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

Dedica

“ 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

"One day the AIs are going to look back on us the same way we look at fossil skeletons on the plains of Africa. An upright ape living in dust with crude language and tools, all set for extinction."

- Ex machina

Abstract

"Happiness can be found even in the darkest of times, when one only remembers to turn on the light."

– Dumbledore

Thanks

First, I would like to express my deepest gratitude to Professor Luigi Di Stefano and Luca De Luigi for the guidance and support during the internship

Second, I would like to thank my parents for their moral support and for their patience.

Third, I would like to thank my girlfriend and friends for being patient with me and to put up with my complainings.

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

To understand why AI systems are vulnerable to the same weakness, we must briefly examine how AI algorithms, or more specifically the machine learning techniques they employ, "learn." Just like the reconnaissance officers, the machine learning algorithms powering AI systems "learn" by extracting patterns from data.

1.2 Problem statement

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 showed that robustness and generalization of ML models can be improved by crafting high-quality adversaries and including them in the training data.

1.3 Research question

1.4 Solution

Our main contributions are summarized as follows: • We propose a simple but strong baseline, TEXTFOOLER, to quickly generate high-profile utility-preserving adversarial examples that force the target models to make wrong predictions under the black-box setting. • We evaluate TEXTFOOLER on three state-of-the-art deep learning models over five popular text classification tasks and two textual entailment tasks, and it achieved the state-of-the-art attack success rate and perturbation rate. Algorithm 1 Adversarial Attack by TEXTFOOLER • We propose a comprehensive four-way automatic and three-way human evaluation of language adversarial attacks to evaluate the effectiveness, efficiency, and utilitypreserving properties of our system. • We open-source the code, pre-trained target models, and test samples for the convenience of future benchmarking

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 stage and the project;

Third chapter presents the datasets used during for the training and the testing of the model;

Fourth chapter presents the experiments did during to develop the system;

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.

The linguistics computational has two branches—computational linguistics and theoretical linguistics. The computational linguistics has been concerned with developing algorithms for handling a useful range of natural language as input. While the 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—principals of grammar which apply to all natural languages [6].

Computational linguistics is concerned with the study of natural language analysis and language generation. Further, the 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 [12].

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 [4].

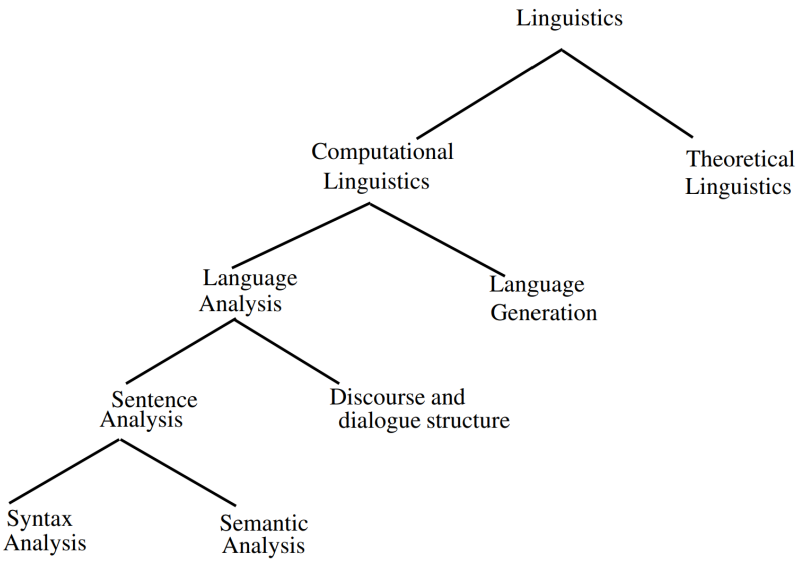


Figure 2.1: Components of NLP

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 [13].

A lexicon which provides an effective combination of traditional lexicographic information and modern computing is called WordNet [25]. 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 vector 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. Authors presented a neural model which obtains word representations as to the product while training language model. The popularity of word representation methods are due to two famous models, Word2Vec [24] and GloVe [30].

2.1.2.3 Contextual embeddings

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

Contextualised word embeddings are variable vector 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 by 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 pretrained 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 whichc 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 into the original text.

Masked Language Modelling is usually solved as 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 [8], which is a bidirectional encoder representation from a particular deep learning architecture called transformer [36]. It uses self-supervised training on the masked language modeling 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 on BERT. Instead of static masking, RoBERTa [22] 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 [34] reduces the size of $BERT_{BASE}$ by 40%, enhances the speed by 60% while retaining 97% of its capabilities. ALBERT [19] 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 modeling 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-ofthe-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 the rules. Most cases of classification in language processing are instead done via supervised machine learning, where an algorithm learn how to map from an observation to a correct output [18].

Many kinds of machine learning algorithms are used to build classifiers. Formerly, statistical and machine learning approaches, such as naïve Bayes, k-nearest neighbors, 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 [27].

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. [18]

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, and pathetic, and 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 [40]. Good results were obtained

using an ensemble, including both LSTMs and CNNs [5]. 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 [23], 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.

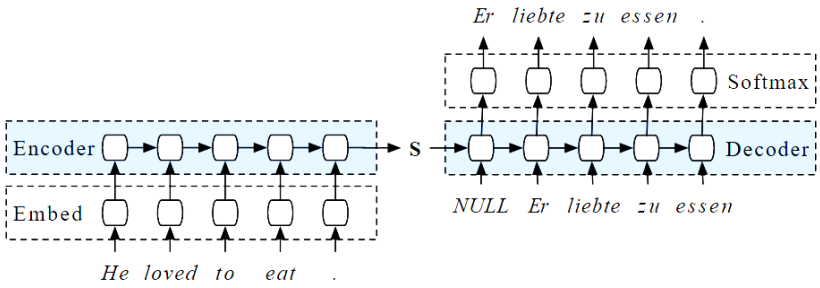


Figure 2.2: An example of a sequence-to-sequence model for machine translation.

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.

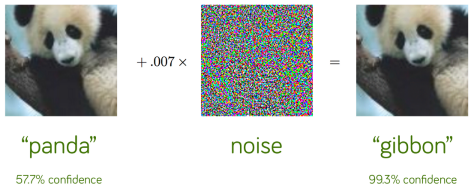
Unlike traditional cybersecurity attacks, these weaknesses are not due to mistakes made by programmers or users. They are just shortcomings of the current state-of-the-art methods. Put more bluntly, the algorithms that cause AI systems to work so well are imperfect, and their systematic limitations create opportunities for adversaries to attack. At least for the foreseeable future, this is just a fact of mathematical life [7].

Two main types of AI attacks can be defined according to the time at which the attack happens [3]:

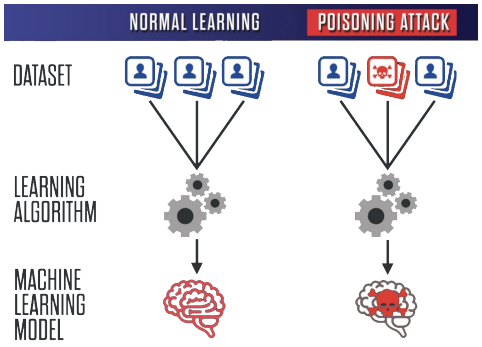
- **Adversarial Attacks (Evasion)**: this is the most common type of attack in the adversarial setting. The adversary tries to evade the system by adjusting malicious samples during testing phase. This setting does not assume any influence over the training data. In Figure 2.3a is depicted how adding an

imperceptible and carefully constructed noise to the input originally regognized 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 [35]



(b) Poisoning attack on training data [7]

Figure 2.3: Examples of Artificial Intelligence Attacks

In this thesis, we will focus on the first type of attack, the Adversarial Attacks. Over past few years, researchers [11, 35] 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. [35] 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 ($f(x') = y'$) [38].

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. [35] was based on L-BFGS, but despite the effectiveness of the method, it was computationally expensive and impractical. Goodfellow et al. [11] proposed a Fast Sign Gradient Method (FSGM) that popularized this research topic. It is a simplification of the L-BFGS method since it add 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 optimizaion methods (eg. JSMA [29], DeepFool [26], C&W [2]) or leveraging Generative Adversarial Network (GAN) to generate adversaries [39].

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 less. 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 defense technologies, the textual adversarial technology has gradually gained researchers' attention.

Papernot et al. [28] is 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

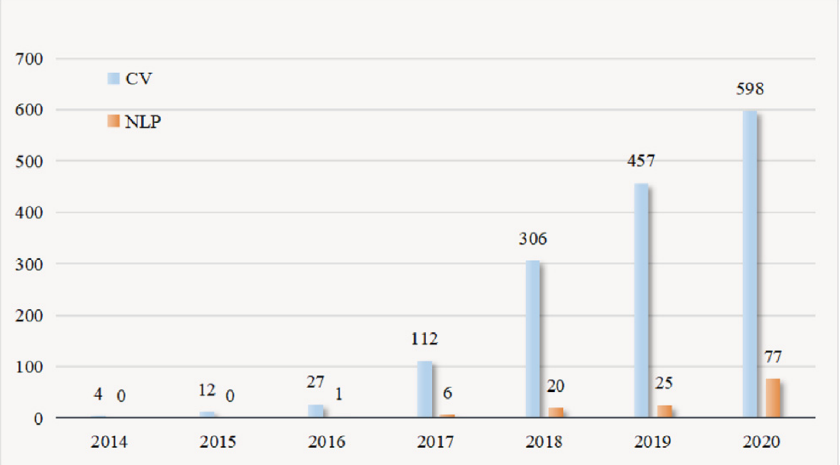


Figure 2.4: Adversarial technology trend in CV and NLP fields [31]

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

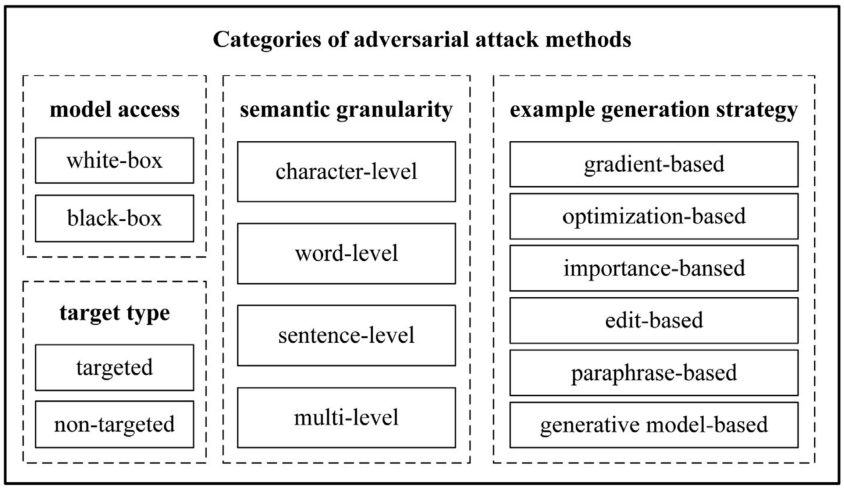


Figure 2.5: Categorization of textual adversarial attack methods [31]

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 [3].

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 [17, 32, 33] 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 [31].

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 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. [31] divide adversarial attacks into six types: gradient-based, optimization-based, importance-based, editbased, 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 optimizationbased 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. [15] divided textual adversarial attacks into four categories: the character-level, word-level, sentencelevel, 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 texted, swapped with a neighbor, 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 attack.

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 [9]), 4 word-level attacks (Probabilistic Weighted Word Saliency (PWWS) [32], TextFooler [17], BERT-based attack [21] and Semese-PSO [37]), 2 sentence-level attacks (Synthetically Controlled Paraphrase Networks (SCPNs [16]), and GAN-based attack [39]), and 1 multi-level attack (TextBugger [20]).

DeepWordBug detemines top critical tokens and modifies them by character-level transformations introducing typos. PWWS is a synonym substitution method that make use of the word saliency and classification probability. TextFooler identifies important words, and replace 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 by 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 consisting 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.

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

2.2.4.1 TextFooler

2.2.4.2 BERT-based attacks

2.3 Machine Learning hardening

2.3.1 Vanilla adversarial training

2.3.2 Attacking to Training

2.4 Text Attack

2.4.1 Framework structure

2.4.2 Attack components

2.4.3 HuggingFace integration

Chapter 3

Methodology

In this chapter we will present the datasets created and used for the visual odometry.

- Introduction

- Research Design
- Research Questions and Hypotheses
- Setting and Sample
- Data Collection
- Data Analysis
- Conclusion

demonstration of fit between methods chosen and research question(s) rationale for choosing materials, methods and procedures details of materials, equipment and procedures that will allow others to: replicate experiments understand and implement technical solutions

Chapter 4

Experimental results

In this chapter we will discuss about different models and different prediction strategies.

Chapter 5

Final discussions

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

A clear answer to your research question or hypothesis Summary of the main findings or argument Connections between your findings or argument to other research Explanation and significance of the findings Implications of the findings Limitations of the research and methodology Recommendations for future research

Summary of Findings Limitations of the research Suggestions for Future Research Conclusion

5.1 Summary of findings

5.2 Limitations

5.3 Future developments

5.4 Conclusions

Bibliopraphy

Bibliography references

[6] 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/ (cit. on p. 4).

[12] Ralph Grishman. *Computational linguistics : an introduction*. Cambridge, Mass.: Cambridge University Press, 1986 (cit. on p. 5).

[18] 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 (cit. on p. 9).

Paper references

[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 (cit. on p. 7).

[2] 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 (cit. on p. 13).

[3] 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 (cit. on pp. 11, 15).

[4] 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 (cit. on p. 5).

[8] 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 (cit. on p. 8).

[9] 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 (cit. on p. 18).

[10] 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.

[13] Louise Guthrie et al. "The role of lexicons in natural language processing". In: *Commun. ACM* 39.1 (1996), pp. 63–72. ISSN: 0001-0782 (cit. on p. 6).

[14] 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.

[15] 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 (cit. on p. 17).

[16] 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 (cit. on p. 18).

[17] 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 (cit. on pp. 16, 18).

[19] 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 (cit. on p. 8).

[20] 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 (cit. on p. 18).

[21] 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 (cit. on p. 18).

[22] Yinhan Liu et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: (2019) (cit. on p. 8).

[24] 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 (cit. on p. 7).

[25] 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 (cit. on p. 6).

[26] 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 (cit. on p. 13).

[27] 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 (cit. on p. 9).

[28] 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 (cit. on p. 13).

[29] 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 (cit. on p. 13).

[30] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: (2014), pp. 1532–1543 (cit. on p. 7).

[31] 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 (cit. on pp. 14–16).

[32] 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 (cit. on pp. 16, 18).

[33] 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 (cit. on p. 16).

[34] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv preprint arXiv:1910.01108* (2019) (cit. on p. 8).

[35] Christian Szegedy et al. "Intriguing properties of neural networks". In: (2013). URL: http://arxiv.org/abs/1312.6199 (cit. on pp. 12, 13).

[37] 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 (cit. on p. 18).

[38] 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 (cit. on p. 13).