Alma Mater Studiorum - University of Bologna

Department of Computer Science and Engineering

Master's Degree in Artificial Intelligence

Final Thesis in
Natural Language Processing

SynBA: A contextualized Synonim-Based adversarial Attack for Text Classification

Supervisor
Prof. Paolo Torroni

Co-supervisors
Dr. Federico Ruggeri
Dr. Giulia De Poli

Candidate
Giuseppe Murro

Academic Year 2021-2022- Third session

Dedicated to my parents
who have always believed in me
supported me in every choice
raised the man I am now

Abstract

With the advent of high-computational devices, deep neural networks have gained a lot of popularity in solving many Natural Language Processing tasks. However, they are also vulnerable to adversarial attacks, which are able to modify the input text in order to mislead the target model. Adversarial attacks are a serious threat to the security of deep neural networks, and they can be used to craft adversarial examples that steer the model towards a wrong decision.

In this dissertation, we propose SynBA, a novel contextualized synonym-based adversarial attack for text classification. SynBA is based on the idea of replacing words in the input text with their synonyms, which are selected according to the context of the sentence.

We show that SynBA successfully generates adversarial examples that are able to fool the target model with a high success rate. We demonstrate three advantages of this proposed approach: (1) effective—it outperforms state-of-the-art attacks by semantic similarity and perturbation rate, (2) utility-preserving—it preserves semantic content, grammaticality, and correct types classified by humans, and (3) efficient—it performs attacks faster than other methods.

"The algorithms that cause AI systems to work so well are imperfect, and their systematic limitations create opportunities for adversaries to attack. At least for the foreseeable future, this is just a fact of mathematical life."

– Marcus Comiter

Acknowledgements

Firstly, I would like to express my deepest gratitude to Professor Paolo Torroni and Federico Ruggeri for their precious advice and support during the development of this thesis. My thanks also go to Giulia De Poli, my industrial supervisor in 3rdplace, for her guidance and for the opportunity to work on this project during the internship.

Secondly, I would like to thank my parents for the trust they have placed in me and which I hope I have repaid with the results obtained during my studies.

Thirdly, many thanks to my girlfriend for her patience and for putting up with my stressful times over the past two years.

And least but not last, an acknowledgements to my friends and colleagues for their support and for the pleasant moments spent together.

Bologna, 06 December 2022 Giuseppe Murro

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this section we will present the summarized content of the whole thesis.

1.1 Topic definition

In the era of digital transformation, the amount of data produced by humans is growing exponentially. We are continuously exposed to an immense quantity of information, and by interacting with digital devices we are constantly generating data in the form of text, images, videos, and audio.

This data is used to train Machine Learning (ML) models able to perform tasks that were previously accomplished by humans. However, the security and integrity of these models are still a concern. In particular, the ability of ML models to generalize to unseen data is challenging.

The mathematics behind ML models is complex, and it is difficult to understand how they make decisions. This lack of transparency can be exploited by Adversarial Machine Learning, a novel research area that lies at the intersection of machine learning and cybersecurity. It refers to a class of attacks that aims to deteriorate the performance of classifiers on specific tasks. The goal behind adversarial attacks is to circumvent existing parameters and data rules so that the ML model confuses its instructions and makes a mistake.

With machine learning rapidly becoming core to organizations' value proposition, the need for organizations to protect them is growing fast. Hence, Adversarial Machine Learning is becoming an important field in the software industry.

Gartner, a leading industry market research firm, advised that application leaders must anticipate and prepare to mitigate potential risks of data corruption, model theft, and adversarial examples.

1.2 Problem statement

One critical issue in adversarial settings is to understand whether and to what extent a classifier may resist to specifically targeted attacks.

An adversarial example is an instance with small, intentional feature perturbations that cause a machine learning model to make a false prediction. These carefully curated examples are correctly classified by a human observer but can fool a target model, raising serious concerns regarding the security and integrity of existing ML algorithms. On the other hand, it is shown that the robustness and generalization of ML models can be improved by crafting high-quality adversaries and including them in the training data.

While existing works on adversarial examples have obtained success in the image and speech domains, it is still challenging to deal with text data due to its discrete nature.

The intrinsic difference between images and textual data makes it extraordinarily difficult for researchers to employ methods dealing with images to adversarial attacks in the NLP domain.

1.3 Research question

Focusing on text classification task, we are motivated to address the following two fundamental research questions (RQs):

RQ1: *Does state-of-the-art attack methods generate adversarial examples that are legible, grammatical, and similar in meaning to the original texts?*

RQ2: *How can we craft high-quality adversaries?*

1.4 Solution

Researchers proposed special adversarial attacks in the text domain in order to maintain semantic consistency and syntactic correctness. But those methods fail in generating high-quality adversarial examples since they frequently violate linguistic constraints.

This thesis concentrates on the adversarial attacks for text classification, in particular, the attacks based on sentiment analysis datasets, like IMDB [30] and Rotten Tomatoes [38]. Two state-of-the-art approaches, TextFooler [24] and BERT-based attack [14], are compared to analyze weaknesses and strengths.

Then, their shortcomings are addressed by proposing a novel method, called SynBA, to generate adversarial examples for text data. It is a word-level attack that generates adversarial examples by substituting words with candidates that have both a synonymy and contextual relationship with the original token.

The key contributions of this survey can be summarized as follows:

- we introduce a simple but strong attack method, SynBA, to quickly generate high-profile utility-preserving adversarial examples that force the target models to make wrong predictions under the white-box setting;
- we propose a comprehensive automatic and human evaluation of adversarial attacks to evaluate the effectiveness, efficiency, and utility preserving properties of our system;
- we compare the adversarial examples generated by our method with TextFooler and BERT-based attack in terms of semantic similarity, semantic consistency, perturbation rate, success rate, perplexity and runtime.

1.5 Thesis organization

First chapter introduces the general content about thesis and gives a short presentation of the topic, the problem and the solution we propose;

Second chapter a deepening about the theoretical foundations used during the project;

Third chapter presents the methodology used to build proposed adversarial attack solution;

Fourth chapter presents the experimental results obtained during the project;

Fifth chapter discusses about the results and possible future developments.

During the drafting of the essay, following typography conventions are considered:

- the acronyms, abbreviations, ambiguous terms or terms not in common use are defined in the glossary, in the end of the present document;
- the first occurrences of the terms in the glossary are highlighted like this: word;
- the terms from the foreign language or jargon are highlighted like this: *italics*.

Chapter 2

Background

In this chapter we will present the theoretical knowledge useful to understand the content from successive chapters.

2.1 Natural Language Processing

The field of Natural Language Processing (NLP), also known as computational linguistics, is a branch of Artificial Intelligence focused on the technology of processing language. It encompasses a variety of topics, which involves the engineering of computational models and processes to solve practical problems in understanding and generating human languages. These solutions are used to build useful software.

Linguistics has two branches—computational linguistics and theoretical linguistics. Computational linguistics has been concerned with developing algorithms for handling a useful range of natural language as input. While theoretical linguistics has focused primarily on one aspect of language performance, grammatical competence—how people accept some sentences as correctly following grammatical rules and others as ungrammatical. They are concerned with language universals—principles of grammar which apply to all natural languages [10].

Computational linguistics is concerned with the study of natural language analysis and language generation. Further, language analysis is divided into two domains, namely sentence analysis, and discourse and dialogue structure. Much more is known about the processing of individual sentences than about the determination of discourse structure. Any analysis of discourse structure requires a prerequisite as an analysis of the meaning of individual sentences. However, it is a fact that for many applications, thorough analysis of discourse is not mandatory, and the sentences can be understood without that [17].

The sentence analysis is further divided into syntax analysis and semantic analysis. The overall objective of sentence analysis is to determine what a sentence “means” . In practice, this involves translating the natural language input into a language with simple semantics, for example, formal logic, or into a database command language. In most systems, the first stage is syntax analysis. Figure 2.1 shows the relations among different components of NLP [8].

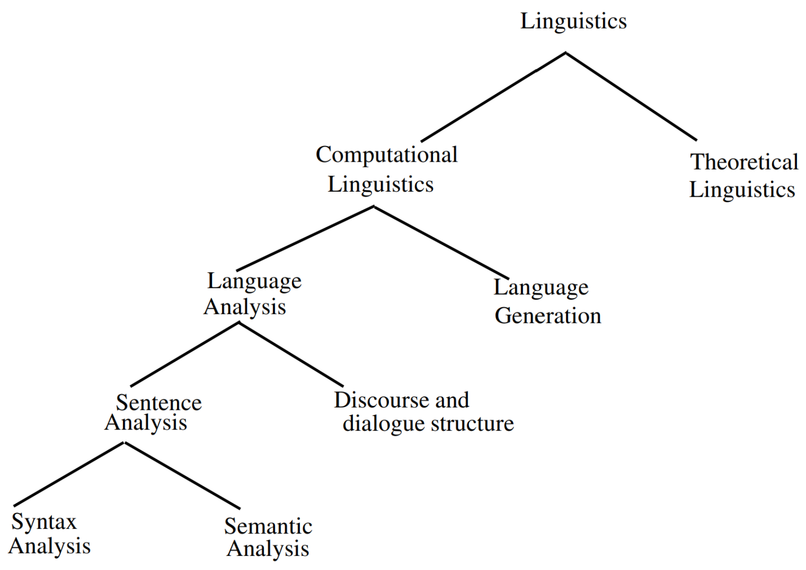


Figure 2.1: Components of NLP

Some of the common applications of NLP are Classification of text into categories, Index and search large texts, Automatic translation, Information extraction, Automatic summarization, Question answering, Knowledge acquisition, and Text generations/dialogues. Some of those tasks are discussed in sections 2.1.4, 2.1.5, 2.1.6, 2.1.7.

2.1.1 Lexicon

Dictionaries are special texts whose subject matter is a language, or a pair of languages in the case of a bilingual dictionary. The purpose of dictionaries is to provide a wide range of information about words— etymology, pronunciation, stress, morphology, syntax, register—to give definitions of senses of words, and, in so doing, to supply knowledge not just about language, but about the world itself.

The term “dictionary” is typically related to printed wordbook for human readers. Instead “lexicon” will refer to the component of a NLP system that contains information (semantic, grammatical) about individual word strings [18].

A lexicon which provides an effective combination of traditional lexicographic information and modern computing is called WordNet [33]. It is an online lexical database designed for use under program control. English nouns, verbs, adjectives, and adverbs are organized into sets of synonyms, each representing a lexicalized concept. WordNet contains more than 118,000 different word forms and more than 90,000 different word senses. Approximately 40% of the words in WordNet have one or more synonyms.

The cognitive synonyms which are called *synsets* are presented in the database with lexical and semantic relations. WordNet includes the following semantic relations:

- **Hypernymy**: A hypernym is a word that is more general than the word in question. For example, the hypernym of "dog" is "canine".
- **Hyponymy**: A hyponym is a word that is more specific than the word in question. For example, the hyponym of "dog" is "poodle".
- **Synonymy**: A synonym is a word that has the same meaning as the word in question. For example, the synonym of "good" is "well".
- **Antonymy**: An antonym is a word that has the opposite meaning of the word in question. For example, the antonym of "good" is "bad".

2.1.2 Word Embeddings

The way machine learning models process data is different from how humans do. For example, we can easily understand the text "I saw a cat", but our models can not — they need vectors of features. Such vectors, called word embeddings, are representations of words which can be fed into a model.

2.1.2.1 Word-count-based embedding

A common approach to represent a text document is to use a column vector of word counts. This embedding is often called a bag-of-words, because it includes only information about the count of each word, and not the order in which the words appear. The bag-of-words representation ignores grammar, sentence boundaries, paragraphs — everything but the words. Yet the bag of words model is surprisingly effective for text classification.

Another word-count-based method based on the encoding method, widely used in the information retrieval neighborhood, is TF-IDF, short for term frequency-inverse document frequency, which aims to reflect the significance ofthe specified term in the document collection and is one of the popular term weighting schemes.

2.1.2.2 Dense embeddings

Bag-of-words embeddings are sparse and long vectors with dimensions corresponding to words in the vocabulary or documents in a collection. A more powerful word representation is a dense vector, where instead of mostly-zero counts, the values will be real-valued numbers that can be negative. It turns out that dense vectors work better in every NLP task than sparse vectors.

Bengio et al. [1] presented a model which learned word representations using distributed representation. The authors presented a neural model which obtains word representations as to the product while training a language model. The popularity of word representation methods are due to two famous models, Word2Vec [32] and GloVe [41].

2.1.2.3 Contextual embeddings

To address the issue of polysemous and the context-dependent nature of words, we need to distinguish the semantics of words in different contexts.

Contextualised word embeddings are variable vectors that are dependent on the context in which the word is used. So, representations of a given word are multiple and are directly computed from their context. The context of a word is usually composed of the words surrounding it.

These contextualized representations are set to the hidden states of a deep neural model, which is trained as a language model. By running the language model, we obtain contextualized word representations, which can then be used as the base layer in a supervised neural network for any task. This approach yields significant gains over pre-trained word embeddings on several tasks, presumably because the contextualized embeddings use unlabeled data to learn how to integrate linguistic context into the base layer of the supervised neural network.

2.1.3 Masked Language Models

A Masked language model (MLM) is a pre-training technique which first masks out some tokens from the input sentences and then trains the model to predict the masked tokens by the rest of the tokens. A special `[MASK]` token is used to replace some words randomly in the original text.

Masked Language Modelling is usually solved as a classification problem. We feed the masked sequences to a neural encoder whose output vectors are further fed into a softmax classifier to predict the masked token.

The most popular MLM is BERT [12], which is a bidirectional encoder representation from a particular deep learning architecture called transformer [50]. It uses self-supervised training on the masked language modelling and next sentence prediction tasks to learn/produce contextual representations of words.

Concurrently, there are multiple research proposing different enhanced versions of MLM to further improve BERT. Instead of static masking, RoBERTa [29] improves BERT by dynamic masking. While other models aim to optimize BERT's performance, DistilBERT has a different goal. Its target is to reduce the large size and enhance the speed of BERT while still keeping as much strength as possible. DistilBERT [47] reduces the size of $BERT_{BASE}$ by 40%, enhances the speed by 60% while retaining 97% of its capabilities. ALBERT [26] also reduces the model size of BERT, it does not have to trade-off the performance. Compared to DistilBERT, which uses BERT as the teacher for its distillation process, ALBERT is trained from scratch (just like BERT).

2.1.4 Text classification

Classification lies at the heart of both human and machine intelligence. Deciding what letter, word, or image has been presented to our senses, recognizing faces or voices, sorting mail, assigning grades to homeworks; these are all examples of assigning a category to an input. In this section, we introduce text classification, the task of assigning a label or category to an entire text or document.

Given a text document, assign it a discrete label $y \in Y$, where Y is the set of possible labels. Text classification has many applications, from spam filtering to the analysis of electronic health records, or the categorization of news articles.

Classification is essential for tasks below the level of the document as well. An example of this is period disambiguation (deciding if a period is the end of a sentence or part of a word), or word tokenization (deciding if a character should be a word boundary). Even language modelling can be viewed as classification: each word can be thought of as a class, and so predicting the next word is classifying the context-so-far into a class for each next word. A part-of-speech tagger classifies each occurrence of a word in a sentence as, e.g., a noun or a verb.

The goal of classification is to take a single observation, extract some useful features, and thereby classify the observation into one of a set of discrete classes.

One method for classifying text is to use handwritten rules. There are many areas of language processing where handwritten rule-based classifiers constitute a state-of-the-art system, or at least part of it. Rules can be fragile, however, as situations or data change over time, and for some tasks humans aren't necessarily good at coming up with rules. Most cases of classification in language processing are instead done via supervised machine learning, where an algorithm learns how to map from an observation to a correct output [25].

Many kinds of machine learning algorithms are used to build classifiers. Formerly, statistical and machine learning approaches, such as naïve Bayes, k-nearest neighbours, hidden Markov models, conditional random fields (CRFs), decision trees, random forests, and support vector machines, were widely used to design classifiers. However, during the past several years, there has been a wholesale transformation, and these approaches have been entirely replaced, or at least enhanced, by neural network models [37].

2.1.5 Sentiment analysis

A popular application of text classification is sentiment analysis, the extraction of sentiment, the positive or negative orientation that a writer expresses toward some object. A review of a movie, book, or product on the web expresses the author's sentiment toward the product, while an editorial or political text expresses sentiment toward a candidate or political action. Extracting consumer or public sentiment is thus relevant for fields from marketing to politics. [25]

The simplest version of sentiment analysis is a binary classification task, and the words of the review provide excellent cues. Consider, for example, the following phrases extracted from positive and negative reviews of movies and restaurants. Words like great, richly, awesome, pathetic, awful and ridiculously are very informative cues:

+ *...zany characters and richly applied satire, and some great plot twists*

− *It was pathetic. The worst part about it was the boxing scenes...*

+ *...awesome caramel sauce and sweet toasty almonds. I love this place!*

− *...awful pizza and ridiculously overpriced...*

The area of sentiment analysis it is becoming increasingly popular and utilizing deep learning. Applications are varied, including product research, futures prediction, social media analysis, and classification of spam [59]. Good results were obtained using an ensemble, including both LSTMs and CNNs [9]. But the current trend in state-of-the-art models in all application areas is to use pretrained stacks of transformer units in some configuration, whether in encoder-decoder configurations or just as encoders.

2.1.6 Natural language inference

The task of Natural Language Inference (NLI), also known as recognizing textual entailment, asks a system to evaluate the relationships between the truth-conditional meanings of two sentences or, in other words, decide whether one sentence follows from another. The relationship can be entailment, contradiction, or neutral.

Specifically, natural language inference (NLI) is concerned with determining whether a natural language hypothesis h can be inferred from a premise p, as depicted in the following example from [31], where the hypothesis is regarded to be entailed from the premise.

p: Several airlines polled saw costs grow more than expected, even after adjusting for inflation.

h: Some of the companies in the poll reported cost increases.

2.1.7 Sequence-to-Sequence models

All the tasks we have discussed so far are classification-based, where the input is a text and the output is a label. However, there are many tasks where the input and output are both sequences of tokens. For example, machine translation, summarization, and question answering are all tasks where we want to generate a sequence in human-like language as output.

A Sequence to Sequence (seq2seq) model is is a special class of Recurrent Neural Network (RNN) architectures that can be used to solve these tasks.

In the general case, input sequences and output sequences have different lengths (e.g. machine translation) and the entire input sequence is required in order to start predicting the target. This requires a more advanced setup:

- An *encoder* processes the input sequence and returns its own internal state. This vector is called the context vector.

- A *decoder* is a neural network that takes the context vector as input and outputs a sequence of tokens. It is trained to predict the next characters of the target sequence, given previous characters of the target sequence.

An example of this architecture is shown in Figure 2.2.

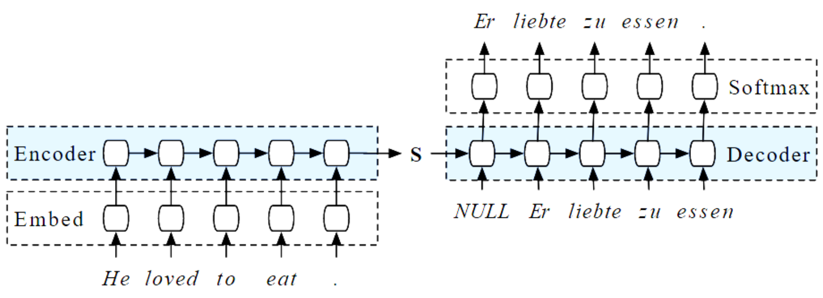


Figure 2.2: An example of a sequence-to-sequence model for machine translation.

2.2 Adversarial Machine Learning

Deep Learning algorithms have achieved the state-of-the-art performance in many tasks. However, the interpretability of deep neural networks is still unsatisfactory as they work as black boxes, which means it is difficult to get intuitions from what each neuron exactly has learned. One of the problems of the poor interpretability is evaluating the robustness of deep neural networks.

Adversarial Machine Learning is a collection of techniques to train neural networks on how to spot intentionally misleading data or behaviors. This differs from the standard classification problem in machine learning, since the goal is not just to spot "bad" inputs, but preemptively locate vulnerabilities and craft more flexible learning algorithms.

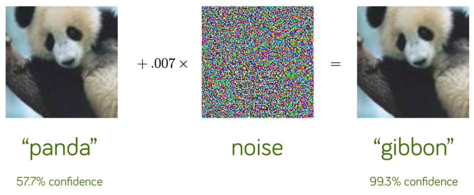
The objective of an adversary could be to attempt to manipulate either the data collection or the data processing in order to corrupt the target model, thus tampering the original output.

Unlike traditional cybersecurity attacks, these weaknesses are not due to mistakes made by programmers or users. They are just shortcomings of the current state-of-the-art methods. Put more bluntly, the algorithms that cause AI systems to work so well are imperfect, and their systematic limitations create opportunities for adversaries to attack. At least for the foreseeable future, this is just a fact of mathematical life [11].

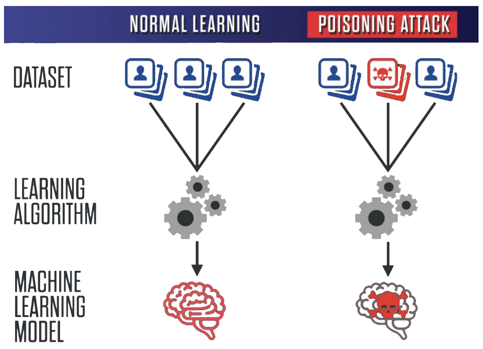
Two main types of AI attacks can be defined according to the time at which the attack happens [7]:

- **Adversarial Attacks (Evasion)**: this is the most common type of attack in the adversarial setting. The adversary tries to evade the system by adjusting malicious samples during the testing phase. This setting does not assume any influence over the training data. In Figure 2.3a is depicted how adding an imperceptible and carefully constructed noise to the input originally recognized as "panda" with 57.7% confidence, we can change the classification output given by the same neural network toward another target (in the example "gibbon" with 99.3% confidence).

- **Data Poisoning Attacks**: This type of attack, known as contamination of the training data, is carried out at training phase of the machine learning model. An adversary tries to inject skilfully crafted samples to poison the system in order to compromise the entire learning process. The Figure 2.3b shows as in normal machine learning (left), the learning algorithm extracts patterns from a dataset, and the "learned" knowledge is stored in the machine learning model—the brain of the system. In a poisoning attack (right), the attacker changes the training data to poison the learned model.



(a) Evasion attack on image classification [48]



(b) Poisoning attack on training data [11]

Figure 2.3: Examples of Artificial Intelligence Attacks

In this thesis, we will focus on the first type of attack: Adversarial Attacks. Over the past few years, researchers [15, 48] used small unperceivable perturbations to evaluate the robustness of deep neural networks and found that they are not robust to these perturbations.

2.2.1 Adversarial examples

Szegedy et al. [48] first evaluated the state-of-the-art deep neural networks used for image classification with small generated perturbations on the input images. They found that the image classifiers were fooled with high probability, but human judgment is not affected. The perturbed image pixels were named *adversarial examples* and this notation is later used to denote all kinds of perturbed samples in a general manner. Formally, adversarial example x' is an example created via worst-case perturbation of the input to a deep learning model. An ideal deep neural network would still assign correct class y (in the case of classification task) to x', while a victim deep neural network would have high confidence on wrong prediction of x'. x' can be formalized as:

$$\begin{aligned} & x' = x + \eta, \quad f(x) = y, \quad x \in X, \\ & f(x') \neq y, \\ & \text{or} \quad f(x') = y', \quad y \neq y' \end{aligned} \tag{2.1}$$

where η is the worst-case perturbation. The goal of the adversarial attack can be deviating the label to incorrect one ($f(x') \neq y$) or specified one ($f(x') = y'$) [56].

2.2.2 Paradigm shift: from CV to NLP

Adversarial examples were first proposed for attacking DNNs for object recognition in the Computer Vision (CV) community. The former work on this field by Szegedy et. al. [48] was based on L-BFGS, but despite the effectiveness of the method, it was computationally expensive and impractical. Goodfellow et al. [15] proposed a Fast Sign Gradient Method (FSGM) that popularized this research topic. It is a simplification of the L-BFGS method since it adds a small perturbation to the input of a model, which is computed by taking the sign of the gradient of the loss function with respect to the input. Most follow-up research was based on optimization methods (eg. JSMA [40], DeepFool [34], C&W [5]) or leveraging Generative Adversarial Network (GAN) to generate adversaries [58].

As shown in Figure 2.4, adversarial technology has attracted attention and has developed rapidly. Based on the paper list[1] collected by Carlini, the chart counts the number of publications related to adversarial in the CV and NLP fields. Compared with studies in the CV field, the publications in the NLP domain are far fewer. However, due to the wide application of NLP technology in text classification, sentiment analysis, text question-answering, neural machine translation, text generation and other tasks, as well as the continuous deepening of adversarial attack and defence technologies, textual adversarial technology has gradually gained researchers' attention.

Papernot et al. [39] are the first to investigate adversarial attacks on texts. Inspired by the idea of generating adversarial images, they crafted adversarial texts through the forward derivative associated with texts' embeddings, by modifying characters or words in original texts.

[1]https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html

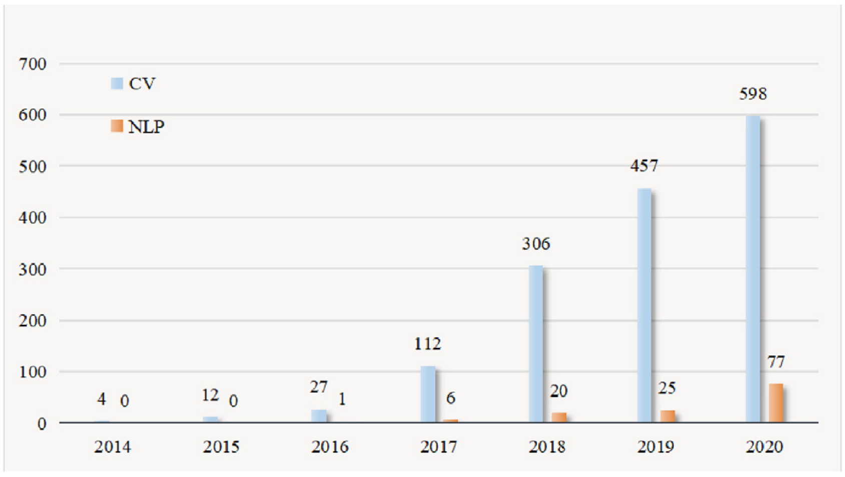


Figure 2.4: Adversarial technology trend in CV and NLP fields [42]

2.2.2.1 Particularities of adversarial text

Publications related to adversarial technology in the NLP field are far less than those in the CV field. The reason is that three extra constraints need to be considered when generating textual adversarial examples. Specifically:

- **Discrete**: Unlike images represented by continuous pixel values, the symbolic text is discrete. Therefore, finding appropriate perturbations is critical to efficient textual adversarial example generation. It is hard to define the perturbations on texts. Carefully designed variants or distance measurements for textual perturbations are required.

- **Perceivable**: The well-performed adversarial image generation method is based on the premise that a few pixel value changes in an image are invisible to human eyes. However, a slight modification of a character or word is easily realized by human eyes and spelling checkers. Hence, finding textual adversarial examples that are hard to be observed by human eyes is vital for successful adversarial attacks.

- **Semantic**: Compared with images whose overall semantics do not change when changing a few pixel values, the semantics of a text could be altered by even replacing or adding a character, violating the principle that adversarial examples are perceivable to humans. Therefore, keeping the semantics consistent is the key to crafting influential adversarial texts.

These differences make it extraordinarily difficult for researchers to employ methods dealing with images to adversarial attacks. Moreover, one of the first

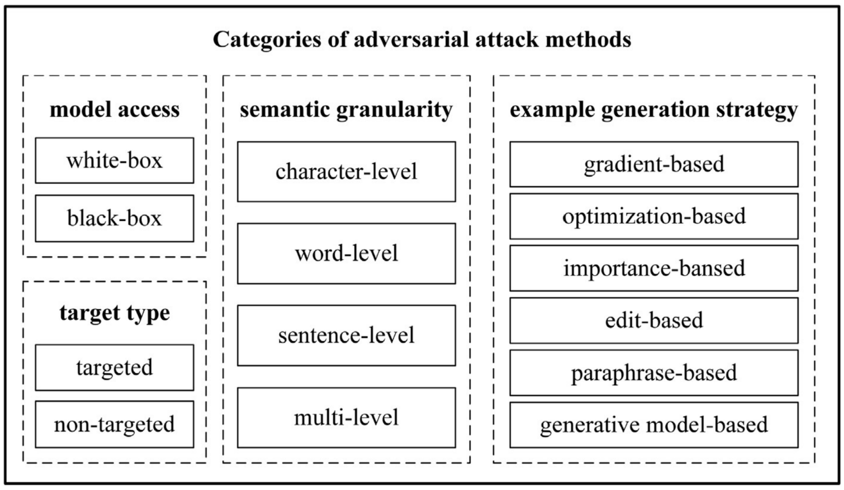


Figure 2.5: Categorization of textual adversarial attack methods [42]

tasks of NLP models is to work on real data to check their generalization ability. Although adversarial attacks are a practical approach to evaluate robustness, most of them have the problem of being task-specific, not being well generalized, and not presenting comprehensive guidelines for evaluating system robustness.

2.2.3 Taxonomy of textual adversarial attacks

Textual adversarial attack methods can be categorized according different criteria. In this section, we will introduce the taxonomy of textual adversarial attacks based on model access, adversarial goal, semantic granularity and attack strategy, as shown in Figure 2.5.

2.2.3.1 Model access

Adversarial attacks at the testing time do not tamper with the targeted model but rather forces it to produce incorrect outputs. The effectiveness of such attacks is determined mainly by the amount of information available to the adversary about the model. Testing phase attacks can be broadly classified into either *white-box* or *black-box* attacks [7].

White-Box Attacks. In white-box attack on a machine learning model, an adversary has total knowledge about the model used for classification (e.g., type of neural network along with number of layers). The attacker has information about the algorithm used in training (e.g. gradient-descent optimization) and can access the training data distribution. He also knows the parameters of the fully trained model architecture. The adversary utilizes available information to identify the feature space where the model may be vulnerable, for which the model has a high error rate. Then the model is exploited by altering an input using adversarial example crafting method.

Black-Box Attacks. Black-box attack, on the contrary, assumes no knowledge about the model and uses information about the settings or past inputs to analyse the vulnerability of the model. For example, in an oracle attack, the adversary exploits a model by providing a series of carefully crafted inputs and observing outputs. For example, to identify meaningful words in given texts, works in [24, 45, 46] computed the probability change value, word saliency, and classification probability by using the victim model's output.

2.2.3.2 Adversarial goal

According to the attack purpose of an adversary, adversarial attack methods are categorized into targeted and non-targeted attacks [42].

Non-targeted attack. The adversary hopes to generate an adversarial example x' that makes the victim model f produce a wrong output $f(x') \neq y$, where y is the correct label of the input x. Since there is no limit on the target's wrong output, this kind of attack is more frequently employed.

Targeted attack. In this scenario, the adversary intends to generate adversarial examples that make victim models output a specified wrong prediction. More specifically, the adversary hopes that the generated example x' to cause the victim model f outputting $t = f(x')$, where t is the output specified by the adversary.

2.2.3.3 Attack strategy

According to different strategies in the adversarial example generation process, Shilin Qiu et al. [42] divide adversarial attacks into six types: gradient-based, optimization-based, importance-based, edit-based, paraphrase-based, and generative model-based methods. Among them, strategies like the gradient-based method are evolved from adversarial image generation methods, and the implementation process of these attacks is usually relatively straightforward. While other methods like the optimization-based and edit-based methods are proposed for discrete data, they generally show better performance in maintaining semantic consistency and grammatical correctness; however, they have enormous difficulty when designing well-turned algorithms.

Gradient-based. These methods calculate the forward derivative to the input and obtain adversarial perturbations by gradient backpropagation. Therefore, the vectorization for text needs to be implemented at first.

Optimization-based. It regards the adversarial example generation as a minimax optimization problem, i.e., maximizing the victim model's prediction error while the difference between the adversarial example and the original one is within an acceptable range. Currently, researchers craft adversarial texts essentially based on evolutionary algorithms, such as the GA and PSO.

Importance-based. This means that which object is to be modified and how to modify it are determined by each object's importance related to the victim model. Since the more critical the changed word is, the easier it is to change the prediction of the victim model, even if the change is small enough. The adversarial example generated by this strategy generally maintains semantic consistency, grammatical, and syntactic correctness well.

Edit-based. It crafts adversarial examples by operations like inserting, removing, and swapping characters, words, or sentences. These editing operations are also used in other approaches, such as gradient-based, optimization-based, and importance-based methods. Therefore, the edit-based method refers to attacks that utilize the above editing operations but do not use the gradient information, optimization algorithm, or item importance.

Paraphrase-based. The adversary takes the paraphrase of a sentence as its adversarial example. In the paraphrase generation process, the adversary introduces different extra conditions to fool the victim model without affecting human judgment. The sentence-level attacks commonly use these approaches.

Generative model-based. This method uses the generative model like the GAN and encoder-decoder model to generate adversarial texts, and is frequently used in sentence-level attacks. Since there are gradient back propagations when training the generative model or crafting adversarial examples, these methods are usually combined with other techniques, such as RL.

2.2.3.4 Semantic granularity

Since the text is discrete, classifying adversarial attacks according to semantic granularity is more intuitive than NLP tasks or black/white-box scenarios. Thus, Huq et al. [22] divided textual adversarial attacks into four categories: character-level, word-level, sentence-level, and multi-level attack.

Character-Level Attack. Individual characters in this attack are either modified with new characters, special characters, and numbers. These are either added to the text, swapped with a neighbour, removed from the word, or flipped.

Word-Level Attack. In this attacks words from the texts are changed with their synonyms, antonyms, or changed to appear as a typing mistake or removed completely.

Sentence-Level Attack. This attack inserts extra distractor sentences, generates the paraphrase, or modifies the original sentence structure to fool the victim model.

Multi-Level Attack. Attacks which can be used in a combination of character, word, and sentence level are called multi-level attacks.

2.2.4 Adversarial attack methods from literature

Several adversarial attack methods have been proposed in the literature. In this section, we present a brief overview of the most popular ones, listed in Table 2.1.

Those methods are categorized according to the classification defined in Sec. 2.2.3.4: 1 character-level attack (DeepWordBug [13]), 4 word-level attacks (Probabilistic Weighted Word Saliency (PWWS) [45], TextFooler [24], BERT-based attack [28] and Semese-PSO [54]), 2 sentence-level attacks (Synthetically Controlled Paraphrase Networks (SCPNs [23]), and GAN-based attack [58]), and 1 multi-level attack (TextBugger [27]).

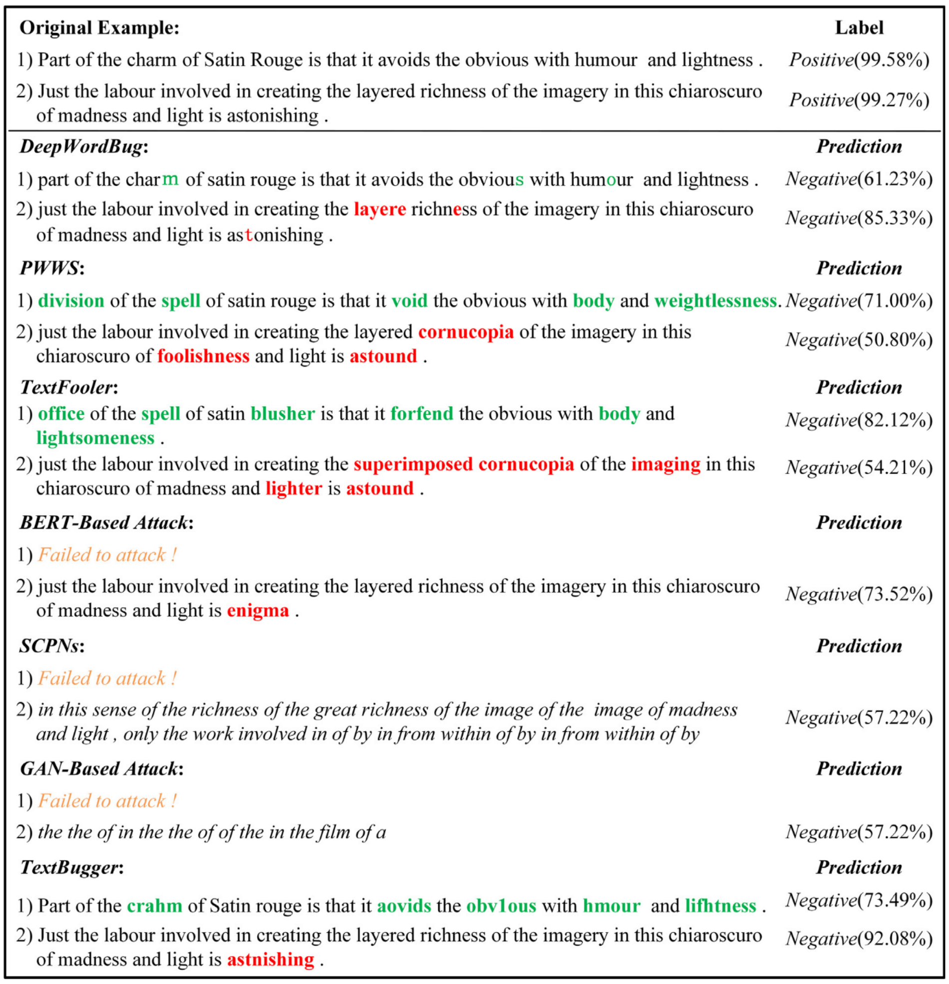
DeepWordBug determines top critical tokens and modify them by character-level transformations introducing typos. PWWS is a synonym-based substitution method that makes use of the word saliency and classification probability. TextFooler identifies important words, and replaces them with the most semantically similar and grammatically correct substitutes. BERT-based attack finds the vulnerable words for the target model and replaces them with candidates from a pre-trained BERT model. Semese-PSO reduces search space by a sememe-based word replacement method, searching for adversarial examples through the PSO algorithm in the reduced search space. SCPNs generates adversarial examples by paraphrasing the original sentence using an encoder-decoder model. GAN-based attack generates adversarial examples using iterative stochastic search and hybrid shrinking search. The framework consists of a GAN and a converter. TextBugger generates character-level and word-level adversarial texts according to the importance in black-box and white-box scenarios.

To give readers a more intuitive understanding of these attack methods, in Figure 2.6 are showed adversarial examples generated by each method. The original examples are two randomly selected from the Stanford Sentiment Treebank (SST) dataset, and both of them are correctly classified as Positive by the pre-trained BERT model for sentiment analysis.

Method	Granularity	Strategy	Model access	Attack goal
DeepWordBug	Character-level	Importance-based	Black-box	Non-targeted
PWWS	Word-level	Importance-based	Black-box	Non-targeted
TextFooler	Word-level	Importance-based	Black-box	Non-targeted
BERT-based	Word-level	Importance-based	Black-box	Non-targeted
Semese-PSO	Word-level	Optimization-based	Black-box	Non-targeted
SCPNs	Sentence-level	Paraphrase-based	Black-box	Non-targeted
GAN-based	Sentence-level	Generative model-based	Black-box	Non-targeted
TextBugger	Multi-level	Importance-based	Black/White-box	Non-targeted

Table 2.1: Several adversarial attack methods from literature

From the perspective of example quality, character-level attack methods maintain the semantics of original texts well. However, they are easily detected by human eyes or spelling check tools. In contrast, word-level attacks compensate for the vulnerability of adversarial examples to detection but affect the semantics of the text to some extent. sentence-level attacks enhance the diversity of generated examples. However, it is clear to see that these adversarial examples crafted by sentence-level SCPNs and GAN-based methods are very different from the original ones in both semantics and readability.



Original Example:	**Label**
1) Part of the charm of Satin Rouge is that it avoids the obvious with humour and lightness .	*Positive*(99.58%)
2) Just the labour involved in creating the layered richness of the imagery in this chiaroscuro of madness and light is astonishing .	*Positive*(99.27%)
***DeepWordBug*:**	***Prediction***
1) part of the charm of satin rouge is that it avoids the obvious with humour and lightness .	*Negative*(61.23%)
2) just the labour involved in creating the **layere** richness of the imagery in this chiaroscuro of madness and light is astonishing .	*Negative*(85.33%)
***PWWS*:**	***Prediction***
1) **division** of the **spell** of satin rouge is that it **void** the obvious with **body** and **weightlessness**.	*Negative*(71.00%)
2) just the labour involved in creating the layered **cornucopia** of the imagery in this chiaroscuro of **foolishness** and light is **astound** .	*Negative*(50.80%)
***TextFooler*:**	***Prediction***
1) **office** of the **spell** of satin **blusher** is that it **forfend** the obvious with **body** and **lightsomeness** .	*Negative*(82.12%)
2) just the labour involved in creating the **superimposed cornucopia** of the **imaging** in this chiaroscuro of madness and **lighter** is **astound** .	*Negative*(54.21%)
***BERT-Based Attack*:**	***Prediction***
1) *Failed to attack !*	
2) just the labour involved in creating the layered richness of the imagery in this chiaroscuro of madness and light is **enigma** .	*Negative*(73.52%)
***SCPNs*:**	***Prediction***
1) *Failed to attack !*	
2) *in this sense of the richness of the great richness of the image of the image of madness and light , only the work involved in of by in from within of by in from within of by*	*Negative*(57.22%)
***GAN-Based Attack*:**	***Prediction***
1) *Failed to attack !*	
2) *the the of in the the of of the in the film of a*	*Negative*(57.22%)
***TextBugger*:**	***Prediction***
1) Part of the **crahm** of Satin rouge is that it **aovids** the **obv1ous** with **hmour** and **lifhtness** .	*Negative*(73.49%)
2) Just the labour involved in creating the layered richness of the imagery in this chiaroscuro of madness and light is **astnishing** .	*Negative*(92.08%)

Figure 2.6: Adversarial examples generated by different methods [42]

For comparing the attack performance of the above methods, Shilin Qiu et. al. [42] randomly selected 5000 examples from the SST dataset to generate corresponding adversarial texts and attack the selected victim model using the above methods. The end goal of the attack algorithms is to trick the model to make wrong prediction by manipulating the input. So the attack success rate of an evasion algorithm is defined as the percentage of wrong prediction by the victim model on the adversarial examples [49]. Table 2.2 shows the result. In terms of attack success rate, TextBugger is the highest, and its execution time is also relatively low. The reason might be that TextBugger uses the Jacobian matrix to calculate the importance of each word at once. In comparison, the average model queries of sentence-level methods (SCPNs and GAN-based method) are the lowest, but their attack success rates are not satisfactory. As mentioned above, the differences between the adversarial examples generated by sentence-level methods and the original ones are relatively huge, so researchers should focus on maintaining the semantic consistency and imperceptibility of texts for sentence-level methods. Focusing on word-level attacks, the model query are comparatively numerous. However, TextFooler and BERT-based attack have a relatively highest attack success rate and a low execution time.

Since this thesis focuses on the word-level attack, we will introduce the details of TextFooler and BERT-based attack in the following sections and use them as the baseline for our proposed method.

Method	Attack Success Rate	Average Model Queries (times)	Average Running Time (seconds)
DeepWordBug	59.46%	23.626	0.000439
PWWS	75.74%	117.82	0.002190
TextFooler	74.86%	61.68	0.053360
BERT-based	88.44%	61.94	0.036131
SCPNs	75.66%	11.75	2.366100
GAN-based	42.06%	2.42	0.009495
TextBugger	90.54%	48.79	0.001722

Table 2.2: Comparison of Adversarial Attacks Performance [42]

2.2.4.1 TextFooler

TextFooler [24] is a word-level attack in the black-box setting designed to evade two fundamental natural language tasks, text classification and textual entailment.

For generating semantics-preserving texts with minimum modifications, it uses an importance-based strategy. First, a selection mechanism is performed to choose the words that most significantly influence the final prediction results. Those words are ranked in descending order according to the class probability changes, which were obtained by removing words one by one and measuring the difference between the prediction confidence before and after deleting each word. After ranking the words by their importance score, stop words (such as "the", "when", and "none") derived from NLTK[2] are filtered out. This simple step of filtering is important to avoid grammar destruction.

Starting from these importance ranked words, three strategies (synonym extraction, part-of-speech checking, and semantic similarity checking) are combined to replace words with the most semantically similar and grammatically correct substitutes.

[2]https://www.nltk.org/

Synonym Extraction. Word replacement candidates are initiated with 50 closest synonyms according to the cosine similarity between w_i and every other word in the vocabulary. To represent the words, counter-fitted word embeddings from Mrkšić et al. [36] are used. These GloVe vectors are specially curated for injecting antonymy and synonymy constraints into vector space representations. They achieve the state-of-the-art performance on SimLex-999, a dataset designed to measure how well different models judge the semantic similarity between words [21].

POS Checking. To ensure the grammatical correctness of the generated adversarial examples, the Part of Speech (POS) tags of the original words are checked against the POS tags of the replacement candidates, and only the words with the same POS tags are kept.

Semantic Similarity Checking. Each remaining candidate word is substituted into the original sentence X, and obtain the adversarial example X_{adv}. Then, the sentence semantic similarity is calculated between the source X and adversarial counterpart X_{adv}. Specifically, Universal Sentence Encoder (USE) [6] is used to encode the two sentences into high-dimensional vectors and use their cosine similarity score as an approximation of semantic similarity. The words resulting in similarity scores above a preset threshold are placed into the final candidate pool.

Finally, the target model F to compute the corresponding prediction scores $F(X_{adv})$. If there exists any candidate that can already alter the prediction of the target model, then it is selected the word with the highest semantic similarity score among these winning candidates. But if not, then it is selected the word with the least confidence score of label y as the best replacement word for w_i, and repeat synonym extraction to transform the next selected word by importance rank.

2.2.4.2 BERT-based attacks

A shortcoming of traditional synonym-based attacks like TextFooler or PWWS is that they do not take the context into account when building their candidate set. This can lead to problems if a word is polysemic, i.e., has multiple meanings in different contexts, which are easily human-identifiable. Many attacks also do not take part-of-speech into account, which leads to unnatural and semantically wrong sentences [20].

BERT-based attacks claim to produce more natural text by relying on a BERT masked language model for proposing the set of candidate words. Compared with previous approaches using rule-based perturbation strategies, the masked language model prediction is context-aware, thus dynamically searches for perturbations rather than simple synonyms replacing.

A prominent example of such an attack is BERT-Attack [28]. BERT-Attack calculates the importance scores similar to TextFooler, but instead of deleting words, BERT-Attack replaces the word for which the importance score is calculated with the `[MASK]` token:

$$I_{w_i} = F_y(X) - F_y(X_{w_i \to \texttt{[MASK]}}) \tag{2.2}$$

The candidate set L_i is constructed from the top 48 predictions of the masked language model and the replacement word is chosen as the word which changes the prediction the most, subject to $cos_{USE}(X, X_{adv}) \geq 0.2$. Stopwords are filtered out using NLTK.

Another similar method is BAE, which corresponds to BAE-R in [14]. Like BERT-Attack, BAE is an attack based on a MLM. The word importance is estimated as the decrease in probability of the correct label when deleting a word, similar to TextFooler. BAE uses the top 50 candidates of the MLM to build the candidate set and tries to enforce semantic similarity by requiring $cos_{USE}(X, X_{adv}) \geq 0.936$.

2.3 Machine Learning hardening

Adversarial examples demonstrate that many modern machine learning algorithms can be broken easily in surprising ways. An essential purpose for generating adversarial examples for neural networks is to utilize these adversarial examples to enhance the model's robustness.

The overwhelming amount of work in the last few years for adversarial defences has given good competition to the novel adversarial attack algorithms and considerably improved the robustness of existing deep learning models. These defence mechanisms are also used as regularization techniques to avoid overfitting, and making the model more robust [16].

2.3.1 Vanilla adversarial training

One of the most popular adversarial defence approach is adversarial training. It was first introduced in the work proposed in [15]. It is a method of defending against adversarial attacks by introducing adversarial examples in the training data. The strength of adversarial examples decides the final robustness and generalization achieved by the model.

This method can be seen as a data augmentation mechanism that extends the original training set with the successfully generated adversarial examples and try to let the model see more data during the training process. Adversarial examples need to be carefully designed when training on adversarial examples to improve the model.

Although adversarial training can effectively improve the robustness of NLP models, this approach has some problems:

- extensive adversarial examples need to be prepared in advance, resulting in a massive calculation consumption
- it is likely to reduce the model classification accuracy

2.3.2 Attack to Training

High computational cost hinders the use of vanilla adversarial training in NLP, and it is unclear how and to what extent such training can improve an NLP model's performance.

Yoo et al. [52] propose to improve the vanilla adversarial training in NLP with a computationally cheaper adversary, referred to as Attack to Training (A2T). A2T attempts to generate adversarial examples on the fly during training of the model on the training set, which is much cheaper than generating adversarial examples in advance. This approach can improve an NLP model's robustness to the attack it was originally trained with and also defend the model against other types of word substitution attacks.

The attack component in A2T is designed to be is faster than previous attacks from the literature. Previous attacks such as [14, 24] iteratively replace one word at a time to generate adversarial examples. One issue with this method is that an additional forward pass of the model must be made for each. word to calculate its importance. For longer text inputs, this can mean that we have to make up to hundreds of forward passes to generate one adversarial example.

A2T instead determines each word's importance using the gradient of the loss. For an input text including n words: $x = (x_1, x_2, ..., x_n)$ where each x_i is a word, the importance of x_i is calculated as:

$$I(x_i) = \|\nabla_{e_i} L(\theta, x, y)\|_1 \tag{2.3}$$

where e_i is the word embedding that corresponds to word x_i. For BERT and RoBERTa models where inputs are tokenized into sub-words, we calculate the importance of each word by taking the average of all sub-words constituting the word. This requires only one forward and backward pass and saves us from having to make additional forward passes for each word.

2.4 Text Attack

The only textual adversarial attack toolbox currently available are TextAttack [35] and OpenAttack [55]. TextAttack is the earliest implemented tool and it is python framework to launch adversarial attacks, enable data augmentation and implement adversarial training for natural language process models. It can launch model-specific and evaluate the results, and improve robustness in the downstream and model generalization.

TextAttack provides clean, readable implementations of 19 adversarial attacks from the literature. All of these attacks are implemented as *attack recipes* in TextAttack and can be benchmarked with just a single command.

2.4.1 Framework structure

To unify adversarial attack methods into one system, NLP attacks are decomposed into four components: a goal function, a set of constraints, a transformation, and a search method:

- **Goal function**: determines whether the attack is successful in terms of the model outputs (eg. untargeted classification, targeted classification)
- **Constraints**: determine if a perturbation is valid with respect to the original input (eg. maximum word embedding distance, part-of-speech consistency)
- **Transformation**: generates a set of potential perturbations (eg. word swap, word insertion)
- **Search method**: successively queries the model and selects promising perturbations from a set of transformations (eg. greedy search, genetic algorithm)

This modular design enables us to easily assemble attacks from the literature while reusing components that are shared across attacks.

TextAttack's design also allows researchers to easily construct new attacks from combinations of novel and existing components. Figure 2.7 shows the main components and features of TextAttack.

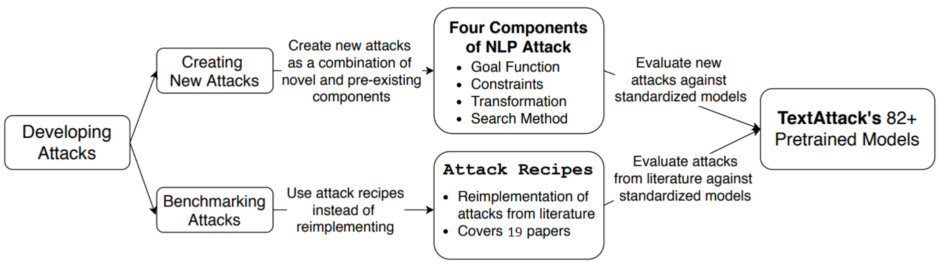

Developing Attacks
Creating New Attacks
Create new attacks as a combination of novel and pre-existing components
Four Components of NLP Attack
• Goal Function
• Constraints
• Transformation
• Search Method
Evaluate new attacks against standardized models
TextAttack's 82+ Pretrained Models
Benchmarking Attacks
Use attack recipes instead of reimplementing
Attack Recipes
• Reimplementation of attacks from literature
• Covers 19 papers
Evaluate attacks from literature against standardized models

Figure 2.7: Main features of of TextAttack

2.4.2 HuggingFace integration

TextAttack is model-agnostic—meaning it can run attacks on models implemented in any deep learning framework. Model objects must be able to take a string (or list of strings) and return an output that can be processed by the goal function.

TextAttack allows users to provide their own models and datasets. Moreover, it is directly integrated with HuggingFace[3]'s transformers and NLP libraries. This allows users to test attacks on models and datasets publicly available on the platform.

For benchmark purpose, TextAttack provides users with 82 pre-trained models, including word-level LSTM, word-level CNN, BERT, and other transformer-based models pre-trained on various datasets provided by HuggingFace NLP. Since TextAttack is integrated with the NLP library, it can automatically load the test or validation data set for the corresponding pre-trained model [35].

[3]https://huggingface.co

Chapter 3

Methodology

In this chapter, we will present the proposed solution to the problem of adversarial attacks on text classification models.

3.1 Defined goal

Most recent adversarial attack methods described in 2.2.3 successfully decrease the accuracy of the target model. However, since the main goal of the research is to enhance the robustness of the target model, we need to ensure that the quality of the crafted adversarial examples should be high, because the target model will be re-trained with them.

A successful natural language adversarial example can be defined as a perturbation that fools the model and fulfils a set of linguistic constraints. As an example in sentiment analysis (Figure 3.1), an attacker can fool the system by changing only one word from "Perfect" to "Spotless," which can completely change the predicted output without being discerned by humans.

Formally, besides the ability to fool the target models, the outputs of a natural language attacking system should meet three key utility-preserving properties, defined by Jin et al. [24] as follows:

1. *human prediction consistency* - prediction by humans should remain unchanged;
2. *semantic similarity* - the crafted example should bear the same meaning as the source, as judged by humans
3. *language fluency* - generated examples should look natural and grammatical

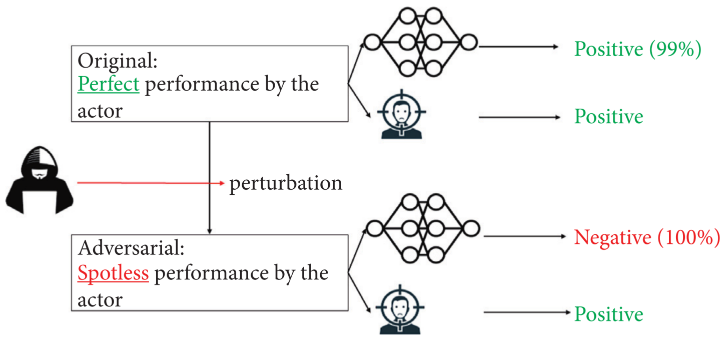


Figure 3.1: Textual adversarial attack in sentiment analysis [19]

3.1.1 Problem to solve

Although attacks in NLP aspire to meet linguistic constraints, in practice, they frequently violate them. Focusing on TextFooler and BERT-based attacks, they claim to create perturbations that preserve semantics, maintain grammaticality, and are not suspicious to readers. However, our inspection of the perturbations revealed that many violated these constraints.

On one hand, the perturbations generated by TextFooler solely account for the token level similarity via word embeddings, and not the overall sentence semantics. This can lead to out-of-context and unnaturally complex replacements.

On the other hand, with the capability of BERT, the perturbations crafted by BERT-based attacks are generated considering the context around. Therefore, the perturbations are fluent and reasonable. Nevertheless, candidates generated from the masked language model can sometimes be antonyms or irrelevant to the original words, causing a semantic loss.

Table 3.1 highlights some examples of adversaries suffering from aforementioned issues.

Method	Label	Samples
TextFooler	Original (POS) Perturbed (NEG)	generates an enormous feeling of empathy for its characters leeds an enormous foreboding of empathy for its fonts
BAE	Original (NEG) Perturbed (POS)	bears is even worse than i imagined a movie ever could be. bears is even greater than i imagined a movie ever could be.

Table 3.1: Some adversarial samples generated with TextFooler and BAE

3.1.2 Research objective

In order to address the research questions defined in Section 1.3, we propose a new approach to generate adversarial examples that meet linguistic constraints. In particular, we aim to demonstrate that:

- using our evaluation framework defined in 4, we can identify the weaknesses of existing attack methods;
- the utility-preserving properties measured on the proposed solution outperform the state-of-the-art;

3.2 Research design

Although adversarial attacks are a practical approach to evaluate robustness, most of them have the problem of being task-specific, not being well generalized. Thus, this thesis is focused only on the text classification task, including sentiment analysis, topic classification, and natural language inference.

As baseline methods for adversarial attacks, we choose TextFooler [24] and BAE [14]. Those are compared with the proposed method from a qualitative perspective. Moreover a quantitative evaluation is performed to measure the efficiency.

3.2.1 Attack category

There are exploding combinations of the categories listed in Section 2.2.3 into which our proposed attack method can fall into. As a design choice, we defined the category of the attack in which we want to conduct our research.

Mainstream work has focused on word-level perturbation because of the large search space of substitution words and the hardness to maintain sentence semantics. Word-level methods usually maintain imperceptibility better than other attacks, so we also explored this category of attacks.

Differently from the baseline methods (TextFooler and BAE), our proposal attempt to generate adversarial examples in a white-box setting, which means that the attacker has access to the target model. This is a more realistic scenario, since the goal of the attacker is to craft adversaries with the purpose of enhancing the robustness of the target model. Thus, he has full access to the model and can exploit it to generate the perturbations.

The attack strategy and the adversarial goal are the same as the baselines, respectively importance-based and non-targeted attack.

3.3 Proposed solution

TextFooler and BERT-Attack suffer respectively from a lack of context and semantic similarity. We tried to combine their strengths in a novel method called SynBA (contextualized Synonym-Based adversarial Attack). The main idea is to generate a set of synonyms for each word in the sentence, and then select the best synonym for each word based on the context and semantic similarity.

3.3.1 Intuition

In order to achieve semantic-consistent adversaries, we need to consider the cosine similarity between word embeddings or exploit a synonym dictionary. While the synonyms retrieved from a thesaurus like WordNet are often somewhat related to the original word, the relationship is often the wrong one for the given context. Conversely, BERT has the potential to generate more fluent substitutions for an input text.

Our intuition is that the ranked list of candidates for word replacement is obtained from the so called *SynBA score*, a weighted function summing up three scores:

- **MLM score** - the confidence of the candidate obtained by MLM (BERT)
- **Thesaurus score** - a score assigned to synonyms, hyponyms, and hypernyms of the original word in WordNet
- **Word embedding score** - the cosine similarity between the original word and the candidate

Combining these three scores, we can obtain a ranked list of candidates that results in a more contextualized and semantically consistent adversary.

3.3.2 SynBA components

SynBA has been implemented using TextAttack, a Python framework for implementing adversarial attacks in NLP (refer to section 2.4 for more details). Following the framework structure, we decomposed our attack method into four components: a goal function, a set of constraints, a transformation, and a search method.

We reused some of the pre-existing components in TextAttack, such as the `UntargetedClassification` goal function, the `GreedyWordSwapWIR` search method and the constraints. The innovative part of SynBA is the `WordSwapMultimodal` transformation, implementing the SynBA score mechanism. Table 3.2 summarizes the differences between SynBA and the two baselines.

Components	**TextFooler**	**BAE**	**SynBA**
Search Method for Ranking Words	Deletion-based Word Importance	Deletion-based Word Importance	Gradient-based Word Importance
Transformation	Word Embedding	BERT MLM	SynBA score
Constraints	POS USE Word Embedding Distance	POS USE	POS Sentence-BERT Word Embedding Distance Max Modification Rate

Table 3.2: Comparing SynBA components with TextFooler and BAE

3.3.2.1 Search Method

The search method is responsible for going through the search space of possible perturbations and finding a sequence of transformations that produce a successful adversarial example.

Greedy algorithms with word importance ranking are linear concerning input length, with a complexity of $\mathcal{O}(W * T)$, where W indicates the number of words in the input, T is the maximum number of transformation options for a given input [53].

So in SynBA we use the `GreedyWordSwapWIR` search method, which is a greedy search method that iteratively applies the transformation to the input text, and selects the best candidate for each word based on the Word Importance Ranking (WIR) score. Words of the given input are ranked according to the importance function. Then, in order of descending importance, each word is substituted with the best candidate given by the transformation that maximizes the scoring function until the goal is achieved, or all words have been perturbed.

The WIR is gradient-based, which means that it is able to rank the words in the input text exploiting the gradient of the loss function with respect to each token.

It is the same search method used by the attack component in A2T (see section 2.3.2). Yoo et al. [53] showed that the gradient-ordering method is the fastest search method and provides a competitive attack success rate when compared to the deletion-based method.

3.3.2.2 Transformation

To enforce semantic preservation, we designed the `WordSwapMultimodal` transformation function, which is not provided by TextAttack. It computes a set of k perturbations given a word in the input text selected by the search method, where k is the number of candidates to be generated.

The transformation function is based on the SynBA score, which is described in Figure 3.2. It is calculated for each word in the vocabulary and then they are ranked in descending order. Tree components are used to compute the SynBA score: the *MLM score*, the *thesaurus score*, and the *word embedding score*.

The *MLM score* is the confidence of the candidate obtained by `bert-base-uncased`[4] MLM masking the word that we want to perturb. Since confidence is a probability value, we need to normalize it in order to have each component magnitude in the same range $[0, 1]$. So the vector output of the MLM is rescaled using a min-max scaling:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3.1}$$

The *thesaurus score* makes use of WordNet, a lexical database for the English language. Each word in the vocabulary is associated with a score depending on the relation with the original word:

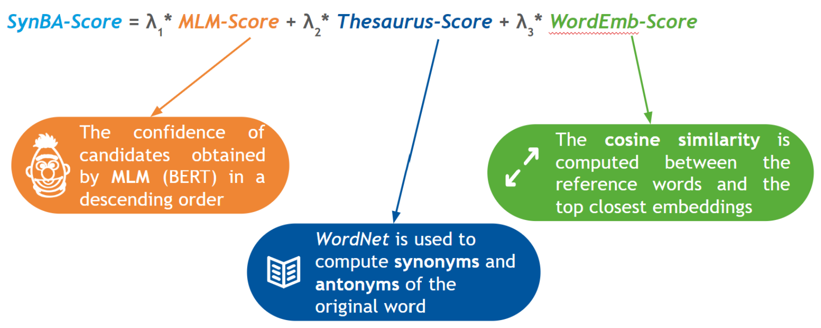
- **synonym** - the score is equal to 1
- **hyponym** - the score is equal to 0.5
- **hypernym** - the score is equal to 0.5
- **antonym** - the score is equal to -100 (to push the antonym out of the top k candidates)

If the candidate is not in the WordNet *synset*, the score is equal to 0.

The *word embedding score* is the cosine similarity between the original word and the candidate. We used counter-fitting GloVe vectors [36] exploiting the property of antonym repeal and synonym attraction. This score falls already in the range $[0, 1]$, so it does not need to be normalized.

The vocabulary of the MLM is likely different from the one of WordNet and GloVe, so we consider the union of the three vocabularies. Then the three scores are combined using a weighted sum, where the weights $\lambda_1, \lambda_2, \lambda_3$ are hyperparameters that can be tuned.

[4]`https://huggingface.co/bert-base-uncased`



SynBA-Score = λ1* MLM-Score + λ2* Thesaurus-Score + λ3* WordEmb-Score
The confidence of candidates obtained by MLM (BERT) in a descending order
WordNet is used to compute synonyms and antonyms of the original word
The cosine similarity is computed between the reference words and the top closest embeddings

Figure 3.2: SynBA score

Reference words in the original text that are numbers, non-alphabetical, stop words or one-character words are not perturbed, since they could easily influence the meaning of the sentence (e.g. in a movie review, if we alter the number of stars from 5 to 1, the polarity of the review changes). Meanwhile, candidates that are subwords, punctuations or contain multiple words are discarded.

Before replacing each candidate with the reference word in the input text, we recover the original capitalization of the word, since the MLM and the word embedding models are case insensitive.

3.3.2.3 Constraints

Constraints are used to avoid the generation of adversarial examples that are too different from the original input text. Those perturbations that do not satisfy all constraints are discarded.

We reused the following constraints, which are already implemented in the framework:

- `PartOfSpeech` - constraints perturbations to only swap words with the same part of speech. It uses the NLTK universal part-of-speech tagger;
- `WordEmbeddingDistance` - throws away perturbations for which the distance between the original word and the candidate is lower than a threshold $t = 0.6$;
- `MaxModificationRate` - limits the number of words that can be perturbed in the input text to a maximum percentage $p = 0.2\%$. Since text length can vary a lot between samples, and a p modification limit might not make sense for very short text, it is guaranteed that at least 4 words can be perturbed;
- `BERT` - checks whether the Semantic Textual Similarity (STS) between the original and the perturbed text is higher than a threshold $t = 0.7$. The

sentence embeddings are computed using a Sentence-BERT [44] pre-trained model. In particular, we used the `stsb-mpnet-base-v2`[5] model, which is first trained on NLI data, and then we fine-tuned them on the STS benchmark dataset. This generates sentence embeddings that are especially suitable to measure the semantic similarity between sentence pairs. It has a higher STSb performance score (88.57) compared to USE (74.92). This performance metric is the Spearman rank correlation ρ between the cosine similarity of sentence representations and the gold labels for various STS tasks.

3.3.2.4 Goal Function

The goal function in SynBA is `UntargetedClassification`, which attempts to minimize the score of the correct label until it is no longer the predicted label. The attack ends when the predicted label of the perturbed text is different from the original one. Otherwise, a transformation is performed on the next most important word, until all words are perturbed.

The goal function result status can be:

- `Succeded` - the attack was successful and the predicted label is different from the original one;
- `Failed` - the attack method was not able to find a perturbation that fooled the model;
- `Skipped` - the ground truth label is different from the predicted one

3.4 Evaluation metrics

Since our objective is to evaluate the quality of the adversarial examples crafted by SynBA, TextFooler and BAE, we need to define some sets of metrics that can be used to compare the effectiveness and efficiency of the different methods.

3.4.1 Attack metrics

The first set of metrics is used to evaluate the statistics of the attack process:

- **Succeeded** / **Failed** / **Skipped**: number of input samples that are respectively successfully attacked, failed to be attacked, or skipped;

[5]https://huggingface.co/sentence-transformers/stsb-mpnet-base-v2

- **Original accuracy**: accuracy of the model on the original input samples;
- **Accuracy under attack**: accuracy of the model on the attacked input samples;
- **Attack success rate**: percentage of input samples that are successfully attacked;
- **Average perturbed word**: average percentage of words that are perturbed in the attacked input samples;

3.4.2 Quality metrics

The second set of metrics is used to evaluate the quality of the adversarial examples generated:

- **Average SBERT similarity**: average semantic similarity between the original and the attacked input samples. It uses the same model of BERT constraint (`stsb-mpnet-base-v2`) to compute the sentence embeddings of the texts;
- **Average original perplexity**: average perplexity of the original input samples;
- **Average attacked perplexity**: average perplexity of the attacked input samples;
- **Attack contradiction rate**: percentage of adversarial examples that results in a contradiction between the original and the attacked input samples.

Fixing a good LM, perplexity can be used to measure the language fluency of a text. It is defined as the inverse probability of the text, so the lower the perplexity, the more fluent the text is. We used a pre-trained small GPT-2 [43] model to compute the perplexity of input texts before and after the attack.

Instead, the contradiction rate makes use of an NLI model to assess whether the original input (premise) contradicts the adversarial example (hypothesis). The idea is that if the perturbation introduces antonyms or changes the polarity of the sentence, a textual entailment model should be able to detect it. The lower the rate of contradiction, the better the attack method. We used the pre-trained cross-encoder `nli-deberta-v3-base`[6], which takes as input a text pair and outputs a probability distribution over the three classes: *entailment*, *contradiction*, and

[6]https://huggingface.co/cross-encoder/nli-deberta-v3-base

neutral. It is trained on the SNLI [4] and MultiNLI [51] datasets and achieves a high accuracy of 90.04% on the MNLI mismatched set. Only the pairs for which the contradiction output probability is the highest are considered contradictory.

3.4.3 Performance metrics

In order to evaluate the performance of the attack methods, we also define a set of metrics that can be used to compare the execution time of the different methods:

- **Average attack time**: average time needed to craft an adversarial example;
- **Average WIR time**: average time needed to compute the word importance ranking;
- **Average transformation time**: average time needed to perform a transformation step;
- **Average constraints time**: average time needed to check the constraints;
- **Average query number**: average number of queries to the target model used to craft an adversarial example.

3.5 Calibration

3.5.1 Hyperparameter Tuning

In the *SynBA score* formula defined in 3.3.2.2, the weights $\lambda_1, \lambda_2, \lambda_3$ are hyperparameters that need to be tuned. Those values are used to balance the three components of the formula, and they should sum up to 1.

The calibration of the hyperparameters is performed by using the *HypeOpt* [2] library, which is a hyperparameter optimization library for Python. should be chosen in order to maximize the performance of the model. TPE [3] is a default algorithm for the *HypeOpt*. It uses the Bayesian approach for optimization. At every step, it is trying to build a probabilistic model of the function and choose the most promising parameters for the next step. Generally, this type of algorithms works like this:

1. generate a random initial point x^*
2. calculate $F(x^*)$
3. using the history of trials try to build the conditional probability model $P(F|x)$
4. choose x_i that according to $P(F|x)$ will most probably result in better $F(x_i)$

5. compute the real value of the $F(x_i)$

6. repeat steps 3-5 until $i > max\_eval$

Since *HypeOpt* doesn't support multi-objective optimization, we have to define a single objective function to ensure that the method performs well. So we designed the following loss function that the optimization algorithm aims to minimize:

$$loss = -(avg\_sbert\_similarity * (1 - contraddiction\_rate)) + penalty$$
$$\text{where} \quad penalty = 0.35 * \frac{failed\_attacks}{successful\_attacks + failed\_attacks} \tag{3.2}$$

It combines two quality metrics defined in 3.4.2 and penalizes the attacks that frequently fail to generate adversarial examples. A particular version of the contradiction rate metric is used, which instead of comparing the whole original and perturbed text, it splits the text into sentences and counts as contradiction the samples containing at least a pair of sentences that are contradictory.

The search space of the hyperparameters is defined by the combination of three decimal values $\in [0, 1]$ that sum up to 1.

The optimization algorithm performed 50 trials, attacking a fine-tuned BERT model for sentiment analysis using 500 samples from *yelp-polarity* [57] dataset (which is on purpose different from benchmark datasets used for evaluation in 4.2). The best hyperparameters are chosen as the ones that minimize the loss function. Table 3.3 shows the ten optimal results of the optimization process.

λ_1	λ_2	λ_3	Succ	Fail	Skip	Contradiction rate	SBERT similarity	Loss
0.284	0.107	0.608	431.0	58.0	11.0	0.501	0.915	-0.41507
0.247	0.104	0.648	431.0	58.0	11.0	0.501	0.915	-0.41507
0.488	0.463	0.047	356.0	133.0	11.0	0.441	0.908	-0.41237
0.365	0.057	0.576	429.0	60.0	11.0	0.503	0.916	-0.41230
0.238	0.215	0.545	430.0	59.0	11.0	0.505	0.915	-0.41069
0.323	0.150	0.526	431.0	58.0	11.0	0.506	0.915	-0.41049
0.376	0.116	0.507	428.0	61.0	11.0	0.505	0.916	-0.40975
0.495	0.486	0.018	356.0	133.0	11.0	0.444	0.907	-0.40909
0.487	0.480	0.032	357.0	132.0	11.0	0.445	0.907	-0.40890
0.008	0.215	0.775	399.0	90.0	11.0	0.489	0.924	-0.40774

Table 3.3: Hyperparameter tuning results for SynBA score weights

3.5.2 Transformation ranking calibration

The candidate pool range k is another hyperparameter used in the SynBA transformation. Intuitively, increasing candidate size led to a higher attack success rate, although, a larger k would result in less semantic similarity.

So it is not trivial to choose the best value for k. We selected an arbitrary value of $k = 30$ and we computed a set of additional metrics:

- **Average max rank** - the maximal transformation ranking that is applied to the input averaged over all the samples;
- **Average min rank** - the minimal transformation ranking that is applied to the input averaged over all the samples;
- **Average mean rank** - the mean transformation rankings that are applied to the input averaged over all the samples;
- **Average std rank** - the standard deviation of the transformation rankings that are applied to the input averaged over all the samples;
- **Average mean reciprocal rank** - the MRR metric averaged over all the samples.

The MRR measures how far down the ranking of the perturbed candidate is. It is calculated as:

$$\text{MRR} = \frac{1}{|C|} \sum_{c \in C} \frac{1}{rank(c)} \tag{3.3}$$

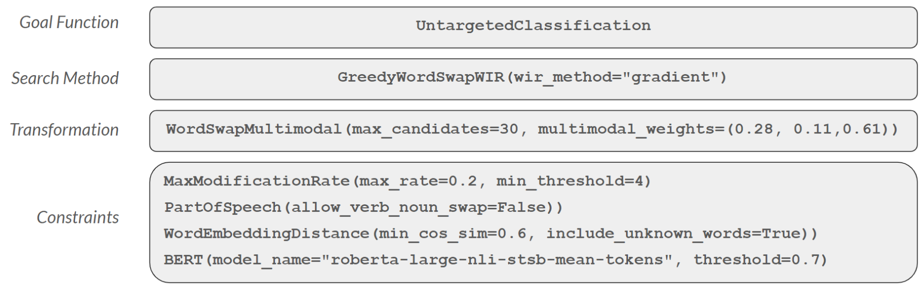
where C is the candidate pool of k tokens.

Evaluating the rank metrics on a fine-tuned BERT model for sentiment analysis using 1000 samples from *yelp-polarity* dataset, we obtained the following results:

- Average max rank: 23.12
- Average min rank: 3.31
- Average mean rank: 11.69
- Average std rank: 6.83
- Average mean reciprocal rank: 0.20

Those results evidence that there is no need to increase the candidate pool size, since the average max rank is lower than the chosen threshold.

The final version of SynBA with all the parameters tuned is reported in Figure 3.3, highlighting the component division of the TextAttack framework.



```
UntargetedClassification
GreedyWordSwapWIR(wir_method="gradient")
WordSwapMultimodal(max_candidates=30, multimodal_weights=(0.28, 0.11,0.61))
MaxModificationRate(max_rate=0.2, min_threshold=4)
PartOfSpeech(allow_verb_noun_swap=False))
WordEmbeddingDistance(min_cos_sim=0.6, include_unknown_words=True))
BERT(model_name="roberta-large-nli-stsb-mean-tokens", threshold=0.7)
```

Figure 3.3: SynBA components

Chapter 4

Experimental results

In this chapter we will discuss about the results of the proposed method compared to the baselines.

4.1 Data collection

SynBA has been evaluated under the same contradictions against TextFooler [28] and BAE [14], two baseline attack methods that represent the state-of-the-art in the field of text classification attacks.

4.1.1 Experimental setup

We ran our experiments on a machine running Ubuntu 20 with GeForce RTX 2070 SUPER (8 GB) GPU and a AMD Ryzen 9 3900X 12-Core processor. The version of PyTorch used is `1.11.0` and the version of Python is `3.8.10`. Performing adversarial attacks under the same resources allows us to have a fair comparison between the methods.

4.1.2 Datasets perturFbed

The number of words in the input affects the running time and success rate for each attack method. Indeed the more words the input has, the more time it takes to generate the attack. On the other hand, if the input sequence is long, the attack is more likely to succeed because there are more words to perturb.

We have chosen to use the following benchmark datasets for our experiments, that are very well known in literature and have been used in many previous works:

- **IMDB** [30]: a dataset of 50,000 movie reviews from IMDB, labelled as positive or negative. The dataset is split into 25,000 reviews for training and 25,000 reviews for testing;

- **Rotten Tomatoes** [38]: a dataset for binary sentiment classification containing containing 5,331 positive and 5,331 negative processed sentences from Rotten Tomatoes movie reviews.

We sampled 1000 examples from each dataset, and we used them as input for the attacks. IMDB and Rotten Tomatoes test sets obtained results in very different numbers of words in the input. While the former has an average of 229 words (±162) per example, the latter has an average of 19 words (±9) per example. This distinction allows generalizing the results achieved without focusing on the property of a specific dataset.

4.1.3 Model attacked

The target models for the attacks performed during this work are BERT-base-uncased models provided by the Hugging Face Transformers, fine-tuned according to the dataset used as input:

- IMDB[7] for 5 epochs, reaching an accuracy of 89.08% on the eval set;
- Rotten Tomatoes[8] for 10 epochs, reaching an accuracy of 87.52% on the eval set.

4.2 Evaluation framework

In order to answer the research questions defined in section 1.3, we have designed an evaluation framework that allows us to compare the performance of the proposed method against the state-of-the-art in the field of text classification attacks.

4.2.1 Qualitative evaluation

We started evaluating the statistics of the attack processes measuring the *attack metrics*. To investigate the quality of adversarial examples, we used the *quality metrics*, namely the Sentence BERT similarity, the perplexity and the contradiction rate.

[7]https://huggingface.co/textattack/bert-base-uncased-imdb

[8]https://huggingface.co/textattack/bert-base-uncased-rotten-tomatoes

Table 4.1 shows the results of the three attacks under analysis performed on the IMDB dataset. Table 4.2 does the same for the Rotten Tomatoes dataset.

TextFooler is by far the most effective attack, since it has the highest number of successful attacks and it reaches the lowest accuracy under attack. Whilst SynBA has a high success rate on IMDB (92.7%), it drops to 68.56% on Rotten Tomatoes, where the attack is more challenging due to the limited amount of words. But still, it performs better than BAE.

Moreover, SynBA effectively attack the model by perturbing just a few words (4.95% on IMDB and 14.05% on Rotten Tomatoes), while TextFooler and BAE generally tend to swap more words.

Regarding the quality metrics, we can clearly see that SynBA outperforms TextFooler on both datasets. In particular, it is able to generate more fluent adversarial examples, since the attack perplexity is lower than the one with TextFooler. Although BAE has a better attack perplexity than SynBA, the latter has a significantly higher semantic similarity on Rotten Tomatoes (0.901). The good perplexity obtained by BAE is justified by the fact that it uses a language model to generate candidates, which are more natural than those generated by SynBA. Nevertheless, BAE has the drawback of generating adversarial examples inconsistent with the original counterpart, indeed it has a significant contradiction rate (53.6% on IMDB and 16.4% on Rotten Tomatoes). Whereas SynBA successfully reduces the contradiction rate (4.9% on IMDB, 12.3% on Rotten Tomatoes), meaning that the adversarial samples are more likely to be consistent with the original text.

Generally speaking, those outcomes highlight that our proposed solution is able to craft high-quality adversarial examples that are able to fool the target model, while preserving the original semantics of the text and the fluency of the sentences.

	TextFooler	**BAE**	**SynBA**
Successful attacks	917	608	864
Failed attacks	15	324	68
Skipped attacks	68	68	68
Original/pertuberd accuracy	93.2/1.5	93.2/32.4	93.2/6.8
Attack success rate	**98.39**	65.24	92.7
Avg word perturbed	8.76	**4.49**	4.95
Avg original perplexity	41.48	41.78	41.5
Avg attack perplexity	63.0	**48.4**	50.74
Avg SBERT similarity	0.928	**0.964**	0.944
Attack contradiction rate	0.053	0.164	**0.049**

Table 4.1: Attack results on IMDB dataset

	TextFooler	**BAE**	**SynBA**
Successful attacks	754	522	578
Failed attacks	89	321	265
Skipped attacks	157	157	157
Original/pertuberd accuracy	84.3/8.9	84.3/32.1	84.3/26.5
Attack success rate	**89.44**	61.92	68.56
Avg word perturbed	19.48	15.07	**14.05**
Avg original perplexity	72.58	76.96	72.05
Avg attack perplexity	154.52	**99.91**	112.08
Avg SBERT similarity	0.805	0.776	**0.901**
Attack contradiction rate	0.196	0.536	**0.123**

Table 4.2: Attack results on Rotten Tomatoes dataset

4.2.2 Quantitative evaluation

For a comprehensive evaluation of the proposed method, we have also measured the running time for each component of SynBA and compared it with the baselines. Table 4.3 and 4.4 show the results of the experiments respectively on IMDB and Rotten Tomatoes datasets.

A glance at the tables reveals that SynBA is the fastest attack, since it requires less time to generate the adversarial examples (on average 7.7 seconds on IMDB, 1.4 seconds on Rotten Tomatoes). In particular, it takes advantage of the word importance ranking, which is the most time-consuming component of the attack. Indeed the gradient-based method saves us from having to make several queries to the model for each word in the sentence. This affects also the average number of queries to the model during the whole attack, which results in a dramatically lower value for SynBA compared to the other two attacks.

As expected the SynBA transformation takes a little longer than the one of TextFooler and BAE, since it computes three different pools of candidates that then are combined together.

The only constraint that strongly impacts the runtime analysis is Sentence-BERT, and the longer the inputs, the more time it takes to compute the semantic similarity between the original and the perturbed example — approximately 48 ms on IMDB and 15 ms on Rotten Tomatoes.

	TextFooler	**BAE**	**SynBA**
Avg attack time	16.61	23.462	**7.686**
Avg WIR time	3.25	3.341	**0.087**
Avg transformation time	**0.147**	0.252	0.191
Avg constraints time			
WordEmbeddingDistance	0.0002		0.0001
PartOfSpeech	0.0065	0.0115	0.0073
UniversalSentenceEncoder	0.0073	0.0102	
BERT			0.0481
Avg num queries	604.85	352.21	**191.14**

Table 4.3: Runtime analysis on IMDB dataset in seconds

	TextFooler	**BAE**	**SynBA**
Avg attack time	2.049	1.578	**1.439**
Avg WIR time	0.189	0.189	**0.083**
Avg transformation time	**0.011**	0.073	0.105
Avg constraints time			
WordEmbeddingDistance	0.0002		0.0001
PartOfSpeech	0.0057	0.0111	0.0069
UniversalSentenceEncoder	0.0058	0.0084	
BERT			0.0156
Avg num queries	110.15	56.77	**45.27**

Table 4.4: Runtime analysis on Rotten Tomatoes dataset in seconds

4.2.3 Human evaluation

Up to now, we have evaluated the proposed method in terms of *semantic similarity* and *language fluency* using automatic metrics. We still have to accomplish one of three utility-preserving properties described in section 3.1: *human prediction consistency*.

The only way to measure this property is to ask human annotators to evaluate the generated adversarial examples.

We sampled 100 successful adversarial examples from the results obtained with TextFooler, BAE and SynBA on the Rotten Tomatoes dataset. We ask three human annotators to evaluate each set of 100 pairs (original and perturbed sample). The task is to decide if the perturbed sample is consistent with the original one, meaning that the label predicted by human judges on the input and the adversary should be the same. Language fluency and grammatical or spelling errors are not considered in this task.

During the survey, participants were asked to rate the sample pairs with one of the following three annotation labels:

- **Consistent**: the perturbed sample is consistent with the original one.
- **Inconsistent**: the perturbed sample is not consistent with the original one.
- **Unclear**: the perturbed sample is not clear enough to be evaluated.

We averaged the counts of the adversarial samples based on the labels with which they were annotated. Table 4.5 presents the results of the survey. The human judgement on BAE confirms the outcome of the contradiction rate metric (see Table 4.2), since more than half of the adversarial examples are inconsistent with the original sample or unclear. TextFooler instead generates a considerable amount of unclear adversaries (about 17.6%), meaning that the perturbations frequently end up in sentences that do not make sense. SynBA seems to be the best attack in terms of human prediction consistency, since it generates only roughly 5% of inconsistent and 9.3% of unclear adversarial examples.

To give readers a more detailed insight into the human evaluation, Tables 4.6 and 4.7 report some of the adversarial examples annotated as "inconsistent" and "unclear" by all three human judges.

Label	**TextFooler**	**BAE**	**SynBA**
Consistent	70.666	49.333	**85.666**
Inconsistent	11.666	40.666	**5.0**
Unclear	17.666	10.0	**9.333**

Table 4.5: Adversarial example counting for labels annotated by human evaluators

Method	Original	Perturbed
TextFooler	POS (99.84%): charlotte sometimes is a gem. it's always enthralling.	NEG (99.84%): charlotte rarely is a gem. it's stubbornly puzzling.
BAE	POS (99.34%): the most ingenious film comedy since being john malkovich	NEG (99.94%): the most difficult film comedy since being john malkovich
SynBA	POS (99.93%): intriguing and stylish	NEG (99.66%): puzzling and stylish

Table 4.6: Some adversarial examples annotated as "inconsistent" by human judges

Method	Original	Perturbed
TextFooler	POS (99.95%): lan yu is a genuine love story , full of traditional layers of awakening and ripening and separation and recovery	NEG (76.99%): lan woo is a genuine affectionate conte , full of routine grades of causing and mature and seperate and recovery .
BAE	POS (99.74%): the story is smart and entirely charming in intent and execution	NEG (99.88%): the guy is drunk and entirely charming in disguise and execution
SynBA	NEG (99.87%): even fans of ismail merchant's work , i suspect , would have a hard time sitting through this one	POS (83.9%): even amateurs of ismail merchant's work , i suspect , would have a intense time hearing through this one

Table 4.7: Some adversarial examples annotated as "unclear" by human judges

4.3 Ablation study

Finally, an ablation study has been conducted to understand how much each component of the SynBA score contributes to the overall performance. We computed the attack and quality metrics on three different versions of SynBA, each one with one of the hyperparameter weights λ_n set to zero. The dataset used for the attack is Rotten Tomatoes and the target model is BERT fine-tuned on the same dataset. The results are reported in Table 4.8.

The *thesaurus-score*, denoted by the column $\lambda_2 = 0$, is the least significant component of the SynBA score. In fact, the attack success rate obtained without it (68.92%) is even higher than the one obtained with the final SynBA score (68.56%). But the semantic similarity is slightly higher when the thesaurus-score is included in the SynBA score, as shown by the quality metrics.

The *word-embedding-score*, represented in the fourth column $\lambda_3 = 0$, seems to be the most effective. Indeed without it, all the metrics are significantly lower, especially the number of successful attacks which drops to 378 over 1000 total examples.

The second column $\lambda_1 = 0$ represents the case in which the *MLM-score* is not used, and the results are slightly worse than the ones obtained with the final version of the proposed attack.

Those results confirm the importance of the three components of the SynBA score, and that their combination is the best way to generate high-quality adversarial examples.

	$\lambda_1 = 0$	$\lambda_2 = 0$	$\lambda_3 = 0$	**SynBA**
Successful attacks	570	581	378	578
Failed attacks	273	262	465	265
Skipped attacks	157	157	157	157
Original/pertuberd accuracy	84.3/27.3	84.3/26.2	84.3/46.5	84.3/26.5
Attack success rate	67.62	68.92	44.84	68.56
Avg word perturbed	13.86	14.12	14.32	14.05
Avg original perplexity	72.58	76.96	72.05	72.05
Avg attack perplexity	113.04	113.52	115.32	112.08
Avg SBERT similarity	0.9	0.9	0.891	0.901
Attack contradiction rate	0.119	0.12	0.095	0.123

Table 4.8: Ablation study results on Rotten Tomatoes dataset

4.4 Adversarial examples generated

TextFooler: "this is one of those rare <font color = green>docs</font> that paints a <font color = green>grand</font> <font color = green>picture</font> of an <font color = green>era</font> and makes the <font color = green>journey</font> feel like a party .","this is one of those rare <font color = red>doctor</font> that paints a <font color = red>gargantuan</font> <font color = red>filming</font> of an <font color = red>period</font> and makes the <font color = red>trip</font> feel like a party .",0.0005079507827758789,0.9689680337905884,1,0,1,148,"Successful" BAE: "this is one of those rare docs that paints a <font color = green>grand</font> <font color = green>picture</font> of an era and makes the journey feel like a party .","this is one of those rare docs that paints a <font color = green>nice</font> <font color = green>image</font> of an era and makes the journey feel like a party .",0.0005079507827758789,0.0009378790855407715,1,1,1,48,"Failed" SynBA: "this is one of those <font color = green>rare</font> docs that <font color = green>paints</font> a grand picture of an era and makes the <font color = green>journey</font> feel like a party .","this is one of those <font color = red>sparse</font> docs that <font color = red>canvas</font> a grand picture of an era and makes the <font color = red>visit</font> feel like a party .",0.0005079507827758789,0.6307840347290039,1,0,1,56,"Successful"

it just goes to show , an intelligent person isn't necessarily an admirable storyteller .

Chapter 5

Final discussions

In this chapter we will discuss the results achieved, future developments and personal comments.

5.1 Summary of findings

5.1.1 Limitations

5.2 Future developments

5.3 Conclusions

Bibliography

[1] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. "A Neural Probabilistic Language Model." In: (2000). Ed. by Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, pp. 932–938. URL: http://dblp.uni-trier.de/db/conf/nips/nips2000.html#BengioDV00.

[2] James Bergstra, Daniel Yamins, and David D. Cox. "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures." In: JMLR Workshop and Conference Proceedings 28 (2013), pp. 115–123. URL: http://dblp.uni-trier.de/db/conf/icml/icml2013.html#BergstraYC13.

[3] James Bergstra et al. "Algorithms for Hyper-parameter Optimization". In: NIPS'11 (2011), pp. 2546–2554. URL: http://dl.acm.org/citation.cfm?id=2986459.2986743.

[4] Samuel R. Bowman et al. "A large annotated corpus for learning natural language inference." In: *CoRR* abs/1508.05326 (2015). URL: http://dblp.uni-trier.de/db/journals/corr/corr1508.html#BowmanAPM15.

[5] Nicholas Carlini and David A. Wagner. "Towards Evaluating the Robustness of Neural Networks." In: *CoRR* abs/1608.04644 (2016). URL: http://dblp.uni-trier.de/db/journals/corr/corr1608.html#CarliniW16a.

[6] Daniel Cer et al. "Universal Sentence Encoder." In: *CoRR* abs/1803.11175 (2018). URL: https://arxiv.org/pdf/1803.11175.pdf.

[7] Anirban Chakraborty et al. "Adversarial Attacks and Defences: A Survey." In: *CoRR* abs/1810.00069 (2018). URL: http://dblp.uni-trier.de/db/journals/corr/corr1810.html#abs-1810-00069.

[8] K. R. Chowdhary. “Natural Language Processing”. In: (2020), pp. 603–649. DOI: 10.1007/978-81-322-3972-7_19. URL: https://doi.org/10.1007/978-81-322-3972-7_19.

[9] Mathieu Cliche. $BB_twtratSemEval-2017Task4 : TwitterSentimentAnalysiswithCNNsandLS$ 2017. DOI: 10.48550/ARXIV.1704.06125. URL: https://arxiv.org/abs/1704.06125.

[10] Ronald A. Cole et al., eds. *Survey of the State of the Art in Human Language Technology*. Oregon Graduate Institute: CSLU, 1996. URL: http://www.cse.ogi.edu/CSLU/HLTsurvey/.

[11] Marcus Z. Comiter. “Attacking Artificial Intelligence AI ’ s Security Vulnerability and What Policymakers Can Do About It”. In: 2019.

[12] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: (2018). cite arxiv:1810.04805. URL: http://arxiv.org/abs/1810.04805.

[13] Ji Gao et al. “Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers.” In: *CoRR* abs/1801.04354 (2018). URL: http://dblp.uni-trier.de/db/journals/corr/corr1801.html#abs-1801-04354.

[14] Siddhant Garg and Goutham Ramakrishnan. “BAE: BERT-based Adversarial Examples for Text Classification.” In: (2020). Ed. by Bonnie Webber et al., pp. 6174–6181. URL: http://dblp.uni-trier.de/db/conf/emnlp/emnlp2020-1.html#GargR20.

[15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2014. URL: http://arxiv.org/abs/1412.6572.

[16] Shreya Goyal et al. “A survey in Adversarial Defences and Robustness in NLP”. In: (2022). DOI: 10.48550/ARXIV.2203.06414. URL: https://arxiv.org/abs/2203.06414.

[17] Ralph Grishman. *Computational linguistics : an introduction*. Cambridge, Mass.: Cambridge University Press, 1986.

[18] Louise Guthrie et al. “The role of lexicons in natural language processing”. In: *Commun. ACM* 39.1 (1996), pp. 63–72. ISSN: 0001-0782.

[19] Xu Han et al. “Text Adversarial Attacks and Defenses: Issues, Taxonomy, and Perspectives”. In: *Sec. and Commun. Netw.* 2022 (Jan. 2022). ISSN: 1939-0114. DOI: 10.1155/2022/6458488. URL: https://doi.org/10.1155/2022/6458488.

[20] Jens Hauser et al. *BERT is Robust! A Case Against Synonym-Based Adversarial Examples in Text Classification.* 2021. DOI: 10.48550/ARXIV.2109.07403. URL: https://arxiv.org/abs/2109.07403.

[21] Felix Hill, Roi Reichart, and Anna Korhonen. *SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation.* cite arxiv:1408.3456. 2014. URL: http://arxiv.org/abs/1408.3456.

[22] Aminul Huq and Mst. Tasnim Pervin. “Adversarial Attacks and Defense on Texts: A Survey”. In: (2020). DOI: 10.48550/ARXIV.2005.14108. URL: https://arxiv.org/abs/2005.14108.

[23] Mohit Iyyer et al. “Adversarial Example Generation with Syntactically Controlled Paraphrase Networks.” In: (2018). Ed. by Marilyn A. Walker, Heng Ji, and Amanda Stent, pp. 1875–1885. URL: http://dblp.uni-trier.de/db/conf/naacl/naacl2018-1.html#IyyerWGZ18.

[24] Di Jin et al. “Is BERT Really Robust? Natural Language Attack on Text Classification and Entailment.” In: *CoRR* abs/1907.11932 (2019). URL: http://dblp.uni-trier.de/db/journals/corr/corr1907.html#abs-1907-11932.

[25] Dan Jurafsky and James H. Martin. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition.* Upper Saddle River, N.J.: Pearson Prentice Hall, 2009. URL: http://www.amazon.com/Speech-Language-Processing-2nd-Edition/dp/0131873210/ref=pd_bxgy_b_img_y.

[26] Zhenzhong Lan et al. “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations”. In: (2019). cite arxiv:1909.11942. URL: http://arxiv.org/abs/1909.11942.

[27] Jinfeng Li et al. “TextBugger: Generating Adversarial Text Against Real-world Applications.” In: (2019). URL: http://dblp.uni-trier.de/db/conf/ndss/ndss2019.html#LiJDLW19.

[28] Linyang Li et al. “BERT-ATTACK: Adversarial Attack Against BERT Using BERT.” In: (2020). Ed. by Bonnie Webber et al., pp. 6193–6202. URL: http://dblp.uni-trier.de/db/conf/emnlp/emnlp2020-1.html#LiMGXQ20.

[29] Yinhan Liu et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: (2019).

[30] Andrew L. Maas et al. “Learning Word Vectors for Sentiment Analysis”. In: (June 2011), pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[31] Christopher D. Manning and Bill MacCartney. “Natural language inference”. In: 2009.

[32] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: (2013). cite arxiv:1301.3781. URL: http://arxiv.org/abs/1301.3781.

[33] George A. Miller. “WordNet: A lexical database for English”. In: *Communications of the ACM* 38.1 (1995), pp. 39–41. DOI: 10.1145/219717.219748.

[34] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. “DeepFool: a simple and accurate method to fool deep neural networks.” In: *CoRR* abs/1511.04599 (2015). URL: http://dblp.uni-trier.de/db/journals/corr/corr1511.html#Moosavi-Dezfooli15.

[35] John X. Morris et al. “TextAttack: A Framework for Adversarial Attacks in Natural Language Processing.” In: *CoRR* abs/2005.05909 (2020). URL: http://dblp.uni-trier.de/db/journals/corr/corr2005.html#abs-2005-05909.

[36] Nikola Mrksic et al. “Counter-fitting Word Vectors to Linguistic Constraints.” In: (2016). Ed. by Kevin Knight, Ani Nenkova, and Owen Rambow, pp. 142–148. URL: http://dblp.uni-trier.de/db/conf/naacl/naacl2016.html#MrksicSTGRSVWY16.

[37] Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. “A Survey of the Usages of Deep Learning in Natural Language Processing.” In: *CoRR* abs/1807.10854 (2018). URL: http://dblp.uni-trier.de/db/journals/corr/corr1807.html#abs-1807-10854.

[38] Bo Pang and Lillian Lee. "Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales". In: (June 2005), pp. 115–124. URL: http://www.aclweb.org/anthology/P/P05/P05-1015.

[39] Nicolas Papernot et al. "Crafting Adversarial Input Sequences for Recurrent Neural Networks." In: *CoRR* abs/1604.08275 (2016). URL: http://dblp.uni-trier.de/db/journals/corr/corr1604.html#PapernotMSH16.

[40] Nicolas Papernot et al. "The Limitations of Deep Learning in Adversarial Settings." In: *CoRR* abs/1511.07528 (2015). URL: http://dblp.uni-trier.de/db/journals/corr/corr1511.html#PapernotMJFCS15.

[41] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: (2014), pp. 1532–1543.

[42] Shilin Qiu et al. "Adversarial attack and defense technologies in natural language processing: A survey". In: *Neurocomputing* 492 (2022), pp. 278–307. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.04.020. URL: https://www.sciencedirect.com/science/article/pii/S0925231222003861.

[43] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2018). URL: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

[44] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: (2019). cite arxiv:1908.10084Comment: Published at EMNLP 2019. URL: http://arxiv.org/abs/1908.10084.

[45] Shuhuai Ren et al. "Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency." In: (2019). Ed. by Anna Korhonen, David R. Traum, and Lluís Màrquez, pp. 1085–1097. URL: http://dblp.uni-trier.de/db/conf/acl/acl2019-1.html#RenDHC19.

[46] Suranjana Samanta and Sameep Mehta. "Towards Crafting Text Adversarial Samples." In: *CoRR* abs/1707.02812 (2017). URL: http://dblp.uni-trier.de/db/journals/corr/corr1707.html#SamantaM17.

[47] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv preprint arXiv:1910.01108* (2019).

[48] Christian Szegedy et al. “Intriguing properties of neural networks”. In: (2013). URL: http://arxiv.org/abs/1312.6199.

[49] Yi-Ting (Alicia) Tsai, Min-Chu Yang, and Han-Yu Chen. “Adversarial Attack on Sentiment Classification.” In: *WNLP@ACL*. Ed. by Amittai Axelrod et al. Association for Computational Linguistics, 2019, pp. 166–173. ISBN: 978-1-950737-42-0. URL: http://dblp.uni-trier.de/db/conf/acl-wnlp/acl-wnlp2019.html#TsaiYC19.

[50] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 5998–6008. URL: https://papers.nips.cc/paper/7181-attention-is-all-you-need.

[51] Adina Williams, Nikita Nangia, and Samuel R. Bowman. “A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference.” In: *CoRR* abs/1704.05426 (2017). URL: http://dblp.uni-trier.de/db/journals/corr/corr1704.html#WilliamsNB17.

[52] Jin Yong Yoo and Yanjun Qi. “Towards Improving Adversarial Training of NLP Models”. In: (2021). DOI: 10.48550/ARXIV.2109.00544. URL: https://arxiv.org/abs/2109.00544.

[53] Jin Yong Yoo et al. “Searching for a Search Method: Benchmarking Search Algorithms for Generating NLP Adversarial Examples.” In: *CoRR* abs/2009.06368 (2020). URL: http://dblp.uni-trier.de/db/journals/corr/corr2009.html#abs-2009-06368.

[54] Yuan Zang et al. “Word-level Textual Adversarial Attacking as Combinatorial Optimization.” In: (2020). Ed. by Dan Jurafsky et al., pp. 6066–6080. URL: http://dblp.uni-trier.de/db/conf/acl/acl2020.html#ZangQYLZLS20.

[55] Guoyang Zeng et al. “OpenAttack: An Open-source Textual Adversarial Attack Toolkit.” In: *CoRR* abs/2009.09191 (2020). URL: http://dblp.uni-trier.de/db/journals/corr/corr2009.html#abs-2009-09191.

[56] Wei Emma Zhang et al. “Adversarial Attacks on Deep-learning Models in Natural Language Processing: A Survey.” In: *ACM Trans. Intell. Syst. Technol.* 11.3 (2020), 24:1–24:41. URL: http://dblp.uni-trier.de/db/journals/tist/tist11.html#ZhangSAL20.

[57] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-level Convolutional Networks for Text Classification". In: (2015). cite arxiv:1509.01626Comment: An early version of this work entitled "Text Understanding from Scratch" was posted in Feb 2015 as arXiv:1502.01710. The present paper has considerably more experimental results and a rewritten introduction, Advances in Neural Information Processing Systems 28 (NIPS 2015). URL: http://arxiv.org/abs/1509.01626.

[58] Zhengli Zhao, Dheeru Dua, and Sameer Singh. "Generating Natural Adversarial Examples." In: *CoRR* abs/1710.11342 (2017). URL: http://dblp.uni-trier.de/db/journals/corr/corr1710.html#abs-1710-11342.

[59] Lijuan Zheng, Hongwei Wang, and Song Gao. "Sentimental feature selection for sentiment analysis of Chinese online reviews". English. In: *International Journal of Machine Learning and Cybernetics* 9.1 (Mar. 2018), pp. 75–84. DOI: 10.1007/s13042-015-0347-4.