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 ...

Abstract

" 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

Thanks

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

1.2 Problem statement

1.3 Research question

1.4 Solution

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.

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 [7].

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 [14].

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 [5].
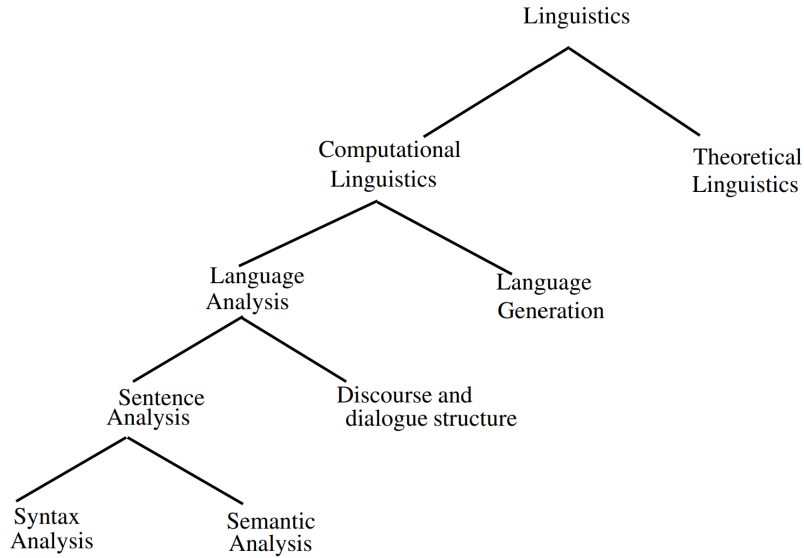
Figure 2.1: Components of NLP

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 [15].

A lexicon which provides an effective combination of traditional lexicographic information and modern computing is called WordNet [29]. 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 [28] and GloVe [36].

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 [9], which is a bidirectional encoder representation from a particular deep learning architecture called transformer [43]. 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 [26] 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 [40] reduces the size of $BERT_{BASE}$ by 40%, enhances the speed by 60% while retaining 97% of its capabilities. ALBERT [23] 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 [22].

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 [33].

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. [22]

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 [49]. Good results were obtained

using an ensemble, including both LSTMs and CNNs [6]. 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 [27], 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.

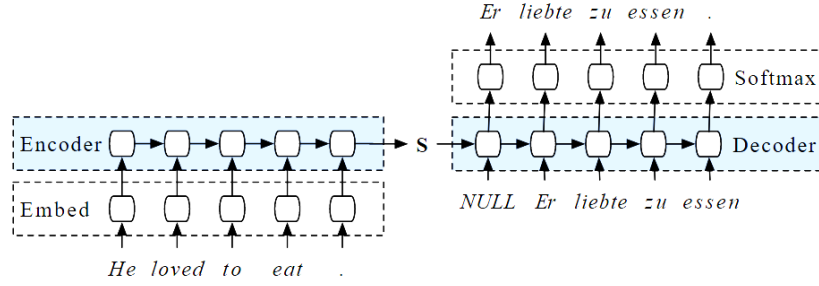An example of this architecture is shown in Figure 2.2.

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 [8].

Two main types of AI attacks can be defined according to the time at which the attack happens [4]:

- **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 regonized 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 [41] (b) Poisoning attack on training data [8]
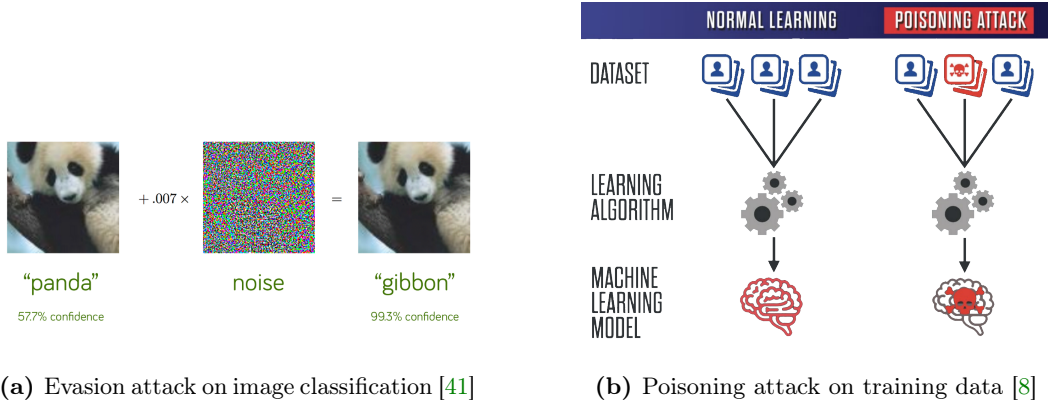
Figure 2.3: Examples of Artificial Intelligence Attacks

In this thesis, we will focus on the first type of attack, the Adversarial Attacks. Over past few years, researchers [12, 41] 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. [41] 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 ($f(x') = y'$) [47].

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. [41] was based on L-BFGS, but despite the effectiveness of the method, it was computationally expensive and impractical. Goodfellow et al. [12] 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 [35], DeepFool [30], C&W [2]) or leveraging Generative Adversarial Network (GAN) to generate adversaries [48].

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. [34] 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.
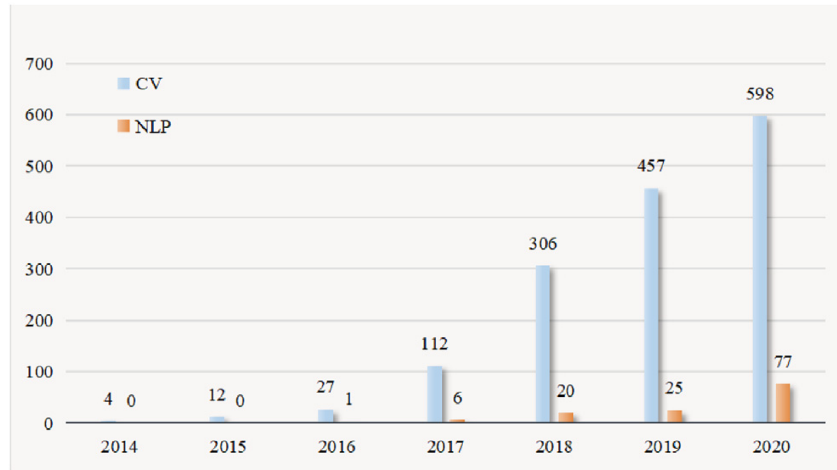
---

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

Figure 2.4: Adversarial technology trend in CV and NLP fields [37]

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
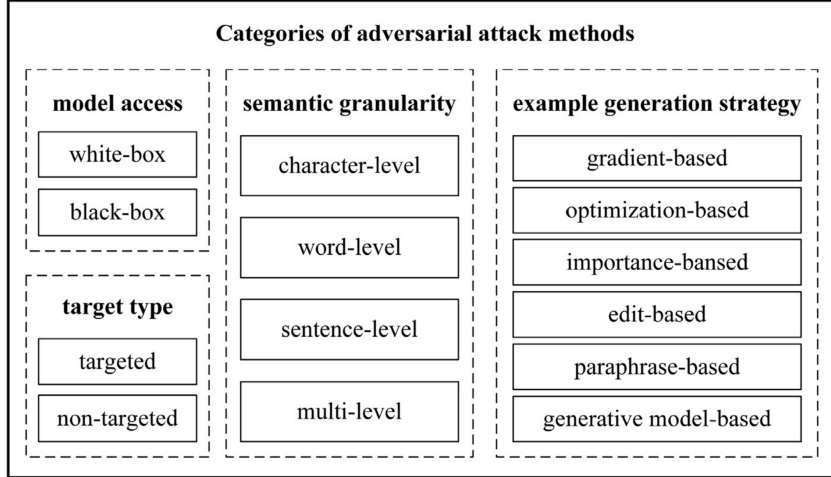
Figure 2.5: Categorization of textual adversarial attack methods [37]

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 [4].

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 [21, 38, 39] 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 [37].

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. [37] 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. [19] 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 [10]), 4 word-level attacks (Probabilistic Weighted Word Saliency (PWWS) [38], TextFooler [21], BERT-based attack [25] and Semese-PSO [45]), 2 sentence-level attacks (Synthetically Controlled Paraphrase Networks (SCPNs [20]), and GAN-based attack [48]), and 1 multi-level attack (TextBugger [24]).

DeepWordBug detemines top critical tokens and modifies them by character-level transformations introducing typos. PWWS is a synonym-based 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 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 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.

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

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

BERT model for sentiment analysis.

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 char**m** of satin rouge is that it avoids the obviou**s** with hum**o**ur and lightness .	*Negative*(61.23%)
2) just the labour involved in creating the **layere** richn**e**ss of the imagery in this chiaroscuro of madness and light is as**t**onishing .	*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 [37]

For comparing the attack performance of the above methods, Shilin Qiu et. al. [37] 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 [42]. 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 [37]

2.2.4.1   TextFooler

TextFooler [21] 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. [32] are used. These GloVe vectors are specially curated for injecting antonymy and synonymy constraints into vector space representations. They achieve the state-of-theart performance on SimLex-999, a dataset designed to measure how well different models judge semantic similarity between words [18].

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) [3] 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 [17].

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 [25]. 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 [11]. 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 defenses has given good competition to the novel adversarial attack algorithms and considerably improved the robustness of existing deep learning models. These defense mechanisms are also used as regularization techniques to avoid overfitting, and making the model more robust [13].

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 [12]. 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 as to what extent such training can improve an NLP model's performance.

[44] 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 literature. Previous attacks such as [11, 21] 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 [31] and OpenAttack [46]. 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.
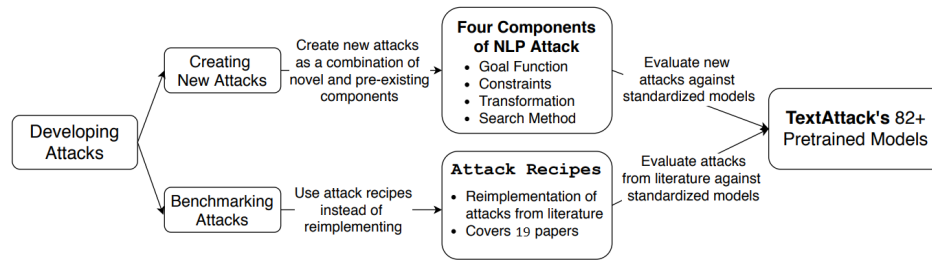
Figure 2.7: Main features of of TextAttack

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 benchmarck 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 [31].

---

[3] https://huggingface.co

Chapter 3

Methodology

3.1   Defined goal

3.1.1   Problem to solve

3.1.2   Research objective

3.2   Research design

3.2.1   Models to attack

3.3   Proposed solution

3.3.1   Intuition

3.3.2   SynBA components

3.3.2.1   Search Method

3.3.2.2   Transformation

3.3.2.3   Constraints

3.3.2.4   Goal Function

3.3.3   Hyperparameter Tuning

3.3.4   Candidates ranking calibration

3.4   Evaluation metrics

3.4.1   Attack metrics

3.4.2   Quality metrics

3.4.3   Performance metrics

Chapter 4

Experimental results

4.1 Data collection

4.1.1 Experimental setup

4.1.2 Datasets perturbed

4.1.3 Model attacked

4.2 Qualitative evaluation

4.2.1 Results on rotten-tomatoes

4.2.2 Results on imdb

4.3 Quantitative evaluation

4.3.1 Performances on rotten-tomatoes

4.3.2 Performances on imdb

4.4 Human evaluation

Chapter 5

Final discussions

5.1 Summary of findings

5.2 Limitations

5.3 Future developments

5.4 Conclusions

Glossary

A2T Attack to Training is a method to generate adversarial examples. It consists in training a model while generate adversarial examples.. 35

Attack success rate The attack success rate is the percentage of adversarial examples that are not correctly classified by the victim model.. 19, 33

Bag-of-words The bag-of-words model is a simplifying representation used in natural language processing and information retrieval. In this model, a text is represented as the bag of its words, disregarding grammar and even word order but keeping multiplicity.. 5, 33

BERT BERT is a method of pre-training language representations, which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. BERT was created and published in 2018 by Jacob Devlin and his colleagues from Google. BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).. 35

CNN Convolutional Neural Networks are a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks, based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series.. 35

CV Computer Vision is the field of study that deals with how computers can gain

high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.. 35

FSGM Fast Sign Gradient Method is a method to generate adversarial examples. It consists in adding a small perturbation to the input of a model. The perturbation is computed by taking the sign of the gradient of the loss function with respect to the input.. 35

GA Genetic algorithms are search heuristics that mimic the process of natural selection. This algorithm repeatedly modifies a population of individual solutions. It does so by using operators such as selection, crossover, and mutation to create new generations of individuals.. 35

GAN Generative Adversarial Networks are a class of machine learning systems invented by Ian Goodfellow in 2014. Two neural networks contest with each other in a game. Given a training set, this technique learns to generate new data with the same statistics as the training set.. 35

Language model A language model is a probability distribution over sequences of words. Given such a sequence, say of length m, it assigns a probability $P(w_1 : w_2 : ... : w_m)$ to the whole sequence.. 6, 33

Lexicon A lexicon is a collection of words along with associated information such as part of speech and sense definitions.. 4, 33

LSTM Long Short-Term Memory is a type of recurrent neural network architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.. 35

MLM Masked language model is a pre-training technique for NLP. It consists in masking some words in a sentence and then train a model to predict the masked words.. 35

NLI Natural Language Inference (NLI) is a text classification task, where given a premise sentence and a hypothesis sentence, the model must predict the

relationship between them. The relationship can be entailment, contradiction, or neutral.. 35

NLP Natural Language Processing is an important field of Artificial Intelligence, linguistics and computer science. It 's about the interactions between computers and human language, in particular, how to program computers to process and analyze large amounts of natural language data.. 35

PSO Particle Swarm Optimization is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formula over the particle's position and velocity.. 35

RL Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward.. 35

RNN Recurrent Neural Networks are a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.. 35

Seq2seq Seq2seq is a model that maps a sequence of symbols to another sequence of symbols.. 35

SST The Stanford Sentiment Treebank is a corpus of sentences labeled with fine-grained sentiment labels. It is used to train models for sentiment analysis.. 35

TF-IDF TF-IDF is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining.. 35

Transformer The Transformer is a model architecture for NLP. It is a neural network that uses attention mechanisms to learn contextual relations between words in a sentence.. 7, 33

USE The Universal Sentence Encoder encodes text into high-dimensional vectors that can be used for text classification, semantic similarity, clustering and other natural language tasks.. 35

Word Example of a term in the glossary. 2, 33

Word embedding Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation.. 5, 33

Acronyms

A2T Attack to Training. 23

BERT Bidirectional Encoder Representations from Transformers. 7

CNN Convolutional Neural Network. 9

CV Computer Vision. 12

FSGM Fast Sign Gradient Method. 12

GA Genetic Algorithm. 16

GAN Generative Adversarial Network. 12

LSTM Long Short-Term Memory. 9

MLM Masked language model. 6

NLI Natural Language Inference. 9

NLP Natural Language Processing. 3

PSO Particle Swarm Optimization. 16

RL Reinforcement Learning. 16

RNN Recurrent Neural Network. 9

seq2seq Sequence to Sequence. 9

SST Stanford Sentiment Treebank. 17

TF-IDF Term Frequency-Inverse Document Frequency. 6

USE Universal Sentence Encoder. 21

Bibliopraphy

Bibliography references

[7] 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. 3).

[14] Ralph Grishman. *Computational linguistics : an introduction*. Cambridge, Mass.: Cambridge University Press, 1986 (cit. on p. 4).

[22] 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. 8).

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. 6).

[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. 12).

[3] Daniel Cer et al. "Universal Sentence Encoder." In: *CoRR* abs/1803.11175 (2018). URL: https://arxiv.org/pdf/1803.11175.pdf (cit. on p. 21).

[4]   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. 10, 14).

[5]   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. 4).

[9]   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. 7).

[10]   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. 17).

[11]   Siddhant Garg and Goutham Ramakrishnan. "BAE: BERT-based Adversar-
        ial 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 (cit. on pp. 22, 23).

[13]   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 (cit. on p. 22).

[15]   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. 5).

[16]   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.

[19]   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. 16).

[20]   Mohit Iyyer et al. "Adversarial Example Generation with Syntactically Con-
        trolled 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. 17).

[21] 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. 15, 17, 20, 23).

[23] 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. 7).

[24] 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. 17).

[25] 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 pp. 17, 22).

[26] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: (2019) (cit. on p. 7).

[28] 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. 6).

[29] 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. 5).

[30] 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. 12).

[31] 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 (cit. on pp. 24, 25).

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

[33]   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. 8).

[34]   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. 12).

[35]   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. 12).

[36]   Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: (2014), pp. 1532–1543 (cit. on p. 6).

[37]   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. 13–15, 19, 20).

[38]   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. 15, 17).

[39]   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. 15).

[40]   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. 7).

[41]   Christian Szegedy et al. "Intriguing properties of neural networks". In: (2013). URL: http://arxiv.org/abs/1312.6199 (cit. on pp. 11, 12).

[44] 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` (cit. on p. 23).

[45] 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. 17).

[46] 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` (cit. on p. 24).

[47] 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. 12).

[48] 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` (cit. on pp. 12, 17).

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