Hindawi
Security and Communication Networks
Volume 2022, Article ID 6458488, 25 pages
https://doi.org/10.1155/2022/6458488

Review Article

Text Adversarial Attacks and Defenses: Issues, Taxonomy, and Perspectives

Xu Han,[1,2] **Ying Zhang**,[2] **Wei Wang**,[1,2] **and Bin Wang**[3]

[1]*Beijing Key Laboratory of Security and Privacy, Intelligent Transportation, Beijing, China*
[2]*College of Computer and Information Technology, Beijing Jiaotong University, Beijing, China*
[3]*School of Electrical Engineering, Zhejiang University, 310058 Hangzhou, China*

Correspondence should be addressed to Wei Wang; wangwei1@bjtu.edu.cn and Bin Wang; bin_wang@zju.edu

Received 16 August 2021; Revised 5 January 2022; Accepted 19 January 2022; Published 23 April 2022

Academic Editor: Yanhui Guo

Deep neural networks (DNNs) have been widely used in many fields due to their powerful representation learning capabilities. However, they are exposed to serious threats caused by the increasing security issues. Adversarial examples were early discovered in computer vision (CV) field when the models were fooled by perturbing the original inputs, and they also exist in natural language processing (NLP) community. However, unlike the image, the text is discrete and semantic in nature, making the generation of adversarial attacks even more difficult. In this work, we provide a comprehensive overview of adversarial attacks and defenses in the textual domain. First, we introduce the pipeline of NLP, including the vector representations of text, DNN-based victim models, and a formal definition of adversarial attacks, which makes our review self-contained. Second, we propose a novel taxonomy for the existing adversarial attacks and defenses, which is fine-grained and closely aligned with practical applications. Finally, we summarize and discuss the major existing issues and further research directions of text adversarial attacks and defenses.

1. Introduction

Regarded as stacked neural networks focusing on emulating the learning approach of human beings, DNNs have been widely studied in the past decade [1]. The ability to autoextract high-dimensional features from massive sensor data makes DNN valuable tools for multiple NLP tasks, such as text classification (TC) [2], natural language generation (NLG) [3], and question answering (QA) [4] in recent years. Nevertheless, DNNs could be tricked by adding some small and subtle perturbations to the original input. These perturbations, known as adversarial examples, were first discovered by Szegedy [5] in image classification tasks. They also exist in many other fields, including almost every subtask in the textual domains. As an example in sentiment analysis (Figure 1), an attacker can fool the system by changing only one word from "Perfect" to "Spotless," which can completely change the predicted output without being discerned by humans. The popularity of DNNs in everyday life has inevitably raised concerns about their security, making textual adversarial attacks and defense an important issue. In the meantime, the study of adversarial attacks offers referential suggestions to the interpretability of DNNs and inspires researchers to design algorithms with high robustness in general [6].

Many methods of generating perturbations have been proposed since Jia and Liang [7]. Despite the interest in this topic in the NLP community, previous reviews of textual confrontation are generally inadequate due to a lack of systematicity and depth in classification. In this work, we introduce the current challenges in textual adversarial attacks and defenses. We also propose a taxonomy of these adversarial attacks and defenses and make further discussion by providing the perspectives of adversarial attacks and defenses in NLP.

1.1. Issues

1.1.1. Issues of the Generation of Adversarial Attacks. Though adversarial attacks are originated and developed relatively into maturity in the computer vision community,

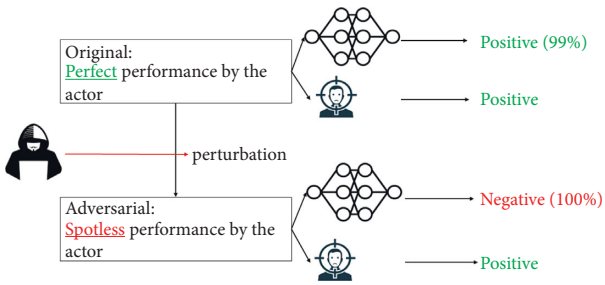

Figure 1: Text adversarial attacks in sentiment analysis.

these methods are inapplicable to the textual domain directly due to three main reasons as follows.

(i) *Discrete vs Continuous*. Unlike image data, the text is inherently discrete. If we want to introduce the methods from the image domain for measuring the distance between the perturbed examples and the original data, e.g., Lp norm, We need to vectorize the text first (the specific method is described in Section 2.1). Otherwise, we need to carefully design the way to compute the perturbation of such discrete text data, which is important for the efficiency of adversarial example generation and quality assessment.

(ii) *Semantic-Intensive vs Semantic-Dense*. The semantics of an image are more sparse than those of a sentence. Moreover, individual pixel points do not describe the overall semantics of the image and are not even sufficient to convey local information. For example, in the transformation of turning a panda into a gibbon, changing the color of just one piece of the panda's fur or even adding noise to the entire image to offer the resolution would not interfere with the human's semantic understanding of the original image, but it is not the same for text. Although the smallest unit of text varies from language to language, its smallest unit itself contains a wealth of semantic information; changing a punctuation mark may subvert the tone, and this semantic density greatly increases the difficulty of evasion attacks.

(iii) *Perceivable vs Imperceivable*. More importantly, the human eyes are robust to local image perturbations. For example, a photograph, even if partially blurred, does not affect one's understanding of the semantics. Text is truly in the opposite case. Text is abstract information, and human processing of such information is so delicate that even minor perturbations can be easily detected. Not only do typos and improper collocations draw attention to themselves but they cannot even bypass increasingly sophisticated spell checker and grammar checker systems.

These differences make it extraordinarily difficult for researchers to employ methods dealing with images to adversarial attacks. Moreover, one of the first tasks of NLP models is to work on real data to check their generalization ability. Although adversarial attacks are a practical approach to evaluate robustness, most of them have the problem of being task-specific, not being well generalized, and not presenting comprehensive guidelines for evaluating system robustness.

1.1.2. Issues of Defense. Compared with the current adversarial attacks with various methods and effects, there is quite little existing work on defense strategies. For defenders, as much as they would like to completely remedy the weaknesses of DNNs, it is no less challenging than redesigning the network architecture. The progress and shortcomings of existing related work can be summarized in the following three categories.

(i) *Ineffectiveness to Unknown Attacks*. To defend against adversarial examples, adversarial training (AT) is a practical and feasible method. Data augmentation, model regularization, and robust optimization are three common methods using the generated adversarial examples. However, a key drawback of this approach is that they assume that the threat model is known, which is often difficult to implement in practice. This leads to this defense method being less effective in the face of unknown attacks.

(ii) *Requirement of Strong Assumptions*. Randomized smoothing and interval bound propagation (IBP) are two common methods to computing robust lower bounds, which is a promising direction for certifying the robustness of the model. However, the former is again a daunting challenge to ensure the integrity of the synonym set to adversarial perturbations under the tight robustness guarantee l_2 norm; the latter requires strong assumptions about the model structure, so it is hard to fit into the recently popular transformer-based language models.

(iii) *Lack of New Defensive Methods*. Robust encoding is also a defensive approach that strengthens the input to the model. Although the existing spell checker can preprocess the input, its ability to map the typos back to the original input is limited. The current immature development of robust coding is an opportunity, considering the trend of NLP systems to generalize the capabilities of models as much as possible, rather than limiting them to a single task.

1.2. Paper Selection. Based on these challenges, we are motivated to address the following two fundamental research questions (RQs):

RQ1: what methods have been used for text adversarial attacks and defenses?

RQ2: how can the existing work on text adversarial attacks and defenses be classified?

We search keywords that include the two fields in Table 1 from security, NLP, and AI top journals and conferences (Figure 2). In addition, we also collected the latest Arxiv articles to ensure the timeliness of the review.

1.3. Contributions. In response to industry concerns, we have conducted comprehensive research and analysis of text adversarial attacks and defenses, in order to provide a reference for related researchers, practitioners, and interested parties. In addition, we hope that readers will have some foundations on the application of DNNs in NLP, since generating perturbation significantly depends on this process. In short, the contributions of this review are as follows:

(1) *Comprehensive Review*. We present a thorough review of text adversarial attacks and defenses, including general background knowledge, victim model architectures, representation approaches, pretraining tasks, perturbation measurements, generation methods, and defense strategies.

(2) *Self-Contained*. We provide all the relevant information in order to render this survey self-contained so that the readers without enough knowledge of NLP can easily understand the relationship between the NLP pipeline and the attack methods.

(3) *Fine-Grained*. We provide a clear perspective on the state-of-the-art adversarial attacks and defenses against NLP systems. According to the in-depth investigation, we classify the minimum units for generating perturbations into char, word, sentence, and multi-level and classify the attacks into gradient, score, decision, and blind-based according to the adversary knowledge, which is the new taxonomy and most detailed survey so far.

(4) *Future Directions*. We discuss a number of open questions and identify some plausible research directions in this research.

The following is the organization of the rest of this article. We describe adversarial attacks on deep learning models in Section 2, including the classifications for adversarial attacks, text representation methods, and DNNs used in NLP. Section 3 first introduces the taxonomy of perturbation generation methods and describes the perturbation measurement methods in detail. The defense strategy is discussed in Section 4, and open issues are presented in Section 5. Section 6 contains related surveys. Section 7 draws the conclusion for this paper.

2. Overview of NLP Systems with Security Issues

Before introducing the details of adversarial attacks and defenses in NLP, first of all we will introduce how to vectorize text, in order to address the three difficult problems of text and image mentioned in the previous issues. This part lays a solid foundation for the feature extraction of deep learning models. Secondly, we will briefly introduce how mainstream deep learning models solve NLP tasks, which makes our survey self-contained. Finally, based on the above background knowledge, we will give a taxonomy of the threat models of adversarial attacks in the mainstream DNN models (Figure 3).

Table 1: Paper selection.

Research domain	Keywords
Adversarial attacks	Text adversarial, Adversarial examples, nlp, Adversarial attack, generating
Defense strategies	Robustness, robust, Verified/certified robustness, Improving robustness Detecting adversarial examples Combating/blocking adversarial

Figure 2: Word cloud of all venue names of the collected papers.

2.1. Vectorizing Textual Inputs. No matter what kind of data, they have to be vectorized before being put into DNN models. For image data, we generally represent the pixel values as vectors or matrices, which is the bottom dimension of an image. Text is fundamentally different from image. It is abstract information that must be vectorized by a special transformation in the first place. Typically, there are three approaches: (1) word-count-based encoding, (2) one-hot encoding, and (3) neural contextual encoding. It is worthy to note that the choice of representation method plays a pivotal role in the performance of NLP tasks. For example, the third approach, also known as the dense encoding, represents the semantics in each dimension, which has evolved into a new era of NLP in the form of pretrained models (PTMs) since it can learn linguistic representations by using the vast amount of unlabeled data existing in the Internet. We will introduce each of these three approaches in the next section.

2.1.1. Word-Count-Based Encoding. Harris [8] proposed the bag-of-words (BOW) model in 1954. As a simplified representation of text, the BOW model has the longest history in text vectorization. This method ignores syntax and order, treats the text as a dictionary of the total size of all occurrences of words, and counts the frequency of each word. Another word-count-based method based on the encoding method, widely used in the information retrieval neighborhood, is TF-IDF [9], short for term frequency-inverse document frequency, which aims to reflect the significance

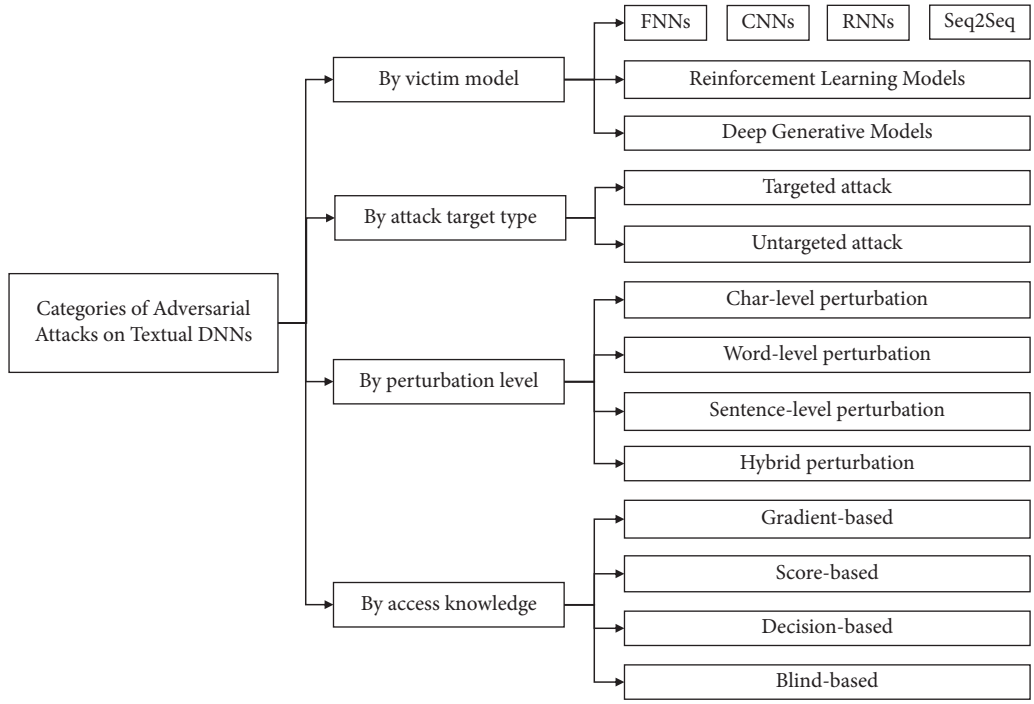

Figure 3: Categories of adversarial attacks on textual DNNs.

of the specified term in the document collection and is one of the popular term weighting schemes.

2.1.2. One-Hot Encoding. Consisting of a fixed-length set of binary bits, one-hot encoding is a vector where only one bit is a 1 and all other bits are 0 [10]. In NLP, a one-hot vector is a $1 * \mathrm{N}$-dimensional matrix/vector that can indicate the occurrence of any word/character (sometimes including symbols) in a vocabulary/alphabet. The specific mathematical definition is as follows.

$$x = \left[\left(x_{11}, \ldots, x_{1n}\right); \ldots \left(x_{m1,\ldots x_{mn}}\right)\right]. \tag{1}$$

In equation (1), for char-level one-hot encoding, let x be the input text of maximum length m, n be the maximum number of characters per word, and $x_{ij} \in \{0, 1\}^{|A|}$, where $|A|$ is the alphabet size.

$$x = \left[\left(x_1, \ldots, x_k, x_{k+1}, \ldots x_m\right)\right]. \tag{2}$$

In equation (2), for word-level one-hot encoding, let x be the input text of maximum length m and $x_{ij} \in \{0, 1\}^{|V|}$, where $|V|$ is the dictionary in the training set corpus. Thus, each word has a fixed vector length, and each sentence has a vector representation of the maximum number of words. As to sentences with words length less than m, zero padding is generally used.

Although one-hot encoding inability to represent semantic links between words and the high sparse matrix is not conducive to DNNs for representation learning, it is typically used in the training process for text preprocessing or as an intermediate variable to pass information because of its simplicity of representation and ease of design and modification.

2.1.3. Neural Contextual Encoding. As Bengio et al. [11] stated, a good representation learning representation should be able to solve generic AI problems rather than only serve for fulfilling specific tasks. It is no exception in the field of NLP. A good linguistic representation should be able to catch the language rules, world common sense, and the knowledge implied by the words in the text. This is exactly why neural contextual encoding is so prevalent today.

Unlike one-hot encoding, neural contextual encoding is also known as dense encoding or distributed representation, which is a method of representing text with a low-dimensional dense vector, where each dimension of the vector can represent a certain semantic meaning without specific meaning, and the whole vector is used to represent a specific notion. The application of generic neural structure in NLP is shown in Figure 4. Based on whether the meaning of the final word vector can be changed or not dynamically according to the context, word embeddings are further divided into two forms: non-contextual and contextual embeddings [12].

(1) *Non-Contextual Embeddings.* The smallest unit to map the input to a word vector can be char, subword, or word. Since using a word to lookup tables in a dictionary often causes the out-of-vocabulary (OOV) problem, we often use char-level representations or subword representations, such as CharCNN [13], byte-pair encoding (BPE) [14], and FastText [15]. For convenience, in the following description, we use the word as input for the mapping of distributed word vectors.

To get a distributed representation of each word, we first search the table in vocabulary V and then map it to a D-dimensional sense vector, where D is a

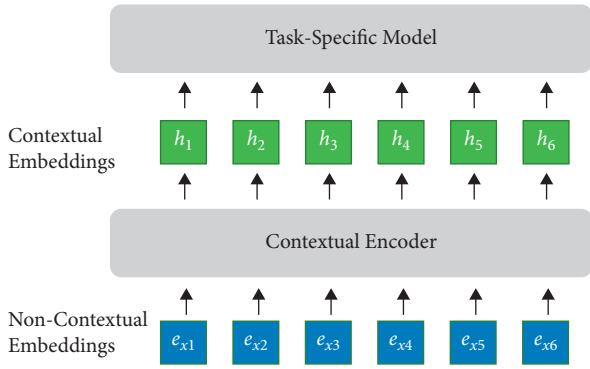

Figure 4: Generic neural architecture for NLP.

hyperparameter representing the dimensionality of the final word vector. This is usually obtained by training the language model together with other parameters of the model. A major flaw of this embedding approach is that the semantics of the word vector is static and does not take its context into account. Therefore, it is also not possible to model polysemous words. To solve this problem and to learn the dependencies between words in their contexts, contextual embedding was proposed.

(2) *Contextual Embeddings*. The neural contextual embedding can be represented by the following equation.

$$[h_1, h_2, \ldots, h_t] = F([x_1, x_2, \ldots, x_3]). \quad (3)$$

For a given input text, it consists of a total of T tokens $[x1, x2 \ldots, x^T]$, where each token can be either word or subword, and F denotes the neural encoder, which is made up of one or more DNN models in the next section. $[h1, h2, \ldots, ht]$ is the dynamic embedding also called contextual embedding of $[x1, x2, \ldots, x^T]$ because the contextual information is included in it.

2.2. Adversarial Attacks towards DNN-Based NLP Models. With the development of computing power and training techniques, DNNs are broadly used as the main technique of deep learning to deal with the NLP tasks [66–68]. There are various neural models, such as graph-based neural networks (GNNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), and attention mechanisms. The ability to alleviate the problem that traditional methods often rely heavily on discrete, artificial features is one of the strengths of these neural models. Deep learning typically employs distributed representations to express the semantic and synthetic characteristics of a language, which are learned automatically in a specific NLP task. Instead, many kinds of research have shown that pretrained models (PTMs) can learn generic language representations on large corpora, and downstream NLP tasks only need fine-tuning to achieve good performance. Currently, the structure of PTMs has evolved from shallow to deep layers. Recent studies have focused on learning contextual word embeddings, such as CoVe [69], ELMo [70], OpenAI GPT [71], and BERT [72]. These learned encoders still need to represent words in context through downstream tasks. In addition, deep models represented by transformer [73] and various pretraining tasks can learn PTMs to facilitate the industry progress in NLP.

However, security issues always arise hand in hand with the development of technology, and these deep learning techniques can become a central part of the research on adversarial attacks and defenses in the process of widespread applications. Thus, we offer a brief overview of the main deep learning models and the scenarios they are adapted to in NLP tasks, along with a brief list of relevant studies on adversarial attack in different tasks (Table 2).

2.2.1. Feedforward Neural Network (FNN). Feedforward neural network (FNN) is the first type of simple artificial neural network invented in artificial intelligence (AI). Parameters propagate unidirectionally from the input layer through the implicit layer to the output layer. Unlike recurrent neural networks, FNN does not form a directed loop inside it. The (single-layer) perceptron has only a single layer of neuronal network and is considered as the simplest form of feedforward neural network, a binary linear classifier. The multi-layer perceptron (MLP) [16], on the other hand, is the most simple neural network. Bengio et al. [17] first used MLP to solve the language model problem in 2003, and although it did not receive much attention at that time, it laid a solid foundation for later deep learning in solving the language model problem and even many other NLP problems. The earlier proposed text represents the word as a low-rank vector instead of one-hot vector. Word embedding, as a by-product of the language model, played a key role in later research. Because feedforward neural networks are easy to implement and have too many variants without standard benchmark architecture to compare with, the model for text classification of FastText [18], considered as MLP structure, is suitable for text classification. Ribeiro et al. [19] carried out the design of semantically equivalent adversarial rules for debugging the NLP model on this basis. The rest of the adversarial attacks on NLP tasks are mostly targeting specific structures in real applications [20–23].

2.2.2. Convolutional Neural Network (CNN). A CNN is essentially a few convolutional layers plus a non-linear activation function, such as ReLu or Tanh. Unlike traditional neural networks, the input of each neuron is connected to the output of the next layer. In CNNs, we use convolution, a sliding window function for matrices, to compute the output of the input layer. This leads to local connections, where each region of the input layer is connected to neurons in the output layer. Typically, each layer has hundreds of filtering operations, and the values of the filters are learned automatically during the training of the CNN.

Another key part of the CNN is the pooling (subsampling) layer, which usually resamples its input after the convolutional layer. It is characterized by providing a fixed

Table 2: Categories of adversarial attacks on textual DNNs.

DNN models	Variants and applications	Representative adversarial attacks
Feedforward neural network	[16–19]	[20–23]
Convolutional neural network	[13, 24–26]	[27–30]
Recurrent neural networks	[25, 31–40]	[7, 27, 41–45]
Sequence-to-sequence	[46–49]	[50, 51]
Attention-based models	[40, 49]	[52–55]
Reinforcement learning models	[56, 57]	[58–60]
Deep generative models	[61–63]	[44, 60, 64, 65]

size output matrix that reduces the dimensionality of the output while retaining the most salient information, and CNN is location invariant, which is necessary for classification. The last concept is the channel, which refers to different views of the input data. In NLP, different channels can be used for different word embeddings (Word2Vec, GloVe) or the same sentence in different languages.

With fast processing speed and the character of location invariance and local compositionality, CNNs are best suitable for the text classification tasks in NLP. Its architecture was evaluated by Kim [24] on various classification datasets including topic classification and sentiment analysis tasks. Experiments show that it achieves high performance on a variety of datasets, even achieving the state-of-the-art results on some of them, despite its simple network structure.

In addition to word-based CNN classification models, some studies use CNNs directly deal with characters. An embedding that learned directly from characters without any pretraining using CNN was explored by Zhang et al. and Zhang and LeCun [25, 26], who used a comparably deep network with a total of 9 layers and achieved good performance on a dataset containing millions of samples. Several adversarial attacks [27–29] were carried out in the above two representative word-level and char-level CNN text classification tasks.

Kim et al. [13] used CNN not only for text classification but also for language modeling. Character embeddings were used as the input of CNN in this paper, and the output of CNN was processed by a highway layer to represent word embedding and then used as the input of RNNLM, which addresses the traditional word embedding which does not work well for low-frequency words. It also constructs a deeper network through the highway network technique, which reduces the optimization parameters very much. Subsequently, the authors in [30] built machine translation tasks based on it and performed adversarial attacks.

2.2.3. Recurrent Neural Network (RNN). As a very important variant of neural network, recurrent neural network (RNN) is significantly different from the standard neural network in that its input is a word rather than a whole sample, which makes it possible to process sentences of different lengths, including many complex structured ones that cannot be handled by standard neural networks. At the same time, RNNs can share features learned at different locations, which are also better than standard neural networks. Naturally suitable for processing sequential data, RNN is widely used in NLP. However, RNNs do not perform well in capturing long text dependencies and have vanishing gradient problems. These drawbacks have given rise to a few RNN architecture variants.

Long short-term memory (LSTM) networks, the first proposed gate control algorithm for RNNs, correspond to a cyclic unit, and the LSTM unit contains three gates: an input gate, a forget gate, and an output gate. In contrast to the recursive computation established by RNN for the system state, the three gates establish a self-loop for the internal state of the LSTM cell [31]. Since the three gates in the LSTM contribute differently to improve its learning ability, omitting the gates with small contributions and their corresponding weights can simplify the neural network structure and consequently improve its learning efficiency. Gated recurrent unit (GRU) network, an algorithm based on the above idea, has only two corresponding recurrent units: update gate and reset gate, where the reset gate functions similarly to the input gate of the LSTM unit, and the update gate implements both forget and output gates [32]. The structure of RNN makes it inherently suitable for processing sequential data, and thus it has a broad application in NLP along with its variants. For example, in the most common text classification tasks [25, 33], LSTM can still achieve good results. There are some works based on this. Lei et al. [41] studied the problem of discrete adversarial attack and submodular optimization. Sato et al. [42] investigated the interpretability of adversarial examples in the input space.

Schuster and Paliwal [34] proposed bidirectional recurrent neural networks (BRNNs) in 1997. The idea of BRNN is dividing the state neurons of traditional RNN into two parts, where one part is responsible for the positive temporal direction (forward state) and the other is responsible for the negative temporal one (backward state). The backward layer is used to obtain future contextual information. When combining bidirectional recurrent neural networks with LSTM modules, bidirectional-LSTM [35] dramatically enhances the performance of tasks such as sentiment-based analysis and machine translation. Related work was carried out; Iyyer et al. [43] used syntactically controlled paraphrase network to attack sentiment analysis system. Jia and Liang [7] also performed black-box attack on text classification. As to the natural language inference (NLI) task, Chen et al. [36] proposed ESIM (enhanced LSTM). Based on chained LSTM elaborated sequence inference model with intra-sentence attention mechanism (intra-sentence attention), ESIM became one of the benchmarks, to

achieve from local inference to global inference. Zhao et al. [44] generated perturbations, which were more natural and grammatical, and explained the local behavior of the black-box model. There are also bidirectional LSTM (cBiLSTM) [37] and decomposable attention model (DAM) [38], two models that are also suitable for solving NLI problems, and Minervini and Riedel [45] attacked the above models.

In addition to classification tasks, LSTM is also adapted to reading comprehension tasks, and common models are Match-LSTM [39] and BiDAF [40], whose robustness has been evaluated by Gao et al. [27].

2.2.4. Sequence-to-Sequence Learning (Seq2Seq) Models. Seq2Seq, a form of encoder-decoder model, is a variant of the recurrent neural network, which consists of an encoder and decoder. The former encodes the information of a sequence as a vector c of any length. The latter gets the contextual information vector c and then decodes the information and outputs it as a sequence. It is also an important model in NLP. For instance, in machine translation, the sentence lengths of the source and the target languages are different, so are the lengths of questions and answers in question answering systems; in automatic summarization, the sentence lengths of target languages are variable and shorter than those of source languages. The Seq2Seq model allows us to use input and output sequences of different lengths and has a fairly wide range of applicability.

In the end-to-end dialogue domain, the VHRED [46] (variational hierarchical encoder-decoder) model was proposed to curb the difficulty of producing meaningful, high-quality responses from RNNLM and HRED [47]. Conventional Seq2Seq architecture tends to produce short secure responses because it lacks the understanding of the semantic information of the responses due to the defined encoding and decoding processes. VHRED addresses this problem by introducing a global (semantic level) random factor, which can strengthen the robustness of the model while capturing high-level concepts. Latent variable makes the response no longer tied to one/several fixed sentences and encourages the diversity of responses. The robustness of this model has been fully tested against the Ubuntu Dialogue Corpus [50]. OpenNMT [48] is a Torch neural network machine translation system open-sourced by Harvard NLP. It is easy to use and scale, while remaining efficient and cutting-edge in translation quality. As a result, it has been widely used as a tool in text summarization and machine translation [49], whose robustness has been tested by Cheng et al. [51] through word-level LSTM and an attention-based encoder-decoder.

2.2.5. Attention-Based Models. The attention mechanism, originated from the study of human vision, focuses on two main aspects: (1) deciding which part of the input needs to be attended to; (2) allocating limited information processing resources to the important parts. The most successful application of the attention mechanism is machine translation [49]. Such neural network-based MT model is also called neural machine translation (NMT), which generally uses the "encoding-decoding" approach for sequence-to-sequence conversion. This approach has two problems: first, the capacity bottleneck of the encoding vector, i.e., all the information in the source language needs to be stored in the encoding vector for effective decoding; second, the long-distance dependency problem, i.e., the information loss problem in the long-distance information transfer during encoding and decoding. With the introduced attention mechanism, we can save the information in each position in the source language. When generating each word in the target language during the decoding process, we select the relevant information directly from the information in the source language as an aid by the attention mechanism. Such an approach can effectively solve the above two problems. For one thing, there is no need to let all the source language information pass through the encoding vector, and the information in all positions of the source language can be accessed directly at each step of decoding. For the other thing, the information in the source language can be passed directly to each step of the decoding process, shortening the distance of information transfer.

BiDAF [40], short for bidirectional attention network, is the benchmark in machine reading comprehension (MRC). A series of works have been carried out to investigate its robustness [52–55].

2.2.6. Reinforcement Learning Models. Reinforcement learning is also known as reactive learning or augmented learning. Unlike supervised learning, reinforcement learning interacts with the environment through intellectual agents to accomplish goals. Its main features are as follows. (a) Reinforcement learning optimizes the entire sequential decision-making procedure with the goal of overall reward. (b) Reinforcement learning optimizes strategies by interacting with the environment without labeling samples.

Because of its applicability to weakly supervised environments, its natural advantage for decision making on discrete spaces, and its ability to cope with many hidden state situations, reinforcement learning has some effective applications for scenarios in NLP [56], such as dialogue systems [57]. Studies on its robustness have also been carried out successively [58–60].

2.2.7. Deep Generative Models. Deep generative models are good at studying how well the model has learned and the domain of the problem. Generative adversarial networks (GANs) [74] and variational autoencoders (VAEs) [62] are the most prevalent text generation models in NLP [63].

GAN [74] is a deep learning model, which is one of the most promising approaches for unsupervised learning on complex distributions in recent years. The model learns by playing each other through (at least) two parts of the framework: one is generate model and the other is discriminate model to produce a fairly good output. In the original GAN theory, both G and D are not required as the neural networks, as long as they fit the corresponding generative and discriminative functions. However, in practice, deep neural networks are generally used as G and D.

VAE [62] is a self-encoder whose coding distribution is normalized during training to ensure good properties in the hidden space, allowing us to generate some new data. "Variational" is derived from the statistical methods of regularization and variational inference.

A good GAN application requires a good training method; otherwise, the output may be unsatisfactory due to the high degree of free nature of the neural network model. Thus, only a few works [44, 60, 64] used GAN to generate sentence-level perturbation attack on TC and NMT tasks. Related techniques can be found in this review [65].

2.3. The General Taxonomy. We give the definitions for the core concepts, the framework for the threat model, and metrics for adversarial attacks.

2.3.1. Definition

(1) *Deep Neural Network (DNN).* As introduced in Section 2.2, they are the core technology for solving NLP problems and the main target of the adversarial attack—the victim model. Deep neural models can be presented in a simplified form: suppose f_θ: $X \longrightarrow Y$ is a non-linear function, where X is a model input for vector text (as introduced in Section 2.1) and Y is the model prediction result, a label or a sequence, depending on the NLP task. θ is the model parameter learned automatically by gradient-based backpropagation during the model training phase. Usually, the model will be defined by the loss function J. At the minimum $J(f_\theta(X), Y)$, θ will be the best parameters.

(2) *Perturbation.* Generally, it is the noise added to the original data, which can be distinguished by different granularity or by the degree of semantic change.

(3) *Adversarial Examples.* When the perturbation refers to a specific sample that is not visible to the human eye, but makes the machine misidentify it (also the definition of a backdoor attack), it refers exclusively to the adversarial example. The formula is described as follows:

$$\begin{aligned} x' &= x + \delta, \\ f(x) &= y, \quad x \in X, \\ f(x) &= Y, \\ f(x') &= Y', \quad Y' \neq Y, \end{aligned} \tag{4}$$

where x is the original input, δ is the small perturbation, $x\prime$ is the generated example, y is the ground truth label, and $Y\prime$ is the misclassification label, and when it is the specified label, it is the targeted attack.

(4) *Evasion Attack.* The adversarial attacks we mentioned in this survey all happen in the testing phase, when the parameters of the victim model could only be accessed and not modified.

2.3.2. Threat Models. According to the definition of Yuan et al. [75], we will discuss several details of the threat model in text adversarial attacks.

(1) *Adversary's Motivation.* The perturbation examples themselves are just a kind of data, which is used to attack or defend depending on the attacker's purpose. For example, it could be used to alter the model output (evasion attack) or it can be used for adversarial training, which makes the model resistant to a specific type of sample. Or it can enhance the overall robustness of the model through data augmentation.

(2) *Targeted vs Untargeted.* As defined in (4), when the output Y' of the perturbation is a specific target, it is called a targeted attack; conversely, it is called a non-specific target. Obviously, the assumption of targeted attack is stronger and relatively more difficult to implement. The two are equivalent only when the model is dealing with a binary classification task.

(3) *Adversary's Knowledge.* According to Zeng et al. [76]'s classification of accessibility to the victim model, we use a fine granularity. Assumptions from strong to weak are as follows: gradient, which is equal to white-box attack (the adversary is fully aware of the victim model); score, the adversary can access the confidence score and the output of the model; decision, the adversary can only access the decision of the model; and blind, which is equal to black-box attack (adversary knows nothing about the model)

(4) *Perturbation Granularity.* For text, the perturbation level can be divided into three granularities: char level, which is the character-level addition, deletion, or exchange of input; word level, when the word is the smallest unit of substitution; and sentence level, which usually means paraphrasing.

2.3.3. Measurement. When evaluation of an adversarial attack is made, there are two aspects to consider: (1) evaluate the validity of the attack and (2) control of the degree of perturbation.

(1) *Attack Evaluation.* Evaluation of the effect of the attack can be divided into three aspects [76]. (1) Attack success rate: this part varies according to the specific task; for classification tasks, it usually includes AUC score, P, *R*, F1 score; for generation tasks, it usually uses blue score; the specific language model prediction score is recommended to be considered in the context of the reader's own task. (2) Adversarial example quality: this part can be referred to as the evaluation of perturbation constraint. (3) Attack efficiency: it includes average query time and average run time for victim models; generally, the smaller these two points are, the better the attack effects will be.

(2) *Perturbation Constraint.* As mentioned earlier, the adversarial example should meet the definition of the

backdoor attack. That is, the human eyes cannot recognize it while the machine misjudges it [77]. For the naturally contiguous pixel information of an image, some research limits the adversarial examples to have the same range of pixels as the original data in vector space [78] or uses norm-based methods, such as $\max_n orm$ [61], L_2 − norm, L_0 − norm, and so on. Also, the above methods cannot be directly introduced into the text. You can vectorize text in the embedding space to calculate its distance first, or see the text as a unit of char level and word level and calculate its edit distance, or judge its similarity at the sentence level and document level (see Section 3.2 for specific details).

3. Generating Adversarial Examples to NLP Systems

3.1. From Perturbation Level to Methods. In this section, we present the current methods for generating adversarial examples. First, we introduce our classification criteria. We classify perturbation into four types: char, word, sentence, and multi-level. Under the different levels, the different stages of the method are generalized according to our abstraction of the stage operation of generating perturbation, considering the access to knowledge possessed by the attacker and also combining the limitations of a specific task.

We need to clarify that perturbation level refers to the smallest unit of operations (add, mask, replace, and swap) in our generating perturbation process, not the final perturbation result we see with our naked eyes. For example, the char-level perturbation can theoretically achieve all levels of perturbation, except that it is more expensive, e.g., the number of operations performed to complete word replacement by char is higher. In most cases, the unit of perturbation is consistent with the presentation effect, but for the rigor of the classification of the paper, it is necessary to illustrate two cases where the perturbation seen by the naked eye is not consistent with the unit of operation.

(1) *Degraded Perturbation*. If the attacker's aim is to destroy, we consider the smallest unit of its operation, for example, disrupting the whole sentence by manipulating char only under specific rules. We consider this as char-level perturbation; similarly, if the attacker locates the keyword first and then perturbs it by char to make misspelling errors, we also consider this as belonging to the char level.

(2) *Escalation Perturbation*. If the attacker's purpose is to reconstruct, then we take the target of its reconstruction as a classification basis. For example, if an attacker implements a semiautomatic paraphrase by word-by-word substitution under artificially set semantic rules, we consider this to be a sentence-level perturbation.

In addition, we consider an operation as multi-level when and only when it can generate both types of perturbations; otherwise, we consider it to classify the operation and its corresponding perturbation in different perturbation levels. This classification is used because we place more emphasis on the pipeline of generated perturbations than on the completeness of the cited paper. We want to modularize the attack process and abstract the purpose of the different modules.

Finally, we refer to [76] for fine-grained classification of access knowledge, listed below.

Gradient: needs to fully comprehend the model of the victim, especially the model structure and parameters, to conduct gradient updates.

Score: needs the prediction scores, e.g., probability distribution over each answer.

Decision: needs only the final output, e.g., classification of predictions.

Blind: does not require knowledge of the model.

Next, we will present the classification of methods under different perturbation levels separately (Table 3).

3.1.1. Char-Level Perturbation. As stated in the previous section, we look at the modularity of the operations in implementing a particular perturbation process. The perturbation can be divided into two phases: location—locate the position of the char to be perturbed; perturbation—the perturbation method. Noticeably, because we look at the purpose, for example, a method that combines *location* and *perturbation* in one phase, i.e., locates the target of the perturbation while determining the operation of the perturbation, we regard it as the perturbation phase since *location* is only an intermediate stage in the generation of perturbations.

Location. Because of the above classification basis, by and large, there is no method of location specifically designed for char level, and a few papers choose to randomly select the char to be perturbed [79].

Perturbation. The pertubation methods are given as follows.

(1) Gradient-based: there is a classical method under char level [28], which defines the flip of characters as an atomic operation, based on the input one-hot vector to find the directional derivative of the target function to get the change that lets the loss grow the most.

$$\max \nabla_x J(x, y)^T \cdot \overrightarrow{v}_{ijb} = \max_{ijb} \frac{\partial J^{(b)}}{\partial x_{ij}} - \frac{\partial J}{\partial x_{ij}}^{(a)}, \quad (5)$$

where $\overrightarrow{v}_{ijb}$ is the operation of flipping from $j-th$ to $i-th$. Therefore, this method can be considered as a one-stage attack that binds location and perturbation. In fact, this approach can also handle words expressed by one-hot vectors. Ebrahimi et al. [80] extended this approach to machine translation tasks and introduced the swap operation.

(2) Blind-based: blind means black-box, i.e., no prior knowledge of the model is required. This is because

Table 3: Categories of adversarial attacks (from perturbation level to methods).

Perturbation level	Attack phase	Access knowledge	Relative work
Char	Location	Blind	[79]
	Perturbation	Gradient	[28, 80]
		Blind	[27, 29, 79, 81–83]
Word	Location	Gradient	[29, 61, 81, 84, 84–89]
		Score	[7, 27, 89–94]
	Perturbation	Gradient	[6, 42, 90, 95–98]
		Score	[92, 99–106]
		Decision	[107, 108]
		Blind	[29, 60, 85, 93, 101, 109–111]
Sentence	Perturbation	Score	[45, 93, 112]
		Decision	[7, 113]
		Blind	[86, 114]
	Generation	Gradient	[115]
		Score	[59, 116]
		Decision	[19, 44, 58, 59, 64, 117]
		Blind	[43, 60, 118]
Hybrid	Generation	Score	[119]

it relies on task form, expert knowledge, and even cross-domain knowledge to provide inspiration. First of all, early approaches tend to be trivial, [27, 29, 81] just randomly processed characters, while they focus on location. Gil et al. [82] used the method of distillation. They treated the perturbation generated by HotFlip in the form of ("Assohle", (4,"n")), which represents the conversion of the 4th character of the first input to "n," resulting in a (19x–39x) speedup. Eger et al. [79] proposed a visual perturbation that uses three char embedding spaces and designed a method from char embedding to word embedding (e.g., scrambled toxic comments like ŭĉk̆ ô,yoŭ âňtî-ŝeîîĉ ĉŭň). This method of inspection is suitable for the current real-world scenario of social networks with a large number of toxic comments, spam detection, etc. Belinkov and Bisk [83], combined with the knowledge of psychology, found that humans can understand the meaning of a sentence by keeping the position of the first and last letters of the word unchanged. They combined this approach to design several forms of perturbation such as swap, middle random, fully random, keyboard error, and natural error to attack the machine translation system.

3.1.2. Word-Level Perturbation. Mainstream work has focused on word-level perturbation because of the large search space of substitution words and hardness to maintain sentence semantics. Similar to char, the preliminary approach focuses on two parts, *location* and *perturbation*. They focus on how to locate the keywords with less knowledge and use common ideas such as random substitution and synonym substitution for the generation of perturbation, even without considering the semantics. The later approaches tend to directly list the candidate set for each word, and the research focuses on how to quickly search for the best perturbation and ensure semantic integrity. In addition to this, there are methods that generate word-level perturbations directly, which are listed in *perturbation*.

Location. The location methods are given as follows.

(1) Gradient-based: some early work borrowed methods from the image domain. The fast gradient sign method (FGSM) was proposed by Goodfellow et al. [61], who devised a method for fast generation of adversarial examples based on the fact that linear behavior is enough to induce adversarial examples in higher dimensional spaces.

$$\frac{x_{\text{FGSM}}^{\text{adv}}}{coloneqx} + \varepsilon \cdot \text{sign}\left(\nabla_x L\left(h(x), y_{\text{true}}\right)\right), \tag{6}$$

where L is the loss function, x and y_{true} denote the input image and the true label, and θ denotes the network parameters: the direction of the gradient can be calculated by the sign function, which is a function used to find the numerical sign. If inputs are greater than 0, the output is 1, if inputs are less than 0, the output is −1, and for inputs equal to 0, the output is 0. It computes only the direction of the gradient of interest and updates the size of ε each time for x. Papernot et al. [84] and Samanta and Mehta [85] used this way to calculate the contribution of each word. $\mathscr{C}_F(w_k, y) = -\nabla_{w_k} J(F, s, y_i)$. This method can also be adopted to locate hot training phrases (HTP) and hot sample phrases (HSP) [29].

The Jacobian saliency map (JSMA) algorithm is inspired by the saliency map in the field of computer vision, which adds a limited number of pixel points to the original image [86]. This method uses the Jacobian matrix to calculate the features from the input to the output, so only a small portion of the input features is modified to achieve a change in the output classification. Jacobian-based attack was adopted in [81, 84].

Later, Yang et al. [87] proposed a probabilistic framework, containing both the greedy attack and the Gumbel attack. The former can select top k important features of input x; the latter uses a concrete variable to approximate the probability distribution, thus requiring fewer model evaluations. Cheng et al. [88] also used a greedy approach to select the nearest candidate words in the gradient direction to the current word vector. The generated results are used to verify the robustness of the NMT model. Additionally, Zheng et al. [89] designed a scoring function for dependency parsing models to measure the difference between true parse tree and any incorrect one:

$$\begin{aligned} \text{set } A &= \left[s(x_h, x_m; \theta) - \max_{j \neq h} s\left(x_j, x_m; \theta\right), -\varepsilon\right], \\ \mathcal{F}(x, \theta) &= \sum_{m=1} \max A, \\ \mathcal{S}(x_i, \theta) &= \left\| \frac{\partial \mathcal{F}(x, \theta)}{\partial e_{x_i}} \right\|_2. \end{aligned} \tag{7}$$

Equation (7) is designed to describe how a custom scoring function can be used to determine the importance of words in a specific task like dependency parsing models, where s is used to determine which of the n words of data x are more likely to be attacked. $s(x_h, x_m, \theta)$ score represents the dependency relationship established in the dependency tree from the head x_h to the modifier x_m likelihood. The perturbation priorities are ordered by $\mathcal{S}(x_i, \theta)$ in descending order.

(2) Score-based: the first score-based approach was proposed by Gao et al. [27]. To determine word saliency, they weighted the results of the output scores of replace, temporal, and temporal tail (Figure 5) on word importance ranking. Meng and Wattenhofer [90] and Jin et al. [91] also used a similar method of masking words to calculate the difference. The main difference of the method in different tasks is the choice of the score function. For example, the unlabeled attachment score (UAS) is used in the dependency parsing model [89], and the F1 score is mostly used in question answering [7]. Shi et al. [92] found that a word pair could be replaced with a shared word, while the ranking was judged by the loss they induce.

Perturbation. For models that introduce the attention mechanism, the attention score is also a basis for evaluation. In MovieQA question answering, Blohm et al. [93] weighted the plot with question and answer and then selected the word with the highest attention weight to be replaced according to the internal attention distribution. Hsieh et al. [94] selected words in transformer based on highest or lowest self-attention score.

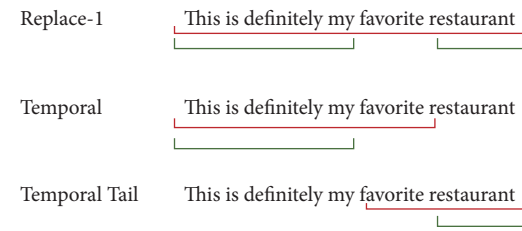

Figure 5: Word saliency judged by score-based method.

(1) Gradient-based: there are many ways to find the optimal perturbation. Behjati et al. [95] employed a projected gradient-based approach to find the nearest one to the current word vector in the embedding space, which is also an early attempt at the universal adversarial trigger. Cheng et al. [96] introduced group lasso on the projected gradient method, with gradient regularization, to achieve two kinds of novel non-overlapping attack and targeted keyword attack. Restricted directions can also provide some interpretability for perturbation generation [42]. Meng and Wattenhofer [90] selected alternative words by iteratively approximating the decision boundary. Wallace et al. [6] also used a HotFlip-inspired approach to design a universal trigger, represented as a one-hot vector. A linear approximation of the task loss is used to adjust the randomly generated noise by minimizing the loss.

Unlike the previous ones, Zhang et al. [97] used the Metropolis–Hastings (MH) sampling method for each round of operation (replacement, insertion, and deletion) of transition proposal, combining language models and gradients to guide the jump direction of the samples in a white-box situation.

Besides, generative methods can be adopted. Song et al. [98] used a pretrained adversarially regularized autoencoder. They first mapped a Gaussian noise vector n to a latent vector z using generator and then generated a trigger t from z using the decoder.

$$\begin{aligned} z &= \text{GENERATOR}(n); \\ t &= \text{DECODER}(z); \\ x' &= [t; x]. \end{aligned} \tag{8}$$

This is also a way to generate a universal trigger, and even some semantics are preserved.

(2) Score-based: because the subsequent work tends to prepare the candidate set for each word and then decide the replacement order, this is equivalent to the process fusion of *location* and *perturbation*. Based on this idea, designed a greedy algorithm, using the method of probability weighted word saliency to design the score function:

$$H(\mathbf{x}, \mathbf{x}_i^*, w_i) = \phi(\mathbf{S}(\mathbf{x}))_i \cdot \Delta P_i^*, \tag{9}$$

where $\phi(\mathbf{z})_i$ is the softmax function.

$$\Delta P_i^* = P(y_{\text{true}}|\mathbf{x}) - P(y_{\text{true}}|\mathbf{x}_i^*), \tag{10}$$

where ΔP_i^* is the optimal word substitution effect.

$$S(\mathbf{x}, w_i) = P(y_{\text{true}}|\mathbf{x}) - P(y_{\text{true}}|\hat{\mathbf{x}}_i), \tag{11}$$

where $S(\mathbf{x}, w_i)$ is the word saliency referring to replacement order. Alzantot et al. [100] first used population-based gradient-free optimization via genetic algorithm. They randomly sampled from the input which word to replace, using Euclidean distance, language model scores, and target label prediction probability to constrain the process of selecting replacement words. The method can be used for selecting both candidate words and transformation operations [101]. In addition to using average-case random operation, they also used the method of greedy search, beam search, and genetic search for the selection of grammatical error generation operation they defined. In fact, the choice of transformation space plays a crucial role in identifying word substitution as a combinatorial optimization problem [102]. Proposed by Zang et al. [103], HowNet is better than wordnet and counterfitted embedding space, which included high-quality and high-efficiency examples. Also particle swam optimization (PSO) has higher attack success rate among the above heuristic search algorithms.

Different from the above ideas, some scholars consider those large-scale pretrained models to have a large corpus, and the training method of the mask makes them inherently capable of generating candidate words. References [104–106] used BERT (RoBERTa) to attack BERT. Shi et al. [92] also generated the replacement phrase pairs by using BERT.

(3) Decision-based: Maheshwary et al. [107] improved the genetic algorithm without using the probability scores of the target tags, but maximizing the semantic similarity between sentences, which makes it work even under hard tagging conditions. Reducing the search space before the optimization is also an important speedup process. Zou et al. [108] used reinforcement learning and incorporated discriminator in the environment. This allows the actor to change the input while attacking the NMT model and preserving the semantics from the discriminator as much as possible.

(4) Blind-based: blind-based word substitution is mostly rule-based and relies heavily on manual manipulation in the early days. Blohm et al. [93] manually selected 106 most frequent words; then they defined lexical substitutions of single words or multi-word expressions. Because adverb tends to have less impact on semantics, Liang et al. [29] used the removal of adjectives and adverbs. Samanta and Mehta [85] also used adding the adjective before the target. There are also some works adopting special rules, for example, combining linguistic phenomena, Fan et al. [101] selected 8 out of 27 grammatical errors of NUCLE [109] and designed the probabilistic transformation. Glockner et al. [110] argued that mods lack lexical and world knowledge, and they used a synonym, hypernym, co-hyponyms, and antonyms to generate a new NLI test set without introducing new vocabulary. Tan et al. [111] used LemmInflect to generate the Morpheus form of the vocabulary. A series of Should-Not-Change and Should-Change strategies were designed to generate paraphrases on the dialogue system [60] with word-by-word replacement strategy to test the over-sensitivity and over-stability of the system, respectively.

3.1.3. Sentence-Level Perturbation. Sentence-level perturbation is divided into two forms: perturbation and generation. *Perturbation* is a two-stage attack in which researchers often locate a specific word first and then paraphrase it by combining the rules. Location is roughly similar to word-level location, and here we will focus on the perturbation rules. *Generation* is a one-stage attack that generates sentence-level perturbations directly with the help of the network.

Perturbation. The perturbation methods are given as follows.

(1) Score-based: judging sentence-level importance should be able to heuristically quantify its score. The question answer model designed by Blohm et al. [93] calculated the attention score for the sentence vector obtained from the concatenate word vector. They deleted the sentences with high attention score to check whether the model can read out the key sentences or not. In addition to the deletion operation, they also designed several methods to add distractor sentences. AddC randomly selects 20 common words from the candidate words; AddQ selects replacement words from the list of all questions; the candidate words of AddQA also include wrong answers. Such concate attack is often used in QA systems. Besides, rules can be designed in conjunction with specific tasks. Minervini and Riedel [45] designed five first-order logic rules for the NLI task (Table 4) by swapping sentence order to obtain the sample that gives the highest model inconsistency score.

Also, this attack introduces external knowledge by adding negation, hyponymy, and antonymy. The same logic is also applied in the generation of adversarial sets for link predictors [112], but only Horn clauses are used in the experimental part.

Table 4: Five first-order logic rules.

NLI rules	
$\mathbf{R}_1$	$\top \Rightarrow \text{ent}(X_1, X_1)$
$\mathbf{R}_2$	$\text{con}(X_1, X_2) \Rightarrow \text{con}(X_2, X_1)$
$\mathbf{R}_3$	$\text{ent}(X_1, X_2) \Rightarrow \text{con}(X_2, X_1)$
$\mathbf{R}_4$	$\text{neu}(X_1, X_2) \Rightarrow \text{con}(X_2, X_1)$
$\mathbf{R}_5$	$\text{ent}(X_1, X_2) \wedge \text{ent}(X_2, X_3) \Rightarrow \text{ent}(X_1, X_3)$

(2) Decision-based: the decision-based attack generally requires only the result of the model call without knowing the confidence scores of the result. This attack can vary with the form of the task. In the QA system, the model response is already quite informative. For example, Jia and Liang [7] designed the attack form of ADDSENT. The model answer is obtained by querying, selecting keywords and wrong answer, converting the original question into a declarative sentence using 50 manually defined rules, and inserting it into the original article after review by crowdsourcing. However, because they specify in advance that the location of the inserted sentences can only be at the beginning and end of the article, this may lead to model cheating. So, Wang and Bansal [113] introduced random location and random keyword and proposed AddSentDiverse attack method, which considered the form of concate attack more comprehensively.

(3) Blind-Based: the earlier articles relied heavily on manual work for sentence-level generation. Moosavi-Dezfooli et al. [86] used a perturbation method of inserting sentences for the found HSPs. Since most of these words are proper nouns, they are suitable for supplementing facts with definite clauses. So, the authors manually added them from Wikipedia or fabricated facts. Zhang et al. [114] found a large number of lexical overlap in paraphrase identification datasets. They used language model-based extraction of NER attributes of words and then exchanged entity word positions to generate negative samples. After that, they used back translation to expand the positive examples and the cross-combination of positive and negative examples to generate label-balanced adversarial-example sentence pairs.

Generation. The generation methods are given as follows.

(1) Gradient-based: for the Quizbowl task, Wallace et al. [115] proposed that participants design the questions causing the model to go wrong. The model calculates the importance of words through a gradient and gives participants clues to design the problem. Since such task format is interesting, it is a clever application of human-computer interaction.

(2) Score-based: Cheng et al. [59] used reinforcement learning to design adversarial agents for the bidding negotiation system. In the white-box scenario, he/she needs to know the dialogue history to guide the agent reward function through logit layer outputs in the white-box scenario. The effect is stronger than the decision-based agent that only knows the final result of the dialogue. Wang et al. [116] designed a controlled text generation model, including an encoder, a decoder, and an attribute classifier. Attribute is encoded with one-hot encoding by a projection that restricts the text to different outputs $s_{a'}$ for different attributes.

$$a^* = \arg \max_{\{a' \neq a\}} \left[-\sum_y y \log p\left(y | s_{a'}\right) \right]. \tag{12}$$

The *optimal* attribute a^* is obtained when the cross-entropy loss between the ground truth labels and the prediction is maximized.

(3) Decision-Based: Wang et al. [117] designed a tree-based autoencoder capable of generating word-level and sentence-level perturbations while maintaining semantic and syntactic information. This attack is oriented towards sentiment analysis and QA tasks, implementing targeted attack for the former and position targeted attack and answer targeted attack for the latter. The embedding approach for word and sentence-level embedding characterizes itself in that sentence level uses the root node of the tree, while word level uses the concatenation of the leaf nodes. The target of the attack is

$$\min \left\| \mathbf{z}^* \right\|_p + cf\left(\mathbf{z} + \mathbf{z}^*\right). \tag{13}$$

Ribeiro et al. [19] first generated semantically equivalent adversaries (SEAs) by the NMT system and then automatically extracted semantically equivalent adversarial rules (SEARs), and finally the process of extracted rules is abstracted into a submodular optimization problem for filtering paraphrase sentences that match the rules, where the semantic score is defined as follows:

$$S\left(x, x'\right) = \min\left(1, \frac{P\left(x' | x\right)}{P\left(x | x\right)}\right). \tag{14}$$

When $\text{Sem}\,Eq\left(x, x'\right) = [S\left(x, x'\right) \geq \tau]$, the resulting sample x' is considered semantically equivalent to x, where τ is defined by crowdsource. The final generated semantically equivalent adversary (SEA) is defined as follows.

$$\text{SEA}\left(x, x'\right) = \nVdash \left[\text{Sem}Eq\left(x, x'\right) \wedge f\left(x\right) \neq f\left(x'\right)\right]. \tag{15}$$

Reinforcement learning also has relevant applications in the study of generative samples. Cheng et al. [59] designed future reward for adversarial agent based on reinforcement learning under the condition that only the outcome of the negotiation is known.

$$R(x_t) = \sum_{x_t \in X^{\text{adv}}} \gamma^{T-t}\left(r^{\text{adv}} - \mu\right), \tag{16}$$

where $[r^{\text{adv}} = S_{\text{adv}} - S_{ori}]$ and the adversary agent's reward is the increment of the score he gets. Because the adversarial agent is set to choose the complement of the target agent, the conversation always agrees.

Han et al. [58] were concerned about the problem that prediction results in structured prediction tasks are vulnerable to small perturbations. This model contains a sentence generator and three parsers, where parser A is the victim model, and parser B and parser C are reference parsers. The quality of the adversarial examples is determined in the following way.

$$s_p(\hat{\mathbf{x}}) = -f\left(\mathbf{y}_{\hat{\mathbf{x}}}^{A}, \mathbf{y}_{\hat{\mathbf{x}}}^{B}\right) - f\left(\mathbf{y}_{\hat{\mathbf{x}}}^{A}, \mathbf{y}_{\hat{\mathbf{x}}}^{C}\right) + f\left(\mathbf{y}_{\hat{\mathbf{x}}}^{B}, \mathbf{y}_{\hat{\mathbf{x}}}^{C}\right), \tag{17}$$

where x is the input and $\mathbf{y}_{\hat{\mathbf{x}}}^{A}, \mathbf{y}_{\hat{\mathbf{x}}}^{B}, \mathbf{y}_{\hat{\mathbf{x}}}^{C}$ is the predicted parse tree of parsers A, B, C respectively. When the result of $\mathbf{y}_{\hat{\mathbf{x}}}^{B}$ is close enough to $\mathbf{y}_{\hat{\mathbf{x}}}^{C}$ and much different from $\mathbf{y}_{\hat{\mathbf{x}}}^{A}$, it is considered as a good adversarial example. The training process is guided by the reinforcement learning way.

There were also researchers who used GANs to generate adversarial examples. Le et al. [64] found that the effect of fake news detector in social media would be affected by fake comments. So, they designed a GAN-based fake comment generation system, where the attack module only needed the output results of victim to guide the comment generation, and combined it with the style module to ensure that the comments were relevant and in a normal style. Zhao et al. [44] also used GANs to generate samples in a range of applications, including image classification, textual entailment, and machine translation tasks.

(4) Blind-Based: Tong and Bansal [60] found that the word-by-word replacement strategy was context-dependent in generating sentences and cannot synthesize effective sentence-level paraphrases. Therefore, they also complemented the generative network to generate perturbations by using the pointer-generator network. Iyyer et al. [43] proposed a syntactically controlled paraphrase network (SCPN). This is a trained encoder-decoder model that produces paraphrased sentences with the desired grammar, which can attack sentiment analysis and text assignment systems. They use back translation and parsers to mark mutations, which makes it possible for them to obtain very large-scale training data. [52] used a novel approach to train the transformer model to generate two test sets on SQuAD problems. One test set is used to verify the robustness of the model to the question paraphrase, and the other is designed to test the dependence of the model on string matching.

3.1.4. Multi-Level Perturbation. Because this review focuses on the basic operation of generating the minimum perturbation unit rather than on the complete experimental flow of the cited paper, if a paper implements multiple perturbation level attacks that can be split, we put it into different levels. Only when the perturbations of different levels are caused by one operation, we consider them as multi-level. Vijayaraghavan and Roy [119] proposed a hybrid word-char encoder-decoder model, which used a reinforcement learning based approach to obtain the victim model through the test phase of feedback design rewards. It can generate both paraphrase and char-level perturbations.

3.2. Perturbation Measurements. As introduced in Section 2.1, image metrics cannot be directly transferred to text. Perturbation level and language measurement are the two parts of metrics.

3.2.1. Perturbation Level. It is divided into vector-based measurement, edit-based measurement, and set-based measurement.

Vector-Based Measurement. After we vectorize the text using the method in Section 2.1, the similarity of the text can be calculated using the distance formula.

(1) *The Euclidean metric* [120] (also known as the Euclidean distance, L_2norm, and RSS distance) is a common definition of distance that calculates the real distance between two points in m-dimensional space, or the natural length of a vector. In two and three-dimensional space, the Euclidean distance is the real distance between two points (i.e., the distance from that point to the origin). The formula for N-dimensional space is as follows.

$$\begin{aligned} d(x, y) :&= \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} \\ &:= \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \end{aligned} \tag{18}$$

(2) *Manhattan distance* (also known as cab geometry) was coined by Hermann Minkowski in the nineteenth century as a geometric term used in geometric metric space to indicate the sum of the absolute axis distances of two points on a standard coordinate system.

$$p = |x1 - x2| + |y1 - y2|. \tag{19}$$

(3) *Cosine Distance*. The cosine of the angle between two vectors was used to measure the magnitude of the difference between two individuals. The cosine distance focuses more on the difference in direction

between two vectors rather than the Euclidean distance. The cosine similarity of N-dimensional vectors $A = [a_1, a_2, \ldots, a_n]$ and $B = [b_1, b_2, \ldots, b_n]$ is calculated as follows:

$$\cos\theta = \frac{A \cdot B}{|A| \times |B|} = \frac{\sum_{i=1}^{n} (A_i \times B_i)}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}. \tag{20}$$

Note that when the above distances are used in computing the sentence vector, they can be interpreted in a semantic-preserved way. Common sentence encoders include the following:

(1) *Skip-Thought Vectors* [121]. Before the advent of BERT, this method was a common way for obtaining high-quality sentence vectors.

(2) *Universal Sentence Encoder(USE)* [122]. The model extended the above multi-task training by adding more tasks and co-training with a skip-thought-like model to predict the sentence context of a given text fragment [122].

(3) *InferSent* used the Stanford natural language inference dataset as a training set to achieve superiority over the skip-thought approach [123].

(4) *Sentence-BERT* [124].

Edit-based measurement mainly measured the smallest unit operation on the text.

(1) *Levenshtein distance*, also known as edit distance [125], calculated the minimum number of individual character edits (such as deletions, insertions, and substitutions) needed to turn one word into another. However, it was only one type of edit distance and was closely related to pairwise string comparisons.

(2) *Word mover's distance (WMD)* was proposed in 2015. It was the pioneering work in the in-depth study of text similarity computation [126]. Based on word2vec, the problem of semantic similarity of text was transformed into a linear programming problem. The formal description was as follows.

First, suppose there is a trained word vector matrix $X \in R^{d*v}$, with a total of v words. The ith column $x_i \in R^d$ represents the d-dimensional word vector of the ith word. Then, the Euclidean distance between word i and word j is

$$c(i, j) = \left\| x_i - x_j \right\|_2, \tag{21}$$

For a given text, the sparse vector $d \in R^n$ can be used as the bag-of-words representation, where the number of words occurring in the ith word i in the text is denoted by c_i; then, the regularized representation of the ith word frequency in d is

$$d_i = \frac{c_i}{\sum_{j=1}^{n} c_j}, \tag{22}$$

Let d, d' represent the bag-of-words representations of the two texts to be computed, respectively, and each word i in d can be transformed to d' in whole or in part. Therefore, define the transfer matrix $T \in R^{n*n}$, where T_{ij} represents how many words i are transferred from d to the word j in $d\imath$, $T_{ij} \geq 0$, thus minimizing the transfer cost and translating to a linear programming problem.

$$\min_{\mathbf{T} \geq 0} \sum_{i,j=1}^{n} \mathbf{T}_{ij} c(i, j) \quad \text{s.t.} \quad \sum_{j=1}^{n} \mathbf{T}_{ij} = d_i, \quad \forall i \in 1, \ldots, n \quad \sum_{j=1}^{n} \mathbf{T}_{ij} = d'_i, \quad \forall i \in 1, \ldots, n. \tag{23}$$

(3) *Word modification rate*: it is the proportion of the total input utterances in which the adversarial example changed the verbs compared to the original input.

(4) *Number of change*: this method only evaluated the number of the editing operation.

Set-Based Measurement. This approach treats words as elements of the set of words in a document to calculate similarity.

(1) *Jaccard index* [127], also known as Jaccard similarity coefficient, was used to compare the similarity and difference between a finite set of samples; the higher the value of the Jaccard coefficient, the higher the similarity of the samples. The similarity of two documents or sentences in a text can be calculated. The formula is as follows.

$$J(A, B) = \frac{A \cap B}{A \cup B}. \tag{24}$$

When the sets A, B are empty, $J(A, B)$ is defined as 1. According to the above formula, when the intersection between $\mathrm{Doc}_1 4$ and the set Doc_2 is larger, it indicates that the degree of similarity between the two is higher. In particular, $J(A, B)$ is 1 when set A and set B are the same.

3.2.2. Language. In addition to the comparison of text similarity, we also need to judge the compliance of the language itself. There are two parts, grammatical level and manual assessment.

Grammatically.

(1) *Number of Grammatical Errors*. This can be evaluated with the help of LanguageTool [128].

Table 5: Categories of adversarial defenses.

Deep learning deployment	Defense strategies	Relative work
Detecting adversarial examples	Local trigger detection	[27, 134–140]
	Universal trigger detection	[6, 141, 142]
Enhancement training phase	Adversarial training	[7, 77, 83, 143–154]
	Robust encoding	[154–157]
	Computing robust lower bounds	[158–165]

Lexical Consistency. It refers to the lexical consistency of the replaced word and the replacing word. This can be achieved with the help of NLKT, ApaCy, and flair.

(2) *Language Model*. Does it fit the context of the following language model [129–131].

Human Evaluation.

(1) Grammatically [132]: determine if there are grammatical errors in the text.

(2) Plausibility [133]: evaluate the fluency of the text and whether it is written by a native speaker.

4. Defense Strategy

For the above attacks, we can summarize the defense ideas in steps by combining the process of deep learning deployment. (1) Data preprocessing stage: we can perform anomaly detection before data are input to the model to determine whether it is an adversarial example. (2) Training stage: we can use the generated adversarial example to secure system using empirical defense strategies, including data augmentation, model regularization, robust optimization, and knowledge distillation. (3) At the same time, we can calculate the robustness bound of the model and use the approximation calculation to approach the lower bound of the robustness bound, and the common methods are interval bound propagation and random smoothing. (4) Finally, we restrict the number of times the adversary accesses the API after the model is employed to reduce the harm of its attacks (Table 5).

4.1. Detecting Adversarial Examples

4.1.1. Local Trigger Detection. In order to protect the system, the first thought on the approach is to filter the raw data to prevent adversarial examples from entering the model and curb the problem before it happens. Related works can be broadly divided into two ideas: (1) detecting the addition, deletion, and exchange of char levels and (2) detecting the substitution of synonyms. Char-level detection can be undertaken with the help of tools, e.g. *Python* autocorrect 0.3.0 package [27]. But those detection tools [134–136] have limited effect. For one thing, they are only effective for partial word-level substitution. For another, they do not work for hieroglyphic languages [137].

Therefore, a great deal of work has been focused on detecting substituted synonyms. Wang et al. [138] assumed that the generalization ability of the model causes the presence of adversarial examples: an insufficiently strong generalization can cause different samples of the same general class to be divided into different classes. Thus, an encoder is used to encode the samples before the classifier, mapping all the points around a sample to the centroid in order to aggregate texts from the same class together. This approach does not require expanding the data or modifying the model architecture but only requires training on the original dataset after adding a mapping in front of the classifier. For an input format like text, which is a discrete symbol in a sentence, finding neighbors is relatively simple. Zhou et al. [139] designed a scrambled discriminator DISP to predict whether each character in the data was scrambled or not. For each potentially scrambled character, it generated an approximate embedding vector and found the closest one to recover the text in the embedding space. In addition, word frequency can also be used to detect synonym substitution. Mozes et al. [140] proposed a frequency-guided word substitution detection method. They assumed that various word substitution methods tend to replace high-frequency words with low-frequency words and designed a rule-based model-independent detection method with statistical data to support this assumption. The experimental results were better than DISP in detecting word substitution, but DISP was more applicable than FGSW in detecting char-level perturbations. However, current methods are adopted to center on the study of char level and word level, while sentence-level paraphrase-type attacks are still under-researched.

4.2. Universal Trigger Detection. In addition to the above detection for the local trigger, there is a defense against the very harmful universal trigger. Wallace et al. [6] looked for universal adversarial triggers that are concatenated-based. Regardless of the input, the model is triggered to produce a specific prediction when a global trigger is added. Le et al. [141] utilized UniTrigger's own strength (the ability to generate a single universal adversarial phrase). Borrowing from the "honeypot" concept in cybersecurity [142], they placed several" trapdoors" on the text DNN classifier to capture and then filter out the malicious examples generated by UniTrigger.

4.3. Enhancement Training Phase. This part of the work aims to enhance model robustness through some empirical training methods. There are three broad categories: adversarial training, robust encoding, and knowledge distillation.

4.3.1. Adversarial Training. Adversarial training (AT) is one of the most effective techniques to improve the robustness of models. It can be further classified into data augmentation, model regularization, and robust optimization [143, 144].

(1) *Data augmentation* uses the generated adversarial examples to extend the original training set. This method is used to enhance the robustness of the reading comprehension system [77]. Experiments show that the system improves the defense against blended adversarial examples to some extent. Similar approaches are also included in [83]. However, such methods are largely static defense methods with limited effectiveness and cannot defend against unknown perturbed data. Jia and Liang [7] performed data-enhanced adversarial training on the text entailment system. Knowledge-based, hand-crafted, and neural-based methods are used to generate data, respectively. They used the idea of the adversarial generative network to integrate the adversarial examples into the optimization process of the classifier. Another idea of data augmentation is that Kang et al. [145] incorporated the average self-negative coding as the input of word embedding into the training data, which has good resistance to char-level perturbations. However, it is not effective for non-character order scrambling.

(2) *Model regularization* treats the adversarial example as a regularization term, which can be expressed as follows:

$$\min\left(J\left(f\left(x\right),y\right)+\lambda J\left(f\left(x'\right),y\right)\right), \tag{25}$$

where λ is the hyperparameter. In a recent study, [146] proposed to improve the robustness of language models by fine-tuning local features with global features from an information-theoretic perspective for word-level adversarial attacks. It contains two mutual information-based regularizers for model training: (1) the information bottleneck regularizer, which is used to suppress the noisy mutual information between the input and feature representations, and (2) the anchored feature tuner, which increases the mutual information between the local stable features and the global features. Miyato et al. [147] used the fast gradient sign method (FGSM) proposed by Goodfellow et al. [166] to compute the perturbation, adding it to the continuous word embedding to generate adversarial input x' and then input it to the model to obtain adversarial loss. By the optimization, the original classification loss (cross-entropy) is added to obtain the new loss, which has achieved good results in text classification tasks. This work is followed by [148–151].

(3) *Robust Optimization*. Madry et al. [152] investigated the problem of adversarial robustness of neural networks from an optimization perspective. It provides a broad, unified view of the study of the adversarial robustness problem of neural networks from an optimization perspective. A saddle-point equation (min-max) is used to describe the adversarial robustness problem rigorously. The following equation is used to precisely describe the security goal we want to achieve:

$$\min_{\theta}\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\Delta x\in\Omega}L\left(x+\Delta x,y;\theta\right)\right]. \tag{26}$$

Al-Dujaili et al. [153] used this method for malware detection DNNs. Previous work uses r l2-or hyper-rectangle. Dong et al. [154] modeled word replacement attacks with the convex hull, using regular terms to reinforce the perturbation of the actual. Also, worst-case adversarial training using generated samples allows the model to achieve the best current robustness against several attacks on sentiment analysis and natural language inference for multiple models.

4.3.2. Robust Encoding. Another part of the work aims to learn the robust word vector, self-supervised pretraining that improves robustness by learning consistent representations under data expansion and adversarial perturbations.

Edizel et al. [155] proposed a method to solve word embeddings for spelling errors. They found that combining fast text with subwords could improve the applicability of existing word embeddings to corpus containing a large number of foreign words in the supervised task of learning spelling error patterns. Jones et al. [156] proposed a framework for robust encoding (RobEn) to build coding. The core part is a function that maps sentences into a smaller discrete encoding space, which satisfies the stability of the encoding mapping while preserving fidelity to unperturbed data. Also, this encoding-based system can easily compute the exact robustness accuracy and avoid the lower bound of the recognized robustness training. Tan et al. [157] merged linguistic information with a data-driven subword encoding scheme, enabling large-scale NLP models to adapt well to non-standard inflected words, preserving the performance of standard English. The method is also effective in improving the vocabulary of existing subword taggers. Dong et al. [154] designed a robust word vector by considering all word vectors around a word.

4.3.3. Computing Robust Lower Bounds. Empirical defenses against attacks carry no guarantee that the robustness of the model is substantially improved, and the robustness of the model needs a quantitative and theoretically guaranteed metric for evaluation. Therefore, we need to calculate model robustness bounds to provide theoretical security guarantees for the robustness of the model. The model robustness bound is for a specific sample, i.e., the model's classification decision for this sample will not vary within this bound.

Most researchers usually use approximation methods to calculate the lower bounds of the model robustness bounds, and the common methods in NLP include randomized smoothing and interval bound propagation.

(1) *Randomized Smoothing*. The concept of randomized smoothing was first introduced to provide stochastic smoothing of top-1 predictions. The robustness of top-1 predictions is guaranteed. The method uses Monte Carlo estimation by sampling n samples $x_1, x_2, \ldots, x_n$ around sample x (equivalent to adding random noise to x) to obtain their $argmax$ category expectations and use them as the final prediction, i.e., $\widehat{f}(x) = 1/n\sum_n f(x_n)$. Recent related work includes [158, 159]. Lecuyer et al. [160] proposed PixelDP, which exploits the connection between the differential privacy and model robustness.

(2) *Interval Bound Propagation (IBP)*. Gowal et al. [161] presented a verifiable robust neural network training method based on the principles of interval boundary propagation (IBP) [162] for the first time. When the input data are perturbed "within the $infty$ parameters" defined by IBP, it is very fast with computational cost comparable to two forward passes of the network, which makes the method highly scalable. Huang et al. [163] demonstrated that the idea of IBP can also be used to analyze the robustness of natural language processing models. Related studies are also included in [164, 165].

4.4. Limitation of API Calls. Finally, to prevent the adversary from repeatedly calling the API multiple times, stealing model information, or even training a shadow model, we could limit the number of API calls. In real scenarios, the attacker cannot often get the training data, model structure, and model parameters of the attacked model and can only achieve the attack purpose through API calls. The current black-box attacks often require a huge amount of API calls, especially for complex AI models. Therefore, some of the unlawful attempts can be circumvented by limiting the amount of API calls from users.

5. Perspectives

By now we have reviewed the relevant works on text adversarial attacks and defensive strategies in the previous sections. The existing problems can be divided into two categories: evasive attacks and defensive strategies, each containing many subsets. Based on our survey, it is easy to find out which approaches are frequently used to solve these problems. We will next discuss about the problems in existing approaches and then give possible implications for research within the field of text adversarial attacks and defenses.

5.1. Discussion. Security analysis of text adversarial attacks and defenses has gained a lot of attention and continues to be a popular theme since 2016. Our survey has the following findings:

(1) *The Trends of Text Adversarial Attacks*. Adversarial examples are essentially data, which can be used for backdoor attacks, testing robustness, or defense. Different purposes require different readability of samples. An attack, for example, requires a higher level of stealth. Also, as DNNs are used in a wider range of applications, there will be a higher demand for research related to improving robustness.

(2) *Performance of Existing Methods*. Most of the earlier approaches used gradient localization of the position to be perturbed before replacement, borrowed from the perturbation generation methods in the image domain. However, its reduction from embedding back to text is not very effective and the strong adversary knowledge is not conducive to deployment. Although the blind-based approach played a great role in expanding the dataset in the early days, it depends on expert knowledge and even manual manipulation. Some later approaches use scores to locate the words to be scrambled. Therefore, it is worth thinking about how to find word-level word replacement methods quickly and accurately using less knowledge, such as decision. Sentence-level perturbations are mostly generative, and it is meaningful and referential to study how to automatically generate sentence-level generations.

5.2. Directions. Although research on textual adversarial attacks and defenses has been undertaken for many years, the discrete and semantic nature of text relative to images leaves many questions to be addressed in this area. In the following, we will discuss future directions and provide some clues for researchers.

(1) *High Quantity*. The discrete nature of text leads to the generation of adversarial examples that can only be hidden by cottage characters or in the form of semantic hiding. The current literature generates adversarial examples with character-level-based perturbations that are easily detected and cannot even bypass grammar checking. Word substitution-based perturbations, based on the substitution of synonyms, slightly change the sentiment tendency or heavily replace the description object. Therefore, we need to generate more stealthy adversarial examples, while making their attack more targeted and even more generalizable as a global trigger.

(2) *High Efficiency*. We can find that early attacks are mostly white-box based, calculating importance by gradient. Later query-based attacks use query results and confidence scores to quantify the importance of words. However, realistic scenarios can often have less knowledge and can only invoke the model interface with limited number of queries. Therefore, we need to design adversarial example generation methods that exploit less access knowledge, fewer queries, and fewer attack times.

(3) *Automation*. The process of generating adversarial examples relies heavily on manual work. From the early stage, we relied heavily on manual retrieval from Wikipedia and other platforms to add relevant and intrusive examples, even in the form of human-in-loop; later on, the sentences were generated according to the rules formulated by human beings; currently, it has evolved into the form of automatic

extraction of rules to generate sentences, which are checked by crowdsourcing review for fluency, or tagged by human beings. It can be seen that the generation process of adversarial examples is gradually developing into automation, and we should also focus on more automated techniques.

(4) *Generation*. Most of the research has focused on the form of word substitution and character substitution, making the sentences generated in this way tend to have a single form and deviate more or less from the semantics of the original sample. A few works have used Seq2Seq, VAE, GAN, RL, and other techniques to generate sentence-level perturbation. Preliminary attempts can transform the form of the sentences, but the results are not very static. Therefore, the next step can be considered to introduce related techniques, even pretrained models, to try to generate semantics-preserving sentence-level adversarial samples, or even consider generating longer adversarial examples, such as paragraph level and chapter level.

(5) *Application*. Current work mostly generates adversarial examples on public datasets such as sentiment analysis, text classification, and machine translation. Compared to the image and video-based adversarial example such as face recognition and autonomous driving, textual adversarial example cause less social impact. Therefore, we can focus on more security-related scenarios [167], such as language style forgery, fake news recognition, malicious code, and other related areas of research. We can also consider studying multi-lingual adversarial instances in the context of the ethnicity of the researchers, as well as studying textual adversarial applications in more practical scenarios such as mental health [168, 169], android applications [170–173], Ethereum smart contract [174], network traffic [175], and discrimination [176].

(6) *Interpretability*. So far, researchers have mostly focused on how they can generate more effective adversarial examples, but few studies have focused on why the phenomenon of adversarial examples is generated. The next step could be research related to interpretability, such as the mechanism of models making predictions, or the attackability of examples [177]. Also consider introducing knowledge graphs to give models common sense reasoning capabilities to fundamentally improve the robustness of models.

(7) *New Phase*. Finally, the attacks discussed in this paper all occur during the testing phase, i.e., the evasion attack. At this point, the adversary has at most white-box knowledge and cannot change the model itself. In the past, this indeed fit the adversary's privilege. But with the advent of federated learning technology, more and more DNN models can be deployed on local clients, which also means that users can control the deployment of local models and even influence the global model by manipulating the gradients uploaded by local models. This is called a poisoning attack, which is undoubtedly a form of attack with a large and more damaging impact. There are already some language models [178], OOV prediction, and other systems that use federated learning architecture, and in consequence, research on poisoning attacks is urgently needed.

6. Related Surveys

Han et al. [179] presented a comprehensive and systematic introduction to adversarial attacks and defenses in DNN models with examples of three data types: images, graphs, and text. In addition to evasion attacks, poisoning attacks are also introduced, taking into account the full cycle of DNNs. It also adds the application of the real world and the importance people attach to such attacks. However, the focus of it is around IMAGES data, and the description of the attack and defense of TEXT is relatively little, almost passing by. DeepSec [180] was the first platform to integrate attack and defense tools, followed by the adversarial robustness toolbox (ART) [181]. Both are comprehensive, but they are about adversarial example generation for images. The textual domain is equally practical, and several related platforms are available for evaluation.

TextAttack [182] is the earliest implemented tool framework for text attacks. They integrated 16 attack forms into the framework and split the attack into components, which can facilitate researchers to design new elements and evaluate the effect of the attack. It can also use perturbed data to train the system against and enhance the data. OpenAttack [76] had a more detailed classification of attacks, classifying adversary knowledge into gradient/score/decision/blind-based and perturbation levels into sentence-level, word-level, char-level, and multi-level. In particular, they designed a more comprehensive form of attack than TextAttack, including both sentence-level and decision/blind-based attacks. TextFlint [183] is a robustness assessment platform. It combines linguistic knowledge to design a set of transformations for generating perturbations and uses these data to analyze the robustness of multiple SOTA models for mainstream NLP tasks. However, the above three articles focus on practice rather than classification, which has some reference value for classification but lacks rigor.

There is no shortage of reviews on textual adversarial attacks and defenses. The framework of Zhang et al. [184] is the most comprehensive one, but their classification of attacks is unsystematic and the granularity is coarse-grained. Although the granularity of the attack classification in [185] is relatively fine, it does not incorporate the adversary's knowledge, and the relationship between related attack methods is not defined clearly enough. The classification of text attacks in [186] is more task-oriented and biased towards the application perspective. At the same time, these articles were generally written early and did not cover the major changes in attack techniques over the years, and thus the cases are of insufficient novelty.

In this paper, not only are the two main lines of NLP pipelines and adversarial attacks highly integrated and self-consistent in terms of knowledge, but it also provides a more systematic summary and refinement of the subcategories including attack, defense, and evaluation. It is both theoretically and experimentally instructive in the field of textual adversarial attacks and defenses and a review with both security and NLP expertise.

7. Conclusion

In this work, we conduct a thorough investigation in the area of text adversarial attacks and defenses of DNNs. For the classification of attack methods, we provide deeper insights, combining fine-grained attack methods, from char to sentence level, with the adversary knowledge, from gradient to blind. For defense methods, we consider their security step by step in a real-life pipeline in conjunction with NLP. Finally, we discuss and provide perspectives for text adversarial attacks and defenses.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported in part by the National Key R&D Program of China under grant no. 2020YFB2103802 and in part by the Fundamental Research Funds for the Central Universities of China under grant no. KKJB320001536.

References

[1] Z. Gu, Le Wang, X. Chen et al., "Epidemic risk assessment by a novel communication station based method," *IEEE Transactions on Network Science and Engineering*, vol. 9, 2021.

[2] K. Kowsari, D. E. Brown, M. Heidarysafa, K. Jafari Meimandi, M. S. Gerber, and L. E. Barnes, "Hdltex: Hierarchical deep learning for text classification," in *Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 364–371, IEEE, Cancun, Mexico, December 2017.

[3] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: state of the art, current trends and challenges," *CoRR, Abs/1708*, 2017, http://arxiv.org/abs/1708.05148, Article ID 05148.

[4] Z. Yu, J. Yu, C. Xiang, J. Fan, and D. Tao, "Beyond bilinear: generalized multimodal factorized high-order pooling for visual question answering," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 12, pp. 5947–5959, 2018.

[5] C. Szegedy, "Google inc, wojciech zaremba, ilya sutskever, google inc, joan bruna, dumitru erhan, google inc, ian goodfellow, and rob fergus. intriguing properties of neural networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, Banff, Canada, April 2014.

[6] E. Wallace, F. Shi, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing Nlp," 2019, https://arxiv.org/abs/1908.07125.

[7] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," 2017, https://arxiv.org/abs/1707.07328.

[8] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[9] A. Rajaraman and J. D. Ullman, *Data Mining*, Cambridge University Press, Cambridge, UK, 2011.

[10] D. Harris and S. L. Harris, *Digital Design and Computer Architecture*, Morgan Kaufmann, Burlington, MA, USA, 2010.

[11] Y. Bengio, L. Pascal, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in Neural Information Processing Systems*, vol. 19, p. 153, 2007.

[12] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Science China Technological Sciences*, pp. 1–26, 2020.

[13] K. Yoon, Y. Jernite, D. Sontag, and R. Alexander, "Character-aware neural language models," *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.

[14] R. Sennrich, H. Barry, and A. Birch, "Neural machine translation of rare words with subword units," 2015, https://arxiv.org/abs/1508.07909.

[15] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[16] R. Frank, *Perceptions and the Theory of Brain Mechanisms*, Springer, Berlin, Germany, 1962.

[17] Y. Bengio, R. . Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[18] J. Armand, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," 2016, https://arxiv.org/abs/1607.01759.

[19] M. T. Ribeiro, S. Singh, and C. Guestrin, "Semantically equivalent adversarial rules for debugging nlp models," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 856–865, Melbourne, Australia, July 2018.

[20] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *Proceedings of the European Symposium on Research in Computer Security*, pp. 62–79, Springer, Oslo, Norway, September 2017.

[21] J. M. Springer, B. Marie Reinstadler, and U.-M. O'Reilly, "Strata: building robustness with a simple method for generating black-box adversarial attacks for models of code," 2020, https://arxiv.org/abs/2009.13562.

[22] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," 2016, https://arxiv.org/abs/1606.04435.

[23] D. Xu, Z. Tian, R. Lai, X. Kong, Z. Tan, and W. Shi, "Deep learning based emotion analysis of microblog texts," *Information Fusion*, vol. 64, no. 1–11, 2020.

[24] K. Yoon, "Convolutional Neural Networks for Sentence classification," *CoRR, Abs/1408*, vol. 5882, 2014, http://arxiv.org/abs/1408.5882.

[25] X. Zhang, J. Zhao, and Y. Le Cun, "Character-level convolutional networks for text classification," 2015, https://arxiv.org/abs/1509.01626.

[26] X. Zhang and Y. LeCun, "Text understanding from scratch," *CoRR, abs/1502*, 2015, http://arxiv.org/abs/1502.01710, Article ID 01710.

[27] Ji Gao, J. Lanchantin, M. Lou Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW)*, pp. 50–56, IEEE, San Francisco, CA, USA, May 2018.

[28] J. Ebrahimi, A. Rao, L. Daniel, and D. Dou, "Hotflip: white-box adversarial examples for text classification," 2017, https://arxiv.org/abs/1712.06751.

[29] B. Liang, H. Li, M. Su, B. Pan, X. Li, and W. Shi, "Deep text classification can be fooled," 2017, https://arxiv.org/abs/1704.08006.

[30] J. Ebrahimi, L. Daniel, and D. Dou, "On adversarial examples for character-level neural machine translation," 2018, https://arxiv.org/abs/1806.09030.

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[32] K. Cho, Bart van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *CoRR, abs/1406*, vol. 1078, 2014, http://arxiv.org/abs/1406.1078.

[33] C. Luo, Z. Tan, G. Min, J. Gan, W. Shi, and Z. Tian, "A novel web attack detection system for internet of things via ensemble classification," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5810–5818, 2020.

[34] M. Schuster, K. K. Paliwal, and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[35] K. Sheng Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," 2015, https://arxiv.org/abs/1503.00075.

[36] Q. Chen, X. Zhu, Z. Ling, S. Wei, H. Jiang, and D. Inkpen, "Enhanced Lstm for natural language inference," 2016, https://arxiv.org/abs/1609.06038.

[37] Tim Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiskỳ, and P. Blunsom, "Reasoning about entailment with neural attention," 2015, https://arxiv.org/abs/1509.06664.

[38] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," 2016, https://arxiv.org/abs/1606.01933.

[39] S. Wang and J. Jiang, "Machine comprehension using match-lstm and answer pointer," 2016, https://arxiv.org/abs/1608.07905.

[40] M. Seo, A. Kembhavi, F. Ali, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," 2016, https://arxiv.org/abs/1611.01603.

[41] L. Qi, L. Wu, P.-Y. Chen, A. G. Dimakis, S. D. Inderjit, and M. Witbrock, "Discrete adversarial attacks and submodular optimization with applications to text classification," 2018, https://arxiv.org/abs/1812.00151.

[42] M. Sato, J. Suzuki, H. Shindo, and Y. Matsumoto, "Interpretable adversarial perturbation in input embedding space for text," 2018, https://arxiv.org/abs/1805.02917.

[43] M. Iyyer, W. John, K. Gimpel, and L. Zettlemoyer, "Adversarial example generation with syntactically controlled paraphrase networks," 2018, https://arxiv.org/abs/1804.06059.

[44] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," 2017, https://arxiv.org/abs/1710.11342.

[45] P. Minervini and S. Riedel, "Adversarially regularising neural nli models to integrate logical background knowledge," 2018, https://arxiv.org/abs/1808.08609.

[46] I. Vlad Serban, A. Sordoni, R. Lowe et al., "A hierarchical latent variable encoder-decoder model for generating dialogues," 2016, http://arxiv.org/abs/1605.06069, Article ID 06069.

[47] I. Vlad Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau, "Hierarchical neural network generative models for movie dialogues," 2015. URL http://arxiv.org/abs/1507.04808, Article ID 04808.

[48] G. Klein, K. Yoon, Y. Deng, S. Jean, M. Alexander, and R. Opennmt, "Open-source toolkit for neural machine translation," 2017, http://arxiv.org/abs/1701.02810, Article ID 02810.

[49] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, https://arxiv.org/abs/1409.0473.

[50] R. Lowe, P. Nissan, I. Serban, and J. Pineau, "The ubuntu dialogue corpus: a large dataset for research in unstructured multi-turn dialogue systems," 2015, http://arxiv.org/abs/1506.08909.

[51] M. Cheng, J. Yi, H. Zhang, P.-Yu Chen, and C.-J. Hsieh, "Seq2sick: evaluating the robustness of sequence-to-sequence models with adversarial examples," *CoRR, Abs/1803*, 2018, http://arxiv.org/abs/1803.01128, Article ID 01128.

[52] W. C. Gan, H. T. Ng, and H. Tou Ng, "Improving the robustness of question answering systems to question paraphrasing," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6065–6075, Association for Computational Linguistics, Florence, Italy, July 2019, https://www.aclweb.org/anthology/P19-1610.

[53] Y. Wang and M. Bansal, "Robust machine comprehension models via adversarial training," *CoRR, abs/*, vol. 1804, 2018 http://arxiv.org/abs/1804.06473, Article ID 06473.

[54] M. T. Ribeiro, S. Singh, and C. Guestrin, "Semantically equivalent adversarial rules for debugging NLP models," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 856–865, Association for Computational Linguistics, Melbourne, Australia, July 2018, https://www.aclweb.org/anthology/P18-1079.

[55] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems. *corr*, abs/1707," 2017, http://arxiv.org/abs/1707.07328, Article ID 07328.

[56] A. R. Sharma and P. Kaushik, "Literature survey of statistical, deep and reinforcement learning in natural language processing," in *Proceedings of the 2017 International Conference on Computing, Communication and Automation (ICCCA)*, pp. 350–354, IEEE, Noida, India, May 2017.

[57] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, "Deep reinforcement learning for dialogue generation," 2016, https://arxiv.org/abs/1606.01541.

[58] W. Han, L. Zhang, Y. Jiang, and K. Tu, "Adversarial attack and defense of structured prediction models," 2020, https://arxiv.org/abs/2010.01610.

[59] M. Cheng, W. Wei, and C.-J. Hsieh, "Evaluating and enhancing the robustness of dialogue systems: a case study on a negotiation agent," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3325–3335, 2019.

[60] N. Tong and M. Bansal, "Adversarial over-sensitivity and over-stability strategies for dialogue models," 2018, https://arxiv.org/abs/1809.02079.

[61] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, https://arxiv.org/abs/1412.6572.

[62] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, https://arxiv.org/abs/1312.6114.

[63] T. Iqbal and S. Qureshi, "The survey: text generation models in deep learning," *Journal of King Saud University - Computer and Information Sciences*, 2020, https://www.sciencedirect.com/science/article/pii/S1319157820303360.

[64] T. Le, S. Wang, and D. Lee, "Malcom: generating malicious comments to attack neural fake news detection models," 2020, https://arxiv.org/abs/2009.01048.

[65] G. H de Rosa and J. P. Papa, "A survey on text generation using generative adversarial networks," *Pattern Recognition*, vol. 119, Article ID 108098, 2021.

[66] X. Zhang, Q. Yang, S. Albaradei et al., "Rise and fall of the global conversation and shifting sentiments during the covid-19 pandemic," *Humanities and Social Sciences Communications*, vol. 8, no. 1, pp. 1–10, 2021.

[67] X. Wei, G. Xu, H. Wang, Y. He, Z. Han, and W. Wang, "Sensing users' emotional intelligence in social networks," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 1, pp. 103–112, 2019.

[68] W. Wang, X. Wei, X. Suo et al., "Hgate: heterogeneous graph attention auto-encoders," *IEEE Transactions on Knowledge and Data Engineering*, vol. 1, p. 1, 2021.

[69] B. McCann, B. James, C. Xiong, and R. Socher, "Learned in translation: contextualized word vectors," 2017, https://arxiv.org/abs/1708.00107.

[70] M. E. Peters, M. Neumann, M. Iyyer et al., "Deep contextualized word representations," 2018, https://arxiv.org/abs/1802.05365.

[71] A. Radford, K. Narasimhan, Tim Salimans, and I. Sutskever, *Improving Language Understanding by Generative Pre-training*, 2018.

[72] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: pre-training of deep bidirectional transformers for language understanding," 2018, https://arxiv.org/abs/1810.04805.

[73] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," 2017, https://arxiv.org/abs/1706.03762.

[74] J. P.-A. Goodfellow, M. Mirza, B. Xu et al., "Generative adversarial networks," 2014, https://arxiv.org/abs/1406.2661.

[75] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[76] G. Zeng, F. Qi, Q. Zhou et al., "Openattack: an open-source textual adversarial attack toolkit," 2020, https://arxiv.org/abs/2009.09191.

[77] T. Hazan, P. George, and D. Tarlow, *Adversarial Perturbations of Deep Neural Networks*, MIT Press, Cambridge, MA, USA, 2017.

[78] C. Szegedy, W. Zaremba, I. Sutskever et al., "Intriguing properties of neural networks," 2013, https://arxiv.org/abs/1312.6199.

[79] S. Eger, G. Gül Şahin, A. Rücklé et al., "Text processing like humans do: visually attacking and shielding nlp systems," 2019, https://arxiv.org/abs/1903.11508.

[80] J. E. Ebrahimi, D. lowd, and D. dou, "On adversarial examples for character-level neural machine translation."" on adversarial examples for character-level neural machine translation," 2018, https://arxiv.org/abs/1806.09030.

[81] J. Li, S. Ji, T. Du, Bo Li, and T. Wang, "Textbugger: generating adversarial text against real-world applications," 2018, https://arxiv.org/abs/1812.05271.

[82] Y. Gil, Y. Chai, Or Gorodissky, and J. Berant, "White-to-black: efficient distillation of black-box adversarial attacks," 2019, https://arxiv.org/abs/1904.02405.

[83] Y. Belinkov and Y. Bisk, "Synthetic and natural noise both break neural machine translation," 2017, https://arxiv.org/abs/1711.02173.

[84] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *Proceedings of the MILCOM 2016-2016 IEEE Military Communications Conference*, pp. 49–54, IEEE, Baltimore, MD, USA, November 2016.

[85] S. Samanta and S. Mehta, "Generating adversarial text samples," in *Proceedings of the European Conference on Information Retrieval*, pp. 744–749, Springer, Grenoble, France, March 2018.

[86] S. Mohsen Moosavi-Dezfooli, A. Fawzi, and F. Pascal, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, Las Vegas, NV, USA, June 2016.

[87] P. Yang, J. Chen, C.-J. Hsieh, J.-L. Wang, and M. I. Jordan, "Greedy attack and gumbel attack: generating adversarial examples for discrete data," *Journal of Machine Learning Research*, vol. 21, no. 43, pp. 1–36, 2020.

[88] Y. Cheng, L. Jiang, and W. Macherey, "Robust neural machine translation with doubly adversarial inputs," 2019, https://arxiv.org/abs/1906.02443.

[89] X. Zheng, J. Zeng, Yi Zhou, C.-J. Hsieh, M. Cheng, and X.-J. Huang, "Evaluating and enhancing the robustness of neural network-based dependency parsing models with adversarial examples," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6600–6610, Singapore, July 2020.

[90] M. Zhao and R. Wattenhofer, "A geometry-inspired attack for generating natural language adversarial examples," 2020, https://arxiv.org/abs/2010.01345.

[91] D. Jin, Z. Jin, J. Tianyi Zhou, J. T. Zhou, and P. Szolovits, "Is bert really robust? a strong baseline for natural language attack on text classification and entailment," *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 5, pp. 8018–8025, 2020.

[92] Z. Shi, T. Yao, J. Xu, and M. Huang, "Robustness to modification with shared words in paraphrase identification," 2019, https://arxiv.org/abs/1909.02560.

[93] M. Blohm, G. Jagfeld, E. Sood, Y. Xiang, and N. Thang Vu, "Comparing attention-based convolutional and recurrent neural networks: success and limitations in machine reading comprehension," 2018, https://arxiv.org/abs/1808.08744.

[94] Y.-L. Hsieh, M. Cheng, D. Cheng Juan, W. Wei, W.-L. Hsu, and C.-J. Hsieh, "On the robustness of self-attentive models," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1520–1529, Florence, Italy, July 2019.

[95] M. Behjati, S. Mohsen Moosavi-Dezfooli, M. S. Baghshah, and F. Pascal, "Universal adversarial attacks on text classifiers," in *Proceedings of the ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7345–7349, IEEE, Brighton, UK., May 2019.

[96] M. Cheng, J. Yi, P.-Y. Chen, H. Zhang, and C.-J. Hsieh, "Seq2sick: evaluating the robustness of sequence-to-sequence models with adversarial examples," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 4, pp. 3601–3608, 2020.

[97] H. Zhang, H. Zhou, M. Ning, and L. Li, "Generating fluent adversarial examples for natural languages," 2020, https://arxiv.org/abs/2007.06174.

[98] L. Song, X. Yu, H.-T. Peng, and K. Narasimhan, "Universal adversarial attacks with natural triggers for text classification," 2020, https://arxiv.org/abs/2005.00174.

[99] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 1085–1097, Florence, Italy, July 2019.

[100] M. Alzantot, Y. Sharma, E. Ahmed, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," 2018, https://arxiv.org/abs/1804.07998.

[101] Y. Fan, Q. Long, T. Meng, and K.-W. Chang, "On the robustness of language encoders against grammatical errors," 2020, https://arxiv.org/abs/2005.05683.

[102] J. X. M. Jin Yong Yoo, E. Lifland, and Y. Qi, "Searching for a search method: benchmarking search algorithms for generating nlp adversarial examples," 2020, https://arxiv.org/abs/2009.06368.

[103] Z. Yuan, F. Qi, C. Yang et al., "Word-level textual adversarial attacking as combinatorial optimization," 2019, https://arxiv.org/abs/1910.12196.

[104] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "Bert-attack: adversarial attack against bert using bert," 2020, https://arxiv.org/abs/2004.09984.

[105] D. Li, Y. Zhang, H. Peng et al., "Contextualized perturbation for textual adversarial attack," 2020, https://arxiv.org/abs/2009.07502.

[106] S. Garg and G. R. Bae, "Bert-based adversarial examples for text classification," 2020, https://arxiv.org/abs/2004.01970.

[107] R. Maheshwary, S. Maheshwary, and V. Pudi, "Generating natural language attacks in a hard label black box setting," 2020, https://arxiv.org/abs/2012.14956.

[108] W. Zou, S. Huang, J. Xie, X. Dai, and J. Chen, "A reinforced generation of adversarial examples for neural machine translation," 2019, https://arxiv.org/abs/1911.03677.

[109] D. Dahlmeier, H. Tou Ng, and S. M. Wu, "Building a large annotated corpus of learner English: the nus corpus of learner English," in *Proceedings Of The 8th Workshop On Innovative Use Of Nlp For Building Educational Applications*, pp. 22–31, Atlanta, GA, USA, June 2013.

[110] M. Glockner, V. Shwartz, and Y. Goldberg, "Breaking nli systems with sentences that require simple lexical inferences," 2018, https://arxiv.org/abs/1805.02266.

[111] S. Tan, S. Joty, M.-Y. Kan, and R. Socher, "It's morphin'time! combating linguistic discrimination with inflectional perturbations," 2020, https://arxiv.org/abs/2005.04364.

[112] P. Minervini, T. Demeester, Tim Rocktäschel, and S. Riedel, "Adversarial sets for regularising neural link predictors," 2017, https://arxiv.org/abs/1707.07596.

[113] Y. Wang and M. Bansal, "Robust machine comprehension models via adversarial training," 2018, https://arxiv.org/abs/1804.06473.

[114] Y. Zhang, J. Baldridge, and L. He, "Paws: paraphrase adversaries from word scrambling," 2019, https://arxiv.org/abs/1904.01130.

[115] E. Wallace, P. Rodriguez, S. Feng, I. Yamada, and J. Boyd-Graber, "Trick me if you can: human-in-the-loop generation of adversarial examples for question answering," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 387–401, 2019.

[116] T. Wang, X. Wang, Q. Yao et al., "Cat-gen: improving robustness in nlp models via controlled adversarial text generation," 2020, https://arxiv.org/abs/2010.02338.

[117] B. Wang, H. Pei, B. Pan, Q. Chen, S. Wang, and B. Li, "T3: tree-autoencoder constrained adversarial text generation for targeted attack," 2019, https://arxiv.org/abs/1912.10375.

[118] W. C. Gan and H. Tou Ng, "Improving the robustness of question answering systems to question paraphrasing," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6065–6075, Florence, Italy, July 2019.

[119] P. Vijayaraghavan and D. Roy, "Generating black-box adversarial examples for text classifiers using a deep reinforced model," in *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 711–726, Springer, Bilbao, Spain, September 2019.

[120] F. Van Der Heijden, R. P. Duin, D. De Ridder, and D. M. J. Tax, *Classification, Parameter Estimation and State Estimation: An Engineering Approach Using MATLAB*, John Wiley & Sons, Hoboken, NJ, USA, 2005.

[121] K. Ryan, Y. Zhu, R. Salakhutdinov et al., "Skip-thought vectors," 2015, https://arxiv.org/abs/1506.06726.

[122] D. Cer, Y. Yang, S.-Y. Kong et al., "Universal sentence encoder," 2018, https://arxiv.org/abs/1803.11175.

[123] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and B. Antoine, "Supervised learning of universal sentence representations from natural language inference data," 2017, https://arxiv.org/abs/1705.02364.

[124] N. Reimers and I. Gurevych, "Sentence-bert: sentence embeddings using siamese bert-networks," 2019, https://arxiv.org/abs/1908.10084.

[125] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, pp. 707–710, 1966.

[126] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *Proceedings of the International Conference on Machine Learning*, pp. 957–966, PMLR, Lille, France, July 2015.

[127] J. Paul, "The distribution of the flora in the alpine zone. 1," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[128] D. Naber, *A Rule-Based Style and Grammar Checker*, GRIN Verlag, Munich, Germnay, 2003.

[129] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[130] A. Holtzman, B. Jan, M. Forbes, B. Antoine, D. Golub, and Y. Choi, "Learning to write with cooperative discriminators," 2018, https://arxiv.org/abs/1805.06087.

[131] R. Jozefowicz, Oriol Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," 2016, https://arxiv.org/abs/1602.02410.

[132] F. J. Newmeyer, *Grammatical Theory: Its Limits and its Possibilities*, University of Chicago Press, Chicago, IL, USA, 1983.

[133] B. Lambert, R. Singh, and B. Raj, "Creating a linguistic plausibility dataset with non-expert annotators," in *Proceedings of the 11th Annual Conference of the International*

Speech Communication Association, Makuhari, China, September 2010.

[134] E. Mays, F. J. Damerau, and R. L. Mercer, "Context based spelling correction," *Information Processing & Management*, vol. 27, no. 5, pp. 517–522, 1991.

[135] A. Islam and D. Inkpen, "Real-word spelling correction using google web 1t 3-grams," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 1241–1249, Singapore, August 2009.

[136] K. Sakaguchi, M. Post, and B. Van Durme, "Grammatical error correction with neural reinforcement learning," 2017, https://arxiv.org/abs/1707.00299.

[137] W. Q. Wang, R. Wang, L. N. Wang, and B. X. Tang, "Adversarial examples generation approach for tendency classification on Chinese texts," *Journal of Software*, vol. 30, no. 8, pp. 2415–2427, 2019.

[138] X. Wang, H. Jin, and K. He, "Natural language adversarial attacks and defenses in word level," 2019, https://arxiv.org/abs/1909.06723.

[139] Y. Zhou, J.-Yu Jiang, K.-W. Chang, and W. Wang, "Learning to discriminate perturbations for blocking adversarial attacks in text classification," 2019, https://arxiv.org/abs/1909.03084.

[140] M. Mozes, P. Stenetorp, K. Bennett, and L. D. Griffin, "Frequency-guided word substitutions for detecting textual adversarial examples," 2020, https://arxiv.org/abs/2004.05887.

[141] T. Le, N. Park, and D. Lee, "Detecting universal trigger's adversarial attack with honeypot," *CoRR*, vol. 10492, 2020, https://arxiv.org/abs/2011.10492.

[142] Y. Sun, Z. Tian, M. Li, Su Shen, X. Du, and M. Guizani, "Honeypot identification in softwarized industrial cyber–physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5542–5551, 2020.

[143] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: a survey," *Ieee Access*, vol. 6, pp. 14410–14430, 2018.

[144] B. Biggio and F. Roli, "Wild patterns: ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018.

[145] D. Kang, T. Khot, A. Sabharwal, and E. Hovy, "Adventure: adversarial training for textual entailment with knowledge-guided examples," 2018, https://arxiv.org/abs/1805.04680.

[146] B. Wang, S. Wang, Y. Cheng et al., "Infobert: improving robustness of language models from an information theoretic perspective," 2020, https://arxiv.org/abs/2010.02329.

[147] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," 2017, https://arxiv.org/abs/1605.07725.

[148] M. Sato, J. Suzuki, H. Shindo, and Y. Matsumoto, "Interpretable adversarial perturbation in input embedding space for text," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI'18*, pp. 4323–4330, AAAI Press, Stockholm, Sweden, July 2018.

[149] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder, "Adversarial training for multi-context joint entity and relation extraction," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2830–2836, Association for Computational Linguistics, Brussels, Belgium, November 2018, https://www.aclweb.org/anthology/D18-1307.

[150] M. Yasunaga, J. Kasai, and D. Radev, "Robust multilingual part-of-speech tagging via adversarial training," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 976–986, Association for Computational Linguistics, New Orleans, LA, USA, June 2018, https://www.aclweb.org/anthology/N18-1089.

[151] Y. Wu, D. Bamman, and S. Russell, "Adversarial training for relation extraction," in *Proceedings of the 2017 Conference on Empirical Methods in Natural*, pp. 1778–1783, Association for Computational Linguistics, Copenhagen, Denmark, September 2017, https://www.aclweb.org/anthology/D17-1187.

[152] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and V. Adrian, "towards deep learning models resistant to adversarial attacks," 2017, https://arxiv.org/abs/1706.06083.

[153] A. Al-Dujaili, A. Huang, E. Hemberg, and U.-M. O'Reilly, "Adversarial deep learning for robust detection of binary encoded malware," in *Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW)*, pp. 76–82, IEEE, San Francisco, CA, USA, May 2018.

[154] X. Dong, L. Anh Tuan, R. Ji, and H. Liu, "Towards robustness against natural language word substitutions," in *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, May 2021.

[155] B. Edizel, A. Piktus, P. Bojanowski, R. Ferreira, E. Grave, and F. Silvestri, "Misspelling oblivious word embeddings," 2019, https://arxiv.org/abs/1905.09755.

[156] E. Jones, R. Jia, A. Raghunathan, and P. Liang, "Robust encodings: a framework for combating adversarial typos," 2020, https://arxiv.org/abs/2005.01229.

[157] S. Tan, S. Joty, L. R. Varshney, and M.-Y. Kan, "Mind your inflections! improving nlp for non-standard english with base-inflection encoding," 2020, https://arxiv.org/abs/2004.14870.

[158] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proceedings of the International Conference on Machine Learning*, pp. 1310–1320, PMLR, Long Beach, CA, USA, June 2019.

[159] M. Ye, C. Gong, and Q. Liu, "Safer: a structure-free approach for certified robustness to adversarial word substitutions," 2020, https://arxiv.org/abs/2005.14424.

[160] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," 2019, https://arxiv.org/abs/1802.03471.

[161] S. Gowal, K. Dvijotham, R. Stanforth et al., "Scalable verified training for provably robust image classification," in *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4841–4850, Seoul, Korea, October 2019.

[162] T. Sunaga, "Theory of an interval algebra and its application to numerical analysis [reprint of res. assoc. appl. geom. mem. 2," *Japan Journal of Industrial and Applied Mathematics*, vol. 26, no. 2-3, pp. 125–143, 2009.

[163] P.-S. Huang, R. Stanforth, J. Welbl et al., "Achieving verified robustness to symbol substitutions via interval bound propagation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*https://www.aclweb.org/anthology/D19-1419, Hong Kong, China, November 2019.

[164] Z. Shi, H. Zhang, K.-W. Chang, M. Huang, and C.-J. Hsieh, "Robustness verification for transformers," in *Proceedings of the International Conference on Learning Representations*, Addis Ababa, Ethiopia, April 2020, https://openreview.net/forum?id=BJxwPJHFwS.

[165] R. Jia, A. Raghunathan, K. Göksel, and P. Liang, "Certified robustness to adversarial word substitutions," in *Proceedings*

of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 4129–4142https://www.aclweb.org/anthology/D19-1423, Hong Kong, China, November 2019.

[166] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA, May 2015, http://arxiv.org/abs/1412.6572.

[167] H. Du, S. Wang, and H. Huo, "Xfinder: detecting unknown anomalies in distributed machine learning scenario," *Frontiers of Computer Science*, p. 83.

[168] N. Wang, M. Wang, X. Xin et al., "Exploring the relationship between anxiety, depression, and sleep disturbance among hiv patients in China from a network perspective," *Frontiers in Psychiatry*, vol. 12, 2021.

[169] W. Xie, M. Ji, M. Zhao et al., "Detecting symptom errors in neural machine translation of patient health information on depressive disorders: developing interpretable bayesian machine learning classifiers," *Frontiers in Psychiatry*, vol. 12, Article ID 771562, 2021.

[170] W. Wang, Y. Li, X. Wang, J. Liu, and X. Zhang, "Detecting android malicious apps and categorizing benign apps with ensemble of classifiers," *Future Generation Computer Systems*, vol. 78, pp. 987–994, 2018.

[171] W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, and X. Zhang, "Exploring permission-induced risk in android applications for malicious application detection," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 11, pp. 1869–1882, 2014.

[172] W. Wang, M. Zhao, and J. Wang, "Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 8, pp. 3035–3043, 2019.

[173] X. Liu, J. Liu, S. Zhu, W. Wang, and X. Zhang, "Privacy risk analysis and mitigation of analytics libraries in the android ecosystem," *IEEE Transactions on Mobile Computing*, vol. 19, no. 5, pp. 1184–1199, 2019.

[174] W. Wang, J. Song, G. Xu, Y. Li, H. Wang, and C. Su, "Contractward: automated vulnerability detection models for ethereum smart contracts," *IEEE Transactions on Network Science and Engineering*, vol. 8, 2020.

[175] W. Wang, Y. Shang, Y. He, Y. Li, and J. Liu, "BotMark: automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors," *Information Sciences*, vol. 511, pp. 284–296, 2020.

[176] P. Rao and M. Taboada, "Gender bias in the news: a scalable topic modelling and visualization framework," *Frontiers in Artificial Intelligence*, vol. 4, 2021.

[177] Z. Yang, Y. Han, and X. Zhang, "Characterizing the evasion attackability of multi-label classifiers," 2020, https://arxiv.org/abs/2012.09427.

[178] M. Li, Y. Sun, H. Lu, S. Maharjan, and Z. Tian, "Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6266–6278, 2019.

[179] X. Han, M. Yao, H.-C. Liu et al., "Adversarial attacks and defenses in images, graphs and text: a review," *International Journal of Automation and Computing*, vol. 17, no. 2, pp. 151–178, 2020.

[180] L. Xiang, S. Ji, J. Zou et al., "Deepsec: a uniform platform for security analysis of deep learning model," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, pp. 673–690, IEEE, San Francisco, CA, USA, May 2019.

[181] N. Maria-Irina, M. Sinn, M. Ngoc Tran et al., "Adversarial robustness toolbox v1.2.0," *CoRR*, vol. 1807, 2018 https://arxiv.org/pdf/1807.01069, Article ID 01069.

[182] J. X. Morris, E. Lifland, Y. Yoo, and Y. Textattack, "A framework for adversarial attacks in natural language processing," 2020, https://arxiv.org/abs/2005.05909.

[183] T. Gui, X. Wang, Q. Zhang et al., "Textflint: unified multilingual robustness evaluation toolkit for natural language processing," 2021, https://arxiv.org/abs/2103.11441.

[184] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep-learning models in natural language processing," *ACM Transactions on Intelligent Systems and Technology*, vol. 11, no. 3, pp. 1–41, 2020.

[185] A. Huq and M. Pervin, "Adversarial attacks and defense on texts: a survey," 2020, https://arxiv.org/abs/2005.14108.

[186] W. Wang, R. Wang, L. Wang, Z. Wang, and A. Ye, "Towards a robust deep neural network in texts: a survey," 2019, https://arxiv.org/abs/1902.07285.