily introduced into a given image, it is more challenging to craft high-quality adversarial text examples. Two significant factors are considered. (1) Efficiency: we should efficiently generate a adversarial text example in a short time. (2) Naturality: generating grammatically correct and semantically coherent adversarial texts is non-trivial in a discrete space.

Recent literature on generating adversarial examples for NLP tasks can generally be classified as character-level and word-level approaches. The character-level methods (Ebrahimi et al., 2018; Gao et al., 2018; Li et al., 2019; Liang et al., 2018) perform substitution, insertion and deletion on selected characters of original examples. However, the generated examples break the naturality principle and can be directly detected by spell checking and adversarial training techniques (Pruthi et al., 2019). The word-level methods enable to generate high-quality adversarial examples through direct transformations on individual words, e.g., substituting a set of words by synonyms that keep original semantic meanings. However, word-level methods, which are formulated as combinatorial optimization problems, suffer from the inefficiency because of the high-dimensional search space (Zang et al., 2020).

Inspired by the above observations, we intend to propose a sentence-level neural network to generate natural and adversarial examples directly. Fig. 1 demonstrates an example. Left and right subfigures denote the adversarial text generations of search-based approach and the proposed approach, respectively. Both approaches have the same

* Corresponding author at: School of Control and Computer Engineering, North China Electric Power University, Baoding, China.
E-mail addresses: 22s151165@stu.hit.edu.cn (A. Li), zfy@ncepu.edu.cn (F. Zhang), sjli@ncepu.edu.cn (S. Li), T.Chen@hud.ac.uk (T. Chen), supan@ncepu.edu.cn (P. Su), wanght@ncepu.edu.cn (H. Wang).

https://doi.org/10.1016/j.eswa.2022.119170
Received 16 November 2021; Received in revised form 27 October 2022; Accepted 27 October 2022
Available online 3 November 2022

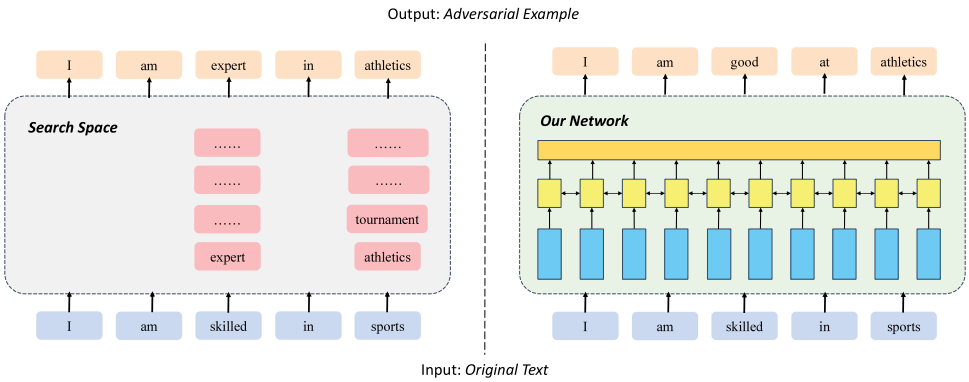


Fig. 1. An example for search based model (left) and the proposed model inference based model (right) in adversarial attack.

input text, i.e., 'I am skilled in sports'. For search-based approach, each word has many candidate synonyms to replace. As a result, finding an adversarial text against target victim model is a combinatorial optimization problem, which consumes high computation cost. But our approach is a sequence-to-sequence (seq2seq) neural network, which directly outputs an adversarial text, e.g., 'I am good at athletics'.

There are three challenges for generating adversarial examples in sentence-level: (1) how to preserve semantic similarity between original and generated examples for imperceptibility; (2) how to generate semantically coherent text for the naturality requirement; and (3) how to bring adversarial factors into generated examples to achieve higher attack success rate. Zhao et al. proposed GANE (Zhao et al., 2018), a state-of-the-art sentence-level attack method. However, they generate examples with poor text coherence and semantic similarity to the original samples.

In this paper, we propose an efficient sentence-level adversarial text generation network called **S**eq2seq **S**tacked **A**uto-**E**ncoder (SSAE) to meet all the above challenges. Specifically, we devise an outer auto-encoder to yield natural text and guarantee semantic similarity between original and generated examples. We stack another inner auto-encoder to transform text embedding into a high-level semantic space for perturbations. In the attack stage, we add small perturbations to text representations that have been encoded in the latent space in order to craft adversarial attributes. Finally, the perturbed text representations are decoded and outputted through the network inference in a very efficient manner. The main contributions of the paper can be summarized as follows:

- A novel and computationally efficient sentence-level solution is proposed that generates textual adversarial examples with high success rates;
- The neural network SSAE hinges on two auto-encoders that enable to generate semantically similar, natural and adversarial examples;
- Evaluated on two NLP tasks, i.e., text classification and textual entailment, the proposed network outperforms alternative methods with promising efficiency and performance.

The reminder of this paper is organized as follows. Section 2 reviews related works. Section 3 presents the proposed novel network. Section 4 introduces the experimental settings, followed by comparative experimental studies reported in Section 5. Finally, Section 6 concludes the paper with future work.

2. Related work

Many existing efforts have been made to generate textual adversarial examples for NLP deep neural networks. Most studies can be classified into three types: character-level, word-level and sentence-level attack.

Character-level attacks generally refer to manipulating characters by transformations including swap, substitution, deletion, and insertion at the granularity of characters (Gao et al., 2018; Li et al., 2019; Liang et al., 2018). Apart from text classification tasks, character-level adversarial attacks have also been investigated for neural machine translation (NMT) tasks (Ebrahimi et al., 2018). Although these methods can achieve high success rates, most generated examples break the naturality principle and can be easily detected by spell checking and adversarial training techniques (Pruthi et al., 2019).

Word-level attack was first studied in Papernot et al. (2016). The key operation is transform several words, e.g., swap, substitution, and insertion. A number of techniques have been proposed at word-level, such as, searching word embeddings space (Sato et al., 2018), selecting synonyms (Jin et al., 2020; Ren et al., 2019) or sememe (Zang et al., 2020), and borrowing language models (Garg & Ramakrishnan, 2020; Li et al., 2020; Zhang et al., 2019). Word-level attacks can generate relatively high-quality natural examples, but the main challenge lies in the high dimensional search space as a result of its combinatorial optimization nature (Zang et al., 2020). Despite many strategies have been proposed such as gradient based methods (Papernot et al., 2016; Sato et al., 2018), sampling based method (Zhang et al., 2019), genetic algorithm (Alzantot et al., 2018), greedy algorithm (Jin et al., 2020; Liang et al., 2018; Ren et al., 2019), the high word search space remains a challenge in efficiently generating adversarial examples.

Similar to the above two approaches, the sentence-level adversarial text generation is also a prevalent research direction, which have been successfully applied to attack NLP models of reading comprehension (Bartolo et al., 2020) and question answering (Wallace et al., 2019). Generally speaking, the semantically equivalent adversarial rules (Ribeiro et al., 2018) and syntactically controlled paraphrase networks (Iyyer et al., 2018) are used to induce changes in the model's predictions. For instance, Zhao et al. (2018) proposed a new latent space framework while incorporating Generative Adversarial Networks to generate grammatically correct and natural adversarial examples that are semantically close to the input. However, these methods also need a greedy algorithm that iteratively searching the optimal results, either in rule space or in semantic space. Sentence-level attacks generate adversarial samples based on the entire original sentence, thus they can generate more diverse adversarial examples than word-level methods. However, our experiments show that adversarial examples generated by sentence-level methods have poor naturality and poor semantic similarity with the original examples (see Fig. 4). Thus, our goal is to find a sentence-level attack method that can have a higher attack success rate, while remain naturality and semantic similarity.

3. Methodology

This section presents the proposed SSAE approach. We start with an introduction to the network, followed by model training description. Then we introduce how it can be utilized to generate adversarial examples efficiently.

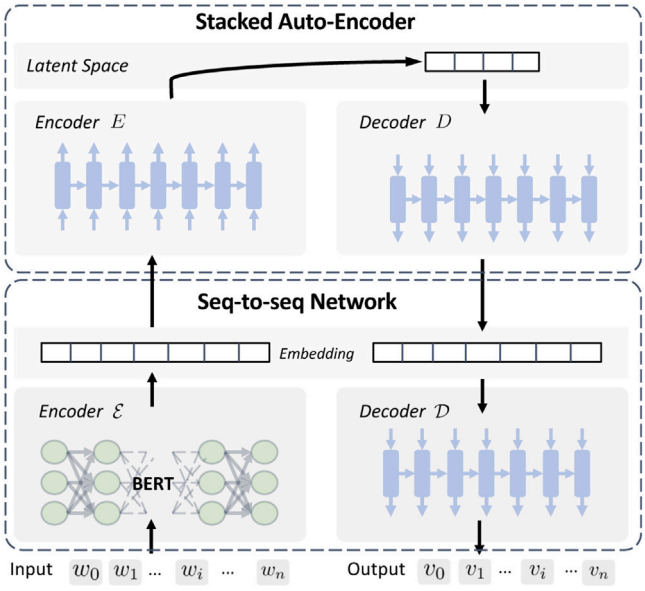


Fig. 2. Network architecture of our SSAE method.

3.1. Proposed network architecture

Assuming a target NLP victim model T and a corpus C, this research aims to establish a neural network model F, such that for a given test example x, an adversarial example $x' = F(x)$ can be generated. The adversarial example would induce a wrong decision to the target model, i.e., $T(x) \neq T(x')$. The key idea of our proposal lies in generating adversarial examples x' directly through conducting inference on the model $F(x)$. Fig. 2 demonstrates the architecture of the proposed SSAE network. SSAE is composed of two auto-encoder networks: a seq-2-seq network and a stacked auto-encoder network. The test example x is transformed to adversarial example x' via encoder $\mathcal{E}$, encoder E, decoder D, and decoder $\mathcal{D}$, respectively.

Seq2seq Network. With the aim to generate adversarial examples directly through the model inference, the proposed method is a sentence-level solution, consisting of two components as illustrated in Fig. 2, where a seq2seq network is first incorporated with an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$. Seq2seq network enables to encode the input text $x = w_0, w_1, \ldots, w_n$ to a text embedding e, and decode it back to the output text $x' = v_0, v_1, \ldots, v_n$.

Practically, in our network architecture, the pretrained BERT is employed as the encoder and LSTM as decoder to ensure a high-quality output to meet the naturality requirement. As denoted in Eq. (1), e' is a perturbed sentence embedding which is slightly different from the original embedding e. $\mathcal{D}_{LSTM}$ recovers sentence embedding e' back to a text x'.

$$\begin{aligned} e &= \mathcal{E}_{BERT}(x) \\ x' &= \mathcal{D}_{LSTM}(e') \end{aligned} \tag{1}$$

Although the seq2seq network could generate an adversarial example by directly adding noise to the embedding e, such approach often induces decoder to produce texts unnatural and semantically meaningless (Zhao et al., 2018). This is due to the embedding e integrates low-level features that are difficult to operate to directly output natural and coherent sentences through perturbations. Therefore, we propose to transform embedding e into a high-level semantic space, which noises can then be added to enable soft perturbation.

Stacked Auto-Encoder. In order to perform soft perturbation with the ultimate goal to preserve semantic similarity of original embedding e, we stack another auto-encoder into the seq2seq network as shown in the upper box of Fig. 2. This inner auto-encoder also includes two networks: an encoder E and a decoder D. Encoder E transforms the sentence embedding e to a high-level semantic embedding z. Then, decoder D transforms it back to sentence embedding e'. Eq. (2) shows the transformation.

$$\begin{aligned} z &= E_{LSTM}(e) \\ e' &= D_{LSTM}(z), \end{aligned} \tag{2}$$

where two three-layer LSTM models are employed for E and D in the stacked auto-encoder network.

Generalizations and Applications. Having a closer examination at the proposed model in Fig. 2, it is worth noting that no target model is involved in the training process. This suggests that the proposed approach is independent of any target model and can thus be applied to various target victim models. Despite the proposed approach belongs to black-box attack, which requires to test whether the generated examples can successfully fool a victim model, experiments as to be introduced later show that the generated examples can perform adversarial attacks to various victim models with high success rate. The application of the proposed method can also be extend to other types of NLP tasks, e.g., toxicity detection (Fan et al., 2021) and hate speech detection (Aldjanabi et al., 2021), since it is an end-to-end text generation framework. This indicates the encoder $\mathcal{E}$ and decoder $\mathcal{D}$ are only required to re-train to fit for different tasks, though we only focus on text classification and textual entailment tasks in this paper.

3.2. Model training

To train our SSAE network, the training phase is guided by the loss function of two specified auto-encoders. For the outer seq2seq network which aims to preserve semantic naturality and similarity, the loss can be defined as its reconstruction error towards original text:

$$\mathcal{L}_{S2S} = \mathbb{E}_{x\sim p(x)} \| D(\mathcal{E}(x)) - x \|. \tag{3}$$

where $p(x)$ denotes the distribution of training data. For the inner stacked auto-encoder (SAE), which intends to map an embedding to a high-level semantic vector, the loss is therefore defined as the reconstruction error between the input embedding e and the output embedding e', as denoted in the following equation:

$$\mathcal{L}_{SAE} = \mathbb{E}_{x\sim p(x)} \| D(E(\mathcal{E}(x))) - \mathcal{E}(x) \|. \tag{4}$$

Thus, the total loss function of SSAE is represented as the sum of seq2seq loss and SAE loss. Our aims to identify the parameter set that minimizes the overall loss as follows:

$$\underset{\Theta}{\operatorname{argmin}} \left(\mathcal{L}_{S2S} + \mathcal{L}_{SAE} \right), \tag{5}$$

where Θ denotes the parameters across all networks.

Specifically, there are four networks to train, namely, $\mathcal{E}$, $\mathcal{D}$, E and D. In general, we have two training strategies: *Pre-train* and *No Pre-train*. The *No Pre-train* strategy simultaneously trains all networks in SSAE and update parameters in a single process. Instead, *Pre-train* is a two-stage strategy. The seq2seq network is trained individually using the original dataset X by Eq. (3). Then the inner stacked auto-encoder network is trained, by fixing the seq2seq model parameters.

In this paper, we adopt the *Pre-train* strategy. For each mini-batch data, we alternatively train SSAE network as follows. (1) **Train network E, D by fixing $\mathcal{E}$ and $\mathcal{D}$.** When $\mathcal{E}$ and $\mathcal{D}$ are fixed, the text embedding e, e', and the generated example x' can be derived by forward computation. The SAE loss can then be computed by Eq. (4) while updating parameters for E and D. (2) **Fine-tune $\mathcal{E}$, $\mathcal{D}$ by fixing E and D.** In this step the seq2seq network is fine tuned guided by Eq. (3) while updating parameters of encoder $\mathcal{E}$ and decoder $\mathcal{D}$ respectively. It is expected that the parameters in Θ converges in a number of training epochs.

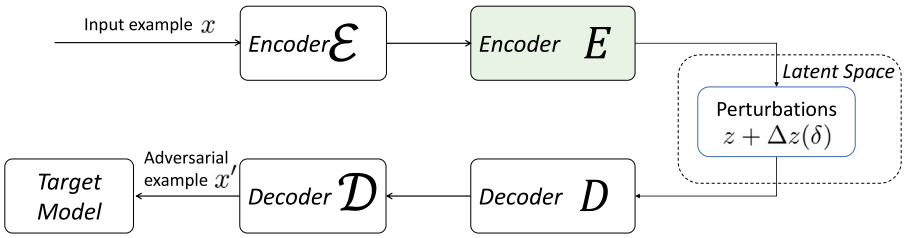


Fig. 3. The attack process of our method.

3.3. Generating adversarial examples

Once the training phase completed, the proposed network enables to efficiently generate adversarial examples directly through performing the network inference. We show this process in Fig. 3, where the input is an original example x and the output is an adversarial example x' to attack the target model. $\mathcal{E}$, D are the encoder and decoder of the seq2seq network of SSAE, and E, D are the encoder and decoder of the stacked auto-encoder network of SSAE. Perturbations are added on high-level semantic vector z in the latent space.

Specifically, given an original input example x, an adversarial example x' is generated by feedforward inference, sequentially going through $\mathcal{E}$, E, D, and D. To further enhance the attack success rate of adversarial examples, we try to bring sufficient adversarial attributes into the latent vector z. Unlike previous study (Zhao et al., 2018) to search a noise, we **randomly** sample a noise $\Delta z(\delta)$ within a hypersphere, where δ denotes the radius. The noise follows the standard normal distribution. Next, we add the noise to z in the latent space as:

$$z' = z + \Delta z(\delta). \tag{6}$$

The perturbed examples z' are then decoded back to sentences via decoder D. We can generate a batch of perturbed examples and then verify whether they could attack successful, given the target victim model. Pseudocode of the generation process is shown in Algorithm 1. For more details, please refer to our source code.

Algorithm 1: Pseudocode for generating adversarial examples

Input: Original sample x, ground-truth label y, noise radius δ, target victim model T, batch size m, max step size t
Output: Adversarial example x'

```
s ← 1;
while s ≤ t do
    for i = 1 to m do
        z ← E(E(x)) ;            // Project x to semantic embedding z.
        z' ← z + Δz(δ) ;         // Add perturbation.
        x' ← D(D(z')) ;          // Decode embedding z' to sentence.
        if T(x') ≠ y then        // Attack successfully.
            return x';
        end
    end
    δ ← δ × 2 ;                  // Increase the noise radius.
end
```

4. Experimental settings

This section introduces the datasets, baseline methods, configurations of the models, and the metrics used for evaluating model performance.

Table 1
The summary of the datasets used.

Dataset	Task	#Class	Avg. SL	#Train	#Test
IMDB	Sentiment analysis	2	230	25 000	25 000
SST-2	Sentiment analysis	2	15	7393	1749
SNLI	Natural language inference	3	10	549 367	9824

4.1. Datasets

We test our SSAE on two tasks: sentiment analysis and natural language inference. For the sentiment analysis task, two representing binary classification datasets are employed: the SST-2 (Socher et al., 2013) and IMDB (Maas et al., 2011). IMDB has longer sentence length in average than SST-2 and is considered more challenging. For the natural language inference task, the Stanford Natural Language Interface (SNLI) dataset (Bowman et al., 2015) is adopted. Each given instance in SNLI has a premise and a hypothesis, with the goal to predict whether the hypothesis is entailment, neural, or contradiction to the premise. The brief summary of these datasets is listed in Table 1, where #Class denotes the number of decision classes, and Avg. SL is the average sentence length in words. #Train and #Test denote the number of training set and test set, respectively.

4.2. Victim models

Three recent and significant classification models are employed as target victim models: BERT (Devlin et al., 2019), LSTM (Howard & Ruder, 2018), and TextCNN (Liang et al., 2018). TextCNN and LSTM are basic deep learning approaches. BERT is an prevalent pretrained language model, and has a better performance in downstream tasks. Although there are many variations of BERT, such as RpBERT (Sun et al.) and Finbert (Liu et al.), we consider the common fine-tuned BERT as our victim model.

4.3. Baseline methods

In order to demonstrate the superiority of the proposed approach, two recent word-level attack methods and a sentence-level attack method are utilized as the comparative baselines. In particular, the word-level baselines are PWWS (Ren et al., 2019) and the state-of-the-art open-source adversarial attack PSO (Zang et al., 2020). They belong to the conventional family of search-based methods that generate adversarial examples through synonym substitution. As for the sentence-level baseline, the GNAE (Zhao et al., 2018) is to generate grammatically and linguistically coherent adversarial sentences via a GAN-based method.

4.4. Model configurations

For the proposed SSAE model, the seq2seq encoder $\mathcal{E}$ is a fine-tuned base-uncased BERT, which has 12-layers, 12 attention heads and 768-dimensional hidden nodes. Across all datasets, the uncased vocabulary of standard BERT with 30,522 words is consistently used.

Table 2
Hyper-parameter settings for victim models.

Dataset	BERT		LSTM			TEXTCNN	
	#Linear layers	ACC	Hidden cell dimension	#Hidden layers	ACC	Kernel size	ACC
IMDB	1	0.91652	128	2	0.86532	[50,50,50]	0.86736
SST-2	2	0.88222	128	2	0.80503	[30,40,50]	0.79646
SNLI	3	0.86136	300	2	0.74338	[30,40,50,60]	0.74287

Table 3
Comparison with word-level baselines.

Methods	IMDB + BERT		SST-2 + BERT		SST-2 + LSTM		SST-2 + TEXTCNN	
	Success rate	Attack time	Success rate	Attack time	Success rate	Attack time	Success rate	Attack time
PSO	0.9310	77 005.5	0.9178	6246.3	0.8784	960.9	–	–
PWWS	0.9136	13 949.7	0.8807	702.6	0.9389	127.2	0.9390	**81.2**
SSAE	**0.9568**	**124.8**	**0.9413**	**114.2**	**0.9616**	**80.9**	**0.9578**	100.9

Table 4
Success rate on IMDB datasets with different bounds (#Samples = 100).

Target model	Metric	Bound					
		0	0.1	0.2	0.3	0.4	0.5
LSTM	Success rate	0.0535	0.3731	0.6667	0.8817	0.9818	0.9977
	Bert-score	0.7858	0.7445	0.6522	0.5875	0.5577	0.5430
	PPL	137.3	159.2	217.8	245.8	238.5	213.3
BERT	Success rate	0.0638	0.6065	0.9568	1	1	1
	Bert-score	0.8480	0.7818	0.6563	0.5912	0.5625	0.5458
	PPL	152.1	199.7	288.4	277.7	235.4	182.2
TextCNN	Success rate	0.0559	0.5222	0.8643	0.9852	0.9989	1
	Bert-score	0.8217	0.7588	0.6318	0.5667	0.5421	0.5301
	PPL	126.2	158.4	243.5	254.8	225.7	180.3

Table 5
Success rate on SST2 datasets with different bounds (#Samples = 100).

Target model	Metric	Bound					
		0	0.1	0.2	0.3	0.4	0.5
LSTM	Success rate	0.3072	0.6640	0.8320	0.9040	0.9408	0.9616
	Bert-score	0.5676	0.5649	0.5511	0.5371	0.5287	0.5181
	PPL	2580.1	2560.4	3057.8	3135.1	3509.6	3507.7
BERT	Success rate	0.2071	0.7054	0.8671	0.9413	0.9756	0.9911
	Bert-score	0.6598	0.6469	0.6240	0.6012	0.5751	0.5554
	PPL	2223.0	2311.0	2682.2	2571.6	2736.3	2770.3
TextCNN	Success rate	0.2455	0.6202	0.7749	0.8657	0.9233	0.9578
	Bert-score	0.5632	0.5600	0.5516	0.5353	0.5275	0.5124
	PPL	2331.9	2382.1	2814.9	3190.8	3219.7	3725.5

The seq2seq decoder D is a LSTM network containing 3 hidden layers with 768-dimensional hidden nodes and a fully connected layer. The inner stacked encoder–decoder network consists of two LSTM networks with 3 hidden layers, and each of them has 500-dimensional hidden nodes. All layers of the SSAE model adopt 0.3 dropout rate. We start by training $\mathcal{E}$ and D on the training corpus for seq2seq model, and then fix their parameters to train encoder E and decoder D according to Eq. (4). For the baselines, we follow the recommended settings (Wang & Wang, 2020; Zang et al., 2020). For victim models, the key settings and prediction accuracy (ACC) are shown in Table 2.

4.5. Hyper-parameter settings

To observe the attack performance of our SSAE model under different hyper-parameter settings, the perturbation bound δ is varied from 0 to 0.5. Considering the proposed model can generate a set of examples given just one input instance, the number of generated samples (#Samples) is varied from 20 to 5000. In case more than one valid adversarial samples among all generated ones, the one with least $\Delta z(\delta)$ is selected as the best adversarial example. All the experiments were performed on a Ubuntu server with an i9-9900K CPU and a Tesla v100 GPU.

Table 6
Success rate on three datasets with different #Samples (bound = 0.2).

Dataset	Target model	#Samples				
		20	100	1000	2000	5000
IMDB	LSTM	0.4266	0.6667	0.8476	0.8794	0.9147
	BERT	0.8422	0.9568	1	1	1
	TextCNN	0.6454	0.8643	0.9749	0.9897	0.9932
SST2	LSTM	0.7120	0.8320	0.9232	0.9392	0.9520
	BERY	0.7276	0.8671	0.9579	0.9712	0.9942
	TextCNN	0.6368	0.7749	0.8721	0.9015	0.9105
SNLI	LSTM	0.6158	0.7141	0.8196	0.8578	0.8768
	BERT	0.6878	0.8153	0.8963	0.9178	0.9392
	Bert-no-seq	0.7890	0.8939	0.9666	0.9833	0.9857

Table 7
Attack Time on three datasets with different #Samples (bound = 0.2, unit: second).

Dataset	Target model	#Samples				
		20	100	1000	2000	5000
IMDB	LSTM	52.41	118.72	2146.06	4973.80	12852.81
	Bert	56.09	124.84	2254.37	8836.55	24124.57
	TextCNN	54.45	133.23	2048.89	4939.91	12534.64
SST2	LSTM	34.59	87.08	1253.76	2558.64	6470.85
	Bert	45.84	114.24	2173.85	8726.03	23875.28
	TextCNN	43.06	100.94	1550.687	3184.28	7903.66
SNLI	LSTM	3.35	6.89	46.07	89.21	200.86
	Bert	4.03	8.74	55.09	105.21	273.94
	Bert-no-seq	3.95	8.46	56.58	108.47	273.75

Table 8
Effect of training strategy on 'SNLI + BERT'. (#Samples = 100).

SSAE	Metric	Bound					
		0	0.1	0.2	0.3	0.4	0.5
Pre-train	Success rate	**0.1216**	0.5375	0.8153	0.9022	0.9178	**0.9404**
	Bert-score	**0.8191**	**0.7866**	**0.7071**	**0.6248**	**0.5639**	**0.5303**
	PPL	664.5	**828.6**	**1306.2**	**1877.3**	**2361.4**	**2378.1**
No Pre-train	Success rate	0.1013	**0.7199**	**0.8939**	**0.9273**	**0.9511**	**0.9404**
	Bert-score	0.8152	0.7471	0.6262	0.5485	0.5092	0.4895
	PPL	**652.5**	963.6	1750.4	2166.1	2459.6	2396.6

4.6. Evaluation metrics

Following on the typical evaluations such as Zang et al. (2020), all models are assessed on 1000 correctly classified instances randomly sampled from the test sets of three datasets. For the textual entailment dataset SNLI, only the hypothesis are perturbed.

In particular, the *attack success rate* and *attack time* are used to evaluate the attack performance, while two common metrics *Bert-score* and *Perplexity (PPL)* are adopted to evaluate the quality of generated adversarial examples. More specifically, the *attack success rate* is defined

Table 9
SSAE Transferability on 'SST-2 + BERT' (#Samples = 20).

Train	Metric	Bound					
		0	0.1	0.2	0.3	0.4	0.5
SST-2	Success rate	**0.2071**	0.7054	0.8671	0.9413	0.9756	0.9911
	Bert-score	0.6598	**0.6469**	**0.6240**	**0.6012**	**0.5751**	**0.5554**
	PPL	2223.0	2311.0	2682.2	2571.6	2736.3	2770.3
IMDB	Success rate	0.1306	**0.7364**	**0.9701**	**0.9978**	**1**	**1**
	Bert-score	**0.6625**	0.5722	0.4736	0.4341	0.4089	0.3872
	PPL	**1432.9**	**1648.8**	**1783.7**	**1424.3**	**983.8**	**481.4**

as the number of successfully attacked examples crafted by attack models against the number of correctly predicted examples with no attack. The *attack time* is measured by the average time in seconds of generating one adversarial sample to successfully fool the victim model. For the SSAE model, the attack time includes *generating time* and *finding time*. The *generating time* denotes generating a batch of examples. Then the generated examples are tested one by one until identifying a valid adversarial example, the duration of which is considered as *finding time*.

Furthermore, *Bert-Score* (Zhang et al., 2020) computes a similarity score between the candidate sentence and the reference sentence using BERT. It correlates words better over human judgment and provides stronger model selection performance than existing metrics. The range of *Bert-Score* varies from −1 to 1, with a larger value suggesting a higher similarity of an adversarial sample to the original text. On the other hand, *PPL* measures the fluency of adversarial samples through GPT-2 (Radford et al., 2019), with larger values indicating poorer sentence fluency.

5. Experimental results and discussions

This section reports the experimental results with detailed discussions. We first demonstrate model performances at both word-level and sentence-level. Then, the impact of hyper-parameters and training strategy is explored, followed by the assessment of model transferability. Finally, some text snippet examples are given to directly show the quality of the generated adversarial examples.

5.1. Comparison with word-level baselines

The comparison with word-level baselines is measured on attack success rate and attack time. Considering the average sentence length and the robustness of victim models vary significantly across different datasets, we set the hyperparameter **bound** to {0.2, 0.3, 0.4, 0.5}, and the hyperparameter **#Samples** to 100. Table 3 presents the results on the attack success rate and attack time. It is observed that the TextCNN was not selected as the victim model in PSO (denoted as '–' in the table), due to the open source code of PSO does not consider TextCNN. As highlighted in bold, the experimental results show that the proposed SSAE achieves the highest attack success rate, and outperforms word-level baselines from 2% to 9% on two datasets and three victim models.

We conduct paired t-test to verify whether the improvements are significant. For the attack success rate, the p-value of SSAE against PSO is: $p < 0.005$, and the p-value of SSAE against PWWS is: $p < 0.0001$. Since the p-values are small, we can safely say that the performance our SSAE outperform word-level baselines significantly on attack success rate. This is because SSAE can generate a large number of candidate examples fast. As a result, it is easy to find out an adversarial example in a limited time.

It is observed that SSAE also achieves the least attack time except 'SST2 + BERT'. For the IMDB dataset, which with longer sentences (the average sentence length is 230), the attack time of SSAE is about 100 times faster than PWWS, and about 700 times faster than PSO. For the SST-2 dataset, which has shorter sentences (the average sentence length is 15), the attack time of SSAE is better under victim model BERT and LSTM. But it is worse under TEXTCNN model. This is because the baselines cost too much time on querying the victim model to find out an adversarial example. The attack time of word-level baseline method reduces exponentially with sentence length decrease. But since our SSAE generates adversarial samples through network inference, the attack time remains stable. To this end, the overall performance of SSAE results in significant reduction in the attack running time.

5.2. Comparison with sentence-level baseline

We perform experiments on sentence-level attack methods through the 'SNLI + LSTM' settings. It is worth noting that we could not directly compare GANE with SSAE for different bounds, as the parameter bounds in the two models have different meanings and specifications. Instead, we vary the bounds to derive a set of pair values: (Success Rate, Bert-Score) and (Success Rate, PPL). We aim to observe the Bert-Score and PPL measurements at the same Success Rate. Fig. 4 demonstrates the relationships of Bert-Score, PPL and Success Rate for GANE and SSAE. It is observed that given the same attack success rate, SSAE achieves higher Bert-Score and lower PPL, indicating that the samples our SSAE generates are more fluent in semantics, and higher similarities with original samples.

Specifically, in Fig. 4(a), the Bert-Score of SSAE is superior to GNAE in each attack success rate. This is because SSAE includes the reconstruction error as the loss function. The reconstruction error loss effectively guarantees high semantic similarity between the generated adversarial samples and the original samples. Similarly, Fig. 4(b) shows that SSAE outperforms GNAE on PPL, especially at high attack success rate, suggesting that the proposed network can generate more natural and fluent sentences than GANE. The reason is that small noises are added in the semantic embeddings, and the decoder of outer Seq2seq module in SSAE guarantees the quality of text generations. We further conduct paired t-test on Bert-score and PPL. For Bert-score, paired t-test gives p-value 4.31×10^{-9} for the difference between our SSAE and GANE. And for PPL, the p-value is 0.007. The t-tests on both metrics indicates that the difference is significant (<0.05).

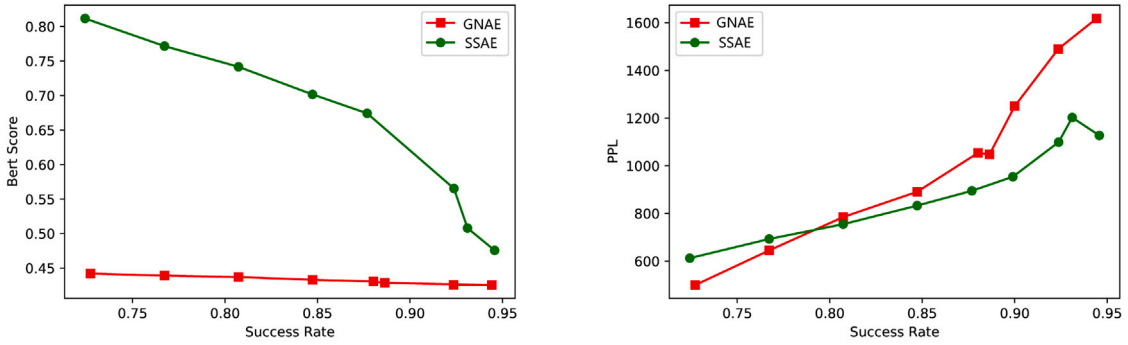
5.3. Impact of hyper-parameters

To test the robustness of the proposed method, the influence of bound and #Samples is further evaluated, by varying the bound in {0, 0.1, 0.2, 0.3, 0.4, 0.5} and #Samples in {20, 100, 1000, 2000, 5000}. Tables 4 and 5 show the impact of bound on the Success Rate, Bert-Score and PPL on the IMDB and SST2 datasets when the #Samples is fixed at 100. Results show that as the bound increases, the Success Rate and PPL increase, but the Bert-Score decreases. These observations indicate that a large bound will lead to good attack performance, but poor generation quality. We should take a balance among Success Rate, Bert-Score and PPL when we select the parameters.

Next, we fix bound at 0.2. The impact of #Samples to Success Rate and Attack Time on three datasets are shown in Tables 6 and 7. We have the following findings: (1) increasing the #Sample leads to an initial boost of the attack Success Rate, and the rate eventually get stable; and (2) attack time rises linearly as the number of sample increases, suggesting that a large number of samples will yield good attack performance, but long attack time. To balance the attack performance, generation quality and attack time, the appropriate bound is selected from {0.2, 0.3, 0.4, 0.5}, with the #Samples from {20, 100}.

Table 10
Illustrative examples, where '**o:**' denotes the original text; '**g:**' denotes the generated adversarial text; and '**h:**' denotes hypothesis in text entailment task for SNLI.

Dataset	Cases	Label
IMDB	**o:** ..... i do wonder why hitchcock never used doris again . at first glance she would fit the profile of blond leading ladies that hitchcock favored . possibly because her wholesome screen image was at odds with the sophistication hitchcock also wanted in his blondes .	Positive
	g: ..... i still wonder why madonna didn gave him again . he first glance the would be the tone of blonde actresses women that her directed . perhaps because her imageness look age was the of with the sensualphinessation in employed in the femmes .	Negative
	o: he has a vicious side to him that is totally unlike a stephen king . he freely mixes in his own homosexuality and odd religious and occultic elements . i love love love love it . i also realize , however , that barker is as much a dark fantasy writer as he is a horror writer	Positive
	g: he has a kind way to him that was almost an a john hitchcock . he uses his his own self and evil supernatural and occultal elements . i love love love love it . i also realize that however , that he is as much a dark fantasy actor than he is a romantic writer	Negative
SST-2	**o:** the film makes a clear look of the cost of the characters in the holocaust story	Positive
	g: the film makes a strong case for the importance of the musicians in creating the motown sound	Negative
	o: a synthesis of persistence and mayhem that is devoid perfect in its own pre from the death of the man ' s hard heart	Negative
	g: a tale of horror and revenge that is nearly perfect in its relentless descent to the depths of one man ' s tortured soul	Positive
SNLI	**h:** A couple is walking together .	
	o: a couple walk hand in hand down a street .	Entailment
	g: a couple walk arm in hand down a street .	Neutral
	h: A man is performing for cash .	
	o: a man playing an electric guitar on stage .	Neutral
	g: a woman playing a electric guitar on stage .	Contradiction
	h: A man is deciding which bike to buy	
	o: a man in a black shirt is looking at a bike in a workshop .	Neutral
	g: a man in a red shirt is looking at a bike in a garage .	Entailment
	h: There is a man tying his shoes .	
	o: a woman is painting a mural of a woman s face .	Contradiction
	g: a woman is painting a picture on a man s face .	Entailment



(a) Bert Score VS. Success Rate (b) PPL VS. Success Rate

Fig. 4. Comparison with sentence-level baseline.

5.4. Impact of training strategy

We test the performance of two training strategies for the SSAE model, i.e., *Pre-train* and *No Pre-train*. The key difference of two strategies lies in whether to pre-train the outer seq2seq network $\mathcal{E}$ and $\mathcal{D}$. Considering SNLI as dataset and BERT as victim model, the results are shown in Table 8.

It is observed that the *No Pre-train* strategy has a larger attack Success Rate than the *Pre-train* strategy. Yet, in terms of the Bert-Score and PPL, *Pre-train* strategy outperforms *No Pre-train* strategy for most bound settings. This indicates that *Pre-train* strategy may generate more natural sentences with higher similarity to original examples. We suggest to adopt the *Pre-train* strategy in the experiments.

5.5. Transferability

As a widely used metric on evaluating the effectiveness of adversarial attack methods, transferability of adversarial examples refers to their ability in fooling a DNN model without any access to it (Kurakin et al., 2017). Tables 4 and 5 suggest that the adversarial examples generated by our SSAE are effective to all victim models, i.e., LSTM, BERT and TEXTCNN, simultaneously. This is because the proposed SSAE network is trained on the same dataset, and is not constrained to any specialized victim models. Therefore, we can safely say that SSAE naturally have transferability among different victim models.

Furthermore, we discuss the transferability of SSAE across different datasets. We first train our SSAE model on IMDB dataset. Then, from this SSAE model we generate adversarial examples on the test set of

SST-2 dataset, and attack BERT trained on the SST-2 dataset. From Table 9 we can see that the generated adversarial examples are also effective, since most Success Rate values of SSAE trained on IMDB are higher than the SSAE trained on SST-2 dataset. We can also find that the transferred PPL values are higher than SSAE trained on SST-2. This is because SSAE trained on IMDB is inclined to generate long sentence. But the Bert-Score values are lower than SSAE trained on SST-2, suggesting that the transferred samples have low similarity to the original samples.

5.6. Analysis and discussion

From the experimental results, the performance of our SSAE network is completely illustrated. Compare with state-of-the-art sentence-level attack method, our method could generate more natural and similar adversarial text, under the same success rate. Compare with word-level attack methods, our work shows significant better performance on attack time. For attack success rate, our approach is still better. As the noise radius δ becomes larger, the success rate of our approach becomes higher, but the performance on text naturality and similarity would get worse. Therefore, SSAE is a comprehensive trade-off considering attack success rate, attack time, and naturality. SSAE also shows strong transferability, since it does not need to query any target victim model in the training process.

5.7. Case study

In order to demonstrate the quality of generated sentences by the proposed SSAE, a number of cases are presented in Table 10, where the substitutions of both the original and generated adversarial texts are highlighted in blue and red respectively. From the table we have the following observations and findings: (1) The proposed SSAE network can generate adversarial texts that are semantically coherent and neural; (2) The generated adversarial texts can not only substitute some words individually, but also at the level of phrases or sub-sentences, e.g., 'a mural of' is replaced by 'a picture on'; and (3) In the long sentence cases such as the IMDB dataset, the adversarial texts substitute a continuous set of words on the original texts. Although this results in a lower Bert-Score for the sentence, it is not easy for people to perceive the changes since the overall text is long.

6. Conclusion and limitations

In this paper, we have proposed SSAE, a novel neural network to generate adversarial text examples at sentence-level. SSAE levarages an end-to-end Seq2seq Stacked Auto-Encoder to generate semantic consistent adversarial examples efficiently through direct network inference. Experiments and discussions show the advantages of our SSAE model: (1) adversarial examples generated by SSAE have high attack success rate than word-level state-of-the-art baselines, by adding perturbations on the semantic embeddings; (2) it is more efficient than word-level baselines since network inference is very fast compared with search-based optimization; and (3) the generated adversarial examples have good qualities according to various metrics, such as Bert-Score, PPL, and transferability.

There are limitations for our SSAE network. First, it is not trivial to determine the appropriate noise radius to perturb on the semantic embeddings for different datasets and victim models. Second, for long text, the changes of many generated adversarial examples are centralized in one sentence, and thus results in lower Bert-Score. In the future work, we plan to devise a module to automatically learn the noise distributions for different datasets. Another potential line is to improve the network to generate more imperceptible adversarial examples for long text, with the changed words distributed in different sentences.

CRediT authorship contribution statement

Ang Li: Methodology, Software, Writing – original draft. **Fangyuan Zhang:** Formal analysis, Data curation. **Shuangjiao Li:** Validation, Visualization. **Tianhua Chen:** Supervision. **Pan Su:** Investigation. **Hongtao Wang:** Conceptualization, Funding acquisition, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Code is availabel at https://github.com/Leon-Francis/SSAE.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant No. 61802124), and the Fundamental Research Funds for the Central Universities, China (Grant No. 2019MS126).

References

Aldjanabi, W., Dahou, A., Al-qaness, M. A. A., Elaziz, M. E. A., Helmi, A. M., & Damasevicius, R. (2021). Arabic offensive and hate speech detection using a cross-corpora multi-task learning model. *Informatics*, *8*(4), 69.

Alzantot, M., Sharma, Y., Elgohary, A., Ho, B., Srivastava, M. B., & Chang, K. (2018). Generating natural language adversarial examples. In *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 2890–2896).

Bartolo, M., Roberts, A., Welbl, J., Riedel, S., & Stenetorp, P. (2020). Beat the AI: investigating adversarial human annotation for reading comprehension. *Transactions on Assocations and Computational Linguistics*, *8*, 662–678.

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 632–642).

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies* (pp. 4171–4186).

Ebrahimi, J., Lowd, D., & Dou, D. (2018). On adversarial examples for character-level neural machine translation. In *Proceedings of the 27th international conference on computational linguistics* (pp. 653–663).

Fan, H., Du, W., Dahou, A., Ewees, A. A., Yousri, D., Elaziz, M. A., Elsheikh, A. H., Abualigah, L., & Al-qaness, M. A. A. (2021). Social media toxicity classification using deep learning: Real-world application UK brexit. *Electronics*, *10*(11).

Gao, J., Lanchantin, J., Soffa, M. L., & Qi, Y. (2018). Black-Box generation of adversarial text sequences to evade deep learning classifiers. In *Proceedings of the 2018 IEEE security and privacy workshops* (pp. 50–56).

Garg, S., & Ramakrishnan, G. (2020). BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 conference on empirical methods in natural language processing* (pp. 6174–6181).

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th annual meeting of the association for computational linguistics* (pp. 328–339).

Iyyer, M., Wieting, J., Gimpel, K., & Zettlemoyer, L. (2018). Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: human language technologies* (pp. 1875–1885).

Jin, D., Jin, Z., Zhou, J. T., & Szolovits, P. (2020). Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *Proceedings of the thirty-fourth AAAI conference on artificial intelligence* (pp. 8018–8025).

Kurakin, A., Goodfellow, I. J., & Bengio, S. (2017). Adversarial examples in the physical world. In *Proceedings of the 5th international conference on learning representations, workshop track*.

Li, J., Ji, S., Du, T., Li, B., & Wang, T. (2019). TextBugger: Generating adversarial text against real-world applications. In *Proceedings of the 26th annual network and distributed system security symposium*.

Li, L., Ma, R., Guo, Q., Xue, X., & Qiu, X. (2020). BERT-ATTACK: adversarial attack against BERT using BERT. In *Proceedings of the 2020 conference on empirical methods in natural language processing* (pp. 6193–6202).

Liang, B., Li, H., Su, M., Bian, P., Li, X., & Shi, W. (2018). Deep text classification can be fooled. In *Proceedings of the twenty-seventh international joint conference on artificial intelligence* (pp. 4208–4215).

Liu, Z., Huang, D., Huang, K., Li, Z., & Zhao, J. FinBERT: A pre-trained financial language representation model for financial text mining. In C. Bessiere (Ed.), *Proceedings of the twenty-ninth international joint conference on artificial intelligence, IJCAI 2020* (pp. 4513–4519).

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies* (pp. 142–150).

Papernot, N., McDaniel, P. D., Swami, A., & Harang, R. E. (2016). Crafting adversarial input sequences for recurrent neural networks. In *Proceedings of the 2016 IEEE military communications conference* (pp. 49–54).

Pruthi, D., Dhingra, B., & Lipton, Z. C. (2019). Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th conference of the association for computational linguistics* (pp. 5582–5591).

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, *1*(8), 9.

Ren, S., Deng, Y., He, K., & Che, W. (2019). Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th conference of the association for computational linguistics* (pp. 1085–1097).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th annual meeting of the association for computational linguistics* (pp. 856–865).

Sato, M., Suzuki, J., Shindo, H., & Matsumoto, Y. (2018). Interpretable adversarial perturbation in input embedding space for text. In *Proceedings of the twenty-seventh international joint conference on artificial intelligence* (pp. 4323–4330).

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1631–1642).

Sun, L., Wang, J., Zhang, K., Su, Y., & Weng, F. RpBERT: A text-image relation propagation-based BERT model for multimodal NER. In *Thirty-fifth AAAI conference on artificial intelligence, AAAI 2021, virtual event, February 2-9,2021* (pp. 13860–13868).

Wallace, E., Rodriguez, P., Feng, S., Yamada, I., & Boyd-Graber, J. L. (2019). Trick me if you can: Human-in-the-loop generation of adversarial question answering examples. *Transactions on Assocation for Computational Linguistics*, *7*, 387–401.

Wang, Z., & Wang, H. (2020). Defense of word-level adversarial attacks via random substitution encoding. In *Proceedings of the 13th international conference on knowledge science, engineering and management* (pp. 312–324).

Yu, H., Luo, H., Yi, Y., & Cheng, F. (2021). A2R2: robust unsupervised neural machine translation with adversarial attack and regularization on representations. *IEEE Access*, *9*, 19990–19998.

Zang, Y., Qi, F., Yang, C., Liu, Z., Zhang, M., Liu, Q., & Sun, M. (2020). Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 6066–6080).

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2020). BERTScore: Evaluating text generation with BERT. In *Proceedings of the 8th international conference on learning representations*.

Zhang, H., Zhou, H., Miao, N., & Li, L. (2019). Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th conference of the association for computational linguistics* (pp. 5564–5569).

Zhao, Z., Dua, D., & Singh, S. (2018). Generating natural adversarial examples. In *Proceedings of the 6th international conference on learning representations*.