Improving the Reliability of Deep Neural Networks in NLP: A Review

Article *in* Knowledge-Based Systems · March 2020

DOI: 10.1016/j.knosys.2019.105210

Basemah Alshemali
University of Colorado Colorado Springs

Knowledge-Based Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Improving the Reliability of Deep Neural Networks in NLP: A Review[☆]

Basemah Alshemali[a,b,*], Jugal Kalita[b]

[a] Taibah University, Al-Medina, Saudi Arabia
[b] University of Colorado at Colorado Springs, Colorado Springs, CO, USA

ARTICLE INFO

Article history:
Received 12 February 2019
Received in revised form 21 October 2019
Accepted 7 November 2019
Available online xxxx

Keywords:
Adversarial examples
Adversarial texts
Natural language processing

ABSTRACT

Deep learning models have achieved great success in solving a variety of natural language processing (NLP) problems. An ever-growing body of research, however, illustrates the vulnerability of deep neural networks (DNNs) to adversarial examples — inputs modified by introducing small perturbations to deliberately fool a target model into outputting incorrect results. The vulnerability to adversarial examples has become one of the main hurdles precluding neural network deployment into safety-critical environments. This paper discusses the contemporary usage of adversarial examples to foil DNNs and presents a comprehensive review of their use to improve the robustness of DNNs in NLP applications. In this paper, we summarize recent approaches for generating adversarial texts and propose a taxonomy to categorize them. We further review various types of defensive strategies against adversarial examples, explore their main challenges, and highlight some future research directions.

1. Introduction

Deep neural networks (DNNs) have gained considerable popularity in tackling a wide variety of machine learning problems such as image classification [1], object detection [2], and malware detection [3]. In addition, deep learning has been employed to learn effective representations for patient records [4] in order to conduct predictive modeling [5] and support disease phenotyping [6]. DNNs are also the most popular option for challenging applications in speech recognition [7], voice synthesis [8], language translation and natural language processing (NLP) [9].

Despite the success of deep learning applications across multiple domains, many of which are life-crucial, a slew of significant safety, security and redundancy concerns have recently come to light. Recent studies have found that neural networks are vulnerable to natural noise, data errors, and to carefully designed input samples that can fool high-performing deep learning models. Only small perturbations in inputs, some of that cannot be detected by humans, are required to create incorrect outputs.

In 2017, an incorrect translation by the Facebook Machine Translation (MT) system led to an inappropriate arrest [10]. Instead of translating an Arabic phrase to "good morning", the Facebook translator interpreted it as "attack them". Arabic is a highly morphological language, and the translator mistook the input word for another that differed by only one letter. Since neural machine translators are commonly used across the Internet, it is important to study their failures in order to gain insights for preventing such incidents from recurring.

In the domain of sentiment analysis, Google's Perspective API[1] uses deep learning models to classify text samples and predict whether a given message is toxic/offensive or not. The Perspective API is widely accessible and querying it is fairly simple, allowing developers and publishers to use Perspective to give real-time feedback to commenters. This tool is frequently used for analyzing social media conversations. However, testing has shown that the API fails to accurately gauge toxicity if typographical errors are found in the original input [11].

In 2014, Szegedy et al. [12] first exposed a vulnerability of DNNs in the context of image classification. They showed that DNNs are susceptible to adversarial examples, where introducing small perturbations into a given input can cause a state-of-the-art deep learning classifier to give invalid output. They also showed that these perturbations can permeate through a multi-network classifier and result in misclassifications there as well. Szegedy et al.'s findings generated additional interest in adversarial examples.

Recently, there has been a marked increase in research investigating adversarial examples in computer vision and NLP. However, adversarial examples in the text domain are harder to

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.105210.

* Corresponding author at: University of Colorado at Colorado Springs, Colorado Springs, CO, USA.
E-mail address: balshema@uccs.edu (B. Alshemali).

1 https://www.perspectiveapi.com/.

https://doi.org/10.1016/j.knosys.2019.105210
0950-7051/© 2019 Elsevier B.V. All rights reserved.

generate. One reason for this is that adding subtle changes to text is much harder than in computer vision, where imperceptible perturbations can be easily introduced into a given image [13]. Furthermore, text sequences live in a discrete space, and word-level perturbations may significantly change the meaning of the text. In this scenario, an adversarial perturbation should change as few words as possible, and so this limitation introduces a sparsity constraint on text changes.

In this review, we briefly discuss the approaches used to create adversarial text examples for the purpose of testing and improving the reliability of DNNs in NLP and make the following contributions.

- We create a taxonomy to categorize the approaches for generating adversarial examples in NLP.
- We probe these approaches and classify them using our taxonomy.
- We discuss various types of defensive strategies against adversarial examples in NLP.
- We highlight main challenges and future research directions for adversarial examples in NLP.

1.1. Organization of the paper

The remainder of this paper is organized as follows. In Section 2, we give an overview of NLP tasks, NLP corpora, and evaluation metrics used in the context of NLP tasks. We also briefly describe different methods for generating adversarial examples against DNNs in the NLP domain, and explore their adversarial goals. In Section 3, we propose a taxonomy to categorize the methods for generating adversarial texts along two axes: the adversary's level of knowledge and the scope for perturbation. We elaborate on these approaches in Section 4, and summarize them in Section 5. In Section 6, we describe representative studies, summarize relatively effective defenses, and provide recommendations for designing defenses against adversarial texts. In Section 7, we analyze current challenges in the field and provide some future research directions. Finally, we conclude our work in Section 8.

2. Brief overview of adversarial examples in NLP

Adversarial examples in NLP are generated using methods that compromise computational models designed to handle NLP tasks. In this section, we provide a brief overview of NLP tasks that are vulnerable to adversarial examples, methods for generating adversarial examples in NLP, characteristics and goals of generating these adversarial examples, and metrics used to evaluate their effects.

2.1. Definition of terminologies

First, we present the frequently encountered technical terms that are relevant to adversarial examples and used within this paper.

- *Adversarial machine learning (AML):* Machine learning in adversarial settings is a field that lies at the intersection of machine learning and computer security. The field focuses on researching the effects of adversarial examples on machine learning algorithms, along with their capabilities and limitations. AML also involves the study of effective designs for machine learning techniques that can resist adversarial samples [14].
- *Adversary:* The entity that creates adversarial examples.
- *Adversarial perturbation:* Noise injected into a clean text sequence to create an adversarial example.
- *Adversarial example/sample:* A modified instance of a text sequence that is deliberately perturbed (e.g., by injecting noise) to foil DNNs. More formally, given a DNN model M and an original text sequence s, an adversary A generates a similar adversarial text sequence s': $s' = A(s)$, such that it results in a different output i.e. $M(s') \neq M(s)$.
- *Robustness:* The robustness of a model is related to its performance and applicability. Robust models (with better performance) are less vulnerable to adversarial examples [15].
- *Adversarial training:* Using perturbed text sequences in addition to clean text sequences to train deep neural networks.
- *Word Embedding:* Since DNN inputs are only vectors of continuous values, this is a learned word representation where words and phrases from the vocabulary are mapped to corresponding vectors comprised of real numbers.

2.2. NLP tasks

Adversarial examples are designed to exploit the weaknesses of DNN models in a wide range of domains. Below, we highlight some NLP tasks involving DNN models that have been shown to be vulnerable to adversarial texts.

- *Classification:* Here, classes, tags, or categories are assigned to text according to its content. Classification applications are wide-ranging, including sentiment analysis, spam detection, and language identification.
- *Machine Translation (MT):* The process of translating input text from one source natural language such as English, to another natural target language such as Spanish. Bilingual datasets are used to build language and phrase models that are employed to translate text.
- *Question Answering (QA):* This task involves answering natural language questions posed by humans, using models that draw information from textual sources or visual sources. Reading comprehension tasks involve answering questions from specific texts.
- *Textual Entailment (TE):* Textual entailment, also referred to as natural language inference (NLI), is a binary relation between two text fragments — the premise (p) and the hypothesis (h). These relations can be entailment, contradiction or neutral. The task is to predict whether the entailed text (hypothesis) can be inferred from the entailing text (premise). For example, *He is snoring* (premise), *A man sleeps* (hypothesis) [16].
- *Tagging:* Text tagging is the process of assigning one or more tags, labels, or categories to each token in a text. For example, Part of Speech (POS) tagging, which classifies individual tokens in a sequence, involves labeling tokens with their part of speech – e.g., noun, verb, adjective, and adverb. On the other hand, morphological tagging assigns morphological knowledge to each token in a context – e.g., singular, plural, tense, and gender.
- *Parsing:* Parsing is the process of breaking a sentence into several structural components and analyzing the relationships among them. In syntactic parsing, we perform syntactic analysis of a sentence, i.e. identifying syntactic constituents such as noun phrases (NP), verb phrases (VP), and prepositional phrases (PP). In dependency parsing, we extract grammatical relations among pairs of words in a sentence – e.g., direct objects (DOBJ), indirect objects (IOBJ), and nominal subjects (NSUBJ).
- *Dialogue Generation:* This involves conversing with human users with coherent and fluent natural language text.

2.3. *Methods for generating adversarial examples in NLP*

Various techniques have been used to create adversarial text examples. Recent methods involve perturbing the text sequences slightly by corrupting component characters. A popular adversarial perturbation is to replace a character with a random character or with an adjacent character from the English keyboard. Inserting random or pre-specified letters, commas, dots, or blank spaces have proven to be effective methods for fooling DNNs in the NLP domain. In addition, deleting a random character or even punctuation marks can change the output of neural models. Swapping two adjacent characters and repeating one character twice does not significantly alter the appearance of a text but can still trick the model into giving an inaccurate response.

Other methods depend on randomizing the order of the token's characters. Here either the order of all characters is randomized, or the order of all the characters except for the first and the last is randomized.

Text semantics are the target of other adversaries. For instance, verbs, adverbs, and adjectives are substituted with their corresponding synonyms or antonyms. The verbs can be negated or their tenses can be manipulated. Although deleting the most frequent words in a language (stop-words) such as *a*, *an*, and *the* seems harmless, recent studies have showed that this too can affect the performance of DNN models [17].

Another technique to generate adversarial texts involves injecting sentences into the text to confuse the DNN models. The inserted sentences can be gibberish, ungrammatical, grammatical, or human-approved. Other attacks paraphrase sentences or delete important sentences from the text.

2.4. *NLP corpora*

Since there are a variety of NLP tasks, different corpora have been collected for specific purposes. We mention the corpora here since adversarial texts have been tested in the context of these corpora, and references to them are included throughout the paper.

- *Text Classification Corpora:* AG's News[2] corpus contains more than one million categorized news articles. The DBpedia ontology classification dataset [18,19] is constructed from Wikipedia articles, classified to 14 non-overlapping classes. The Yahoo! Answers[3] topic classification dataset [19] includes questions and their corresponding answers, categorized to 10 topic classifications. The Amazon Review[4] and Yelp Review[5] corpora have user ratings of products and places, respectively. IMDB [20] and RottenTomatoes [21] are two corpora of movie reviews along with their associated sentiment labels. The Stanford Sentiment Treebank (SST) [22] comprises sentences labeled into five sentiment classes: very negative, negative, neutral, positive, or very positive. The Twitter[6] dataset for gender classification includes 20,000 samples, each sample with multiple fields such as name and location. The Kaggle Toxic Comment Classification dataset[7] contains a vast number of Wikipedia comments, labeled by humans as toxic, severe toxic, obscene, threat, insult, or identity hate. Enron Spam [23] is a collection of spam and non-spam emails.
- *Machine Translation Corpora:* The TED parallel corpora is a collection of bilingual and multilingual corpora extracted from TED Talks for 109 languages [24]. WMT16[8] provides several corpora for machine translation such as the News Translation corpus and the Multimodal Machine Translation corpus.
- *Question Answering Corpora:* The Stanford Question Answering Dataset (SQuAD) [25] has been created using 150,000 human-written questions on Wikipedia passages along with their answers. The MovieQA corpus [26] was obtained from movie subtitles, scripts, and plots and consists of about 15,000 multiple choice questions.
- *Text Tagging Corpora:* The English Penn Treebank [27] corpus has about 7 million words of part-of-speech tagged text and 3 million words of skeletally parsed text originally collected from Wall Street Journal articles. Universal Dependencies[9] (UD) is also a collection of grammatically annotated treebanks for several languages.
- *Textual Entailment Corpora:* The Stanford Natural Language Inference (SNLI) [28], Multi-Genre Natural Language Inference (MultiNLI),[10] and SciTail [29] are three preferred corpora used in textual entailment. The SNLI corpus is a collection of 570,000 human-written English sentence pairs manually labeled as entailment, contradiction, or neutral. MultiNLI contains 433,000 sentence pairs annotated with textual entailment information, while SciTail consists of 3,234 annotated questions and 115,564 annotated sentences.
- *Dialogue Generation Corpora:* Reddit[11] datasets, including Reddit Politics, Reddit News, and Reddit Movies, have been scraped to construct 4 million conversations. Each conversation holds 3 to 6 messages between two users. The Collaborative Communicating Agents (CoCoA) [30] corpus consists of 11,000 human-to-human dialogues. The Ubuntu Dialogue [31] corpus has approximately one million multi-turn dialogues, with a total of over 7-million utterances obtained from the chat logs of Ubuntu technical support.

2.5. *Evaluation metrics in NLP*

To evaluate the efficiency of adversarial examples, researchers have used various metrics depending on the task. Since these metrics are used throughout the paper, we discuss them here explicitly for the benefit of the reader.

- *Classification Accuracy*: This is a metric for evaluating the performance of classification models. It is the ratio of the number of correct predictions to the total number of predictions and is normally reported as a percentage. Besides classification models, accuracy is also used to evaluate textual entailment models and tagging models for both morphological tagging and POS tagging.
- *Misclassification Rate*: Misclassification rate or "error rate" is the ratio of the number of incorrect predictions to the total number of predictions and is normally reported as a percentage. In the context of adversarial examples, the performance of an adversary can be measured using the misclassification rate which is, in this setting, the "success rate" metric.

2 http://www.di.unipi.it/~gulli/.
3 https://webscope.sandbox.yahoo.com/catalog.php?.
4 http://snap.stanford.edu/.
5 https://www.yelp.com/dataset/challenge.
6 https://www.kaggle.com/crowdflower/twitter-user-gender-classification.
7 https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/.

8 http://www.statmt.org/wmt16/.
9 http://universaldependencies.org/.
10 https://www.nyu.edu/projects/bowman/multinli/.
11 https://www.reddit.com/.

- *Bilingual Evaluation Understudy (BLEU)*: This is used to evaluate the quality of a text that is translated from one natural language to another natural language using a machine translator [32]. This quality represents the correspondence between a human's translation (the reference) and that of a machine. Higher quality machine translations are closer to the human reference. The quality of the translation is measured between 0 and 1 and is often reported as a percentage.
- *Labeled Attachment Score (LAS)*: This is a standard evaluation metric to evaluate parser performance. It indicates the percentage of tokens that were parsed correctly and can be calculated as the ratio of the number of tokens that were correctly parsed to the total number of parsed tokens. LAS is normally reported as a percentage.

2.6. Characteristics of adversarial examples in NLP

In this section, we discuss the properties of adversarial examples in NLP and the differences between creating adversarial examples for computer vision and for NLP tasks.

- *Perceivability:* In computer vision, small perturbations to the pixels of an image usually cannot be easily detected by the human eye but can still fool DNN models. This is not commonly the case with adversarial texts. Small changes to text samples like adding a character or replacing a word are recognizable by humans [33] and limit the effectiveness of adversarial texts.
- *Semantic Dependencies:* Small perturbations to an image usually do not change the semantics of the image, as changing a few individual pixels does not turn an image of a car into a cat. In contrast, small textual changes can easily change the semantics of a sentence, which may also affect a DNN's output. For instance, adding a negation word ("not") to a sentence changes its sentiment completely.
- *Transferability:* Adversarial examples generated to fool a specific DNN model can fool other models even if their architectures are different or they were trained on different training sets, provided that the models perform the same task. This phenomenon is not restricted to deep learning models. Adversarial examples exploit this transferability against different machine learning algorithms such as DNNs, logistic regression (LR), support vector machines (SVM), and decision trees (DT), including commercial machine learning classification systems like Amazon Machine Learning[12] and Google Machine Learning.[13]

2.7. Goals of adversarial applications in NLP

Adversarial examples in NLP can be employed to either improve DNN models or render them ineffective. In this section, we discuss two types of adversarial text based on their goals. One type is dedicated to improve the efficacy and the applicability of DNN models while the other is used to duplicate or steal DNN models deployed in the real world.

2.7.1. Adversarial applications to improve DNNs

In 2014, Szegedy et al. [12] demonstrated that DNNs can be easily exploited by adversarial examples. Following their observations, many researchers further investigated applications of adversarial samples in the context of improving DNNs, revealing concrete problems in the safety of machine learning models and encouraging researchers to find solutions in the process. Within this context, adversarial examples can be used to achieve the following goals: (1) Evaluating DNN performance in the face of adversaries; (2) Detecting over-sensitivity and over-stability in DNN models; and (3) Reducing model vulnerability to adversarial examples. In essence, these adversarial examples can be seen as a game theory artifact to improve the robustness of DNN models [34]. All adversarial applications discussed in this paper (except Section 2.7.2) are used for improving the efficacy of neural network-based systems.

2.7.2. Adversarial applications to steal DNNs

The impact of deep learning on several domains of computer science such as computer vision, speech recognition, and NLP has led to the invention of machine learning as a service (MLaaS). Such platforms provide application programming interfaces (APIs) to query their trained deep learning models, which are usually built over cloud services.

These APIs allow users to access the trained models and charge them on a pay-per-query basis. This is useful for developers but constitutes a fertile ground for adversarial attacks. Intelligent adversaries can employ such queries to steal machine learning models as shown in several recent works [35–38]. Stealing machine learning models allows attackers to: (1) Query the models an unlimited number of times without paying full query charges; (2) Potentially leak sensitive information concerning private training data; (3) Steal intellectual property related to the stolen model; and (4) Evade security systems such as spam, fraud, or malware detectors. Another serious risk arises when operating at a national level, where information that is sensitive to national security might be compromised, which has even greater ramifications [36].

Tramèr et al. [35] successfully extracted equivalent or near equivalent models from the solutions provided by Amazon Machine Learning, Google Machine Learning, Microsoft Azure,[14] BigML,[15] and PredictionIO[16] via public APIs. Their idea was to feed in the data points and estimate the model from the outputs, hence duplicating the functionality of (or stealing) the model.

Similarly, Shi et al. [36] developed a three-stage method to steal machine learning classifiers: (1) Prepare the test data; (2) Obtain the labels; and (3) Train the model to predict equivalent or similar results. They demonstrated that developing a stolen model could save the attacker significant training time, provide access to sensitive information, and provide ways to trick the model.

Hitaj et al. [37] proposed stealing DNNs even with embedded watermarking techniques [39]. A watermark is a hidden mark that the owner of the model can use to verify data authenticity or track copyright violations of the model. They showed that there are at least two ways to avoid detection when stealing models. The first attack obtains (i.e. steals) a few models from different owners and provides access to an ensemble of these models rather than any specific one. The other attack develops a detector algorithm, that attempts to identify the possible watermark inputs and either rejects them or outputs labels randomly. Through experiments, they showed that ensembles achieve at most a 25.5% watermark recognition rate, while properly trained detectors can bring that number down to 10% or less.

Wang and Gong [38] alternatively proposed methods to steal hyperparameters. The direct monetary value of such stealing is derived from identifying hyperparameters that are obtained from a resource intensive selection process. Their methods not only successfully extract the hyperparameters, but can also be

12 https://aws.amazon.com/machine-learning/.

13 https://cloud.google.com/products/ai/.

14 https://azure.microsoft.com/.

15 https://bigml.com/.

16 http://predictionio.apache.org/.

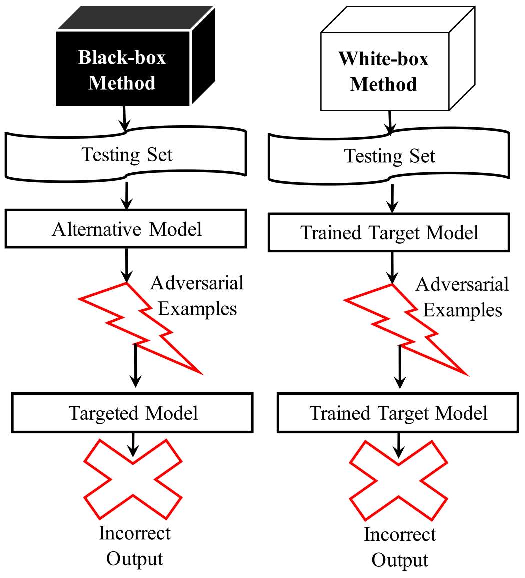

Fig. 1. Black-box vs. white-box adversarial methods. In the black-box settings, the adversary can only query and observe the targeted model's output. Consequently, it generates its adversarial examples using an alternative model. In white-box settings, the adversary uses the targeted model which has full access to for generating its adversarial examples.

combined with other model parameter extraction tools to completely copy models from MLaaS solutions like Amazon Machine Learning and Microsoft Azure.

Within this paper, we focus on adversarial examples that are used to improve the robustness of neural network models. Our goal is to provide an up to date, extensive study of these adversarial examples in a single document and our discussion of stealing attacks is rooted within this context.

3. Taxonomy of adversarial examples in NLP

Here, we put forth a taxonomy to categorize the methods for generating adversarial texts along two axes.

1. Adversary's knowledge level:
   - *Black-box methods:* In these methods, the adversary is only able to query the target model and manipulate its input samples by testing and observing the model's output [40].
   - *White-box methods:* Here, the adversary possesses complete knowledge of the target model, and the adversarial examples are generated while having access to the model's parameters, along with the model's architecture, training method, input features, and in some situations the training data as well [41]. Fig. 1 shows the differences between black-box and white-box methods.
2. Adversary's perturbation level:
   - *Character-level perturbations:* Here, the adversary perturbs the input by changing, inserting, or deleting an internal character within a token, like converting "Team" to "Texm" [43].
   - *Token-level perturbations:* The adversary perturbs the input by inserting, deleting, or replacing a token with another token, e.g., replacing "definitely" with "certainly".
   - *Sentence-level perturbations:* The adversary perturbs the input by inserting, replacing, or deleting a whole sentence [13].

Table 1
Taxonomy of recent adversarial examples in NLP along two axes: adversary's knowledge level and adversary's perturbation level.

Knowledge Level	Perturbation level		
	Character-level perturbation	Token-level perturbation	Sentence-level perturbation
Black-box adversaries	Belinkov and Bisk [40], Heigold et al. [42], Henderson et al. [33], Rodriguez and Rojas-Galeano [11], Gao et al. [43], Naik et al. [44], Sogaard et al. [45], Li et al. [46].	Rodriguez and Rojas-Galeano [11], Naik et al. [44], Samanta and Mehta [47], Alzantot et al. [48], Glockner et al. [49], Blohm et al. [50], Niu and Bansal [17], Li et al. [46].	Jia and Liang [13], Henderson et al. [33], Blohm et al. [50], Ribeiro et al. [51], Niu and Bansal [17].
White-box adversaries	Liang et al. [52], Ebrahimi et al. [53], Ebrahimi et al. [54], Li et al. [46].	Blohm et al. [50], Liang et al. [52], Ebrahimi et al. [53], Ebrahimi et al. [54], Li et al. [46], Behjati et al. [55].	Blohm et al. [50], Liang et al. [52], Mudrakarta et al. [56].

It is worth mentioning that some methods have perturbations at more than one level. Perturbations can be at the character-level, the token-level, the sentence-level, the character-token-level, the character-sentence-level, the token-sentence-level, or character-token-sentence-level. In the sections that follow, we consider recent studies on adversarial examples in light of our taxonomy.

4. Methods for generating adversarial texts

Within this section, we methodically go over various approaches for generating adversarial texts. We present how the methods work, and the effects current state-of-the-art adversarial examples can achieve. Table 1 presents these methods along the two axes discussed in our taxonomy (Section 3).

4.1. Black-box adversarial applications

Since white-box adversarial methods cannot be launched without accessing the model, most investigations of the robustness of NLP models have been conducted using black-box adversarial methods.

4.1.1. Character-level black-box adversarial applications

Heigold et al. [42] evaluated character-based models on several types of random noise for two NLP tasks: morphological tagging and machine translation. They experimented with three different types of noise: (1) Randomly swapping two neighboring characters in a word; (2) Randomly scrambling all characters within a word except for the first and the last characters; (3)

Randomly flipping a character with another character at a pre-specified rate. They studied the impact of noisy input on recurrent neural networks (RNNs) and convolutional neural networks (CNNs) trained on the UD English dataset and the TED (German–English) parallel corpora. For morphological tagging, when the researchers scrambled all the words, swapped the characters with 10% probability, and flipped the characters with 10% probability, the models showed performance degradation from around 95% to around 80% tag accuracy, a decrease that was fairly consistent across all noise types. For machine translation, the authors again scrambled the words, swapped, and flipped the characters with 10% probability. Here, the models' performance decrease was drastic and strongly varied by the type of noise. Scrambling the words introduced the greatest noise — the BLEU score went down from around 30 to around 05.

Belinkov and Bisk [40] investigated the sensitivity of neural machine translation models to natural and synthetic noise containing common misspellings. Naturally occurring errors were collected from some corpora and inserted into their datasets. In addition, four types of synthetic noise were used: (1) Swapping two adjacent letters (one swap per word); (2) Randomizing the order of all the letters in a word except for the first and the last; (3) Completely randomizing words; (4) Randomly replacing one letter in the word with an adjacent key on the traditional English keyboard, e.g., *noise* to *noide*. They experimented with the char2char [57], Nematus [58] and character-level CNN (charCNN) [19] models and used the TED (French, German, and Czech to English) parallel corpora. They demonstrated that character-level machine translation systems are overly sensitive to random character manipulations. On average, the French corpus BLEU score degraded from 42.5 to 7.5, while that of the German corpus went from 33.0 to 4.0 and that of the Czech corpus went from 27.1 to 3.5.

Gao et al. [43] fooled sentiment classification models in a black-box setting. They designed a black-box adversary (called DeepWordBug) to generate adversarial modifications directly on the input tokens using a two-step approach. First, determine the important tokens to modify, according to the effect on classification from a classifier. Then, modify those tokens slightly to fool the target classifier. Once the primary tokens were determined, a simple algorithm was used to transform those tokens and form adversarial samples. Four transformation methods were proposed: (1) Swapping two contiguous letters in the word; (2) Substituting a letter for a random letter within the word; (3) Deleting a letter at random; (4) Inserting a random letter. The researchers tested their approach on eight text datasets (AG's News, Amazon Review full and polarity, DBPedia, Yahoo! Answers , Yelp Review full and polarity, and Enron Spam), which together cater to a variety of NLP tasks (text classification, sentiment analysis and spam detection) targeting a word-level LSTM (Word-LSTM) model and a charCNN model [19]. Their approach reduced the model's prediction accuracy by 64.38% on average for the Word-LSTM model and 30.35% on average for the charCNN model, using their combined scoring function.

Sogaard et al. [45] studied the sensitivity of neural network-based dependency parsers for English punctuation. The authors evaluated three neural network dependency parsers: the Uppsala parser (UUPARSER) [59,60], the graph-based parser (KGRAPHS) [61], and the STANFORD parser [62], as well as two, older, non-neural alternatives: the arc-eager MALTPARSER [63] and the TURBOPARSER [64]. Two perturbation maps were introduced: (1) Remove all punctuation marks, and (2) Inject commas and dots. Their results showed that state-of-the-art dependency parsers are excessively reliant on punctuation, and therefore suffered from absent or creatively used punctuation. Another important finding was that neural network dependency parsers are more sensitive to punctuation than vintage non-neural parsers. With their no-punctuation strategy, the drop in LAS was quite dramatic. For instance, with the English Penn Treebank, the neural parsers had an average increase in error of 50.43% LAS, while non-neural parsers had an average increase in error of 38.45% LAS. Similarly, with medium punctuation injection rates, all parsers were vulnerable and performed worse than in the absence of punctuation. However, neural parsers suffered higher increases in errors than vintage parsers.

Algorithm 1: Creating adversarial samples using Samanta and Mehta's method [47]

```
input  : Sample input text $s$, Classifier $F$, Word $w$ with high
         contribution towards determining the class-label $y$ of the
         input sample $s$.
output: Adversarial text $s$.
1  while $y$ does not change do
2      if $w$ is Adverb then
3          Remove $w$ from $s$;
4      else
5          Consider a candidate pool $P$ for $w$;
6          if $w$ is Adjective and $p$ is Adverb where $p \in P$ then
7              Add $p$ before $w$ in $s$;
8          else
9              Replace $w$ with $p$ in $s$;
10         end
11     end
12     $y = F(s)$;
13 end
14 Return $s$;
```

4.1.2. Token-level black-box adversarial applications

Samanta and Mehta [47] proposed an algorithm to fool sentiment classifiers while preserving the meaning of the sentences. The authors introduced three kinds of word modifications: (1) Replace adjectives with adverbs; (2) Insert adverbs; (3) Remove adverbs. The contribution of every word toward determining the sample's class label was approximated, and sorted in decreasing order. They used synonyms and real word typos of high contributing words, as well as genre or category specific keywords to build a candidate pool for each word and obtain adversarial adverbs. Algorithm 1 explains Samanta and Mehta's method. They employed these modifications to fool CNN models using a Twitter dataset for gender classification and the IMDB movie review dataset for sentiment analysis. Testing on adversarial samples showed that, for the IMDB dataset, their model's accuracy decreased from 74.53% to 32.55%. On the other hand, using the Twitter dataset, their model scored 63.43% when tested on clean data but only 50.10% when tested on adversarial data.

Alzantot et al. [48] developed a black-box adversary to perturb tokens in a text. They used a population-based optimization algorithm to generate semantically and syntactically similar adversarial samples for sentiment analysis. Their core algorithm randomly selects a word in the sentence and determines a suitable replacement that has a similar meaning, fits within the surrounding context, and maximizes the target's label prediction scores. They followed three steps to achieve their goals. *First step:* Compute the nearest neighbors of the selected word based on the Euclidean distance in the GloVe embedding space [65]. *Second step:* Rank the candidate words based on the absolute difference between their language model's scores, and the score assigned to the original selected word given the surrounding context. Keep only the top words with the minimum absolute difference. *Third step:* Pick the word that maximized the target label prediction probability when replacing the original word. Samples of the

Table 2
An example of Alzantot et al.'s adversarial text [48]. Modified words are highlighted.

Text	Class
Original: great little short film that aired a while ago on sbs here in burge get a copy if you can probably only good for a few viewings as you will end up remembering the script and it is the twists that make this film so funny **well** directed and intriguingly scripted it is an example of just **how** good low budget **short** films can be.	Positive. (Confidence: 100.00%).
Perturbed: great little short film that aired a while ago on sbs here in burge get a copy if you can probably only good for a few viewings as you will end up remembering the script and it is the twists that make this film so funny **alright** directed and intriguingly scripted it is an example of just **why** good low budget **brief** films can be.	Negative. (Confidence: 57.89%).

adversarial examples produced by Alzantot et al.'s method are borrowed from their paper and shown in Table 2. They evaluated their adversary by randomly sampling 100 correctly classified reviews from the IMDB dataset and then modified 8.7% of the words. Their adversarial examples successfully changed the LSTM model's output with a 100% success rate.

Glockner et al. [49] designed testing sets that showed the deficiency of natural language inference (NLI) models. To isolate lexical knowledge, they obtained the premises from the SNLI training set. For each premise, the authors constructed several adversarial hypotheses by substituting a single word within the premise with a different word. They focused on producing only entailment and contradiction examples, with neutral examples created as a by-product of this process. Entailment examples were created by substituting a word with a synonym or hypernym, while contradiction examples were generated by replacing words with their hyponyms and antonyms. All the introduced words were already present in the training set of SNLI. They evaluated three models with no external knowledge: Residual-Stacked-Encoder [66], Enhanced Sequential Inference Model (ESIM) [67], and Decomposable Attention [68]. All of these models were based on pre-trained GloVe embeddings [65]. They also used WordNet baseline and the Knowledge-based Inference Model (KIM) [69] with external knowledge from WordNet [70]. All models were trained on SNLI, MultiNLI, and SciTail training sets. Their results showed that the accuracy drop in their models' performance was substantial, ranging from 11% to 33%. They also showed that the KIM considerably outperformed the other models, demonstrating that lexical knowledge is "the only requirement for good performance" on the adversarial testing set, exposing the limitations of the other neural models.

4.1.3. *Sentence-level black-box adversarial applications*

Jia and Liang [13] proposed adversarial examples for reading comprehension systems. They defined an adversary that added a new distracting sentence to the paragraph's end and left the answer and the question unchanged. Four methods to add adversarial sentences were introduced: (1) ADD-SENT, which added grammatical sentences that appear similar to the question but do not conflict with the correct answer; (2) ADD-ANY, which added arbitrary sequences of English words, regardless of grammaticality. In practice, this adversary generated gibberish sentences that used many words from the question but had no meaning; (3) ADD-COMMON, which worked exactly like ADD-ANY except that it only added common English words; (4) ADD-ONE-SENT, which added a human-approved sentence, selected at random, to the paragraph. The researchers' experiments demonstrated that across sixteen published models trained on SQuAD dataset, adding grammatically correct adversarial sentences decreased the average F1-score from 75% to 36%. In addition, when the adversary added un-grammatical sequences of words, the average F1-score on four models decreased further to just 07%. In this adversarial setting, all the models suffered not only from over-sensitivity, but also from over-stability (the model's inability to distinguish a sentence that actually answered the question from a sentence that had words in common with the question).

Ribeiro et al. [51] introduced semantically equivalent adversaries (SEAs) to reveal local and global over-sensitivity bugs in state-of-the-art neural classifiers. SEAs were proposed to generate semantically equivalent adversarial samples, based on paraphrase generation techniques. These techniques included translating sentences into other languages, and back again to create adversarial examples [71]. They identified adversarial examples by generating paraphrases of a sentence x and continuously retrieving predictions from a black-box model f until the original prediction changed. Using $SemEq(x, \hat{x})$ to produce 1 if the sentence x is semantically equivalent to $\hat{x}$ and 0 otherwise, they defined SEA as a semantically equivalent instance that changed the prediction of the model in the following equation:

$$SEA(x, \hat{x}) = 1[SemEq(x, \hat{x}) \wedge f(x) \neq f(\hat{x})]. \tag{1}$$

They trained a fastText model [72] on Rotten Tomatoes movie reviews and tested it on IMDB sentence-sized reviews [73]. They also tested the model of Zhu et al.'s [74] using their dataset. The results showed that neural classifiers are susceptible to such adversarial examples for a large fraction of predictions.

A summary of the character-level, token-level, and sentence-level black-box adversarial applications is presented in the supplementary material[17]: Section 1.

4.1.4. *Character-token-level black-box adversarial applications*

Naik et al. [44] proposed an evaluation methodology for NLI models based on adversarial techniques, calling them "stress tests". To evaluate the robustness of NLI models, the authors constructed three classes of stress tests: (1) Competence tests: This class includes (a) Numerical Reasoning: Randomly choose numbers from the premise sentences and either change them, or prefix them with "less than" or "more than". (b) Antonymy: Substitute words with their antonyms; (2) Distraction tests: This category contains (a) Word overlap: Append the tautology "and true is true" to the end of every hypothesis sentence; (b) Negation: Append the tautology "and false is not true", which contains a negation word "not" to the end of every hypothesis sentence; (c) Length mismatch: Append the tautology "and true is true" five times to the end of every premise sentence; (3) Noise tests (spelling errors): This class involves (a) Randomly swapping adjacent characters within a word; (b) Randomly substituting a single character with a character next to it on the English keyboard. Fig. 2 summarizes Naik et al.'s stress tests. They evaluated the performance of six neural models with the MultiNLI dataset: Nie and Bansal [66], Chen et al. [75], Balazs et al. [76], Conneau et al. [77], BiLSTM [78], and CBOW [79]. Their evaluation revealed weaknesses within these models, and the performance of all models dropped across all stress tests. They also showed that, Chen et al.'s model had the best performance, suggesting that their gated-attention mechanism handled shallow word-level perturbations to some extent.

4.1.5. *Token-sentence-level black-box adversarial applications*

Blohm et al. [50] applied adversarial strategies on machine reading comprehension systems to investigate the behavioral

[17] https://ars.els-cdn.com/content/image/1-s2.0-S0950705119305428-mmc1.pdf.

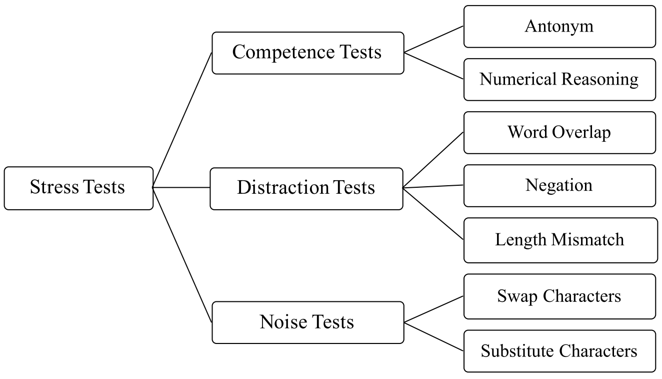

Fig. 2. Stress tests proposed by Naik et al. [44]: Competence tests, distraction tests, and noise tests.

Table 3
An example of a question and the adversarial sentence generated by the AddQA adversary proposed by Blohm et al. [50]. Here, the adversarial sentence has a high overlap with the question and one of the wrong answers (answer 4). As a result, the model picked the wrong answer (4) instead of the correct answer (1).

Adversarial sentence:	what aziz what do do what clothing opens do do.
Question:	What does Aziz do after he moves to Kashmir?
Answers:	(0) He opens a mosque.
	(1) He opens a clinic. (the correct answer).
	(2) He opens a school.
	(3) He becomes a monk.
	(4) He opens a clothing store.

differences between RNNs and CNNs. They designed two types of black-box adversaries: (1) Word-level black-box adversaries, where they inspected the most frequent words of their validation set's questions and manually defined lexical substitutions. These lexical substitutions were required to preserve meaning and result in grammatical sentences; (2) Sentence-level black-box adversaries, where a distracting sentence was added to the plot, regardless of grammaticality. This distracting sentence contained random commonly-used English words (AddC), the question words (AddQ), or words from the question and incorrect answers (AddQA). They applied their adversaries to Wang and Jiang's model [80], Liu et al.'s model,[18] Dzendzik et al.'s model [81], and their six proposed models trained on the MovieQA dataset. Their results demonstrated that the models were robust against meaning-preserving, lexical substitutions due the use of pre-trained GloVe embeddings. These embeddings helped the models to generalize to semantically equivalent tokens. Additionally, the models' performance dropped notably under sentence-level black-box adversaries in general and under AddQA in particular. Table 3 shows an example of a question and the adversarial sentence crafted by the AddQA adversary.

Besides sentiment classifiers, dependency parsers, reading comprehension systems, NMT, and NLI, dialogue models are also vulnerable to adversarial examples. Niu and Bansal [17] proposed two categories of black-box adversarial strategies that revealed the limitations of dialogue models (see Fig. 3): Should-Not-Change strategies (i.e., non-semantic perturbations to the source input that should not change the response) and Should-Change strategies (semantic perturbations to the source input that should change the response). Should-Not-Change strategies evaluated over-sensitivity of the model while Should-Change strategies tested the model's over-stability. On the Should-Not-Change side, five perturbation methods were introduced: (1) Random swap, which randomly swapped adjacent tokens; (2) Stop-word dropout, which randomly removed the most frequent words in a language from the inputs – e.g., "Ben ate the carrot" to "Ben ate carrot"; (3) Data-Level paraphrasing, which replaced words and phrases in the original inputs with a paraphrased version – e.g., "She bought a bike" to "She purchased a bicycle"; (4) Grammar errors, which replaced some grammatically correct words or phrases with grammatically incorrect ones. For example, altering verbs' tenses incorrectly: "He does not like cake" to "He do not like cake"; (5) Generative-Level paraphrasing, which generated paraphrases of the source input – e.g., "How old are you?" to "What is your age?". On the Should-Change side, they tested two perturbation methods: (1) Add negation, which negated the root verb of the source input sequence; (2) An antonym strategy, which changed verbs, adverbs, or adjectives to their antonyms; They evaluated their adversarial strategies on the Variational Hierarchical Encoder–Decoder (VHRED) model [82], the Reinforcement Learning (RL) model [83], and the Dynamic Knowledge Graph Network (DynoNet) [30] using two datasets: the Ubuntu Dialogue Corpus and the CoCoA dataset. Their results showed that both VHRED and the RL models were vulnerable to all Should-Change strategies and most Should-Not-Change strategies. On the other hand, DynoNet model was robust to Should-Change strategies which implied that it was not paying attention to the natural language context, except for the entities contained in it.

4.1.6. Character-sentence-level black-box adversarial applications

Henderson et al. [33] investigated several crucial aspects of security in dialogue systems, including adversarial examples, privacy violations, and safety concerns. Two kinds of input noise were considered: (1) Misspelt words where a character in a word had been removed or replaced or an extra character had been inserted; (2) Paraphrased sentences. To determine the effect of adversarial texts, Henderson et al. generated 1000-character edits and 6 paraphrased adversarial examples for 20 base example sentences, and used each example as an input to the VHRED dialogue model trained on the Reddit Movies and the Reddit Politics datasets. They found that their adversaries can significantly change semantic meaning and output distributions, even though the adversarial examples themselves resemble the base input. Their preliminary analysis showed that current neural dialogue systems are susceptible to adversarial examples, and further work is needed to study adversarial examples in the context of natural language and dialogues.

A summary of the black-box adversarial applications with character-token-level, token-sentence-level, and character- sentence -level perturbations is shown in Section 2 of the supplementary material.

4.2. White-box adversarial applications

The general idea of white-box adversaries includes perturbing the input in a direction that maximizes the model's loss function. This means that these adversaries compute the gradient of the model's loss function with respect to the input and use the result to alter the input, with the goal of increasing the loss value. Small perturbations like these are often sufficient to produce incorrect outputs. Adversaries, however, need to make a tradeoff between generating adversarial examples that are visually similar to the original input using small perturbations, and generating stronger adversarial examples that do not retain this visual similarity.

In this section, we study adversarial examples within white-box settings, wherein the adversary has access to the model's parameters and can leverage the gradients of the model to create more damaging manipulations for a larger degradation in the target model's performance. In the white-box setting, the adversary

[18] arXiv:1709.05036.

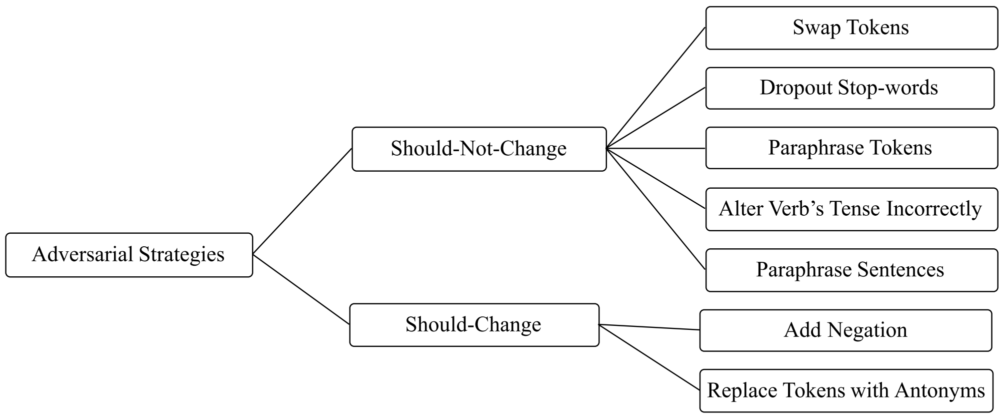

Fig. 3. Adversarial strategies proposed by Niu and Bansal [17]: Should-Not-Change strategies and Should-Change strategies.

possesses complete information on the model, which it can use to develop worst-case adversarial samples and reveal much larger vulnerabilities.

4.2.1. *Token-level white-box adversarial applications*

Behjati et al. [55] proposed a universal adversary for text classifiers, based on iterative gradient projection. They generated data-independent, perturbed text that could be concatenated with any input sample (from a corresponding domain) to fool the classifier with high success rate. They evaluated their adversary on three RNN-based architectures (LSTM, Bi-LSTM, and mean-LSTM) and two text classification datasets (AG's news and SST). Their experiments showed the vulnerability of text classifiers to such adversaries. For instance, inserting the adversarial word "abandonware" to the beginning of every input sample dropped the accuracy of their LSTM model, trained on the AG's news dataset, from 93.42% to 49.72%.

4.2.2. *Sentence-level white-box adversarial applications*

Mudrakarta et al. [56] analyzed the sensitivity of a reading-comprehension deep learning model. Their attribution technique, called Integrated Gradients (IG), computed the importance of question words and isolated the particular words used by deep learning systems to generate answers. The authors used this technique to improve the Jia and Liang ADD-SENT adversary's success rate [13] on reading comprehension models. Mudrakarta et al. used the SQuAD dataset and Yu et al.'s model [84] to test the impact of correlating ADD-SENT adversaries with their technique. They found that the adversaries were successful an additional 50% of the time when the injected sentence included every question word deemed important by the model (the top-attributed nouns in the question). For instance, in the question: "Where, according to gross state product, does Victoria rank in Australia?", "Australia" received a high attribution. Including the adversarial sentence: "According to net state product, Adelaide ranks 7 in New Zealand." did not confuse the model. However, including "Australia" in the adversarial sentence altered the prediction given by the model.

4.2.3. *Character-token-level white-box adversarial applications*

Ebrahimi et al. [53] proposed a gradient-based optimization method, called HotFlip, to generate adversarial samples and evaluate neural text classifiers. Hotflip generated adversarial samples by an atomic flip operation, which replaced one token with another using the model's gradient with respect to the one-hot vector input. Three perturbation approaches were proposed: (1) Flipping a character; (2) Inserting a character; (3) Deleting a character. Hotflip estimated which character modification had the highest estimated loss, and leveraged a beam search to find a set of modifications (character flips) that worked well together to fool the classifier. Ebrahimi et al. employed their technique to fool the charCNN-LSTM model [85] trained on the AG's news dataset. They showed that in a white-box setting, using a beam search to find a set of manipulations that fool the classifier was more effective than using a greedy search. Furthermore, they demonstrated that only a few single character changes were needed to trick the classifier. Their adversary had success rates in the 90 s with acceptable confidence scores. Additionally, they provided ways to adapt their proposed method to word-level models by computing derivatives with respect to one-hot word vectors. They tested their technique on Kim's word-level CNN (Word-CNN) [86], which was trained for binary sentiment classification on the SST dataset.

In addition to text classifiers, Ebrahimi et al. [54] also explored the space of adversarial examples for neural machine translation (NMT), and proposed new types of adversaries that remove or change a word in a translation. They extended their HotFlip white-box adversary [53] to include a larger set of adversaries and improved beam search. Ebrahimi et al. proposed a gradient-based optimization method that performed four types of untargeted operations: (1) Flip: Replacing one character with another; (2) Swap: Replacing two adjacent characters with each other; (3) Delete: Deleting one character; (4) Insert: Adding one character. They employed derivatives with respect to the one-hot representation of the input to rank the candidate change operations, in order to find adversarial examples that satisfied the adversary's goal. In addition to the untargeted operations, they implemented controlled operations where the adversary removed a specific word from the translation by maximizing the loss function of the target word. Targeted operations were also implemented where the adversary not only removed a word but also replaced it with another by minimizing the predictive loss function of the new word, chosen specifically to replace the old one. Fig. 4 summarizes Ebrahimi et al.'s adversarial operations. They used the TED Talks parallel corpus and the model of Costa-Jussa et al. [87] to evaluate their adversaries. They implemented their white-box adversaries and compared their results with those of black-box adversaries proposed by Belinkov and Bisk [40]. Their results showed that by taking advantage of information about the model's gradients, white-box adversarial perturbations significantly outperformed black-box perturbations in all kinds of operations.

4.2.4. *Token-sentence-level white-box adversarial applications*

In addition to the black-box adversaries proposed by Blohm et al. [50], discussed in Section 4.1.5, they also applied two types

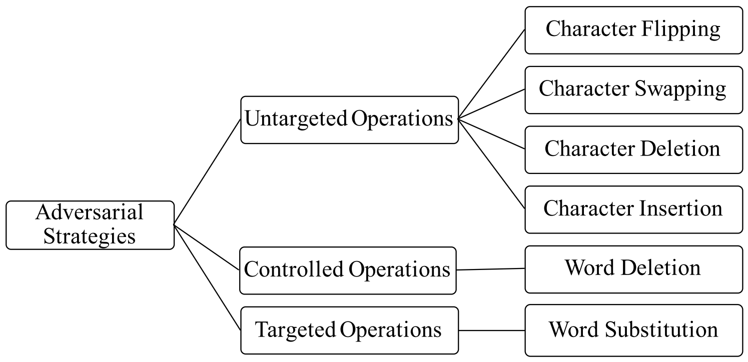

Fig. 4. Adversarial strategies proposed by Ebrahimi et al. [54]: Untargeted operations, controlled operations, and targeted operations.

of white-box adversaries to machine reading comprehension systems, for the purpose of investigating the behavioral differences between RNNs and CNNs: A word-level white-box adversary and a sentence-level white-box adversary. The word-level adversary used the models' sentence-level attention distribution to find the sentence that gave the most weight, conditioned on the correct answer. Then, it substituted the words that received most attention with randomly chosen words from their dataset's vocabulary. The sentence-level adversary removed the sentences with the highest attention. Their results showed that removing important words or important sentences from the plots made the models fail quickly, since they were no longer able to draw correct answers for the questions without the necessary plot contexts.

4.2.5. Character-token-sentence-level white-box adversarial applications

Liang et al. [52] developed a method to craft adversarial text samples against character-level text classifiers. Three perturbation strategies were employed: insertion, modification and removal. For insertion, they used their model's cost gradient to find the top 100 hot (important) characters for each sample. They scanned their training samples and found the phrases that contained a certain concentration of hot characters. From these phrases, they determined the most frequent phrases, the Hot Training Phrases (HTPs), and used them to construct adversarial payloads. These payloads could contain one or more words or even complete sentences. They inserted the adversarial payloads near the phrases with high contribution to the original class. For modification, similar to the process used to determine the HTPs, they determined phrases of each sample that had high contribution to the original classification and termed them Hot Sample Phrases (HSPs). They modified these HSPs in two ways: (1) Replacing the token or the phrase with a common misspelling of it obtained from related corpora, and (2) Replacing some characters with ones that had a similar visual appearance, e.g., replacing the lower-case letter "l" (el) with the numeral "1" (one). For removal, they deleted the HSPs that decreased the confidence of the original class but did not compromise the meaning. Table 4 shows the result of inserting a forged fact, removing an HTP, and modifying an HSP (adapted from Liang et al.'s paper). They employed these perturbation strategies to trick Zhang et al.'s model (charCNN) [19] using the DBpedia ontology dataset. Their results illustrated that by applying their adversarial strategies, the classification of a given sample can be altered to any desired target class, or to an arbitrary target class.

Section 3 in the supplementary material summarizes all white-box adversarial applications with all kinds of perturbation.

Table 4
An example of adversarial samples generated by combining the three strategies proposed by Liang et al. [52]. By removing an HTP (strikethrough), inserting a forged sentence (bold), and modifying an HSP (underline), the output classification changed from label "Building" to label "Means of Transportation". In the word "Cast1e": the lower-case letter "l" (el) is replaced with the numeral "1" (one).

The Old Harbor Reservation Parkways are three ~~historic~~ roads in the Old Harbor area of Boston. **Some exhibitions of Navy aircrafts were often held here.** They are part of the Boston parkway system designed by Frederick Law Olmsted. They include all of William J. Day Boulevard running from Cast1e Island to Kosciuszko Circle along Pleasure Bay and the Old Harbor shore. The part of Columbia Road from its northeastern end at Farragut Road west to Pacuska Circle (formerly called Preble Circle). Old Harbor Reservation

4.3. Adversarial examples against real-world applications

Besides local, offline DNNs, adversarial examples are used to evaluate real-world online NLP applications, including the Google Perspective API, Google Machine Learning, Microsoft Azure, and Amazon Machine Learning. The existence of such threats creates concerns for these applications and undermines their usability. In this section, we explore recent studies that have attempted to undermine real-world applications using a variety of adversarial strategies.

Google's Perspective API, designed to determine whether a certain comment would be considered toxic/offensive within a conversation, is imperfect. By "subtly" modifying the words that comprise the toxic comment, the machine learning model can be tricked into labeling a toxic comment as "healthy". Rodriguez and Rojas-Galeano [11] adapted Hosseini et al.'s[19] adversarial perturbations and formally characterized them as either obfuscation adversaries or polarity adversaries. In obfuscation adversaries, the adversary modifies a word's characters to hide toxicity in the input comments. Such adversarial methods can include misspellings or character substitutions, character repetitions and unnecessary punctuation (inserting commas, dots, or blanks within letters in the words). In polarity adversaries, the adversary attempts to obtain high toxicity scores for inoffensive comments. This is accomplished by simply negating toxic words, effectively inverting the comments' meaning. Their results showed that the Google toxicity detection model can be defeated by adversarial perturbations that alter comments with typographic or polarity manipulations. The obfuscation adversaries were reported to effectively reduce the toxicity scores of aggressive comments to benign levels, while polarity adversaries were reported to effectively increase the toxicity scores of non-aggressive comments to a high level.

19 arXiv:1702.08138.

Table 5
Recent studies and the character-level adversarial strategies used.

Strategy	[42]	[33]	[11]	[40]	[43]	[44]	[45]	[52]	[53]	[54]	[46]
Swap two adjacent characters.	✓			✓	✓	✓				✓	✓
Scramble characters.	✓			✓							
Delete letters.		✓			✓				✓	✓	✓
Delete punctuation marks.							✓				
Insert letters.		✓			✓				✓	✓	
Insert commas.			✓				✓				
Insert dots.			✓				✓				
Insert spaces.			✓								✓
Repeat characters.			✓								
Change numbers.						✓					
Replace characters with other characters.	✓	✓	✓		✓				✓	✓	
Replace characters with keyboard characters.				✓		✓					✓
Replace characters with visually similar characters.								✓			✓

Table 6
Recent studies and the token-level adversarial strategies used.

Strategy	[11]	[44]	[47]	[48]	[49]	[50]	[17]	[52]	[53]	[46]	[55]
Negate verbs or phrases.	✓	✓					✓				
Swap two adjacent tokens.							✓				
Delete stop-words.							✓				
Insert tokens.			✓					✓			✓
Remove tokens.			✓					✓			
Replace adjectives with adverbs.			✓								
Replace tokens with random tokens.						✓					
Replace tokens with their synonyms.				✓	✓	✓	✓				
Replace tokens with their antonyms.		✓			✓		✓				
Replace tokens with semantically related tokens.									✓	✓	
Replace tokens with misspelled version of them.								✓			

Li et al. [46] proposed TEXTBUGGER, a general framework to generate utility-preserving adversarial texts in both black-box and white-box settings. They captured the words that were significant to the classifiers first, and then manipulated them. The authors introduced five kinds of bugs: (1) Insert a space into the word; (2) Randomly delete a character from the word (excluding the first and the last character); (3) Randomly swap two adjacent characters in the word (excluding the first and the last character); (4) Replace characters with visually similar characters, e.g., replace a with @, or with adjacent characters on the keyboard; (5) Replace a word with a semantically similar word using the nearest neighbor in a context-aware word embedding space. They evaluated their framework on the IMDB, RottenTomatoes movie reviews, and the Kaggle Toxic Comment Classification dataset using Kim's CNN and charCNN [19]. They also assessed their framework on ten sentiment analysis real-world applications: Google Machine Learning, Microsoft Azure, IBM Waston Natural Language Understanding (IBM Watson),[20] Facebook fast-Text,[21] Amazon Machine Learning, ParallelDots,[22] Aylien Sentiment,[23] TheySay Sentiment,[24] TextProcessing,[25] and Mashape Sentiment.[26] For toxic content detection, they evaluated TEXTBUGGER on five real-world applications: Google Perspective, Facebook fast-Text, IBM Watson, ParallelDots AI, and the Aylien Offensive Detector. On average, TEXTBUGGER scored 89.66% and 66.12% success rates in fooling sentiment analysis and toxic content detection, respectively, in real-world applications.

A summary of all real-world applications that have been compromised by adversarial examples is shown in the supplementary material: Section 4.

20 https://www.ibm.com/watson/services/natural-language-understanding/.
21 https://fasttext.cc/.
22 https://www.paralleldots.com/.
23 https://aylien.com/text-api/sentiment-analysis/.
24 http://www.theysay.io/.
25 https://text-processing.com/.
26 https://market.mashape.com/features/consumer/.

5. Outlook of adversarial strategies

In this paper, we have presented a comprehensive review of the types of adversarial strategies to compromise deep learning models in the realm of NLP. While various technical details have already been mentioned, below we provide a summary and make more holistic observations regarding these strategies.

5.1. Obfuscation strategies

Most character-level adversarial perturbations focus on swapping two adjacent characters or replacing characters with others as shown in Table 5. In real-world scenarios, transposition errors like these are not uncommon, as we sometimes mistype a character by an adjacent character on the English keyboard. This is an easy to implement adversary but one that may not be truly potent. Scrambling the characters is a strong adversary but has not received significant attention in recent studies. One reason for this may be that this adversarial method results in text samples that humans can easily identify as distorted.

5.2. Semantic strategies

Token replacement has been attempted in most black-box and white-box token-level adversarial methods as shown in Table 6. The easiest way to obtain semantics-preserving adversarial text is to substitute tokens with their synonyms, making this method a popular perturbation choice. This can create a potent adversary if the synonym is an ambiguous word. Negating verbs, adverbs and adjectives, and substituting tokens with their antonyms are also preferred techniques for changing the meaning of text and confusing DNN models.

5.3. Injection strategies

Among sentence-level adversaries, the injection of grammatical sentences has received the lion's share of attention, as can

Table 7
Recent studies and the sentence-level adversarial strategies used.

Strategy	[13]	[33]	[50]	[17]	[51]	[52]	[56]
Remove sentences.						✓	
Paraphrase sentences.		✓		✓	✓		
Insert grammatical sentences.	✓				✓	✓	✓
Insert ungrammatical sentences.	✓		✓				
Insert human-approved sentences.	✓		✓				

Table 8
Recent adversaries in NLP along with their targeted tasks. The tasks are discussed in Section 2.2.

Adversary	Text classification	Machine translation	Question answering	Textual entailment	Text tagging	Dialogue generation	Parsing
Jia and Liang [13]			✓				
Belinkov and Bisk [40]		✓					
Heigold et al. [42]		✓			✓		
Henderson et al. [33]						✓	
Rodriguez and Rojas-Galeano [11]	✓						
Gao et al. [43]	✓						
Naik et al. [44]				✓			
Sogaard et al. [45]							✓
Samanta and Mehta [47]	✓						
Alzantot et al. [48]	✓						
Glockner et al. [49]				✓			
Blohm et al. [50]			✓				
Niu and Bansal [17]						✓	
Ribeiro et al. [51]	✓						
Liang et al. [52]	✓						
Ebrahimi et al. [53]	✓						
Ebrahimi et al. [54]		✓					
Mudrakarta et al. [56]			✓				
Li et al. [46]	✓						
Behjati et al. [55]	✓						

be seen in Table 7. Although removing important sentences from the text is easy to implement in the white-box setting and has a high chance of confusing targeted models, the capabilities of this adversary have not been extensively investigated.

5.4. Targeted NLP tasks

Adversarial methods have been proposed to compromise models that are designed to handle a variety of tasks. Text classification has a finite and fixed number of outputs (labels), which makes it easy to generate adversarial examples in this area. As seen in Table 8, this ease has been displayed investigation of text classification problems, including sentiment analysis and spam detection. Researchers' attention has been drawn repeatedly to the tasks of text classification, machine translation and question answering. However, less attention has been given to the task of dialogue generation, text tagging and parsing, as these tasks are more difficult [88]. Since DNNs are increasingly being used to solve nearly all conceivable tasks in NLP, the potential for adversarial perturbations abound almost everywhere. After decades of research, certain NLP tasks such as machine translation have achieved widespread success and acceptance due to the capabilities of modern DNNs. Cleverly crafted adversarial examples may pose a threat to further success, if such adversaries are not researched and thwarted through appropriate defensive strategies.

6. Defensive strategies against adversarial texts

To the best of our knowledge, existing defensive strategies for adversarial examples mainly focus on the vision domain and have not been thoroughly studied in the NLP domain. A few studies have attempted to employ adversarial training or pre-processing techniques such as spelling correction, to mitigate adversarial texts [13,40,43,46,48,53]; however, no effective defense has been proposed yet. In this section, we introduce recent strategies to resist adversarial examples and enhance the robustness of DNN models. Although some of these strategies have not been applied in the NLP domain yet,we describe representative studies and relatively effective defenses in hopes of inspiring researchers to investigate defenses in the area of NLP.

Defensive approaches employed to mitigate adversarial examples can be categorized into three main areas:

1. Modified training or testing, such as adversarial training strategies and spelling correction.
2. Modifying models by means such as changing the architecture, parameters, or loss function of the model during the training stage.
3. Appending add-ons such as using external models during the testing stage.

Fig. 5 summarizes the defensive strategies discussed in this paper. While the methods in the first category do not directly affect the models, those in the latter two categories do.

6.1. Modified training or testing

After exploring the various methods for generating adversarial texts, two natural questions arise. Is it possible to detect and recover the perturbed text using a spell checker? Is it feasible to make the DNN more robust by exposing it to perturbed texts during the training stage? In this section, we discuss the effectiveness of these strategies against adversarial examples.

6.1.1. Adversarial training

Goodfellow et al. [34] introduced adversarial training with the goal of improving the robustness of DNNs. The basic idea is to craft a large set of adversarial examples using various strategies and inject them into the training datasets. The training data then

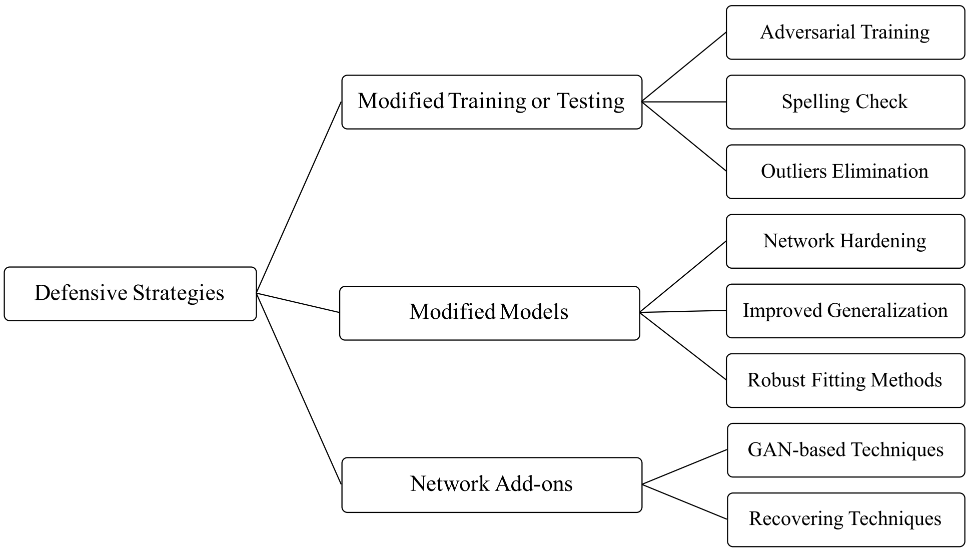

Fig. 5. A summarization of the defensive strategies discussed in Section 6.

includes adversarial samples mixed with corresponding clean ones. Utilizing adversarial training could provide regularization for DNNs [34,89] and reduce over-fitting [90], which in turn could enhance the robustness of DNNs against adversaries.

In the context of text, most studies that have proposed new methods to generate adversarial texts have evaluated their methods with adversarial training as the first line of defense against their adversarial examples [43,46,53]. The adversarial training technique, however, is not always useful and might only be effective on its corresponding adversarial samples. Sato et al. [91] experimented with adversarial training in neural machine translation systems and noticed an average of 2.31% BLEU score improvement. Pruthi et al. [92] indicated that adversarial training provides only marginal benefits. Their evaluation on the BERT model [93] for sentiment classification tasks, showed that adversarial training could only restore 5.10% classification accuracy when tested with a character-level swap adversary. Jia and Liang [13] employed adversarial training to improve the performance of reading comprehension models. They demonstrated that adversarial training is effective only as long as no new adversarial examples are encountered. In addition, effective black-box adversarial strategies can be used to reduce the performance of networks that have undergone adversarial training [94].

6.1.2. Spelling check

Another defense strategy against adversarial texts is to pre-process the data using a spell checker. Most character-level adversarial operations proposed in recent studies produce misspelled words [11,33,42,43,46,53,54]. These misspellings can be detected by applying a spell checker.

Gao et al. [43] used the Python autocorrector[27] to mitigate their adversarial examples. The Python autocorrector increased their model's prediction accuracy from an average of 13.34% to an average of 74.17%. Belinkov and Bisk [40] used Google's spell-checker to correct the misspellings generated by their adversarial methods. Google's spell-checker increased the French corpus' BLEU score from 16.7 to 21.4 and the German corpus' BLEU score from 18.6 to 25.0, reducing the effectiveness of adversarial texts in both cases.

Recent studied have proposed spelling correctors that outperform off-the-shelf spell checkers. Alshemali and Kalita [95] proposed a defense against character-level adversaries, using a spelling correction system that utilized frequency and contextual information. They demonstrated that their spelling corrector outperformed six state-of-the-art spelling checkers. Pruthi et al. [92] combated adversarial misspellings by placing a word recognition model in front of their classifier. Their proposed recognition system was built upon the semi-character RNN-based word recognizer by Sakaguchi et al. [96]. They showed that their defense mechanism outperformed off-the-shelf spell checkers and adversarial training.

Although utilizing spell checkers could mitigate the adversarial examples generated by character-level modifications, it may not be useful on word-level operations, such as replacing tokens with their synonyms. In this case, contextual information can be helpful in detecting real-word misspellings and grammatically sound adversarial texts. Context-sensitive spell-checking systems can identify misspelt words, even if the words themselves are in the dictionary, based on the context [97,98].

6.1.3. Eliminating outliers

Typically, in the domain of data science, more data is better. However, outliers can disturb the learning process in the primary stages. Outliers can be described as a portion of data that has different characteristics from the other data (usually defined by some threshold value(s)) [99]. Due to the critical insights it might provide, outlier detection plays an important role in a wide range of applications. Adversarial examples and outliers share an important feature [100] – both are outside of normal distribution. Therefore, detecting the outliers/adversarial samples in the testing data is likely to protect the DNN models.

Ko and Fujita [101] proposed evidential analytics to filter out noisy samples from Big Data. Noisy data could be comprised of either outliers or doubtful elements, which makes the data non-identifiable. Their approach performed like a data pre-processor for data representation and manifold learning. Based on dominance and similarity characteristics, they clustered the samples and detected the noisy ones, which were subsequently removed using an inductive operation.

Liu and Shen [102] alleviated the effect of noise when predicting sentiment polarity, by designing a filter gate that utilizes a gating mechanism to control the flow of information signal and obtain precise sentiment representations. When the signal carries valuable sentiment information, it goes through the filter gate, otherwise, the signal is rejected.

[27] https://github.com/phatpiglet/autocorrect/.

In Pota et al.'s work [103], word-level and character-level representations of the input words were concatenated and fed to Bi-LSTM layers [104], trained to associate a sequence of POS tags to a sequence of words. They demonstrated that their proposed method not only improved the overall accuracy on POS tagging, but showed superior performance on detecting rare and out of vocabulary (OOV) words.

6.2. Modified models

Researchers have shown that altering the models in certain ways proves beneficial to resistance against potential adversarial examples [105]. In this section, we discuss some approaches that modify DNNs to counter adversarial samples.

6.2.1. Network hardening

In addition to pre-processing the input, it is advisable to explore the possibility of hardening the networks themselves so that they cannot be fooled easily. This hardening process requires an understanding of what happens within DNNs when adversarial examples are processed by a trained network. In other words, it is necessary to understand the true technical cause of DNN malfunction or failure. Human intelligence is resilient in that minor distractions and garbling of sensory inputs do not usually lead to a failure of our processing capabilities [106]. Capable DNNs, therefore, should ideally be resistant to such adversarial texts.

In computer vision, network optimization has been applied in the context of defending neural networks against adversarial images. Raghunathan et al. [107] proposed a certifiable, scalable, and trainable method to defend two-layer networks against small adversarial perturbations. Based on a semidefinite relaxation, they computed an upper bound on the worst-case loss for neural networks. This upper bound serves as a robustness certificate such that, for a given input and a network, no adversary can force the error to exceed a specific value. They optimized this certificate with the parameters of the network to provide an adaptive regularizer, that boosts robustness against adversarial inputs. The work of Raghunathan et al. [107] is one of a few provable defenses against popular adversaries [108].

Sinha et al. [109] used distributionally robust optimization to both defend against adversaries and develop an effective optimization procedure that guarantees better performance on adversarial samples. They considered a Lagrangian penalty formulation for perturbing the underlying distribution of data in a Wasserstein ball, to provide a procedure for adversarial training that updated the parameters of the model with the worst-case perturbations of the training data.

Quantization is another approach for hardening DNNs in the field of computer vision. Dynamic Quantized Activation (DQA) treats thresholds of the quantization activation function as tuning parameters for improving DNN robustness during adversarial training. The effects of adversarial examples can be reduced by quantizing the output at each activation layer [110].

Xia et al. [111] adopted a Bayesian model to deal with two classes of opinion-level features: Intra-opinion features such as opinion target and words indicative of opinion, and inter-opinion features such as correlative words in a sentence [112]. They showed that their approach is capable of resolving polarity ambiguity in sentence classification.

6.2.2. Generalization where possible

In the clinical environment, Fujita and Cimr [113] proposed an 8-layer CNN model to detect three of the most frequently occurring and difficult to distinguish types of heart disorders using only basic data normalization of electrocardiogram (ECG) signals. Their CNN model contains three convolutional layers, each one followed by a max-pooling layer. The last two layers are fully-connected. The use of max pooling layers ensures that no single feature gets to be the deciding point as it is leveraged by its surroundings. This step can also be viewed as expanded utilization of the contextual information.

Yang et al. [114] proposed a model to first pre-process data, then perform analysis to obtain information on similarity and dominance. Their model considers not only a data point's information, but also information from its surrounding environment (i.e., neighbors). Thus, the model not only uses the data of the input, but also takes into consideration the dominating class in the area. These techniques could be adapted to NLP models to allow the model to better grasp relevant information from a context and not overestimate the importance of a single item in the sequence.

Cheng et al. [115] proposed an approach to improve the robustness of neural machine translation systems with doubly adversarial inputs. Their method first attacks the model with adversarial source inputs, then defends the model with adversarial target inputs, aiming to improve its robustness against the adversarial source inputs. Their results substantiated that their method improved the generalization performance of their model, over both clean and perturbed datasets.

In the context of sentiment analysis, Li et al. [116] improved the generalization capability of their model by ensembling reinforcement learning (RL), GANs, and RNNs. Their model is able to expand any given dataset using RL and GANs and learn the structure of sentences directly with RNNs.

6.2.3. Robust fitting methods

Robust fitting methods lead to better parameters and enhance the performance of the model. Lai et al. [117] proposed a robust fitting method called GSS, that estimates the model hypothesis for data with high outlier rates. The aim of GSS is to reduce the occurrence of incorrect parameters and hence improve the robustness of the model fitting. It does this by iteratively performing an improved greedy search (IGS) with a parameter detector until a correct estimation is obtained. IGS estimates accurate model parameters by re-initializing the model hypothesis using the inliers of the current best hypothesis. The parameter detector detects the incorrect model parameters of the generated hypothesis, discards them, and generates new hypotheses based on the IGS and a specified inlier threshold, that is used to segment the inliers from the outliers.

In later work, Lai et al. [118] developed a global greedy search (GGS) strategy to generate accurate model hypotheses. This strategy initially samples from the currently obtained inliers and then iteratively samples data subsets from the whole input dataset until adequate parameters are found or the search process is reinitialized. If more than one hypothesis is found, a mutual information theoretical method is applied to fuse the hypotheses.

Liao et al. [119] developed a high-level representation guided denoiser (HGD) method to eliminate perturbed samples. The loss function of the HGD compares the model's outputs produced by clean samples and denoised samples. The clean sample is reconstructed by subtracting the perturbation from the noisy sample. HGD improves the model's immunity against both white-box and black-box adversaries. It also can be trained on a small dataset and can be employed to defend models other than the one guiding it.

6.3. Network add-ons

DNN models can be protected from adversarial examples even after being deployed. Add-on mechanisms can be used with DNN models to defend them against adversarial perturbations. These add-ons do not require modifying the training procedure or the model's structure.

6.3.1. *GAN-based defensive techniques*

Goodfellow et al. [120] proposed Generative Adversarial Networks (GANs) that involve two models: a generator and a discriminator. The generative model creates realistic samples aiming to fool the discriminator. On the other hand, the discriminative model detects the original samples from the perturbed ones. A min–max loss function is used to train the two models simultaneously. Later, Gulrajani et al. [121] extended GANs to WGANs such that the same approach can be utilized for text data, where the data is normally presented in discrete fashion.

Samangouei et al. [122] developed a defense strategy (Defense-GAN) that leverages the capability of GANs [120] to defend neural classifiers against white-box and black-box adversaries. With Defense-GAN, the distribution of clean images is modeled. It expects clean samples to be close to this distribution, while adversarial samples will be further away from it. At testing time, when Defense-GAN detects a suspicious image, it utilizes gradient descent to find the closest output to this suspicious image that does not include any adversarial changes, then feeds this projected output to the classifier instead of the original suspicious image.

The Defense-GAN approach can be used to process textual input, where in the case of a suspicious sample, it searches for the closest match within the data points. This could be done with the use of nearest neighbor or cosine similarity (or some other distance measure for vectors) and would need to be computed on the sample-size vectors in addition to the word-size vectors.

6.3.2. *Perturbation recovering techniques*

Akhtar et al. [123] proposed a Perturbation Rectifying Network (PRN) to defend models against universal perturbations. They appended “pre-input” layers to the targeted model, such that the targeted model and its parameters need no modifications. A PRN is trained to recover adversarial samples in such a way that the targeted model’s output becomes the same as it would be for the clean versions of the same samples. A test sample is first passed through the PRN, and if an adversarial sample is detected, the output of the PRN is used as the final result.

Zhou et al. [124] proposed the Learning to Discriminate Perturbations (DISP) framework to identify and recover perturbed text inputs for text classification models. Given a perturbed textual sample, a perturbation discriminator first detects a set of potential perturbations that could have been applied. Then, for each potential perturbed token, an embedding estimator, optimized with a corpus of token embeddings, calculates an approximate embedding vector. Finally, the perturbed token is replaced with the closest token in the embedding space based on approximate k-nearest neighbors (kNN) search. According to their work, DISP can recover perturbed text for any NLP model without adjusting the model or the training procedure.

7. Challenges and future directions

In the previous sections, a detailed description of adversarial methods and suggestions for defense were given to help readers gain a better understanding of the phenomenon of adversarial examples in NLP, and to motivate researchers to pursue research in this domain. Although several methods for constructing adversarial texts have been proposed in recent years, many related challenges need to be discussed and addressed. In this section, we discuss some challenges related to creating and mitigating adversarial examples in NLP, highlighting avenues for future research.

7.1. *Discrete data*

The process of generating adversarial examples for NLP tasks has a relatively shorter history than that of creating them for vision tasks. Until recently, research on both adversarial strategies and defenses has focused mainly on images, and little attention was paid to texts. One primary reason is that it is more challenging to deal with discrete data. Normally, we vectorize the textual data before feeding them into DNNs, however, adopting adversarial methods or defenses from the vision domain and applying them to word embeddings will not be a direct process [125]. Applying the Fast Gradient Sign Method (FGSM) [34] on word embeddings will result in invalid characters, invalid word sequences [126] or out of vocabulary (OOV) problems. Defense-GAN [122] leverages the capability of GANs [120] to defend neural classifiers against adversarial examples. The way Defense-GAN works is that it adds random noise to the input and lets the discriminator learn to detect it. Since we cannot add random noise to discrete input, this defense has not been applied to textual data in an efficient way yet and needs attention from researchers.

7.2. *Perceivablity*

In contrast to the vision domain, where adding small perturbations to images leads to incorrect classification by DNNs yet usually correct classification by humans, generating imperceptible adversarial texts is nearly impossible [48]. Several studies have proposed character-level modification operations such as flipping characters [11,33,42,53,54], deleting characters [33,46,53,54], and inserting characters [33,53,54] to create misspellings or invalid words. Besides the fact that humans are extremely robust against misspellings [106], such misspellings can be easily detected by spelling or grammar checkers, thus rendering the adversary a failure [40,43]. From a practical point of view, carefully designed imperceptible textual perturbations that are syntactically and semantically correct are needed.

7.3. *Transferability*

Szegedy et al. [12] first observed the transferability property of adversarial samples. They found that adversarial examples can transfer to fool several neural networks that have the same architecture but are trained on different datasets. Subsequently, Papernot et al.[28] found that the same adversarial examples can be effective even against neural networks with different architectures. They also showed that adversarial examples can transfer to fool classifiers trained by different machine learning algorithms, as long as they are trained to perform the same task. Because of this property, an adversary can generate adversarial examples using a substitute model, and then use these examples to compromise other models or/and datasets. This property is more critical for black-box adversaries, where the target model and its training set are not accessible [13,43].

Defensive distillation, proposed by Papernot et al. [127,128], is a popular defensive procedure in which a secondary model is trained to predict the probabilities of the labels, that are generated by the main model. This defensive mechanism has succeeded in hardening DNNs against adversarial examples in the image domain. Soll et al. [129] examined the performance of the defensive distillation strategy on text classification tasks. Their results demonstrated that defensive distillation does not help with improving the robustness of their neural networks against

28 arXiv:1605.07277.

adversarial examples, nor does it block the transferability of adversarial text examples between NNs. This indicates that different methods for hardening neural networks for NLP tasks need to be examined with transferability in mind. In some situations, completely new methods must be developed.

7.4. Universality

In image classification tasks, Moosavi-Dezfooli et al. [94] proposed the Universal Adversarial Perturbations (UAP) approach to fool classifiers with high confidence. UAPs are data independent and can be concatenated to any input sample (from the corresponding domain). Following Moosavi-Dezfooli et al. [94], Behjati et al. [55] introduced universal adversarial perturbations to fool textual classifiers. There is no need to separately optimize the perturbation for each input, as the generated perturbation can be applied to any input sequence.

A suggested, more powerful universal adversary would fool any model on any text in any language. Most recent methods of generating adversarial texts focus on English datasets, but other languages such as Arabic [130] and Chinese have been neglected thus far in adversarial research.

On the other hand, the existence of universal adversaries is another important issue to be taken into consideration when developing defenses. A universal defensive technique that can defend the model against any adversarial perturbation is also required. Such a universal defense would need to be resistant to any adversary and not be broken by building new adversaries as defending only against existing adversarial strategies is insufficient. Some defenses that have claimed to guard against existing adversarial examples were later shown to be vulnerable to new adversaries [131].

7.5. Performance evaluation

Most studies that have proposed methods to craft adversarial texts evaluated the performance of their methods using the accuracy metric (the lower the accuracy, the more effective the adversarial perturbations). Some researchers calculated the difference in accuracy before and after applying their methods. More meaningful evaluation metrics measure features beyond accuracy such as semantic similarity, transfer rate, and imperceptibility rate. Li et al. [46] evaluated semantic similarity of their adversarial examples using edit distance, Euclidean distance, and cosine similarity. Michel et al. [132] calculated the semantic similarity of their adversarial examples using both manual and automatic evaluation metrics. They found that chrF [133], which is based on character n-gram F-score, correlated most with human judgment, followed by METEOR [134] and BLEU. It is apparent that more standardized methodologies for evaluating the transferability, imperceptibility, and semantic similarity of adversarial examples are required.

8. Conclusion

Although the performance of DNNs on a variety of natural language tasks is outstanding, researchers have shown that DNN models are vulnerable to adversarial examples, which carry subtle input perturbations that can result in incorrect output. As deep learning remains at the core of recent machine learning innovations, this discovery has prompted considerable study into adversarial examples in NLP. This paper presents a comprehensive review of the use of adversarial examples to improve the reliability of DNN models in NLP. In this review, we investigated recent approaches for generating adversarial texts from two aspects — the adversary's level of knowledge and the scope for perturbation. We also discussed defensive strategies to improve robustness in the wake of adversarial examples, highlight current challenges and possible future research directions. While it is evident that adversarial examples are a significant threat to deep learning, high activity in this domain makes us hopeful that the research community will be able to achieve even greater robustness against adversarial perturbations in the near future.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.knosys.2019.105210.

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[2] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems, 2015, pp. 91–99.

[3] N. Papernot, P. McDaniel, A. Sinha, M. Wellman, Towards the science of security and privacy in machine learning, in: IEEE European Symposium on Security and Privacy, 2018.

[4] E. Choi, M.T. Bahadori, E. Searles, C. Coffey, M. Thompson, J. Bost, J. Tejedor-Sojo, J. Sun, Multi-layer representation learning for medical concepts, in: Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1495–1504.

[5] Z. Che, S. Purushotham, R. Khemani, Y. Liu, Interpretable deep models for ICU outcome prediction, in: American Medical Informatics Association Symposium, Vol. 2016, 2016, p. 371.

[6] Z. Che, D. Kale, W. Li, M.T. Bahadori, Y. Liu, Deep computational phenotyping, in: Proceedings of the 21th ACM International Conference on Knowledge Discovery and Data Mining, 2015, pp. 507–516.

[7] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal Process. Mag. 29 (6) (2012) 82–97.

[8] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A.W. Senior, K. Kavukcuoglu, Wavenet: A generative model for raw audio., in: The ISCA Speech Synthesis Workshop, 2016, p. 125.

[9] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: Advances in Neural Information Processing Systems, 2014, pp. 3104–3112.

[10] Y. Berger, Israel arrests palestinian because facebook translated good morning to attack them, 2017, https://www.haaretz.com/israel-news/palestinian-arrested-over-mistranslated-good-morning-facebook-post-1.5459427.

[11] N. Rodriguez, S. Rojas-Galeano, Fighting adversarial attacks on online abusive language moderation, Appl. Comput. Sci. Eng. 915 (2018) 480–493.

[12] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in: International Conference on Learning Representations, 2014, URL https://openreview.net/forum?id=kklr_MTHMRQjG.

[13] R. Jia, P. Liang, Adversarial examples for evaluating reading comprehension systems, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2017, pp. 2021–2031.

[14] L. Huang, A.D. Joseph, B. Nelson, B.I. Rubinstein, J. Tygar, Adversarial machine learning, in: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, 2011, pp. 43–58.

[15] D. Su, H. Zhang, H. Chen, J. Yi, P.-Y. Chen, Y. Gao, Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 631–648.

[16] R. Mitkov, The Oxford Handbook of Computational Linguistics, Oxford University Press, 2005.

[17] T. Niu, M. Bansal, Adversariasl over-sensitivity and over-stability strategies for dialogue models, in: Proceedings of the 22nd Conference on Computational Natural Language Learning, 2018, pp. 486–496.

[18] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al., Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia, Semant. Web 6 (2) (2015) 167–195.

[19] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: Advances in Neural Information Processing Systems, 2015, pp. 649–657.

[20] A.L. Maas, R.E. Daly, P.T. Pham, D. Huang, A.Y. Ng, C. Potts, Learning word vectors for sentiment analysis, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 2011, pp. 142–150.
[21] B. Pang, L. Lee, Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales, in: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, 2005, pp. 115–124.
[22] R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1631–1642.
[23] V. Metsis, I. Androutsopoulos, G. Paliouras, Spam filtering with naive Bayes-which naive Bayes? in: Proceedings of the Third Conference on Email and Anti-Spam, Vol. 17, 2006, pp. 28–69.
[24] C. Mauro, N. Jan, S. Sebastian, B. Luisa, C. Roldano, F. Marcello, The iwslt 2016 evaluation campaign, in: International Workshop on Spoken Language Translation, 2016.
[25] P. Rajpurkar, J. Zhang, K. Lopyrev, P. Liang, Squad: 100, 000+ questions for machine comprehension of text, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2016, pp. 2383–2392.
[26] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, S. Fidler, Movieqa: Understanding stories in movies through question-answering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4631–4640.
[27] A. Bies, J. Mott, C. Warner, English news text treebank: Penn treebank revised, 2015, https://wakespace.lib.wfu.edu/handle/10339/64068.
[28] S.R. Bowman, G. Angeli, C. Potts, C.D. Manning, A large annotated corpus for learning natural language inference, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2015, pp. 632–642.
[29] T. Khot, A. Sabharwal, P. Clark, SciTail: A textual entailment dataset from science question answering, in: Proceedings of the Association for the Advancement of Artificial Intelligence, 2018.
[30] H. He, A. Balakrishnan, M. Eric, P. Liang, Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, 2017, pp. 1766–1776.
[31] R. Lowe, N. Pow, I. Serban, J. Pineau, The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems, in: Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2015, pp. 285–294.
[32] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, BLEU: a method for automatic evaluation of machine translation, in: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, 2002, pp. 311–318.
[33] P. Henderson, K. Sinha, N. Angelard-Gontier, N.R. Ke, G. Fried, R. Lowe, J. Pineau, Ethical challenges in data-driven dialogue systems, in: AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society, 2018, pp. 123–129.
[34] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: International Conference on Learning Representations, 2015, URL https://iclr.cc/archive/www/doku.php%3Fid=iclr2015:main.html.
[35] F. Tramèr, F. Zhang, A. Juels, M.K. Reiter, T. Ristenpart, Stealing machine learning models via prediction APIa, in: USENIX Security Symposium, 2016, pp. 601–618.
[36] Y. Shi, Y. Sagduyu, A. Grushin, How to steal a machine learning classifier with deep learning, in: IEEE International Symposium on Technologies for Homeland Security (HST), 2017, pp. 1–5.
[37] D. Hitaj, B. Hitaj, L.V. Mancini, Evasion attacks against watermarking techniques found in mlaas systems, in: International Conference on Software Defined Systems, 2019, pp. 55–63.
[38] B. Wang, N.Z. Gong, Stealing hyperparameters in machine learning, in: IEEE Symposium on Security and Privacy, 2018, pp. 36–52.
[39] Y. Nagai, Y. Uchida, S. Sakazawa, S. Satoh, Digital watermarking for deep neural networks, Int. J. Multimedia Inf. Retr. 7 (1) (2018) 3–16.
[40] Y. Belinkov, Y. Bisk, Synthetic and natural noise both break neural machine translation, in: International Conference on Learning Representations, 2018, URL https://openreview.net/forum?id=BJ8vJebC-.
[41] Y. Liu, X. Chen, C. Liu, D. Song, Delving into transferable adversarial examples and black-box attacks, in: International Conference on Learning Representations, 2017, URL https://openreview.net/forum?id=Sys6GJqxl.
[42] G. Heigold, G. Neumann, J. van Genabith, How robust are character-based word embeddings in tagging and MT against wrod scramlbing or randdm nouse? in: Proceedings of the 13th Conference of the Association for Machine Translation in the Americas, 2018, pp. 68–80.
[43] J. Gao, J. Lanchantin, M.L. Soffa, Y. Qi, Black-box generation of adversarial text sequences to evade deep learning classifiers, in: IEEE Security and Privacy Workshops, 2018, pp. 50–56.
[44] A. Naik, A. Ravichander, N. Sadeh, C. Rose, G. Neubig, Stress test evaluation for natural language inference, in: Proceedings of the International Conference on Computational Linguistics, 2018, pp. 2340–2353.
[45] A. Søgaard, M. de Lhoneux, I. Augenstein, Nightmare at test time: How punctuation prevents parsers from generalizing, in: Proceedings of the EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, 2018, pp. 25–29.
[46] J. Li, S. Ji, T. Du, B. Li, T. Wang, TEXTBUGGER: Generating adversarial text against real-world applications, in: Proceedings of Network and Distributed System Security Symposium (NDSS), 2019.
[47] S. Samanta, S. Mehta, Generating adversarial text samples, Adv. Inf. Retr. 10772 (2017) 744–749.
[48] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, K.-W. Chang, Generating natural language adversarial examples, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2018, pp. 2890–2896.
[49] M. Glockner, V. Shwartz, Y. Goldberg, Breaking NLI systems with sentences that require simple lexical inferences, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 2018, pp. 650–655.
[50] M. Blohm, G. Jagfeld, E. Sood, X. Yu, N.T. Vu, Comparing attention-based convolutional and recurrent neural networks: Success and limitations in machine reading comprehension, in: Proceedings of the 22nd Conference on Computational Natural Language Learning, 2018, pp. 108–118.
[51] M.T. Ribeiro, S. Singh, 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), Vol. 1, 2018, pp. 856–865.
[52] B. Liang, H. Li, M. Su, P. Bian, X. Li, W. Shi, Deep text classification can be fooled, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-18), 2017.
[53] J. Ebrahimi, A. Rao, D. Lowd, D. Dou, Hotflip: White-box adversarial examples for NLP, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 2017, pp. 31–36.
[54] J. Ebrahimi, D. Lowd, D. Dou, On adversarial examples for character-level neural machine translation, in: Proceedings of the 27th International Conference on Computational Linguistics, 2018, pp. 653–663.
[55] M. Behjati, S.-M. Moosavi-Dezfooli, M.S. Baghshah, P. Frossard, Universal adversarial attacks on text classifiers, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 7345–7349.
[56] P.K. Mudrakarta, A. Taly, M. Sundararajan, K. Dhamdhere, Did the model understand the question? in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 2018, pp. 1896–1906.
[57] J. Lee, K. Cho, T. Hofmann, Fully character-level neural machine translation without explicit segmentation, Trans. Assoc. Comput. Linguist. 5 (2017) 365–378.
[58] R. Sennrich, O. Firat, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. Läubli, A.V.M. Barone, J. Mokry, et al., Nematus: a toolkit for neural machine translation, in: Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics, 2017, pp. 65–68.
[59] M. de Lhoneux, Y. Shao, A. Basirat, E. Kiperwasser, S. Stymne, Y. Goldberg, J. Nivre, From raw text to universal dependencies-look, no tags!, in: Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, 2017, pp. 207–217.
[60] M. de Lhoneux, S. Stymne, J. Nivre, Arc-hybrid non-projective dependency parsing with a static-dynamic oracle, in: Proceedings of the 15th International Conference on Parsing Technologies, 2017, pp. 99–104.
[61] E. Kiperwasser, Y. Goldberg, Simple and accurate dependency parsing using bidirectional LSTM feature representations, Trans. Assoc. Comput. Linguist. 4 (1) (2016).
[62] D. Chen, C. Manning, A fast and accurate dependency parser using neural networks, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2014, pp. 740–750.
[63] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, E. Marsi, Maltparser: A language-independent system for data-driven dependency parsing, Nat. Lang. Eng. 13 (2) (2007) 95–135.
[64] D. Fernández-González, A.F. Martins, Parsing as reduction, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, 2015, pp. 1523–1533.
[65] J. Pennington, R. Socher, C. Manning, Glove: Global vectors for word representation, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1532–1543.
[66] Y. Nie, M. Bansal, Shortcut-stacked sentence encoders for multi-domain inference, in: Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, 2017, pp. 41–45.
[67] Q. Chen, X. Zhu, Z. Ling, S. Wei, H. Jiang, D. Inkpen, Enhanced LSTM for natural language inference, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, 2017, pp. 1657–1668.
[68] A.P. Parikh, O. Täckström, D. Das, J. Uszkoreit, A decomposable attention model for natural language inference, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2016, pp. 2249–2255.

[69] Q. Chen, X. Zhu, Z.-H. Ling, D. Inkpen, S. Wei, Neural natural language inference models enhanced with external knowledge, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vol. 1, 2018, pp. 2406–2417.

[70] C. Fellbaum, Wordnet: Wiley online library, Encyclopedia Appl. Linguist. (1998).

[71] J. Mallinson, R. Sennrich, M. Lapata, Paraphrasing revisited with neural machine translation, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics.

[72] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of tricks for efficient text classification, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, 2017, pp. 427–431.

[73] D. Kotzias, M. Denil, N. De Freitas, P. Smyth, From group to individual labels using deep features, in: Proceedings of the 21th ACM International Conference on Knowledge Discovery and Data Mining, 2015, pp. 597–606.

[74] Y. Zhu, O. Groth, M. Bernstein, L. Fei-Fei, Visual7w: Grounded question answering in images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4995–5004.

[75] Q. Chen, X. Zhu, Z.-H. Ling, S. Wei, H. Jiang, D. Inkpen, Recurrent neural network-based sentence encoder with gated attention for natural language inference, in: Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, 2017, pp. 36–40.

[76] J.A. Balazs, E. Marrese-Taylor, P. Loyola, Y. Matsuo, Refining raw sentence representations for textual entailment recognition via attention, in: Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, 2017, pp. 51–55.

[77] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, A. Bordes, Supervised learning of universal sentence representations from natural language inference data, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2017, pp. 670–680.

[78] N. Nangia, A. Williams, A. Lazaridou, S.R. Bowman, The Repeval 2017 shared task: Multi-genre natural language inference with sentence representations, in: Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, 2017, pp. 1–10.

[79] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: International Conference on Learning Representations, 2013, URL https://openreview.net/forum?id=idpCdOWtqXd60s.

[80] S. Wang, J. Jiang, A compare-aggregate model for matching text sequences, in: International Conference on Learning Representations, 2017, URL https://openreview.net/forum?id=HJTzHtqee.

[81] D. Dzendzik, C. Vogel, Q. Liu, Who framed Roger Rabbit? multiple choice questions answering about movie plot, in: Proceedings of the the Joint Video and Language Understanding Workshop: MovieQA and the Large Scale Movie Description Challenge (LSMDC), 2017.

[82] I.V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A.C. Courville, Y. Bengio, A hierarchical latent variable encoder-decoder model for generating dialogues, in: The Association for the Advancement of Artificial Intelligence, 2017, pp. 3295–3301.

[83] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, D. Jurafsky, Deep reinforcement learning for dialogue generation, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2016, pp. 1192–1202.

[84] A.W. Yu, D. Dohan, Q. Le, T. Luong, R. Zhao, K. Chen, Fast and accurate reading comprehension by combining self-attention and convolution, in: International Conference on Learning Representations, 2018, URL https://openreview.net/forum?id=B14TlG-RW.

[85] Y. Kim, Y. Jernite, D. Sontag, A.M. Rush, Character-aware neural language models, in: The Association for the Advancement of Artificial Intelligence, 2016, pp. 2741–2749.

[86] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1746–1751.

[87] M.R. Costa-Jussà, C. España-Bonet, P. Madhyastha, C. Escolano, J.A. Fonollosa, The TALP–UPC Spanish–English WMT biomedical task: Bilingual embeddings and char-based neural language model rescoring in a phrase-based system, in: Proceedings of the First Conference on Machine Translation, Vol. 2, 2016, pp. 463–468.

[88] Y. Belinkov, J. Glass, Analysis methods in neural language processing: A survey, Trans. Assoc. Comput. Linguist. 7 (2019) 49–72, http://dx.doi.org/10.1162/tacl_a_00254.

[89] S. Sankaranarayanan, A. Jain, R. Chellappa, S.N. Lim, Regularizing deep networks using efficient layerwise adversarial training, in: The Association for the Advancement of Artificial Intelligence, 2018.

[90] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial machine learning at scale, in: International Conference on Learning Representations, 2017, URL https://openreview.net/forum?id=BJm4T4Kgx.

[91] M. Sato, J. Suzuki, S. Kiyono, Effective adversarial regularization for neural machine translation, in: Proceedings of the Annual Meeting of the Association for Computational Linguistics, 2019, pp. 204–210.

[92] D. Pruthi, B. Dhingra, Z.C. Lipton, Combating adversarial misspellings with robust word recognition, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 5582–5591.

[93] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.

[94] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, Universal adversarial perturbations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1765–1773.

[95] B. Alshemali, J. Kalita, Toward mitigating adversarial texts, Int. J. Comput. Appl. 178 (50) (2019) 1–7, http://dx.doi.org/10.5120/ijca2019919384.

[96] K. Sakaguchi, K. Duh, M. Post, B. Van Durme, Robsut wrod reocginiton via semi-character recurrent neural network, in: The Association for the Advancement of Artificial Intelligence, 2017, pp. 3281–3287.

[97] P. Fivez, S. Šuster, W. Daelemans, Unsupervised context-sensitive spelling correction of english and dutch clinical free-text with word and character n-gram embeddings, in: Biomedical Natural Language Processing Workshop, 2017.

[98] C.J. Lu, A.R. Aronson, S.E. Shooshan, D. Demner-Fushman, Spell checker for consumer language (CSpell), J. Amer. Med. Inf. Assoc. 26 (3) (2019) 211–218.

[99] Y. Liu, Z. Li, C. Zhou, Y. Jiang, J. Sun, M. Wang, X. He, Generative adversarial active learning for unsupervised outlier detection, IEEE Trans. Knowl. Data Eng. (2019).

[100] G. Lerman, T. Maunu, An overview of robust subspace recovery, Proc. IEEE 106 (8) (2018) 1380–1410.

[101] Y.-C. Ko, H. Fujita, An evidential analytics for buried information in big data samples: Case study of semiconductor manufacturing, Inform. Sci. 486 (2019) 190–203.

[102] N. Liu, B. Shen, Aspect-based sentiment analysis with gated alternate neural network, Knowl.-Based Syst. (2019) 105010.

[103] M. Pota, F. Marulli, M. Esposito, G. De Pietro, H. Fujita, Multilingual POS tagging by a composite deep architecture based on character-level features and on-the-fly enriched word embeddings, Knowl.-Based Syst. 164 (2019) 309–323.

[104] W. Ling, T. Luís, L. Marujo, R.F. Astudillo, S. Amir, C. Dyer, A.W. Black, I. Trancoso, Finding function in form: Compositional character models for open vocabulary word representation, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1520–1530.

[105] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: International Conference on Learning Representations, 2018, URL https://openreview.net/forum?id=rJzIBfZAb.

[106] G. Rawlinson, The significance of letter position in word recognition, IEEE Aerosp. Electron. Syst. Mag. 22 (1) (2007) 26–27.

[107] A. Raghunathan, J. Steinhardt, P. Liang, Certified defenses against adversarial examples, in: International Conference on Learning Representations, 2018, URL https://openreview.net/forum?id=Bys4ob-Rb.

[108] A. Athalye, N. Carlini, D. Wagner, Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples, in: International Conference on Machine Learning, 2018.

[109] A. Sinha, H. Namkoong, J. Duchi, Certifiable distributional robustness with principled adversarial training, in: International Conference on Learning Representations, 2018, URL https://openreview.net/forum?id=Hk6kPgZA-.

[110] G. Goswami, A. Agarwal, N. Ratha, R. Singh, M. Vatsa, Detecting and mitigating adversarial perturbations for robust face recognition, Int. J. Comput. Vis. 127 (6–7) (2019) 719–742.

[111] Y. Xia, E. Cambria, A. Hussain, H. Zhao, Word polarity disambiguation using bayesian model and opinion-level features, Cogn. Comput. 7 (3) (2015) 369–380.

[112] I. Chaturvedi, E. Cambria, R.E. Welsch, F. Herrera, Distinguishing between facts and opinions for sentiment analysis: survey and challenges, Inf. Fusion 44 (2018) 65–77.

[113] H. Fujita, D. Cimr, Computer aided detection for fibrillations and flutters using deep convolutional neural network, Inform. Sci. 486 (2019) 231–239.

[114] X. Yang, T. Li, D. Liu, H. Fujita, A temporal-spatial composite sequential approach of three-way granular computing, Inform. Sci. 486 (2019) 171–189.

[115] Y. Cheng, L. Jiang, W. Macherey, Robust neural machine translation with doubly adversarial inputs, in: Proceedings of the Annual Meeting of the Association for Computational Linguistics, 2019, pp. 4324–4333.

[116] Y. Li, Q. Pan, S. Wang, T. Yang, E. Cambria, A generative model for category text generation, Inform. Sci. 450 (2018) 301–315.

[117] T. Lai, H. Fujita, C. Yang, Q. Li, R. Chen, Robust model fitting based on greedy search and specified inlier threshold, IEEE Trans. Ind. Electron. (2018).

[118] T. Lai, R. Chen, C. Yang, Q. Li, H. Fujita, A. Sadri, H. Wang, Efficient robust model fitting for multistructure data using global greedy search, IEEE Trans. Cybern. (2019).
[119] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, J. Zhu, Defense against adversarial attacks using high-level representation guided denoiser, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1778–1787.
[120] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[121] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A.C. Courville, Improved training of Wasserstein GANs, in: Advances in Neural Information Processing Systems, 2017, pp. 5767–5777.
[122] P. Samangouei, M. Kabkab, R. Chellappa, Defense-GAN: protecting classifiers against adversarial attacks using generative models, in: International Conference on Learning Representation, 2018, URL https://openreview.net/forum?id=BkJ3ibb0-.
[123] N. Akhtar, J. Liu, A. Mian, Defense against universal adversarial perturbations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3389–3398.
[124] Y. Zhou, J.-Y. Jiang, K.-W. Chang, W. Wang, Learning to discriminate perturbations for blocking adversarial attacks in text classification, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2019.
[125] W.Y. Wang, S. Singh, J. Li, Deep adversarial learning for NLP, in: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials, 2019, pp. 1–5.
[126] Z. Zhao, D. Dua, S. Singh, Generating natural adversarial examples, in: International Conference on Learning Representations, 2018, URL https://openreview.net/forum?id=H1BLjgZCb.
[127] N. Papernot, P. McDaniel, X. Wu, S. Jha, A. Swami, Distillation as a defense to adversarial perturbations against deep neural networks, in: IEEE Symposium on Security and Privacy, IEEE, 2016, pp. 582–597.
[128] N. Papernot, P. McDaniel, Extending defensive distillation, in: IEEE Symposium on Security and Privacy, 2017.
[129] M. Soll, T. Hinz, S. Magg, S. Wermter, Evaluating defensive distillation for defending text processing neural networks against adversarial examples, in: International Conference on Artificial Neural Networks, 2019, pp. 685–696.
[130] B. Alshemali, J. Kalita, Adversarial examples in arabic, in: International Conference on Computational Science and Computational Intelligence, 2019.
[131] N. Carlini, G. Katz, C. Barrett, D.L. Dill, Ground-truth adversarial examples, 2018.
[132] P. Michel, X. Li, G. Neubig, J. Pino, On evaluation of adversarial perturbations for sequence-to-sequence models, in: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 3103–3114.
[133] M. Popovic, chrf: character n-gram F-score for automatic MT evaluation, in: Proceedings of the Workshop on Statistical Machine Translation, 2015, pp. 392–395.
[134] M. Denkowski, A. Lavie, Meteor universal: Language specific translation evaluation for any target language, in: Proceedings of the EACL Workshop on Statistical Machine Translation, 2014, pp. 376–380.