# A Query-Efficient Black-box Adversarial Attack on Text Classification Deep Neural Networks

by

Mohammad Mehdi Yadollahi

**Master of Computer Software Engineering, University of Tehran, 2015**

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF**

**Master of Computer Science**

In the Graduate Academic Unit of Computer Science

| | |
|---|---|
| Supervisor(s): | Ali A. Ghorbani, PhD, Faculty of Computer Science<br>Arash Habibi Lashkari, PhD, Faculty of Computer Science |
| Examining Board: | Huajie Zhang, Ph.D., Faculty of Computer Science, Chair<br>Sajjad Dadkhah, Ph.D., Faculty of Computer Science<br>Mahdi S. Hosseini, Ph.D., Electrical and Computer Engineering |

This thesis is accepted by the
Dean of Graduate Studies

**THE UNIVERSITY OF NEW BRUNSWICK**

**2022**

# Abstract

Recent work has demonstrated that modern text classifiers trained on Deep Neural Networks are vulnerable to adversarial attacks. There are insufficient studies on text data compared to the image domain, and the lack of investigation originates from the special challenges of the NLP domain. Despite being extremely effective, most adversarial attacks in the text domain ignore the overhead they induced on the victim model. In this research, we propose a Query-Efficient black-box adversarial attack named EQFooler on text data that tries to attack a textual deep neural network while considering the amount of overhead that it may produce. The evaluation of our method shows that the results are promising .

We demonstrate the impact of keyword extraction methods in generating query-efficient adversarial attacks. Four variants of the EQFooler mode are developed based on different keyword extractors and importance score strategies. We compare the performance of these variants in terms of four evaluation metrics, namely original accuracy, adversarial accuracy, change rate, and number of queries. All the variants of the proposed attack significantly reduce the accuracy of the targeted models. Among those variants, EQFooler-Rake-MS has the best functionality in terms of adversarial accuracy, change rate and the number of queries needed. Also, multiple experiments are designed to compare the outcomes of the proposed method with the state-of-the-art adversarial attacks as a baseline. The results show that the EQFooler is as powerful as the state-of-the-art adversarial attacks while requiring

fewer queries to the victim model. In addition, we study the transferability of the generated adversarial examples. Compared to the baseline in any transfer setting, at least one of the variants has better outcomes than the baseline.

# Dedication

This thesis work is dedicated to my wife, Farzaneh, who has been a constant source of support and encouragement during graduate school and life challenges. I am genuinely thankful for having you in my life. This work is also dedicated to my parents, Mahbobeh and Mahmood, who have always loved me unconditionally and whose good examples have taught me to work hard for what I aspire to achieve.

# Acknowledgements

I would like to express my gratitude and appreciation for my patient and supportive supervisors, Dr. Ali A. Ghorbani and Dr. Arash Habibi Lashkari, whose guidance, support, and encouragement have been invaluable throughout this study.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Introduction

Deep neural networks (DNNs) are artificial neural networks designed as a series of layers of neurons. In this structure, computing components serve as individual neurons. They are attached to the other neurons by links with specific weights. The primary duty of the neurons is to convey the results of activation functions to the neurons of the next layer. The DNN structure is inspired by the human brain, which learns data and produces knowledge based on massive input samples. Since they use real-valued vector representations, they are talented at working with various data like images, texts, videos, and audios.

The security of machine learning models can be categorized into multiple levels. These categories include authenticating the owner of the model, preserving the privacy of the training data, defending against training time attacks (e.g. data poisoning), and test time attacks (e.g. adversarial examples). In this work, we mainly focus on the test time attacks. An adversarial example is an instance with small and intentional perturbations that cause a machine learning model to make a wrong decision while going unnoticed by the human.

DNNs are being used in many fields like natural language processing, image classification, and object detection with outstanding results. However, research has proven that DNN models are vulnerable to adversarial example attacks, causing incorrect predictions by adding imperceptible perturbations into standard inputs[64]. Studies on adversarial examples in the image domain have been well investigated, but research in the textual domain is inadequate [38, 52, 83]. Many researchers have shown that object detection, sound recognition, malware detection, and sentiment analysis systems are prone to adversarial example attacks.

Adversarial example generations in textual data is more challenging than in common computer vision tasks due to two main reasons: the discrete nature of input space and ensuring semantic coherence with the original sentence. While most of the prior efforts focus on word misspelling, word changing, and phrase removal and insertion, some other works generate adversarial examples in natural sentences. They craft an adversarial sample that is semantically and syntactically similar to the original sample [34]. None of the previous research concerns their resource consumption, such as time and number of queries needed to build a single adversarial example, to the best of our knowledge.

In the literature, adversarial attacks are categorized into three groups: white-box, black-box, and hard-labeling. In white-box attacks, adversaries have full access to architectures, weights, and parameters of target models. Therefore, these attacks have better performance than other two attacks. In the black-box scenario, adversaries have access only to the prediction labels and confidence score, which is part of the output model. A hard-label method is sometimes considered to be a black-box approach because the adversaries have access to the predicted label. However, in this kind of attack, the predicted label is the only information an adversary has access to. It means there is no confidence score in this scenario. Thus, in comparison with the other two categories, using a hard-label attack is more complicated.

A common approach to defending against black-box attacks is using a firewall to block the source, which frequently issues a high number of queries. So, reducing the number of requests needed for generating adversarial attacks is essential to have an efficient approach. On the other hand, one possible defence approach in the textual domain is to correct typos and grammatical errors that a given text might have. Thus, there are two crucial conditions while generating adversarial documents in the text-domain: 1) the crafted documents must be grammatically correct. 2) they should be semantically similar to the original documents. Here, we present an approach named EQFooler, which is designed to be query efficient and generate high-quality adversarial documents in terms of grammatical and semantics. It generates adversarial examples for the textual deep neural network models with comparable adversarial accuracy to the state-of-the-art methods. Additionally, it reduces the resources needed for developing adversarial examples by determining the critical words from a given document using keyword extraction methods without considering the model. After extracting keywords as the candidates, we prioritize them based on either their keyword extractor's score or the target model confidence scores. Then, a set of synonyms will be extracted for each keyword using word embedding. The candidates will be replaced with the most similar semantic synonyms that are grammatically correct. We continue this process until the predicted label changes and the adversarial document be found.

## 1.2 Summary of Contributions

The main purpose of this research is to develop a threat model for generating adversarial examples for deep learning models in the NLP domain. In summary, the following are the contributions of this thesis:

- **Black-box adversarial attack**: Our approach has black-box access as query

access. It means that the adversary can supply any input $x$ and get the predicted class probabilities, $P(y|x)$ for all classes $y$. So, the adversary cannot analytically compute the gradient $\nabla P(y|x)$ as is allowable in the white-box case.

- **Query-limited approach**: The proposed framework can perturb the original model so that it causes misclassification using a limited number of queries. In this research work, we propose a query-efficient algorithm for generating adversarial examples on text classification models. This approach can be applied when there is a limit on time or money, especially if inference time is a bottleneck or the attacker incurs a cost for each query.

- **Partial-information setting**: Based on the definition presented in [29], in the partial information setting, the attacker has access only to the probabilities $P(y|x)$ for $y$ in the top $k$ (e.g., $k = 5$) classes $\{y_1, ..., y_k\}$. Our proposed threat model has access only to the top label and its probability. So, we can consider it as a special case of this setting where $k = 1$ and a partial-information attack should succeed in this case as well.

- **Implementation and Evaluation**: We develop a library that performs a query-efficient adversarial attack on a text classification deep neural network model. We evaluate the effectiveness of the proposed method by comparing the outcomes with one of the state-of-the-art adversarial attacks in terms of adversarial accuracy, change rate and number of queries.

## 1.3 Thesis Organization

The rest of the thesis is organized as follows.

- chapter 2: Literature Review first discusses the required definitions for under-

standing the previous and current studies, including machine learning, neural networks and deep learning. Then it reviews the recent related work in adversarial attacks, especially adversarial attacks on the textual content. Also, it explains the classification of different adversarial attacks proposed in this domain.

- chapter 3: Proposed System Design introduces a comprehensive overview of the proposed framework, followed by an explanation of each stage. It explains how an adversarial attack efficiently happens through three phases of the proposed method. Also, this chapter presents the principal parts implemented in the adversarial attack framework and describes their primary duties.

- chapter 4: Experiments and Evaluations depicts the results of the proposed efficient adversarial attack implementation. This chapter reports the dataset, evaluation metrics, and results of the proposed framework considering different settings. Also, a comparison between the proposed attack and some other approaches is provided.

- chapter 5: Conclusion and Future Work concludes the thesis with a summary of its contributions, challenges and limitations of conducting the research, and remarks on future work.

# Chapter 2

# Literature Review

## 2.1 Introduction

In recent years, machine learning has become a deniable part of human lives. It can be found anywhere around a person's daily life, from their cell phones, text editors and even a car that they drive. Machine learning has many branches and techniques, but recently deep neural networks have become more effective in many machine learning tasks. They have been used in different recognition problems in the text, image and speech, with extraordinary achievements. Because of these successes, many researchers and industrial companies applied deep learning techniques to their safety-critical tasks.

These deep learning techniques should be secure, accurate and stable. However, many attempts have proven that DNN models are vulnerable to adversarial examples [64]. Therefore the adversarial example attack can be formally defined as – "Adversarial examples are inputs to machine learning models that an attacker intentionally designed to cause the model to make mistakes." [24]

This chapter presents principle information about machine learning. Specifically, it focuses on the deep neural networks that work on the text content. Then, it

discusses the adversarial attacks on the textual model and tries to categorize them based on their nature. Afterward, it provides a comprehensive study on the topic of adversarial attacks on textual deep neural network models. Also, it explains the well-known adversarial attacks and defences proposed in the literature.

## 2.2 Machine Learning Background

Machine learning (ML) is an application of artificial intelligence (AI) providing systems that can automatically learn and improve themselves from experience without being explicitly programmed for a specific task. The primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust actions accordingly. In other words, machine learning focuses on developing computer programs that can access data and use it to learn for themselves. Like other computer science systems, a machine learning model can target security attacks on any part of its processes, i.e. training data, the learned parameters, the input data, the test output. The most common security attacks on ML models and solutions to prevent them are explained in the following.

- **Adversarial Machine Learning Attack:** Here, malicious attackers aim to find slight variations in the model data inputs, resulting in redirected and undesired model outputs. For example, an adversary can fool an image classification model by only modifying one pixel in the input image to output an incorrect label. This modification can cause the model to misidentify the image with a degree of certainty.
  - **Solution**: A robust DataOps and MLOps [35] solution can introduce randomness and intentional noise into the training set and the model tuning to reduce the model's vulnerability to such attacks.

- **Data Poisoning Attack:** Data poisoning attacks happen when the attacker recognizes the source of raw data used to train the model, then tries to bias or "poison" the data to compromise the resulting model's performance. Here, the hacker needs access to raw data.
  - **Solution**: The best solutions are prevention and detection. Production model monitoring of any data or concept drift can identify any phase shifts, imbalances, or shifts in data density and distribution. We can prevent the model from learning from compromised data through consistent monitoring.
- **Online Adversarial Attack:** If the model is online learning from a continuous stream of new data, an attacker can manipulate the model's learning by sending false data. If the model is learning from "fake news" or new, incorrect data, this can have ethical implications, such as learning from sexist, racist, or other hate speech. The manipulations of the attack are irrevocable since they operate on a transient data stream.
  - **Solution**: A robust data ingestion process in the MLOps solution would identify such a concentration of anomalous new data streams. Data versioning in the MLOps solution would allow the model to roll back to an earlier unaffected model version quickly.
- **Data Phishing Privacy Attack:** Hackers take advantage of specific existing parameters to reverse-engineer the dataset and break confidentiality through a data phishing attack. The main reasons for this type of attack are
  - The training set includes confidential data.
  - The model is overfitted and unable to generalize the new data.
  - The quantity of training data is deficient.

An example of this type of dataset is healthcare datasets. A patient record may contain extensive data, but the dataset may contain only a small number of patient records.

- **Solution**: Statistical techniques, such as randomized data hold-out or differential privacy, can be considered a layer of privacy protection. There are also promising discoveries being made in fully encrypted machine learning where the user of the ML model submits only encrypted input data, and the encrypted model is only exposed to the user. This ensures the privacy of the model and training data are fully protected.

### 2.2.1 Artificial Neural Network

Artificial neural networks (ANNs) are a subset of machine learning algorithms whose name and structure, inspired by the human brain, mimic how biological neurons signal to one another. They rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity. So, they are also at the heart of deep learning techniques.

Artificial neural networks (ANNs) comprise an input layer, one or more hidden layers, and an output layer (Figure 2.1). Each node connects to another with an associated weight and threshold. A node will be activated and send data to the next layer of the network if its output is above the specified threshold value. Otherwise, no data are passed along to the next layer of the network. Figure 2.1 shows an example of artificial neural network with the input ($x_1$ - $x_n$), the corresponding weights ($w_1$ - $w_n$), a bias $b$, and activation function $f$ applied to the weighted of sum input.

Each individual node acts as a linear regression model composed of input data, weights, a bias (or threshold), and an output which is generated as bellow:

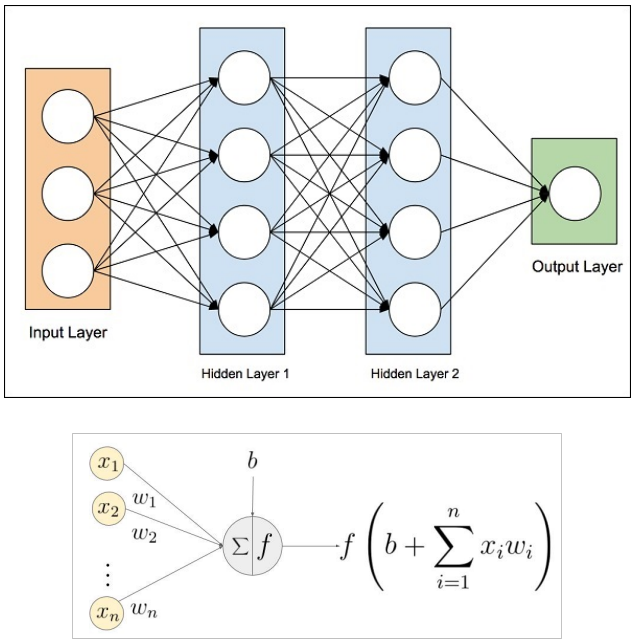



Figure 2.1: An artificial neural network with two hidden layers, and an example of neuron

$$output = f(x) = f(b + \sum_{i=1}^{n} x_i w_i) \tag{2.1}$$

where $x$ is input to neuron, $w$ is weights, $n$ is the number of inputs from the incoming layer, $b$ is bias, and $f(.)$ is the activation function. There are two broad categories of activation, linear and non-linear. Some important non-linear activation functions $f(z)$ are listed below:

- **Rectified Linear Units (ReLU):** ReLU ensures that the output does not go below zero (or negative). Therefore if $z$ is greater than zero, the output remains $z$, otherwise it is zero. The formula is $f(z) = max(0, z)$.
- **Tanh:** it returns the hyperbolic tangent of z as the output, $f(z) = tanh(z)$.
- **Sigmoid:** the output is calculated as $f(z) = 1/(1 + e^{(-1 \times z)})$.

Neural networks can be classified into different types, which are used for various

purposes. While there is not a comprehensive list of different types of ANNs, we describe the most common types of neural networks that are well-known because of their common use cases:

1) Perceptron: It is the oldest neural network, created by Frank Rosenblatt in 1958. It has a single neuron and is the simplest form of a neural network.

2) Feedforward neural networks, or multi-layer perceptrons (MLPs): They are comprised of an input layer, a hidden layer or layers, and an output layer with a sigmoid activation function, as most real-world problems are nonlinear. MLPs are the foundation for different well-known techniques in computer vision and natural language processing domains.

3) Convolutional neural networks (CNNs): They are similar to feedforward networks, but they are usually utilized for pattern recognition, image recognition, and computer vision. These networks harness principles from linear algebra, particularly matrix multiplication, to identify patterns within an image.

4) Recurrent neural networks (RNNs): They are identified by their feedback loops. They are commonly used in speech recognition, natural language processing, and other similar domains where dealing with sequential data is the bottleneck. Recurrent neural networks recognize data's sequential characteristics and use patterns to predict the next likely scenario.

### 2.2.2 Deep Learning

Deep learning is a type of machine learning and artificial intelligence that imitates the way humans gain certain types of knowledge. In contrast to traditional linear machine learning algorithms, deep learning algorithms are stacked in a hierarchy of increasing complexity and abstraction. A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. The name DNNs evolved from the use of many more hidden layers making it a 'deep'

network to learn more complex patterns. The wide range of models for deep neural networks includes DNNs, CNNs, RNNs, and LSTMs. Recent studies have even brought us attention-based networks that focus on specific parts of a deep neural network. The larger the network and the more layers it has, the more complex the network becomes and the more resources and more time it needs to train. Deep neural networks work best with GPU-based architectures that take less time to train than classical CPUs, while recent developments have shortened training times considerably [74].

While DNNs can model complex non-linear relationships, many issues can arise with naively training them. Two common issues are overfitting and computation time.

- **overfitting**: Since DNNs have added layers of abstraction for modelling rare dependencies in the training data, they are prone to overfitting. Some regularization techniques have been proposed for combating overfitting during the training phase, such as Ivakhnenko's unit pruning [31], weight decay ($\ell_2$-regularization) or sparsity ($\ell_1$-regularization) [5], and dropout regularization, which excludes rare dependencies by randomly omitting units from the hidden layers during training [14]. Also, data can be augmented to increase the size of smaller training sets to reduce the chances of overfitting.

- Since DNNs have many training parameters, such as the learning rate, initial weights, and the number of layers/units per layer, it is impossible to sweep through the parameter space for optimal parameters because of both time and computational complexities. So, several methods, such as batching [27] (computing the gradient on several training examples at once rather than individual examples), have been proposed to speed up computation. Alternatively, significant speedups have been produced in the training model by using large processing capabilities of many-core architectures (such as GPUs [79] or the Intel Xeon Phi [67]) because of their suitability for the matrix and vector

computations.

### 2.2.3 Deep Learning in Natural Language Processing

Natural language processing is a subfield of linguistics, computer science, and artificial intelligence that uses algorithms to interpret and manipulate human language. NLP techniques help data analysts turn unstructured text into usable data and insights. These techniques are able to understand, analyze, manipulate, and potentially generate human language. NLP is one of the most broadly applied areas of ML. It has several subsets, including natural language understanding (NLU), machine reading comprehension, and natural language generation (NLG), transforming data into human words.

For a long time, most methods used to study NLP problems employed shallow machine learning models and time-consuming, hand-crafted features. Most traditional machine learning models faced a problem called the "curse of dimensionality" because they primarily used hand-crafted features, and the linguistic information was represented with sparse representations. This motivated the study of distributed representations of words in low-dimensional space. With the success of word embeddings as low dimensional and distributed representations, NLP models have achieved higher results on various tasks as compared to traditional models.

#### 2.2.3.1 Word Embeddings

Word embeddings or distributional vectors necessarily follow the distributional hypothesis suggesting that words with similar meanings tend to occur in a similar context. Thus, distributional vectors capture the characteristics of the neighbours of a word. Capturing the similarity between words is the main advantage of distributional vectors. It is possible to measure similarity between vectors using different measuring metrics such as cosine similarity. In a deep learning model, word embed-

dings are often used as the first data processing layer. Typically, they are pre-trained in a large unlabeled corpus by optimizing an auxiliary objective, such as predicting a word based on its context where general semantic and syntactical information can be captured by learned word vectors. These per-tained word embeddings have proven to be efficient in capturing context similarity, analogies and they are efficient and fast because of their smaller dimensionality.

Collobert and Weston [12] were the first to show the utility of pre-trained word embeddings. Their proposed neural network architecture formed the foundation of many current approaches. After that, Word embeddings were revolutionized by Mikolov *et al.* [46, 45], who proposed the CBOW and skip-gram models to construct high-quality distributed vector representations efficiently. As shown in Figure 2.2, the CBOW model tries to estimate the conditional probability of a target word given the context words surrounding it across a window of size $k$. On the other hand, the skip-gram model does the exact opposite of the CBOW by predicting the surrounding context words given the central target word. Pennington *et al.* [55] developed an extension to the word2vec method for efficiently learning word vectors, called Global Vectors for Word Representation, or GloVe. Their proposed model benefits from the global statistics of matrix factorization techniques like Latent Semantic Analysis (LSA) with the local context-based learning in word2vec. FastText [1] is another word embedding method that is an extension of the word2vec model proposed by Facebook in 2016. It can understand suffixes and prefixes by capturing the meaning of shorter words. To do so, fastText represents each word as an n-gram of characters (sub-words) instead of learning vectors for words directly. So, for example, take the word “learning” with n=4; the fastText representation of this word is <lear, earn, arni, rnin, ning>. The word embedding vector for "learning" will be the sum of all these n-grams.

[1] https://fasttext.cc/

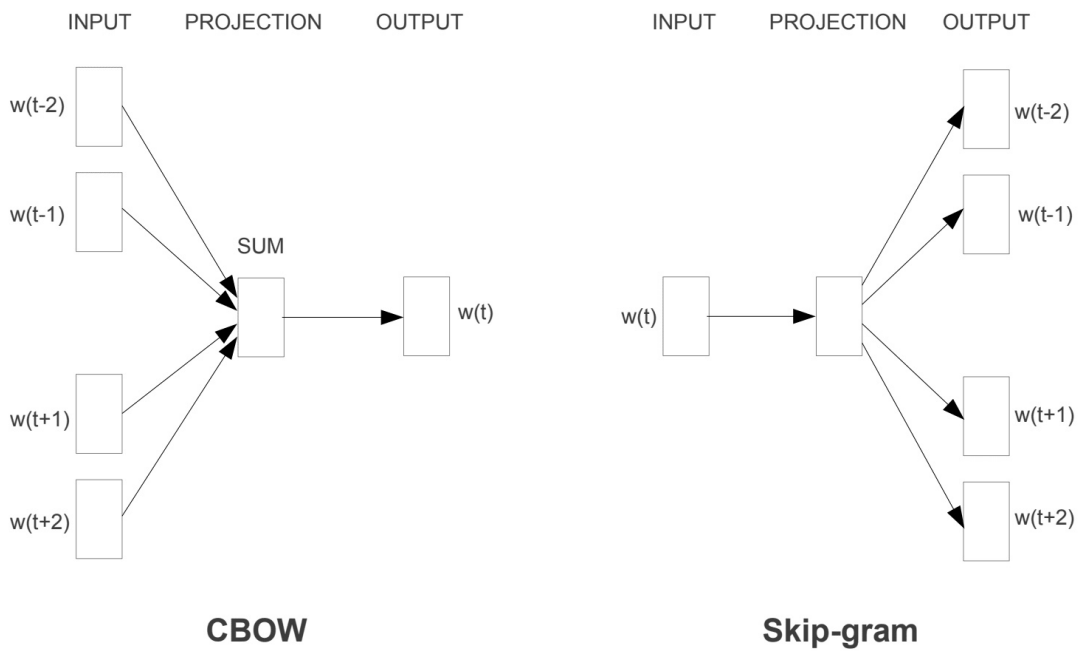



Figure 2.2: Word2Vec Training Models [45].

#### 2.2.3.2 Deep Learning models

Recently, deep neural networks have been gaining increasing popularity in the NLP community. Various DNN models have been proposed and adopted in different tasks of NLP domain. The most common deep neural networks in NLP are Convolutional Neural Networks (CNN), Recurrent/Recursive Neural Networks (RNN), and their variants. We will briefly overview the DNN architectures and techniques applied in NLP that are closely related to this research.

**Convolutional Neural Networks (CNN):** CNNs consist of some convolutional layers and pooling (down-sampling) layers and a final fully-connected layer. They use activation functions to connect the down-sampled layer to the fully-connected layer or the next convolutional layer. The convolution operation in the convolutional layer tries to extract meaningful local patterns of input. The pooling layer allows the network to be deeper with less overfitting by reducing the parameters and computation in the network. Generally speaking, CNN models learn a number

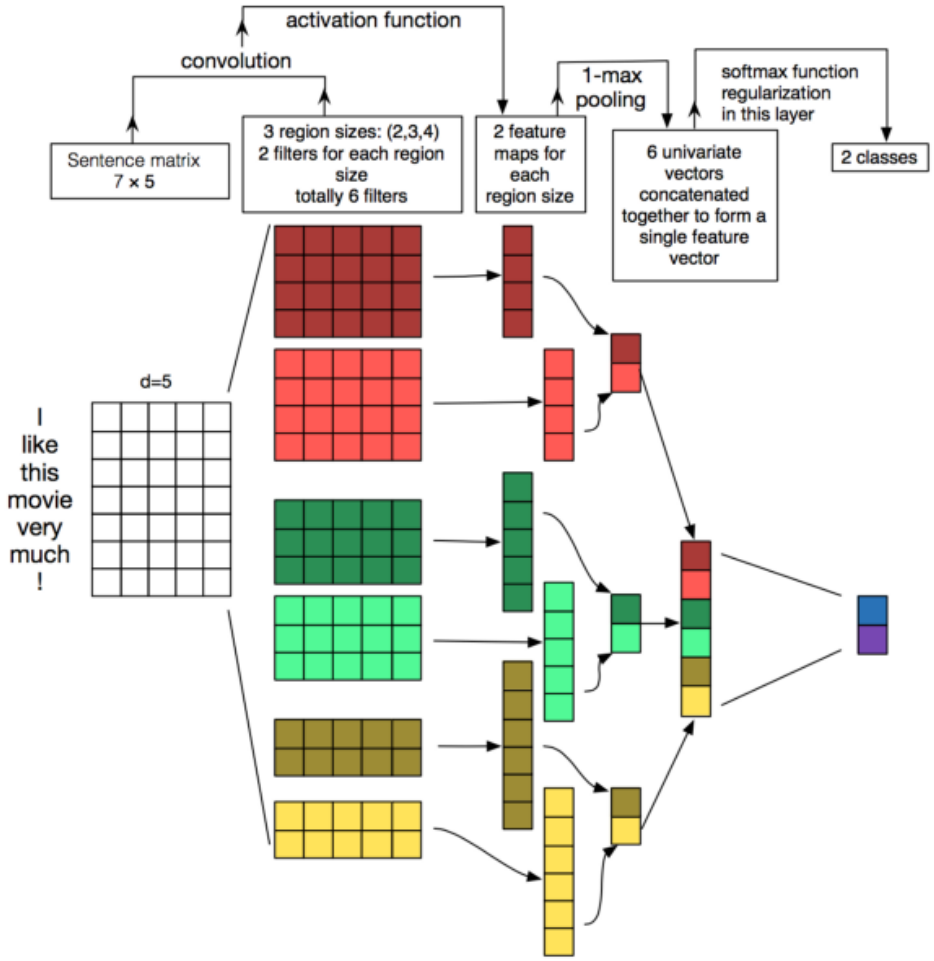



Figure 2.3: An example of what happens when using CNN on text data [82].

of local predictors and combine them to define a fixed-sized vector for the inputs, representing the most important informative aspects for the application task. In addition, CNN allows arbitrarily-sized inputs, and it is order-sensitive. Figure 2.3 shows an example of what happens when using CNN on text data. CNN models and their different variants excel in computer vision tasks and are later widely adopted in NLP applications. In [37], Yoon Kim adopted a CNN model for sentence classification, and it became a benchmark work in adopting CNN in NLP applications. His proposed model utilizes Word2Vec to represent words as the input vector. Then the convolutional operation is limited to the direction of word sequence rather than

the word embeddings. Multiple filters in pooling layers deal with the variable length of sentences. The model demonstrated outstanding performance on several benchmark datasets. Later, another CNN model was proposed by Zhang *et al.* [81] for text classification at the character level. It uses the one-hot representation in the alphabet for each character. It additionally performed data augmentation by replacing words and phrases with their synonyms to control the generalization error. These two well-known CNN models are evaluated via adversarial examples in many applications [4, 16, 17, 19, 40].

**Recurrent/Recursive Neural Networks:** Recurrent NNs are adapted from feed-forward NNs for learning mappings between sequential inputs and outputs [58]. They allow data with arbitrary length, which introduces cycles in their computational graph to model efficiently the influence of time [23]. This kind of model demonstrates impressive performance in dealing with sequential data [20] since recursive NNs do not suffer from statistical estimation problems originating from data sparsity. Recursive NNs [21] extend recurrent NNs from sequences to tree, respecting the hierarchy of the language. Additionally, Bi-directional RNNs were introduced for looking at sentences in both directions, forwards and backwards, utilizing two parallel RNN networks and combining their outputs. RNN models were first applied in NLP tasks by Bengio *et al.* [6]. Specifically, they employed RNN in the language model, where the probability of a sequence of words is computed in a recurrent fashion. The input is the feature vectors for all the preceding words, and the output is the conditional probability distribution over the output vocabulary. Therefore, RNNs have been applied to many NLP tasks because of their ability to model various kinds of sequential data. Consequently, RNN has also attracted attention for its role in countering adversarial attack [53]. RNN has many variants, among which Long Short-Term Memory (LSTM) network [51] is the most popular. LSTM, as a specific RNN, was designed to learn the long-term dependencies by its hidden states. The

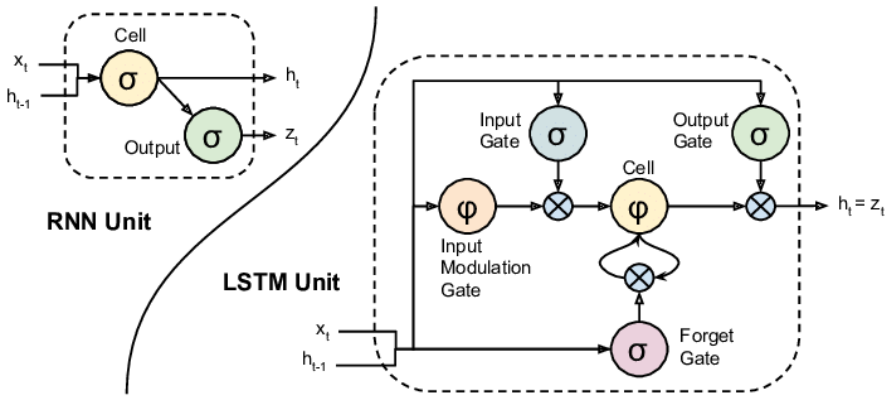



Figure 2.4: A basic RNN cell (left) and an LSTM memory cell (right) [15]

hidden states are computed by combining three gates, namely input gate, forget gate, and output gate. These gates control information flow which draws on the logistic function. Figure 2.4 illustrates the RNN cell and LSTM memory cell. Some popular LSTM variants are introduced to solve different NLP tasks [11, 28, 56, 65, 71, 78]. LSTM networks have subsequently been demonstrated to be more effective than conventional RNNs [25]. These representative works have recently been evaluated against adversarial examples recently [19, 32, 33, 47, 53, 56, 60, 63, 83].

## 2.3 Adversarial Attacks

In the literature, adversarial attacks are categorized into three types: white-box, black-box, and hard-labeling. In white-box attacks, adversaries have full access to architectures, weights, and parameters of target models. Therefore, these attacks have better performance than black-box attacks. In the black-box scenario, adversaries have only the prediction labels and confidence score, which is part of the output model. A hard-labeling method is sometimes considered to be a black-box approach because the adversaries have access to the predicted label. However, in this kind of attack, the predicted label is the only thing an adversary has access to. It means there is no confidence score in this scenario. So, in comparison to other

categories, attacking a model with this approach is more complicated than the two others.

In addition, it is possible to classify adversarial attacks according to the modified objects. This modified object can be a character, word, or sentence. So, adversarial attacks in the text domain can be divided into three categories, named character-level, word-level, and sentence-level [72].

### 2.3.1 White-box Attack

To the best of our knowledge, the first study that has been done on the text was conducted by Papernot *et al.* [53]. By using computational graph unfolding, they could evaluate the forward derivative to embed inputs of word sequences. After that, an FGSM method generates the adversarial perturbations. This approach has a main flaw which is the generated vector may not represent a valid word. To overcome this problem, they use an identical dictionary to find the substitution words. Their results show that, although their method fools an LSTM model to predict wrongly, the generated adversarial document has grammatical errors.

Samanta *et al.*[59] applied the same approach that Papernot did. They also employed an FGSM to select those words that profoundly affected the classification results when they were eliminated from the original text. To build a modified document, they applied three alteration strategies (insertion, substitution, and deletion) on some of the words with the highest importance. They required a dictionary in the insertion and replacement approaches to find the proper surrogate word for substitution. This dictionary has been made with synonyms, typos, and type-specific keywords for each word. Despite the performance, their method has two main disadvantages. First, converting a document to an adversarial one takes too long, and it is not effective. Second, for some important words, it is not able to find even one candidate for replacement.

A method named HotFlip was proposed by Ebrahimi *et al.* [17], similar to one-pixel attack [62]. HotFlip depends on an atomic flip operation like replacing one character using gradient computation in the white-box scenario. By doing some modifications, their approach is capable of generating word-level perturbation. The main disadvantage of this approach is not suitable for large-scale data because most successful adversarial examples have more than one or two flips.

### 2.3.2 Black-box Attack

Gao *et al.* [19] introduced DeepWordBug, a method to create adversarial documents in black-box scenarios. Their approach has two sections, finding important words and applying imperceptible perturbations for deceiving a trained model. Some works proposed a framework to attack a deep neural network in black-box and white-box scenarios like the work that Li *et al.* [39] have done. In the white-box setting, they calculate a Jacobian matrix to find the importance score of each term, and then, they generate both character-level and word-level adversarial examples. In the black-box setting, the authors divided records into multiple sentences. Then they separated the sentences that had different labels. After that, isolated sentences were sorted by their confidence score, and the most influential words were calculated. The most influential words were calculated by removing operation. The modification process in the black-box setting was the same as the white-box.

Based on genetic algorithm [3], Alzantot *et al.* [2] proposed an approach to generate adversarial examples. The authors picked words from the documents and calculated their nearest neighbors in GloVe embedding space[55] by Euclidean Distance. Then they filtered out words with high language model scores to create a new generation. If the predicted label of a modified sample was different from the original one, they chose two examples randomly as parents to build the next generation. Although the predicted label of a sample may change after the substitution process, these

substitutions may not be correct grammatically or semantically.

One of the most powerful attack methods on textual deep neural networks was published by Jin *et al.*[34] They proposed a baseline to create adversarial text from an original document. They evaluated their framework on two major natural language tasks, text classification and textual entailment. To generate an adversarial example, they first determined the influential words in the document and then replaced them with their synonyms. They applied some constraints on their method to ensure that the generated document semantic-wise and grammatic-wise are correct. They showed that their approach could decrease the performance of many DNNs such as pre-trained BERT, convolutional, and recurrent neural networks.

### 2.3.3 Hard-labeling Attacks

Maheshwary *et al.* [43] developed an adversarial attack approach using search space reduction and population optimization. Their proposed method is a hard-labeling (decision-based) attack strategy that fools text classification and entailment Deep Neural Network models. Their attack strategy leverages a bio-inspired optimization algorithm to create semantically similar adversarial examples by only using the predicted label of the target model. The fitness function of the optimization algorithm is expanding the semantic similarity between the original document and crafted adversarial text. They demonstrated the efficacy of their attack by comparing it to the state-of-the-art attack strategy.

Wallace *et al.* [68] noted that Machine Translation (MT) systems are vulnerable to be stolen by querying them and training models to simulate their outputs. They demonstrated that MT model stealing is possible in the hard-labeling setting. To prove their statement, they trained imitation models that achieve the effectiveness of the original. They also proposed a mitigation method for this kind of attack.

### 2.3.4 Targeted

Adversarial attacks can be categorized into targeted and non-targeted attacks based on the adversary's intention. For targeted attacks, an adversary creates an adversarial example $x'$ to target a specific class t by increasing the targeted class's confidence score. In this kind of attack, attackers intentionally manipulate the results of the model based on their desires. Since targeted attacks need more knowledge about the model, the costs for applying this kind of attack are higher than for non-targeted ones.

### 2.3.5 Non-targeted

In contrast to targeted attacks, the purpose of non-targeted attacks is to fool the model in any possible way. The predicted label for a crafted example $y'$ can be any of the available classes of he model except $y$, which is the original label of the sample. Non-targeted attacks rely on decreasing the confidence score of the correct category, unlike the targeted attack. Thus, the distribution of misclassified samples is the least concern. The non-targeted attacks in the textual area have three main approaches to generate an adversarial example from an original one: FGSM-based, optimization-based, and importance-based. All of these approaches affect the prediction result, but the influence of the important-based method on prediction results is stronger than those of the other two [73].

## 2.4 Classification of Adversarial Attacks on Textual Model

Since the data in the text domain are different from the image or audio data, attack strategy and attack types are somewhat different. So, we can classify attacks on

text mining models differently based on which components are modified. We can classify attack techniques into four types: character-level attacks, word-level attacks, sentence-level attacks, and multi-level attacks. Text data are generally inserted, removed, swapped/replaced, or flipped in these adversarial attacks.

### 2.4.1 Character-Level Attack

The character-level attacks create an adversarial example by modifying one or more characters in a document. These characters include letters, special symbols, and numbers. For instance, in HotFlip [17], the authors substitute one character based on gradient computation to convert a document to an adversarial one. Also, in DeepWordBug [19], after finding significant words, the authors apply slight perturbations (character-level) to generate an adversarial example.

In contrast to these two methods, iAdv-Text [60] applies a character-level attack to produce a modified word interpretable by a person. iAdv-Text uses an optimization algorithm to find a substitution that exists in the word embedding space.

### 2.4.2 Word-Level Attack

Unlike character-level attacks, word-level attacks alter words instead of modifying characters. Some approaches used in character-level attacks perform similar modifications to those used in word-level attacks. HotFlip and iAdv-Text are some of those algorithms. These algorithms find the candidates for replacement between synonyms, typos, genre-specific keywords.

Gong *et al.* [22] proposed a similar approach to iAdv-Text regarding their optimization approach. They looked for a perturbation example in embedding space. They applied WMD for measuring the similarity. Compared to iAdv-Text, the generated adversarial by Gang *et al.*'s model is less readable than iAdv-Text.

### 2.4.3 Sentence-Level Attack

Some researchers extend the idea behind the word-level attack to achieve another type of attack that adds perturbation to the sentences of a document. Jia *et al.* [33] proposed an adversarial attack that adds a crafted sentence to the original paragraph to fool the reading comprehension systems. They targeted the adjective of a sentence and replaced them with antonyms. Also, they leveraged GloVe embedding space [55] to find a substitution for named entities and numbers. In addition to ADDSENT, they introduced another approach to generate adversarial sentences named ADDANY. This approach chose the word for crafting randomly and thus does not validate the generated sentences grammatically.

## 2.5 Attacking Image DNNs vs Attacking Textual DNNs

In contrast to the image adversarial example generation algorithm, adversarial example attacks on text-domain data could not leverage the gradient-based attack. The main reason is that the nature of input data in the textual content is discrete. So, an adversary has to generate discrete perturbations to craft an adversarial example. The other difference between generating adversarial examples in the image and text domain is their metrics. Most of the studies in the image domain use $L0$, $L2$, and $L\infty$ to evaluate the similarity and undetectability of crafted samples with the original document. Most evaluation metrics in the image domain cannot be directly applied to text data, and they need some modification or assumption.

Linguistically, a text document is readable if it is grammatically and semantically correct. So, every crafted or modified text should be grammatically and semantically valid. This validation does not exist in the image domain, but any generated text documents must be evaluated to ensure they are human-readable.

To conclude this comparison, Zhang et. al [80] identified three main differences between generating adversarial examples in the text and image domain.

- Discrete vs Continuous Inputs: Input data for images is continuous, but we have discrete data in texts. This difference leads to building new text generating approaches and using different similarity metrics.

- Perceivable vs Unperceivable: Changing some few pixels in an image is not critical and can be ignored, but small changes in a word even though on the character level can cause an unreadability and be detectable by spell checkers and grammar checkers

- Semantic vs Semantic-less: the semantic integrity of an image is not changed by tampering with a few pixels. But changing a word can alter the semantic value of a sentence or the whole document.

## 2.6 Adversarial Defenses

The primary goal of developing adversarial attacks to machine learning techniques like deep neural networks is to improve their robustness. To the best of our knowledge, there are two common ways to accomplish this purpose adversarial training and knowledge distillation. In adversarial training, authors try to improve the robustness of their model by generating adversarial example for their model and combine them with the traing data. In this case the trained model learns how to deal with the real attack in production environments. Knowledge distillation approaches create a new neural network model based on the existing model's knowledge. This method helps the final model to improve its flaws.

### 2.6.1 Adversarial Training

The idea behind the adversarial training was first used by Szegedy *et al.* [64] and then followed by Goodfellow *et al.* [24]. They built a model that used original data and adversarial examples as its training set to improve the robustness of the trained model against multiple attacks. This section explains some well-known methods like data augmentation, model regularization, and robust optimization.

#### 2.6.1.1 Data Augmentation

The data augmentation method is the most common and most efficient technique against black-box adversarial attacks. Data augmentation methods create some adversarial examples based on original data, and then they combine the crafted samples with the original dataset to enlarge the training set. This approach helps the deep neural network models to see more similar data in the training phase. Although data augmentation improves the model's robustness, it does not entirely resolve the adversarial attack problem. This statement has been shown in Jia [33] and William's studies [75]. Both studies show that data augmentation is effective unless an adversary uses some adversarial example that does not exist in the combined training set.

#### 2.6.1.2 Model Regularization

Some other adversarial attack defences try to add the crafted samples into the model's parameters instead of extending the training set. They implement the generated adversarial examples as the regularizer. The learning objective of the model after adding the regularizer is shown in Equation 2.2.

$$min(J(f(x), y) + \lambda J(f(x'), y)) \tag{2.2}$$

where $\lambda$ is a hyperparameter.

Miyato *et al.* [48] introduced a model regularization adversarial training by using a linear approximation. This study was the baseline of multiple approaches in this area, and many researchers extended their work. The authors in [60] applied the concept of Miyato's research into the LSTM. They leverage FGSM to include the adversarial training as a regulator to the model's objective. They proposed a direction vector to resolve the interpretability of the crafted text. This approach perturbs the word embedding to generate a valid word in the embedding space. The researchers in [77] added noise to the pre-trained embedding model to evaluate their defence in the relation extraction task, but the central core of their study has been adopted from [48].

#### 2.6.1.3 Robust Optimisation

Madry *et al.* [42] proposed a deep neural network model learning that includes attack and defence simultaneously. They considered the learning process as a robust optimization with the min-max formulation. They have an inner non-concave as maximization problem to perform the attack. Also, they have an outer non-convex as the minimization problem for the defence part. The study confirmed the successfulness of training a DNN model that is robust against adversarial attacks in the image area.

Al-Dujili *et al.* [1] proposed a malware detection DNN that learned based on the mentioned idea. The main contribution of their study is that it can handle discrete data. They form the learning objective in Equation 2.3.

$$\theta^* = \arg\min_{\theta} E_{(x,y)\sim D}[\max_{x' \in S(x)} L(\theta, x', y)] \tag{2.3}$$

Where $D$ denotes the distribution of data, $y$ is the groundtruth label, and $\theta$ is the learnable parameters, $S(x)$ denotes the set of binary indicator vectors showing the

functionality of malware $x$, $L$ indicates the classification model's loss function.

### 2.6.2 Distillation

In the literature, distillation as another possible defence against adversarial examples has been proposed by Papernot *et al.* [54]. The idea behind this method is to use the softmax output of the DNN to train another DNN. These two DNNs have the same structure. It is worth mentioning that the softmax out of the first DNN is the class probabilities in the classification task. In distillation methods, the raw output of the softmax layer has not been used. A temperature parameter $T$ has been used to modify the softmax of the first DNN. They proved that the model sensitivity to minor disorders had been reduced when they used high T. The equation 2.4 shows the formula for modifying the softmax output.

$$q_i = \frac{exp(z_i/T)}{\sum_k exp(z_k/T)} \tag{2.4}$$

where $z_i$ donates the softmax layer's input and the level of knowledge distillation controls by $T$.

The authors of [26] utilize distillation defence with a high temperature T for DNNs that work on discrete data. They made a comparison of the robustness of the model with two proposed defences. They also applied an augmentation method on the original training set to improve the robustness of their model. They found that adversarial training builds a more robust model than distillation.

## 2.7 Concluding Remarks

Among all the topics under machine learning and deep learning, security attracts lots of attention. Many researchers have been conducted on the security of machine learning in the past few years. Although many studies have been done to

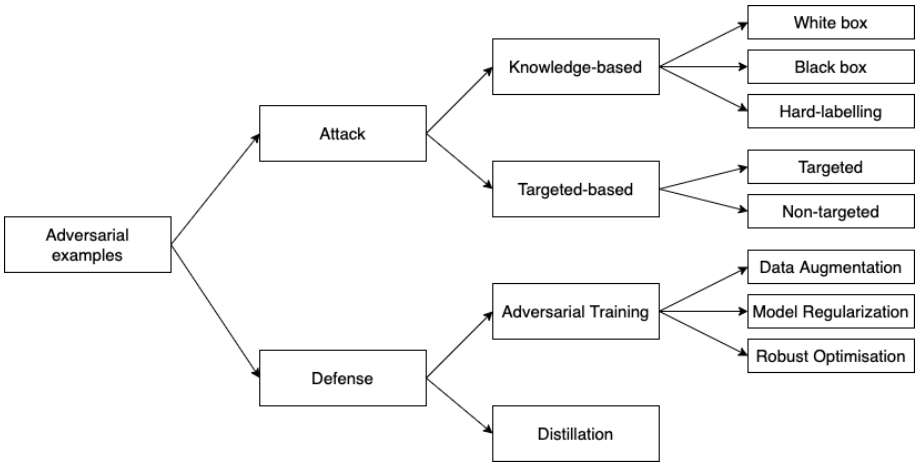



Figure 2.5: Classification of adversarial attack and defenses.

improve the robustness of deep learning models, they could not train a fully robust model against adversarial attacks. Moreover, they could not explain this happening comprehensively.

Developing new adversarial attacks and defences has gotten much attraction from researchers, but their presence and prevalence are unclear yet. Ilyas *et al.* [30] tried to explain the primary source of this happening. They demonstrated that some features in the data could cause a model to be vulnerable to an adversarial example. These features are acquired from the data distribution's pattern and have two main characteristics. First, they are very beneficial for the model's predictivity. Second, they are impenetrable to humans.

The existing adversarial attacks can be categorized based on two features: available information and targeted class. The adversary's knowledge has classified attacks into white-box, black-box, and hard-labeling classes. In the literature, adversarial attacks are categorized into targeted and non-targeted groups. Figure 2.5 shows the classification of adversarial attacks based on knowledge and target. In addition, it organizes the existing defences that have been proposed in the literature.

Wang *et al.* [73] summarized the methodology of the adversarial attacks in the textual area, character-level, word-level, and sentence-level. In addition, they show

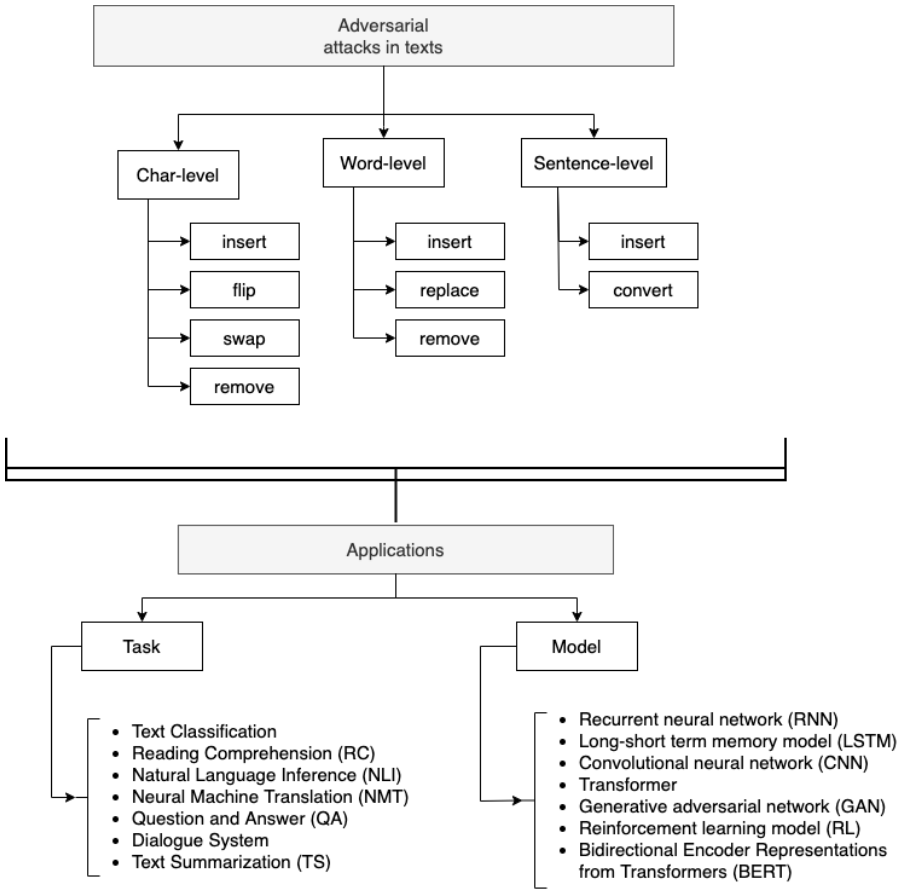



Figure 2.6: Classification of adversarial texts. [73]

the available approaches in each of these methods. Also, they demonstrate tasks and deep neural network models that those attacks have targeted. Figure 2.6 depicts Wang's summarization of the taxonomy of adversarial text attacks.

Table 2.1 shows the most important of the existing adversarial text attacks that have been proposed in recent years. It provides the characteristics like the white-box/black-box/hard-labeling, Targeted and Non-targeted models that they attacked, and the metric used. Some of the frameworks that proposed more than one attack approach that is highlighted in Table 2.1. As Table 2.1 presents, the distribution of non-targeted attacks is higher than the targeted one. The amount of information that each type of attack needs is the primary reason for this distribution. Table 2.1 also shows that LSTM and CNN are the two most favourite targets among deep

Table 2.1: Attributes of adversarial attackers in textual DNNs

| Method | White/Black box/ Hard labeling | Targeted/Non-targeted | Model | Metric |
|---|---|---|---|---|
| TextBugger [39] | Both white and black box | Non-targeted | LR, char-CNN, CNN | multiple metrics |
| Papernot *et al.* [53] | White box | Non-targeted | LSTM | |
| DeepWordBug [19] | Black box | Non-targeted | LSTM, char-CNN | Edit Distance |
| Alzantot *et al.* [2] | Black box | Targeted | LSTM, RNN | Euclidean Distance |
| HotFlip [17] | White box | Targeted | CNN, char-CNN, LSTM | Consine Similarity |
| iAdv-Text [60] | White box | Non-targeted | LSTM | Consine Similarity |
| Samanta *et al.* [59] | White box | Non-targeted | CNN | |
| Gong *et al.* [22] | White box | Targeted | CNN | Word Mover Distance(WMD) |
| Textbugger [39] | Both white and black box | Non-targeted | LSTM, CNN | |
| Maheshwary *et al.* [43] | Hard Labeling | Non-targeted | CNN, LSTM | |
| Wallace *et al.* [68] | Hard labeling | Non-targeted | LSTM | |
| Jia *et al.* [33] | Black box | Non-targeted | CNN, char-CNN | |

neural networks. We can find the reason for this tendency among the performance of these two models in textual tasks.

One of the critical aspects of adversarial example attacks is transferability. Transferability is the ratio of generated samples for a model that can fool another machine learning model. Szegedy *et al.*[64] stated that transferability between two models with different architecture and training sets is possible. This property able an adversary to attack a model and disrupt the results without having any information about the architecture or having access to its training set.

There are multiple definitions for the transferability of adversarial examples. Tramer *et al.* [66] measure the similarity of decision boundary of various models to describe this property. Based on their explanation, Moosavi *et al.* [49] proposed a method for generating universal adversarial examples. They suggested a universal perturbation that can be applied to any sample to turn into an adversarial example that can fool any model. Preventing adversarial examples from transferring to another model and measuring their transferability has remained an open problem even though many studies have done great work on this topic.

# Chapter 3

# Proposed System Design

## 3.1 Overview

This research proposes a new framework named EQFooler for attacking textual deep neural networks trained for text classification. The core of the proposed framework is based on extracting keywords from a text document. Well-known challenges in generating adversarial examples for text domains will be addressed. These challenges include discrete input space and having semantically correct samples. We propose an approach for creating adversarial text examples that need fewer queries compared to the other methods.

## 3.2 Problem Definition

In this section, we will formalize the problem of finding an adversarial example for a textual deep neural network efficiently in the case of black-box, word-level and non-targeted attacks. First, generating an adversarial example will be defined, and after that, we will explain the attack's efficiency to formulate the problem.

In any adversarial attacks scenario, there is a pre-trained model $F : x_i \implies y_i$ that links an input $x_i$ to a predicted label $y_i$. This model was learned based on

a corpus of $N$ documents $X = \{x_1, x_2, x_3, ..., x_N\}$ and a set of annotated labels $Y = \{y_1, y_2, y_3, ..., y_N\}$ that are related to each document. The classifier decides to assign a label $y_k$ to an input $x_k$ based on a posterior probability $P$.

$$\arg\max_{y_i \in Y} P(y_i|x_k) = y_k \tag{3.1}$$

An adversarial example $x'$ is a crafted sample made from an original document $x_k$ by adding an imperceptible perturbation $\Delta x_k$, if $F(x_k) \neq F(x')$ and the difference between two samples ($\Delta x$) is negligible by human observation. A similarity function has calculated the difference between original and crafted samples. So, the two conditions can be interpreted as:

$$\arg\max_{y_i \in Y} P(y_i|x') \neq y_k$$

$$Sim(x_k, x') \leq \epsilon$$

The similarity function in natural language processing usually refers to the grammatic and semantic similarity.

Consider $x = w_1 w_2 w_3 .. w_m$ as a document constructed by a list of words. We can modify any of these words by inserting, removing, changing some characters or words to craft an adversarial example $x' = w'_1 w'_2 w'_3 .. w'_{m'}$

## 3.3 Threat model

An adversarial example attack is when an adversary tries to induce a training model to return a wrong label for specific input. In other words, an adversary attempts to convert a single document $x$ into $x'$ by applying a negligible perturbation to fool a trained model in label prediction. All adversarial attack methods have unique characteristics defined by their attributes and features. However, they can be categorized

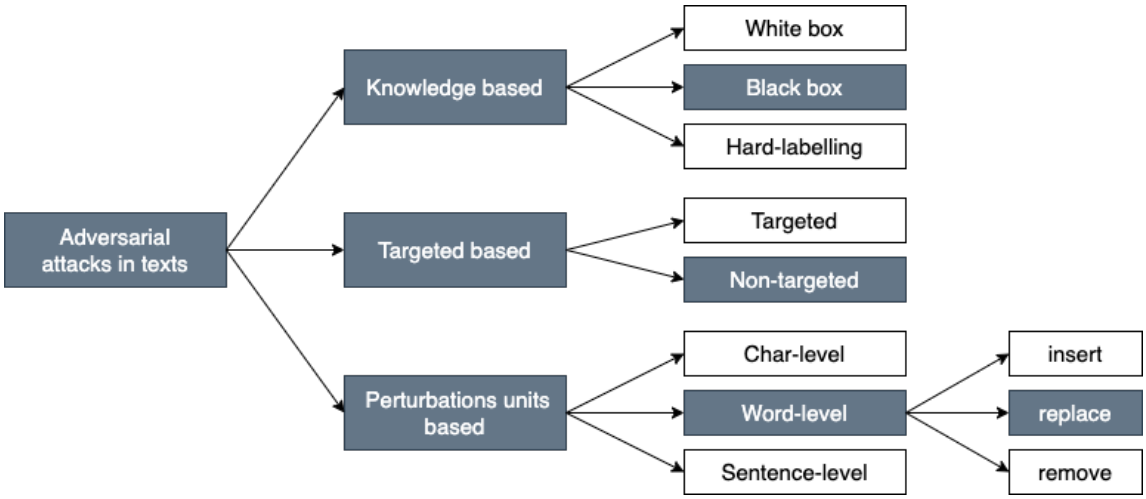



Figure 3.1: Our threat model in the textual adversarial attacks taxonomy

into one of the classes listed in Figure 2.5 and Figure 2.6.

Figure 3.1 depicts the position of the proposed adversarial attack from three different points of view. The proposed method has been placed in the black-box attack category. Without a doubt, black-box attacks are more practical than white-box attacks because they are very similar to real-world situations. A sample, the predicted label, and the confidence score of that prediction are the only information the adversary has in this setting. So, the attacker is trying to find the relation between the input sample and the output label by using the confidence score. Seeing this relationship helps the adversary to identify the model weakness for generating a more potent adversarial example.

The proposed adversarial attack replaces the most influential words of a document with some other phrases, so it is placed on word-level methods from the perturbation granularity aspect. Also, from the targeted-based point of view, the proposed method is categorized as a non-targeted attack. In this attack, we try to decrease the confidence score of classification.

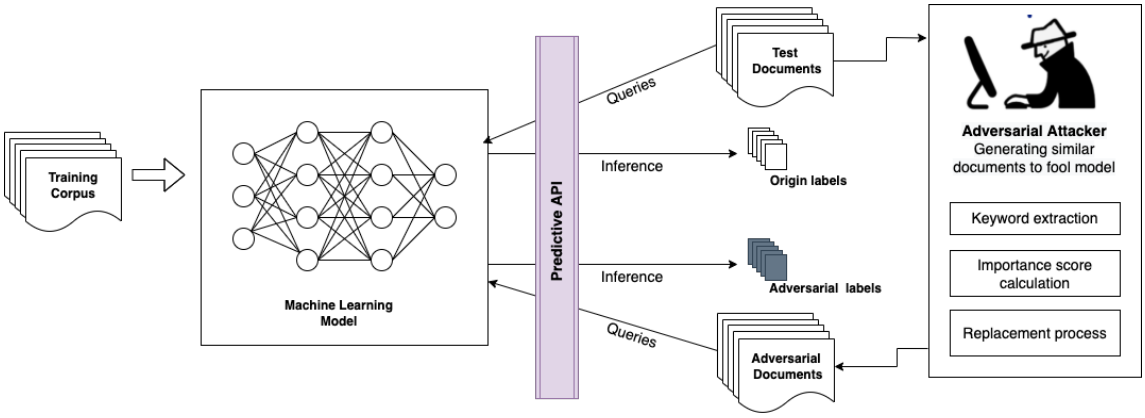



Figure 3.2: workflow of the proposed adversarial attack framework

## 3.4 Proposed Method

This research presents EQFooler as a framework to generate an efficient adversarial example for deep neural networks used in text classification tasks. Figure 3.2 shows the workflow of this framework. The proposed method consists of three main sections for targeting a pre-train model: keyword extraction, importance score calculation, and replacement process. The proposed method gets test documents and their inference labels and, through its three components, converts them to adversarial documents that can fool the pre-train model. In the following subsections, we explain each of these components comprehensively.

Algorithm 1 captures the proposed framework and shows how these components are connected and work with each other. The keyword extraction component has been shown in line 2 as a function call. Lines 3-5 denote the important score calculation. Finally, lines 9-19 represent the replacement process and validation of the modified document, including the syntactic and semantic inspection.

### 3.4.1 Keyword Extraction

Keyword extraction uses machine learning and natural language processing (NLP) techniques to understand and analyze a text document. It is used to identify key-

**Algorithm 1:** The proposed adversarial attack framework

**input :** x: a given example
**output:** x': an adversarial example if generated

$K = Extractkeywords(x)$
**for** *each word* $k_i$ *in* $K$ **do**
    Calculate the importance score $I_{k_i}$
**end**
$K_{Sorted} = Sortkeywords(K, model)$
$score, label = model.predict(x)$
threshold = 0
**for** *each* $k_i$ *in* $K_{Sorted}$ **do**
    $w = findBestSimilarWord(k_i)$
    $x'$ = replace $k_i$ with $w$ in $x$
    **if** $ValidateSyntactic\&Semantic(x, x') == True$ **then**
        $score', label' = model.predict(x')$

        **if** $label \neq label'$ **then**
            **return** $x'$
        **end**
        $x = x'$
    **end**
**end**

words from all kinds of text like business reports, social media content, online blogs and forums, news articles, and more. The output of this technique is a sorted list of words and phrases along with their scores in descending order.

Having an approach for finding the most influential words independent of the targeted model is one of the main contributions of the algorithms. In the baseline, the authors calculated the importance score of all the words in a document without any order. However, we use keyword detection methods to identify the most important words and expressions in a text.

When a keyword extraction technique is applied to a given document $x$, the output is sorted list $K = \{k_1, k_2, k_3, ..., k_n\}$, where $k_i$ is the $i^{th}$ important word or phrase in that document. The main idea is that these keywords are more influential on the output of a deep neural network that is predicted label than the other words. In the next step, we only calculate the importance score of these keywords rather than all the words in a document.

Many keyword extraction techniques are used to solve real-world problems. It is necessary to study which one of these algorithms has better performance in the proposed attack. Some state-of-the-art keyword extraction techniques have been presented in Table 3.1. Also, features such as supervised/unsupervised, graph-based, stopword list, POS, stemming and needing extra corpus for those keyword extraction methods are summarized in Table 3.1. We select unsupervised methods that do not require an additional corpus for running among these keyword extraction methods. So, we examine eight keyword extraction techniques to determine the best option for generating an adversarial example.

Table 3.1: Summary features of different keyword extraction methods.

| Keyword Extractor | ref. | Unsupervised | Graph-based | language Dependence | | | external Corpus |
|---|---|---|---|---|---|---|---|
| | | | | Stop word list | POS | Stemming | |
| YAKE | [9] | ✓ | | ✓ | | | |
| RAKE | [57] | ✓ | | ✓ | | | |
| TextRank | [44] | ✓ | ✓ | | ✓ | | |
| SingleRank | [69] | ✓ | ✓ | ✓ | ✓ | | |
| TopicRank | [8] | ✓ | ✓ | ✓ | ✓ | | |
| PositionRank | [18] | ✓ | ✓ | ✓ | ✓ | | |
| MultipartiteRank | [7] | ✓ | ✓ | ✓ | ✓ | ✓ | |
| RakUN | [61] | ✓ | ✓ | | | ✓ | |
| TF.IDF | [36] | ✓ | | | | | ✓ |
| TopicalPageRank | [41] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ExpandRank | [70] | ✓ | ✓ | | ✓ | | ✓ |
| KEA | [76] | | | | | ✓ | ✓ |

- **Yake:** Campos *et al.* [9] proposed a lightweight unsupervised automatic keyword extraction method named YAKE that supports texts of different sizes, domains, or languages. To select the most important keywords of a text, Yake rests on statistical text features extracted from single documents without training on a particular set of documents. It depends neither on dictionaries, external-corpus, size of the text, language, nor domain. Their proposed method has five main steps: (1) text pre-processing and candidate term identification; (2) feature extraction; (3) computing term score; (4) n-gram generation and computing candidate keyword score; and (5) data deduplication and ranking. Detailed explanations of each of these steps are provided in [9]. The authors

compared Yake against 11 state-of-the-art keywords extractors, applying them to twenty datasets. The results indicate that Yake can significantly outperform other methods on documents with different sizes and languages.

- **TextRank:** It is an unsupervised and graph-based ranking model for text processing that can also extract the most influential keywords [44]. TextRank revolves around representing the concerned lexical unit in a graph and later using the famous PageRank[1] algorithm to rank those chunks. To form the graph, each word in the vocabulary serves as a vertex for the graph and edge shows the connection between two words. There is a connection between two words $w_i$ and $w_j$ if the words co-occur within a window of $N$ words, $W_N$, in the processed documents. These edges are used to obtain an unweighted undirected graph, and then PageRank algorithm will apply to rank the words. The idea of ranking words originated from PageRank algorithm of Google with a basic idea of voting or recommendations, which means the graph's edges are considered votes. The more votes a node gets, the higher its score, and the more critical it is.

- **SingleRank:** It is a keyword extraction method that applies a clustering algorithm to a document to obtain appropriate clusters. Then it uses the graph-based ranking algorithm for keyphrase extraction within each group used [69]. Similar to TextRank, the SingleRank method tries to rank words based on constructing a graph from the documents where the weights of the edges show the strength of the semantic relationship between two words. But, unlike the TextRank, this algorithm builds a weighted graph to rank words of documents. In this graph, the value of the weights can be calculated by the number of times the two words appeared in the window of $W_N$ at the same time.

[1] https://en.wikipedia.org/wiki/PageRank

- **TopicRank:** Bougouin *et al.* proposed a keyphrase extraction method named TopicRank [8] that is similar to TextRank method. It uses a graph-based ranking model to assign a significance score to each topic and selects a candidate from each top-ranked group [8]. This algorithm firstly divides the document into multiple topics by applying hierarchical agglomerative clustering, then it uses PageRank to score each topic. It forms the keyphrases set by selecting the first candidate word from each top-ranked topic.

- **PositionRank:** Florescu et al. [18] proposed an unsupervised model for keyphrase extraction from educational documents that combines information from all positions of a word's occurrences into a biased PageRank. This algorithm introduces location information based on SingleRank according to the idea that the earlier the candidate words appear in a document, the more important they are. The proposed graph structure called a multipartite graph represents documents as tightly connected sets of topic-related candidates. In this graph, nodes are keyphrase candidates, and they are connected if they belong to different topics. This representation allows the ranking algorithm to fully use the mutually reinforcing relationship between topics and candidates.

- **MultipartiteRank:** In 2018, Boudin proposed an unsupervised keyword extraction method named MultipartiteRank [7] that builds a multipartite graph structure topical information to represent the keyphrases of a document. This model represents topics and keyphrase candidates in a single graph and exploits their mutually reinforcing relationship to improve candidate ranking.

- **Rapid Automatic Keyword Extraction (RAKE):** Stuart Rose *et al.* [57] introduced is a domain-independent keyword extraction algorithm named Rapid Automatic Keyword Extraction (RAKE)[2]. Their model determines key-

[2]https://pypi.org/project/rake-nltk/

words in a document based on word frequency and its co-occurrence with other words. The main idea of this method is that keywords frequently contain multiple words but rarely contain punctuation, stop words, or other words with minimum lexical meaning. RAKE splits the text into a list of words known as Content Words by removing stop words from the same list. Then, it builds the word co-occurrence graph for content words. Each row in the adjacency matrix of this graph shows the number of times a given content word co-occurs with another content word in candidate key phrases. By considering such a graph, the method calculates three main matrices: word degree, word frequency, and the ratio of the degree to frequency. Then, it considers the candidate key phrases, and the score for the new keyword is the sum of its member keyword scores. A new candidate keyword is then created as a combination of those keywords and their interior stop words.

- **RakUN[3]:** It is a keyword detection algorithm that employs graph-based language representations for efficient denoising and keyword detection[61]. The key ideas of the RaKUn method are transforming texts to graphs, then pruning graphs based on token similarity by introducing meta vertex aggregation, and finally, ranking nodes of graphs to identify the keywords. To rank nodes, the authors exploited the "load centrality" as a fast measure to estimate the importance of nodes in the graph. The load centrality of a node indicates the fraction of all shortest paths that pass through that node. After ranking the words based on load centrality metric, the keyword extraction is extended from unigrams to bigram and threegram keywords extraction based on load centrality scores computed for considered tokens.

[3]https://github.com/SkBlaz/rakun

### 3.4.2 Importance score calculation

The importance score of a word is equal to its effect on the prediction process. Measuring this effect in a white-box scenario is trivial because the gradients of the model are available. However, in a black-box setting, such information is not accessible. So, we create a mechanism to calculate the importance score of each extracted keyword based on what Jin *et al.* have done in [34].

The importance score of a keyword $I_{k_i}$ calculates the impact of that keyword in the prediction process. To calculate the importance score keyword $k_i$, first, we query the model with document $x = \{w_1, w_2, w_3, ..., k_i, ..., w_n\}$ and store its label $y$ and confidence score $C_y$. Then, we find the selected keyword in the document and remove it to have a new copy, $x' = \{w_1, w_2, w_3, ..., w_n\}$. After that, we send the modified document, $x'$, to the model and get the inference label, $y'$, along with its confidence score, $C_{y'}$. We consider the difference between these two confidence scores as the importance score of the specific keyword, $I_{k_i}$. Equation 3.2 shows this calculation mathematically.

$$I_{k_i} = \begin{cases} C_y(x) - C_y(x') & \text{if } y = y' \\ (C_y(x) - C_y(x')) - (C_{y'}(x) - C_{y'}(x')) & \text{if } y \neq y' \end{cases} \tag{3.2}$$

In Equation 3.2, $y'$ represents the predicted label for the modified document, $x'$. Also, $C_y(.)$, and $C_{y'}(.)$ denote the confidence score of label $y$, and $y'$, respectively. Equation 3.2 has another interpretation in which the $I_{k_i}$ calculates the prediction change in the case of removing $k_i$.

### 3.4.3 Replacement process

In the replacement process, we have two central parts, finding the best candidate for replacement and validating the modified document from a syntactic and semantic

perspective. However, before running both of the above parts, we sort the keywords based on their importance score.

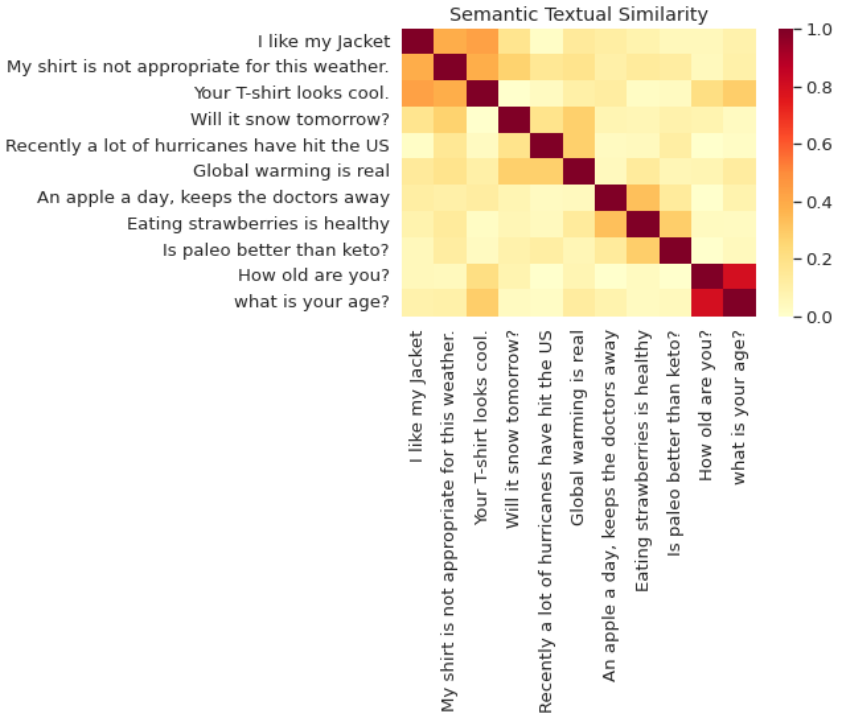
```
# Clothes
"I like my Jacket",
"My shirt is not appropriate for this weather.",
"Your T-shirt looks cool.",

# Weather
"Will it snow tomorrow?",
"Recently a lot of hurricanes have hit the US",
"Global warming is real",

# Food and health
"An apple a day, keeps the doctors away",
"Eating strawberries is healthy",
"Is paleo better than keto?",

# Asking about age
"How old are you?",
"what is your age?",
```

(a) Sentences and categories




(b) Similarity matrix

Figure 3.3: Sentence similarity scores using embeddings from the universal sentence encoder

In the first part, we are looking for a candidate close to the original keyword that changes the predicted label or decreases the prediction's confidence score. We used

word embeddings from [50], which has promising results in synonym finding in the word embedding area. We pick the top $N$ synonymous words close in measure of cosine distance to the keyword in the word embedding space for each word. After this part, we drop all the candidates with different part-of-speech (POS) tags than the keyword.

In the second part, we evaluate the generated text syntactically and semantically. We use POS tagging as an approach to keep the generated text grammatically perfect. On the other hand, we use Universal Sentence Encoder (USE) [10] as a semantic measurement of generated text.

**Universal Sentence Encoder (USE)** is a model for sentence encoding that is used in multiple NLP tasks. It converts a sentence into an embedding vector. It has two encoders with different purposes. One of them is built upon the transformers architecture that provides high accuracy, notable complexity, and increased resource cost. The other encoder is more efficient, although it produces a lower accuracy. It computes the cosine similarity of two embedding vectors, $u$ and $v$, related to the original sentences for semantic similarity calculation of two sentences. After that, it leverages arccos to transform the cosine similarity into an angular distance. Equation 3.3 shows the formula of this conversion [10]. Figure 3.3 shows the similarity matrix between 11 sentences in four different categories that their sentences are semantically related to each other.

$$Sim(u, v) = (1 - accros(\frac{u \,.\, v}{||u||\ ||v||})/\pi) \tag{3.3}$$

In this research, we select those generated samples whose similarity to the original document based on the USE result is higher than a threshold and drop the rest of the candidates. If we have multiple candidates on our list, we pick the candidate that changes the predicted label. If none of the generated adversarial examples changes

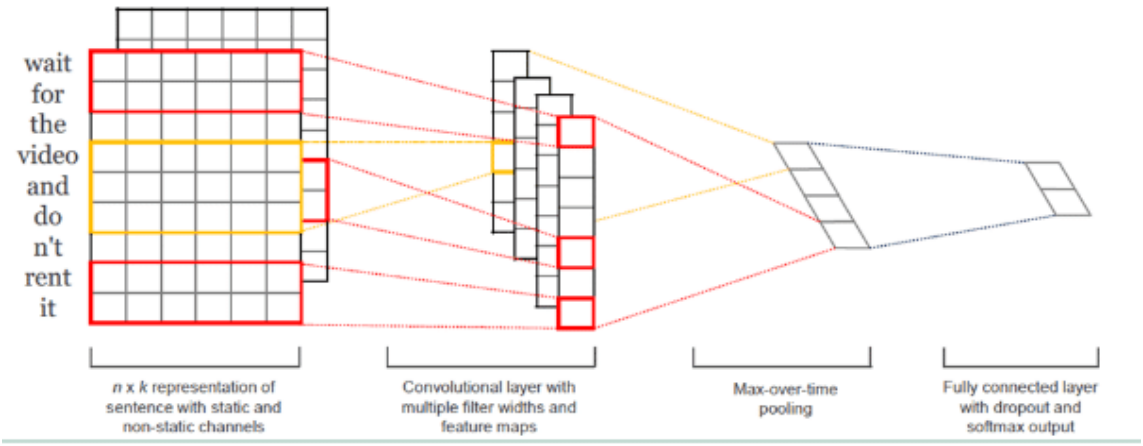



Figure 3.4: WordCNN model architecture with two channels for an example sentence, [37].

the output class, we select the one that drastically decreases the confidence score.

## 3.5 Target models

In our framework, we target two state-of-the-art sentence classification models: word-based convolutional neural network (WordCNN) [37] and word-based long-short term memory (WordLSTM) [28]. There are two main reasons for selecting these two DNN models. First, these two models have an outstanding performance on the four benchmark datasets that we choose, and their stunning results have been shown in many different studies. Also, to the best of our knowledge, they are the state-of-the-art DNN models for text classification tasks. Second, since most researchers use these two models as their targets and want to compare the proposed attack framework to them and have a fair comparison setting, we choose WordCNN and WordLSTM as target models for the proposed framework.

**WordCNN:** For the WordCNN model, we used three window sizes of 3, 4, and 5, and 100 filters for each window size with a dropout of 0:3. Figure 3.4 shows the WordCNN model architecture which is a slight variant of the CNN architecture proposed by Collobert *et al.* [13].

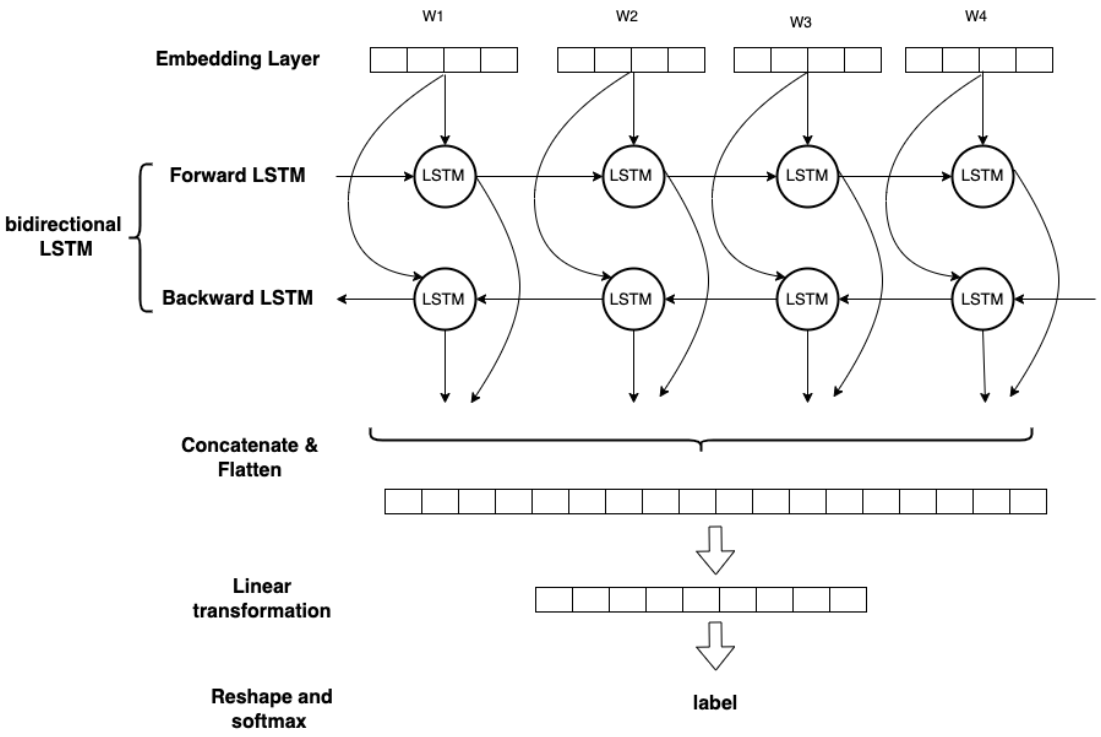



Figure 3.5: An illustration of the BiLSTM architecture.

**WordLSTM:** Our WordLSTM model consists of a 1-layer bidirectional LSTM (BiLSTM) with 150 hidden units and a dropout of 0:3. Figure 3.5 provides an illustration of the BiLSTM architecture, where each bidirectional LSTM unit consists of two LSTM units, namely forward and backward LSTM cells.

For both models, we utilized the 200-dimensional Glove word embeddings pre-trained on 6B tokens from Wikipedia and Gigawords [55].

## 3.6 Concluding Remarks

In real word problems, it is essential to have an efficient query model to attack a deep learning model because of the time complexity of generating adversarial examples, especially in black-box settings. Besides, the generated adversarial example should be similar to the original samples as much as possible. An adversary should consider additional constraints on adversarial documents in the text-domain. The crafted text

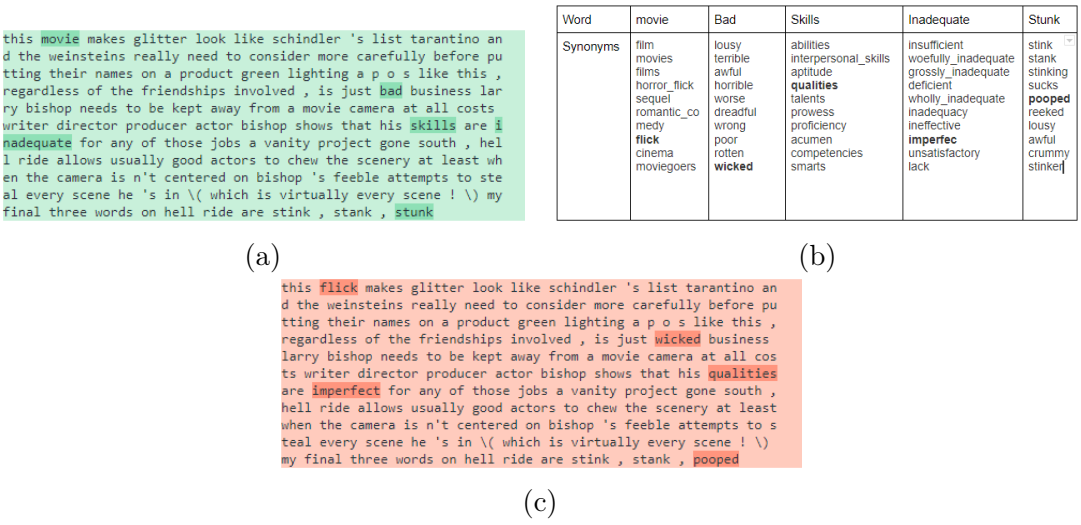



Figure 3.6: An example of generating an adversarial example by EQFooler

should be grammatically correct and semantically similar to the original document. This section proposes an efficient query adversarial attack named EQFooler, which leverages keyword extraction to find the best word candidates for perturbation. So, the EQFooler has three main steps: keyword extraction, importance score calculation, and replacement process. Additionally, we consider two state-of-the-art text classifiers named WordCNN and WordLSTM as the target model. We suggest using the confidence score of the target model to find the importance score of the extracted keywords and then check the quality validation of the crafted document based on the language grammar rules and the semantic similarity with the original document. Figure 3.6 shows an example of how EQFooler generates an adversarial document through these steps: 1) keyword extraction on a given text, 2) finding the closest ten synonyms for each keyword and selecting the best substitution, and 3) replacing keywords with their best synonyms and generating the adversarial document.

According to adversarial taxonomy, the proposed attack model lies in the black-box category because it has access only to the model's confidence scores and predicted labels output and lies on the non-targeted type because it tries to decrease the confidence score of the prediction. Alternatively, as it replaces the most influential

keywords, it is a word-level method in terms of the granularity of the modification unit.

# Chapter 4

# Experiments and Evaluations

## 4.1 Overview

This chapter is organized into three sections. First, this chapter presents the details of the designed experiments that help us verify the effectiveness of our proposed adversarial attack framework. We will present the benchmark datasets, evaluation metrics, the required settings for training the deep learning model on text classification, along with four different variants of EQFooler with different settings to generate adversarial examples.

In the second section, we focus on the experimental results of different variants of the EQFooler model and analyze their behaviour while applying them to the selected benchmarks. The results in this category of experiments demonstrate the effectiveness of the proposed method and emphasize the power of calculating keyword scores in the replacement process of our framework.

In the third section, we compare the best variants of EQFooler models with the baseline in terms of different evaluation metrics to show the effectiveness of the generated adversarial documents.

## 4.2 Design Experiments

### 4.2.1 Benchmark Datasets

We will apply the proposed attack framework to different text classification tasks with two Deep Neural Networks (LSTM and CNN) to assess the efficacy of the designed method. For these classification tasks, we chose the four datasets described bellow (Table 4.1 summarizes their characters).

- **IMDB:** This dataset is crawled from IMDB website that provides lots of information about films and TV series. Each record in this dataset includes a review of a movie and a label that indicates its sentiment polarity, and it is usually used for sentiment classification. The IMDB dataset contains 50,000 positive and negative reviews. The average length of each records is nearly 200 words.[1]

- **Yelp:** The Yelp dataset was built in 2015 for the Yelp Dataset Challenge by Zang *et al.* [81]. This dataset contains 560K samples that have polarity reviews of texts. There are two classification tasks for the original dataset: predicting the number of stars for each user and polarity detection by considering stars 1 and 2 negative and 3 and 4 positive. Since our usage lies on the second task, we convert the labels to negative and positive before training the model.

- **MR:** Pang and Lee first introduced the MR dataset in 2005 [51]. They built a dataset of sentence-level sentiment classification on positive and negative movie reviews. We use this dataset as one of our targets, with 90% and 10% for training and testing, respectively.

- **AG's News:** AG's News is a sentence-level topic classification dataset. It has four news topics including World, Sports, Business, and Science/Technology.

[1] https://datasets.imdbws.com/

Table 4.1: Properties of text classification datasets

| Dataset | Train | Test | Avg Length |
|---|---|---|---|
| IMBD | 25K | 25K | 215 |
| Yelp | 560K | 38K | 152 |
| MR | 9K | 1K | 20 |
| AG | 120K | 7.6K | 43 |

Each news has a separate title, description, and body. We concatenate all these parts to build a joint content.

### 4.2.2 Evaluation Metrics

To evaluate the attack framework, we will calculate four metrics: Original Accuracy, Adversarial Accuracy, Change Rate, and Number of Queries. A brief description for each evaluation metric is provided below.

- **Original Accuracy:** It describes the production of a model in an original setting without any malicious activities. A higher value shows that a model has a better performance. Equation 4.1 formulates this metric.

$$Acc_{org} = \frac{|\{(x,y) \mid (x,y) \in D_{test} \ \ \& \ \ F(x) = y\}|}{|D_{test}|} \tag{4.1}$$

where $F(.)$ represents the model and $D_{test}$ is a part of whole dataset that we consider as a testset. It includes documents, $x$, and their grand truth labels, $y$.

- **Adversarial Accuracy:** This metric shows the performance of the trained model after the adversarial attack happened. The lower value indicates the power of the adversarial attack. It can be calculated by Equation 4.2.

$$Acc_{adv} = \frac{|\{(x,y) \mid (x,y) \in D_{test} \ \& \ F(G_F(x)) = y\}|}{|D_{test}|} \tag{4.2}$$

where $G_F(.)$ denotes the adversarial example generator function regarding model $F$, means $x' = G_F(x)$ where $x$ is an origin document, and $x'$ is the corresponding crafted document that can fool model $F$.

- **Change Rate:** It states the percentage of the original document that has been changed to convert it from a benign sample to an adversarial example. We formulate the change rate in Equation 4.3.

$$CR = \frac{1}{|D'_F|} \sum_{(x,x') \in D'_F} \frac{|\{w|w \in x \ \ \& \ \ w \notin x'\}|}{|\{w|w \in x\}|} \tag{4.3}$$

where $w$ indicates words of a document, $x = w_1, w_2, w_3, ...w_m$, and $D'_F$ shows the modified samples that have fooled a given model, $F$. It can be defined by Equation 4.4.

$$D'_F = \{(x, x', y)|(x, y) \in D_{test} \ \ \& \ \ x' = G_F(x) \ \ \& \ \ F(x) \neq F(x') \ \ \& \ \ F(x) = y\} \tag{4.4}$$

- **Number of Queries:** It indicates the number of queries that the adversarial attack method performs to fool the model and achieve the better adversarial accuracy result. Equation 4.5 illustrates the upper and lower bound of this metric, $|Q|$. In the worst case, the modified document would be obtained after trying all $s$ synonyms of all extracted keywords, $K = k_1, k_2, k_3, ...$, and in the best case, the adversarial example would be found by replacing the first keyword with one of its synonyms.

$$\frac{1}{|D'_F|} \sum_{(x,x') \in D'_F} (|K| + 1 + s) \leq |Q| \leq \frac{1}{|D'_F|} \sum_{(x,x') \in D'_F} (|K| + 1 + s \times |K|) \tag{4.5}$$

- **Transferability:** This metric measures the potential of crafted adversarial

examples based on a model, $F_1$ to fool another unknown model, $F_2$. Equation 4.6 explains it mathematically. Please note that $T(.,.)$ is an asymmetric metric, i.e. $T(F_1, F_2) \neq T(F_2, F_1)$.

$$T(F_1, F_2) = \frac{|\{(x, x', y)|(x, x', y) \in D'_{F_1} \ \ \& \ \ F_2(x) \neq F_2(x') \ \ \& \ \ F_2(x) = y\}|}{|D'_{F_1}|} \tag{4.6}$$

### 4.2.3 Settings

This section describes all the parameters we have used while running the proposed framework. In addition to those parameters that have been discussed in Section 3.5, we set the training batch size to 128 for both WordCNN and WordLSTM. The extracted keywords from each document are 100 to have a powerful adversarial examples and an efficient attack method. We conduct a gap analysis to determine the best value for this parameter, as discussed in Subsection 4.3.2. The two other parameters that influence the performance of the proposed framework are the number of synonyms that we pick from the word embedding for each extracted keyword and the similarity threshold for acceptance of the crafted document. We use the mentioned amount for these parameters in the baseline method, which are 0.7 for the similarity threshold and 50 for the number of synonyms. We chose these numbers because we want to have a fair comparison between the proposed framework and the baseline.

We consider the following four different variants of the proposed framework based on their keyword extraction method and the strategy for ranking replacement candidates.

- **EQFooler-Rake-KS:** is the variant of the proposed model that utilizes the Rake keyword extractor and finds the candidate words for replacement based on their scores returned by the Rake method.

Table 4.2: Comparison of different keyword extractors in EQFooler, keyword extractors' scores, using WordCNN, IMDB dataset.

| Keyword Extractor | Original Accuracy | Adv Accuracy | Number of Queries | Change Rate |
|---|---|---|---|---|
| yake | 89.2 | 0.1 | 632.7 | 8.07 |
| multipartite_rank | 89.2 | 4.2 | 627.5 | 7.2 |
| position rank | 89.2 | 10.9 | 589.1 | 6.3 |
| single rank | 89.2 | 3.5 | 552.5 | 6.7 |
| text rank | 89.2 | 42.4 | 439 | 3.85 |
| topic rank | 89.2 | 7.1 | 610.4 | 6.96 |
| rake | 89.2 | 0.1 | 468.2 | 6.6 |
| RaKUn | 89.2 | 3.9 | 637.9 | 7.33 |

- **EQFooler-Yake-KS:** is the variant of the proposed model that utilizes the Yake keyword extractor and finds the candidate words for replacement based on their scores returned by the Yake method.

- **EQFooler-Rake-MS:** is the variant of the proposed model that utilizes the Rake keyword extractor and finds the candidate words for replacement based on the important scores (see Subsection 3.4.2).

- **EQFooler-Yake-MS:** is the variant of the proposed model that utilizes the Yake keyword extractor and finds the candidate words for replacement based on the importance scores (see Subsection 3.4.2).

## 4.3 Experimental Results

### 4.3.1 Keyword Extraction Techniques

The first element of the proposed adversarial attack framework is keyword extraction, which helps the system generate adversarial examples efficiently. To achieve this, we used eight well-known keyword extraction techniques (see chapter 3).
As we described in subsection 4.2.3, we have four variants in the proposed framework. Table 4.2 and Table 4.3 summarize the effectiveness of keyword extraction methods with two different scoring methods, keyword scoring (KS) and target's model scoring

Table 4.3: Comparison of different keyword extractors in EQFooler, target model's scores, attacking WordCNN, IMDB Dataset.

| Keyword Extractor | Original Accuracy | Adv Accuracy | Number of Queries | Change Rate |
|---|---|---|---|---|
| yake | 89.2 | 0 | 464.1 | 3.7 |
| multipartite_rank | 89.2 | 4.2 | 446 | 4.46 |
| position rank | 89.2 | 10.7 | 530.1 | 5.22 |
| single rank | 89.2 | 3.6 | 443.8 | 4.521 |
| text rank | 89.2 | 42.4 | 430.5 | 3.55 |
| topic rank | 89.2 | 6.8 | 482.7 | 4.83 |
| rake | 89.2 | 0.1 | 386.3 | 3.63 |
| RaKUn | 89.2 | 3.2 | 442.5 | 4.42 |

(MS), respectively. In all experiments, we attacked the WordCNN model that was trained on the IMDB dataset. As shown in Table 4.2 and Table 4.3, Yake and Rake keyword extractors have better performance than other keyword extraction methods in terms of adversarial accuracy. For instance, both scoring methods could decrease the model's accuracy to below 2%. However, when we use the target's model scoring, the change rate drops by more than 60% compared to keyword extractor scoring. Also, the target's model scoring variants need fewer queries for generating adversarial examples than other variants.

Similarly, Table 4.4 and Table 4.5 show the adversarial attack results while attacking WordLSTM trained on IMDB by the same variants of the proposed framework. We also see the same behaviour of keyword extraction methods in this setting. Although the best performance of keyword extraction methods in terms of change rate in the WordLSTM model belongs to the textRank, its adversarial accuracy is 64, which means this version of the proposed method cannot fool the model correctly.

### 4.3.2 Gap Analysis

We design a gap analysis on this parameter to evaluate the effect of the number of extracted keywords in the first phase of the proposed framework. We find the best value of the number of extracted keywords during this analysis. In this analysis, we vary the number of extracted keywords from 10 to the length of the documents. We

Table 4.4: Comparison of different keyword extractors in EQFooler, keyword extractors' scores, using WordLSTM, IMDB Dataset.

| Keyword Extractor | Original Accuracy | Adv Accuracy | Number of Queries | Change Rate |
|---|---|---|---|---|
| yake | 89.8 | 1.2 | 1312 | 15.2 |
| multipartite_rank | 89.8 | 11.6 | 988.4 | 10.7 |
| position rank | 89.8 | 23.7 | 911.7 | 8.96 |
| single rank | 89.8 | 10.5 | 944.2 | 10.28 |
| text rank | 89.8 | 63.8 | 564.7 | 4.5 |
| topic rank | 89.8 | 18.6 | 939.7 | 18.6 |
| rake | 89.8 | 1.2 | 1000.3 | 12.16 |
| RaKUn | 89.8 | 9.9 | 954.2 | 10.34 |

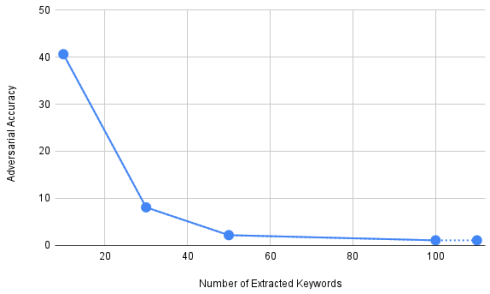
Table 4.5: Comparison of different keyword extractors in EQFooler, target model's scores, using WordLSTM, IMDB Dataset.

| Keyword Extractor | Original Accuracy | Adv Accuracy | Number of Queries | Change Rate |
|---|---|---|---|---|
| yake | 89.8 | 1.9 | 5.958 | 725.1 |
| multipartite_rank | 89.8 | 10.8 | 688.7 | 6.1 |
| position rank | 89.8 | 22.7 | 802.6 | 6.93 |
| single rank | 89.8 | 10 | 681.2 | 6.18 |
| text rank | 89.8 | 64 | 537.4 | 3.82 |
| topic rank | 89.8 | 18.1 | 734.2 | 6.253 |
| rake | 89.8 | 1.1 | 575.3 | 5.56 |
| RaKUn | 89.8 | 9 | 674.3 | 6.06 |

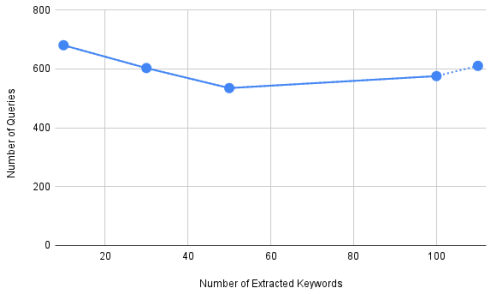
select the IMDB dataset for the experiment because it has the longest records among all the benchmark datasets. We chose Word LSTM as the target model because it is more robust than WordCNN based on tests previously conducted.

Table 4.6 demonstrates the gap analysis results based on adversarial accuracy, number of queries, and change rate. The proposed method has the best performance in terms of adversarial accuracy when we extract the top 100 important words from a document. Although the number of queries needed for $k = 50$ is the lowest amount in Table 4.6, we have better change rate and adversarial accuracy values with different $k$.

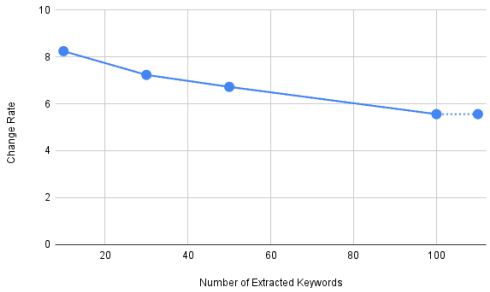
We did not consider the value more than 100 for the $k$ parameter for two reasons. First, based on the Table 4.1, the average document length for two benchmark datasets is lower than 100. Second, the number of queries needed to generate adversarial examples increases when we use values more than 100 for the $k$ parameter.




(a) Adversarial Accuracy with different number of extracted keywords




(b) Number of Queries with different number of extracted keywords




(c) Change Rate with different number of extracted keywords

Figure 4.1: Parameter tuning of $K$, IMDB dataset, using LSTM.

Figure 4.1 depicts these results based on the three metrics separately. By considering all the results, we decided to extract 100 important words from documents with more than 100 words. For those documents with less than 100 words, we returned all their phrases sorted based on the keyword extraction score.

### 4.3.3 Evaluation Results

Different variants of the proposed method have been evaluated in this section. Table 4.7 and Table 4.8 show the evaluation of these variants based on four mentioned metrics when they have trained on four benchmark datasets for two target models, WordLSTM and WordCNN. The promising results in Table 4.7 and Table 4.8 indicate that the proposed method can attack state-of-the-art deep neural networks without considering how they are accurate. The proposed method decreases the

Table 4.6: Gap analysis on number of extracted keywords, IMDB dataset, using LSTM

| Number of Extracted Keywords | Adversarial Accuracy | Number of Queries | Change Rate |
|---|---|---|---|
| Maximum length of document | 1.1 | 610 | 5.56 |
| 100 | 1.1 | 575.3 | 5.56 |
| 50 | 2.2 | 534.6 | 6.72 |
| 30 | 8.1 | 602.5 | 7.23 |
| 10 | 40.6 | 680 | 8.24 |

Table 4.7: The performance of different variants of EQFooler while attacking WordLSTM.

| datasets | models | Original Accuracy | Adv Accuracy | Number of Queries | Change Rate |
|---|---|---|---|---|---|
| IMDB | EQFooler-Yake-KS | 89.8 | 1.2 | 1312 | 15.2 |
| | EQFooler-Rake-KS | 89.8 | 1.2 | 1000.3 | 12.16 |
| | EQFooler-Yake-MS | 89.8 | 1.9 | 725.1 | 5.958 |
| | EQFooler-Rake-MS | 89.8 | 1.1 | 575.3 | 5.56 |
| YELP | EQFooler-Yake-KS | 96 | 6.4 | 1318.7 | 27.7 |
| | EQFooler-Rake-KS | 96 | 9.7 | 910 | 18.6 |
| | EQFooler-Yake-MS | 96 | 6.6 | 700.6 | 11.4 |
| | EQFooler-Rake-MS | 96 | 4.5 | 605 | 11.14 |
| MR | EQFooler-Yake-KS | 80.7 | 4.3 | 180.1 | 23.07 |
| | EQFooler-Rake-KS | 80.7 | 7.4 | 135.3 | 18.25 |
| | EQFooler-Yake-MS | 80.7 | 4.1 | 134.5 | 14.9 |
| | EQFooler-Rake-MS | 80.7 | 4.4 | 117.6 | 14.8 |
| AG | EQFooler-Yake-KS | 91.3 | 7.1 | 601 | 41.06 |
| | EQFooler-Rake-KS | 91.3 | 23.7 | 342.7 | 23.24 |
| | EQFooler-Yake-MS | 91.3 | 13.2 | 392 | 21.67 |
| | EQFooler-Rake-MS | 91.3 | 12.4 | 334.8 | 21.7 |

model's accuracy to below 2% for the IMDB dataset and below 5% for Yelp and MR. The average change rate of the adversarial example is acceptable for all datasets. Table 4.7 and Table 4.8 also show that no matter the dataset, the WordCNN model is more vulnerable than the WordLSTM.

As shown, the average change rate and adversarial accuracy for the WordCNN are lower than the WordLSTM model. Also, the IMDB dataset is a favourite target for our framework since the adversarial accuracy is close to zero with a bit of change rate. Table 4.7 and Table 4.8 offers another insight that, generally, when a model has higher original accuracy, it is harder to decrease its initial accuracy after an adversarial attack.

On average, those variants that use the target's models scoring perform better than

Table 4.8: The performance of different variants of EQFooler while attacking WordCNN.

| datasets | models | Original Accuracy | Adv Accuracy | Number of Queries | Change Rate |
|---|---|---|---|---|---|
| IMDB | EQFooler-Yake-KS | 89.2 | 0.1 | 632.7 | 8.07 |
| | EQFooler-Rake-KS | 89.2 | 0.1 | 468.2 | 6.6 |
| | EQFooler-Yake-MS | 89.2 | 0 | 524.2 | 3.51 |
| | EQFooler-Rake-MS | 89.2 | 0.1 | 386.3 | 3.63 |
| YELP | EQFooler-Yake-KS | 94 | 3.3 | 1008 | 22.55 |
| | EQFooler-Rake-KS | 94 | 5 | 687.3 | 15.4 |
| | EQFooler-Yake-MS | 94 | 4.2 | 560.5 | 9.33 |
| | EQFooler-Rake-MS | 94 | 2.8 | 462.8 | 9.1 |
| MR | EQFooler-Yake-KS | 78 | 3.4 | 158 | 20.6 |
| | EQFooler-Rake-KS | 78 | 5.6 | 126.3 | 17.2 |
| | EQFooler-Yake-MS | 78 | 3.6 | 130.7 | 14.23 |
| | EQFooler-Rake-MS | 78 | 4.1 | 113.8 | 14.08 |
| AG | EQFooler-Yake-KS | 91.5 | 0.8 | 314.1 | 24.17 |
| | EQFooler-Rake-KS | 91.5 | 6.5 | 227.2 | 17.15 |
| | EQFooler-Yake-MS | 91.5 | 1.7 | 267.8 | 15.32 |
| | EQFooler-Rake-MS | 91.5 | 1.7 | 216 | 15.33 |

keyword extractor scoring in both target models. For example, the adversarial accuracy of the target's models scoring variants for the WordLSTM trained on the MR dataset is 4.1 and 4.2 for EQFooler-Yake and EQFooler-Rake variants, respectively, which have a better performance than EQFooler-Yake and EQFooler-Rake variants which use keyword extractor scoring. Figure 4.2 illustrates the comparison between all variants in terms of adversarial accuracy. As shown, EQFooler-Rake-MS has the best performance in all the datasets except Ag when we target the WordLSTM model. However, when we target the WordCNN model, EQFooler-Yake-KS has a better performance than other variants.

Regarding the number of queries metric, those variants that use the target's model scoring show their ability to decrease the number of queries needed to convert an original document to an adversarial one. For instance, as shown in Table 4.7 for targeting the WordLSTM model, EQFooler-Rake-MS and EQFooler-Yake-MS need around 40% fewer queries to generate adversarial examples. As Table 4.8 shows, our proposed framework has a similar behaviour while selecting WordCNN as the target model. Figure 4.3 depicts a comprehensive comparison between the performance of all variants of the proposed framework regarding the number of queries needed for

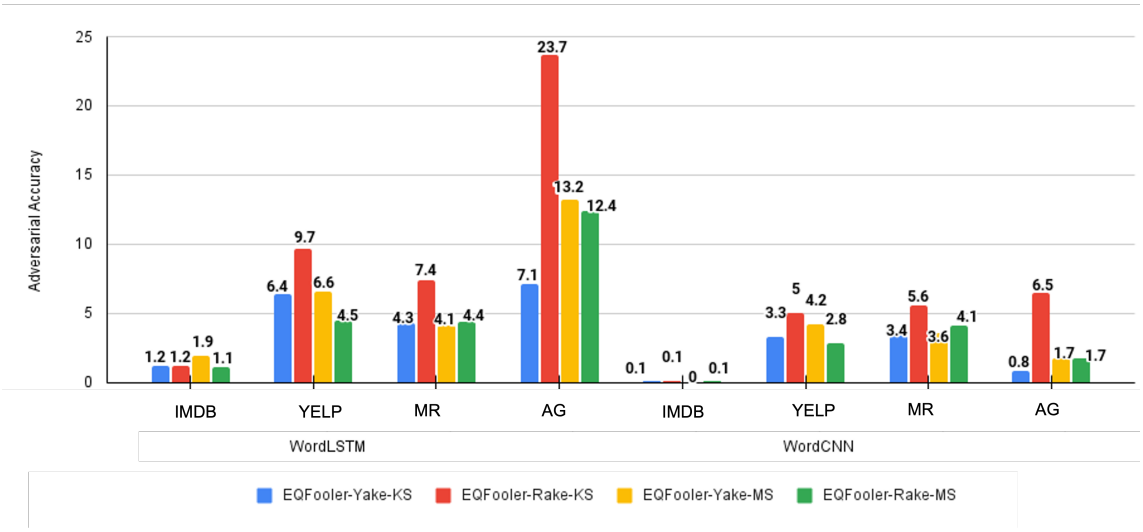



Figure 4.2: Comparison of the adversarial accuracy of different variants of EQFooler, attacking WordLSTM and WordCNN, IMDB, YELP, MR, and AG datasets.

generating adversarial examples.

Table 4.7 and Table 4.8 present the change rates in targeting WordLSTM and WordCNN, respectively. For example, the change rate of EQFooler-Yake-MS for the IMDB dataset is 5.95, which is around 50% lower than EQFooler-Yake-KS when it targets the WordLSTM. Table 4.8 shows the same results when we attack WordCNN. The change rate of EQFooler-Yake-MS for the IMDB dataset is around 60% lower than EQFooler-Yake-Ks's change rate. Figure 4.4 demonstrates the same behaviour through all the benchmark datasets.

Generally speaking, the variants of the proposed framework using target's model scoring for the replacement phase outperform the other variants. Therefore, we consider EQFooler-Yake-MS and EQFooler-Rake-MS for further analysis in the following sections.

### 4.3.4 Transferability

We investigated the success rate of generated adversarial example transferability, investigating those created on one model can induce another to predict wrongly. We

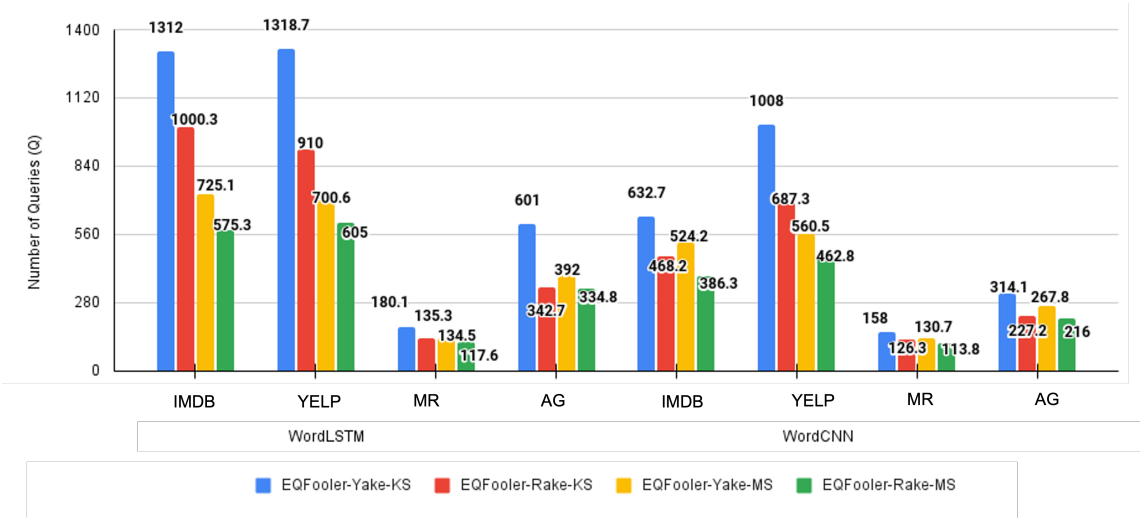



Figure 4.3: Comparison of the number of queries needed of different variants of EQ-Fooler, attacking WordLSTM and WordCNN, IMDB, YELP, MR, and AG datasets.

Table 4.9: Transferability $T(F_1, F_2)$ of different variant of the proposed framework using keyword extractors' scores.

| Generated adversarial based on,$F_1$ | | EQfooler-Yake (WordCNN) | EQfooler-Rake (WordCNN) | EQfooler-Yake (WordLSTM) | EQfooler-Rake (WordLSTM) |
|---|---|---|---|---|---|
| Applied to, $F_2$ | | WordLSTM | | WordCNN | |
| Datasets | IMDB | 9.315 | 7.744 | 34.537 | 34.650 |
| | Yelp | 6.284 | 4.944 | 13.951 | 17.305 |
| | MR | 40.349 | 41.436 | 44.895 | 43.520 |
| | AG | 4.631 | 5.529 | 19.240 | 21.746 |

gathered the successful adversarial examples for all datasets (incorrectly classified by a model) for this experiment. Then, we measured the success rate of collected crafted samples against another target model.

Table 4.9 and Table 4.10 demonstrate the transferability of four variants of the proposed framework. The first row of both tables shows the model and variants that the adversarial examples generated based on them. The second row shows the target model we want to attack with existing samples. In other words, we transfer the generated adversarial example for the first row to attack the model mentioned in the second row.

As shown in Table 4.9, the WordLSTM model is more robust than WordCNN. For example, the success rate of adversarial examples generated by EQFooler-Yake-KS

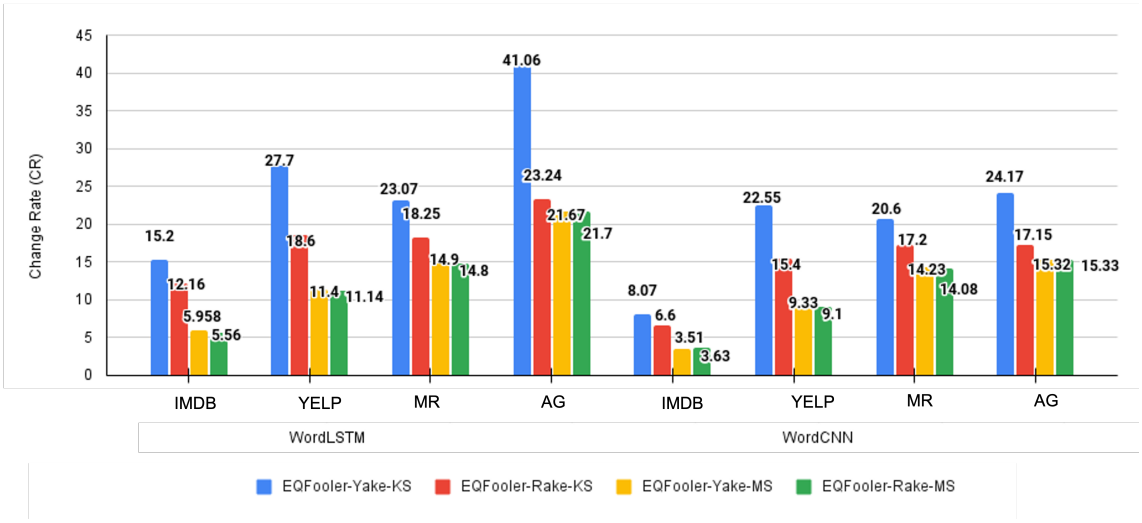
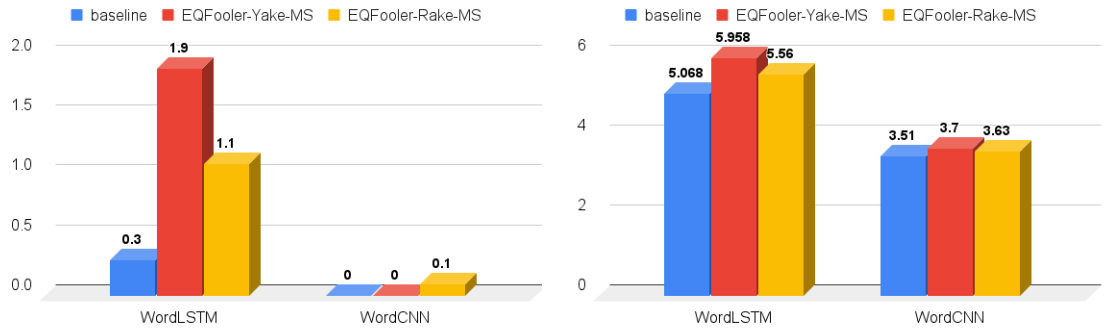



Figure 4.4: Comparison of the change rate of different variants of EQFooler, attacking WordLSTM and WordCNN, IMDB, YELP, MR, AG datasets.

Table 4.10: Transferability $T(F_1, F_2)$ of different variant of the proposed framework using target models' scores.

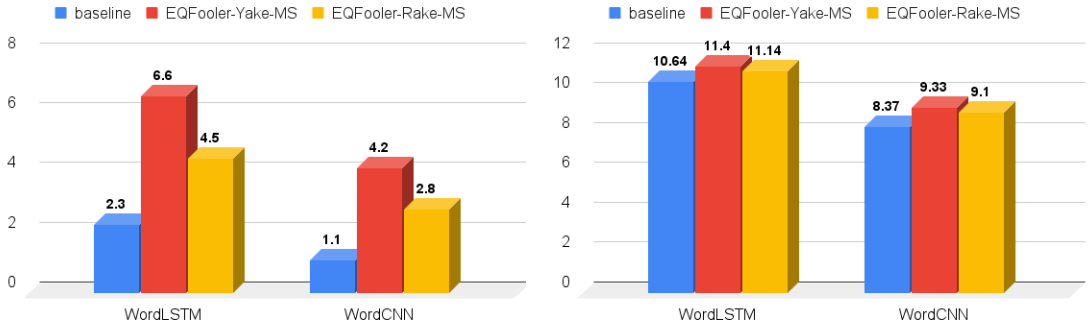
| Generated adversarial based on,$F_1$ | | EQfooler-Yake (WordCNN) | EQfooler-Rake (WordCNN) | EQfooler-Yake (WordLSTM) | EQfooler-Rake (WordLSTM) |
|---|---|---|---|---|---|
| Applied to, $F_2$ | | WordLSTM | | WordCNN | |
| Datasets | IMDB | 13.453 | 14.703 | 27.531 | 26.719 |
| | Yelp | 10.134 | 10.746 | 18.009 | 18.907 |
| | MR | 41.532 | 41.678 | 43.995 | 43.512 |
| | AG | 6.125 | 6.111 | 18.950 | 17.490 |

and EQFooler-Rake-KS based on WordLSTM is 34.53% and 34.65%, respectively, when we attack the WordCNN model. However, the success rate transferability for the vice versa situation is 9.3%, 7.74%. We see a similar behaviour for other datasets except for MR, for which success rate transferability for both models is the same. Table 4.10 shows the success rate transferability for those variants using target model scoring. The higher robustness of WordLSTM compared to the WordCNN model is proved. For example, the adversarial examples generated for WordCNN models can not be transferred to the WordLSTM model with a high success rate.




(a) Adversarial accuracy (b) Change rate

Figure 4.5: A comparison between EQFooler and baseline, using WordLSTM and WordCNN, IMDB dataset

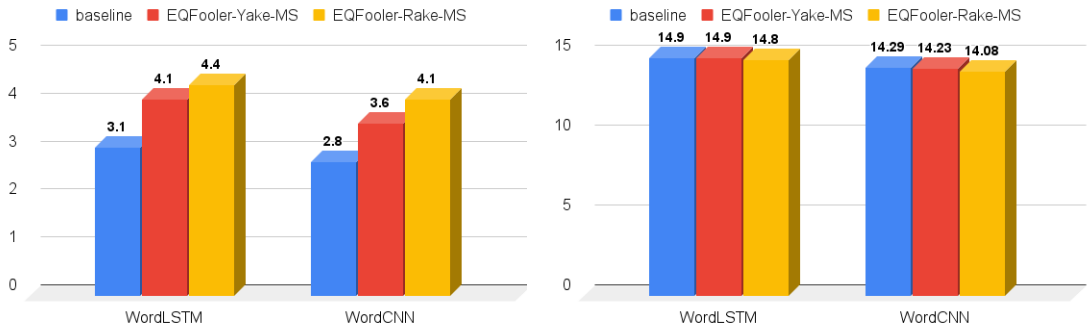



(a) Adversarial accuracy (b) Change rate

Figure 4.6: A comparison between EQFooler and baseline, using WordLSTM and WordCNN, YELP dataset.

## 4.4 Result Comparison and Discussion

To comprehensively evaluate the proposed method, we consider the TextFooler [34] as the baseline for our framework and compare it with the two variants, EQFooler-Yake-MS and EQFooler-Rake-MS of the proposed method. The comparison between these three attack methods has been made regarding adversarial accuracy, average change rate, and Number of Queries. Figure 4.5, Figure 4.6, Figure 4.7, and Figure 4.8 show the adversarial accuracy and change rate of these three attack methods on four different datasets. Figure 4.5(a) shows the comparison results for the IMDB




(a) Adversarial accuracy

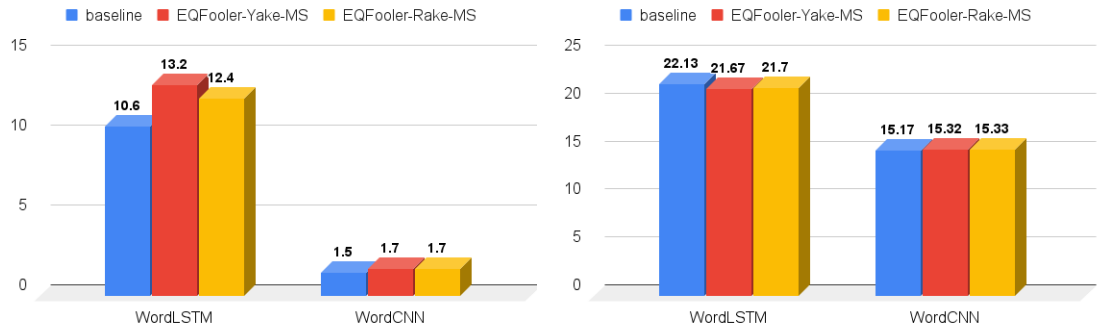
(b) Change rate

Figure 4.7: A comparison between EQFooler and baseline, using WordLSTM and WordCNN, MR dataset.

dataset regarding the adversarial accuracy. For attacking WordCNN, the baseline and EQFooler-Yake-MS reduce the accuracy to zero. It means that they can induce the WordCNN model to predict the label completely incorrectly. EQFooler-Rake-MS has similar adversarial accuracy; it is 0.1% which is very close to zero. Although in targeting the WordLSTM model, TextFooler approach has better performance in adversarial accuracy, EQFooler-Yake-MS and EQFooler-Rake-MS variants have stunning results, which are very close to the baseline's results.

Figure 4.5(b) provides a comparison regarding the average change rate using IMDB dataset. In general, the baseline has a better performance in this metric, but this superiority is not significant, i.e., 0.5% and 0.1% for WordLSTM and WordCNN target models respectively. Since documents in IMDB dataset have 212 words on average, these amounts would be equal to one word.

Figure 4.6(b) show the comparison between three attack methods for the Yelp dataset. The first observation obtained from this figure is that TextFooler has better adversarial accuracy than two other attack methods. However, the EQFooler-Rake-MS adversarial accuracy is very close to the baseline with a negligible difference (around 1.5%) for targeting the WordCNN model. Regarding the average change rate, we saw the same IMDB's results here, the EQFooler-Rake-MS and EQFooler-




(a) Adversarial accuracy (b) Change rate

Figure 4.8: A comparison between EQFooler and baseline, using WordLSTM and WordCNN, AG's News dataset

Table 4.11: A comparison between proposed adversarial attack and baseline regarding the number of needed queries to fool the WordLSTM and WordCNN models.

| Target Model | Datasets | Baseline | EQFooler-Yake-MS | EQFooler-Rake-MS | *Improvement(%) |
|---|---|---|---|---|---|
| WordLSTM | IMDB | 666.1 | 725.1 | **575.3** | 13.6% |
| | Yelp | 630.5 | 700.6 | **605** | 4% |
| | MR | 125.9 | 134.5 | **117.6** | 6.6% |
| | AG | 353.5 | 392 | **334.8** | 5.2% |
| WordCNN | IMDB | 524.2 | 464.1 | **386.3** | 26.3% |
| | Yelp | 492.3 | 560.5 | **462.8** | 6% |
| | MR | 123.3 | 130.7 | **113.8** | 7.7 |
| | AG | 228 | 267.8 | **216** | 5.2% |

*Improvement(%) shows the percentages of the EQFooler-Rake-MS improvement in terms of number of queries with baseline

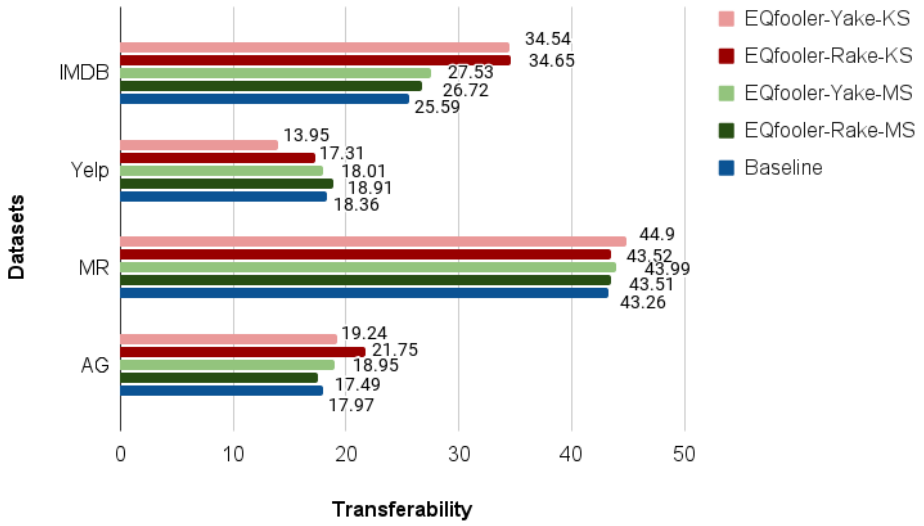
Yake-MS obtain almost the same results as the baseline's result.

The behaviour of the three attack methods in the MR dataset in the adversarial accuracy is very similar to the IMDB and Yelp datasets (see Figure 4.7(a)). However, in the average change rate (Figure 4.7(b)), the situation is different, and the EQFooler-Rake-MS has a better performance.

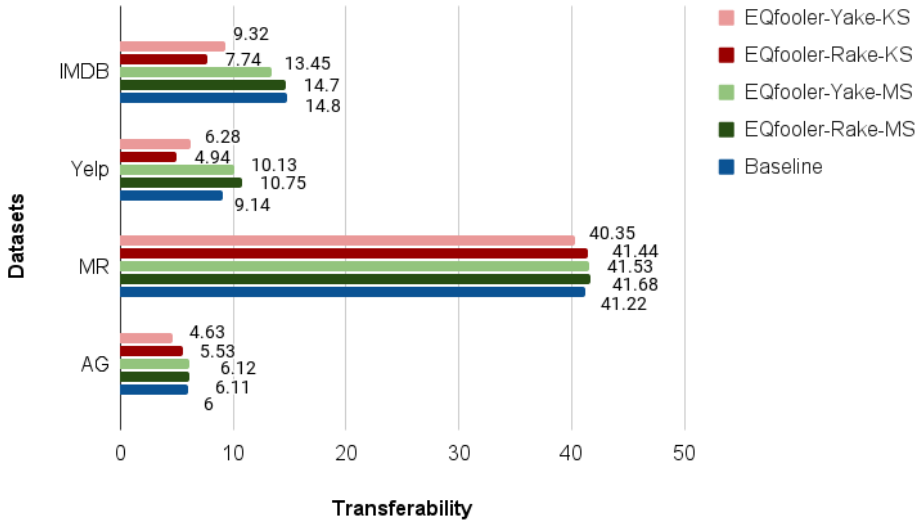
Figure 4.8 shows the results of the comparison of on AG dataset. We almost get the same results as the baseline in terms of adversarial accuracy targeting the WordCNN model. However, in the average change rate, we saw a bit of improvement; for example, EQFooler-Rake-MS achieves a 21.7% average change rate of 0.5%, which is lower than the baseline.

When considering the Number of Queries as the evaluation metric, the proposed method shows its capability. Table 4.11 shows these promising results. In all combinations of dataset and target models, EQFooler-Rake-MS has superiority by a substantial difference. Sometimes the difference between EQFooler-Rake-MS and TextFooler reaches up to 25%. For example, in the IMDB dataset and the WordLSTM model, we saw that EQFooler-Rake-MS could reduce the Number of Queries to 575, which is around 100 less than the baseline amount. This reduction for the IMDB dataset and the WordCNN model is more significant and reaches about 150. To compare the transferability of the proposed framework with the baseline, we designed the same experiment that we explained in subsection 4.3.4. Figure 4.9 has two subfigures showing the two different success rate transferability comparisons between the two models. Figure 4.9a shows the results of transferring adversarial examples generated by the four variants of the proposed framework and baseline based on WordLSTM to attack the WordCNN model. Figure 4.9b shows the vice versa situation, which means attacking the WordLSTM model by the adversarial example generated based on WordCNN.

In any comparison situation, at least one of the variants of the proposed framework has a better performance than the baseline. For example, in transferring WordLSTM adversarial example to the WordCNN model on the IMDB dataset, all the four variants of the proposed method have better performance than baseline. EQFooler-Rake-KS and EQFooler-Yake-KS improve the success rate of transferability by 10% compared to the baseline. Generally speaking, those variants which use target model scoring have a better success rate transferability, although in some cases, variants that use keyword extractor scoring have promising results.




(a) $F_1$: WordLSTM , $F_2$: WordCNN




(b) $F_1$: WordCNN, $F_2$: WordLSTM

Figure 4.9: A comparison between EQFooler's variants and baseline regarding Transferability $T(F1, F2)$

## 4.5 Concluding Remarks

The experiment's results reported in this chapter demonstrate the following points:

- We evaluated the performance of eight different keyword extraction methods in the adversarial example generation process. Rake and Yake have a better performance than other keyword extraction methods. So, we have designed all the experiments based on utilizing them.

- We defined four variants of the proposed framework that differ in keyword extraction method and scoring and ranking strategy. Variant EQFooler-Rake-MS leverage Rake as its keyword extraction and the target model's scoring as its scoring strategy outperforms other variants in most of the capacities. However, in some cases, different variants have a comparable result.

- We considered TextFooler [34], one of the most potent adversarial attacks in the text domain, as our baseline to evaluate the performance of the proposed framework variants. The experiment demonstrated comparable outcomes with baseline results in every evaluation metric. The promising results indicated that the proposed variants have a significant superiority over the baseline in terms of the number of queries needed to generate adversarial examples.

- We Showed that WordLSTM models are more robust than WordCNN models regarding transferability. The proposed framework has a better performance in the transferability of adversarial attacks compared to the baseline. For example, the proposed adversarial attack outperformed the baseline by 10% in the case that we attacked a WordCNN model trained on the IMDB dataset with the crafted samples for the WordLSTM model.

# Chapter 5

# Conclusions and Future Works

## 5.1 Conclusions

Adversarial examples are the consequences of non-robust data features and the vulnerabilities of machine learning models that an adversary uses to attack the model in the test time. Currently, deep neural networks are being used in many fields like natural language processing, image classification, and object detection, with outstanding results. Hence, adversarial attacks and their defences have become research interests for many machine learning and cybersecurity researchers.

Adversarial example generation for textual data is more challenging than in other domains such as computer vision for two main reasons: the discrete nature of input space and ensuring semantic coherence with the original sentence. While most of the prior efforts focus on word misspelling, word changing, and phrase removal and insertion, some other works generate adversarial examples in natural sentences. They generate crafted documents that are semantically and syntactically similar to the original ones [34]. However, none of the previous research concerns their resource consumption, such as running time and the number of queries needed to build a single adversarial example.

This work investigated how to create an efficient query black-box adversarial attack to induce a textual deep neural network to predict incorrectly. One of the most common approaches for measuring the efficiency of an adversarial attack is measuring the original document's change rate and the number of queries needed to craft an adversarial example. In this regard, we developed an Efficient Query black-box adversarial attack named EQFooler. While the proposed method mainly focused on decreasing the overhead on the victim models, it tried to generate grammatically and semantically correct documents. To do so, EQFooler has three phases, namely keyword extraction, importance score calculation, and replacement process.

Since EQFooler is a word-level approach, it leverages keyword extraction methods to find the best word candidates independently from the target model. The importance scores of a selected candidate have been calculated by considering the difference of confidence scores before and after eliminating the candidate word from the original document. The replacement process should be grammatically correct and semantically coherent. Therefore, we first extracted each candidate's synonyms using word embedding. Then, we used a POS tagger to check the generated document grammatically and Universal Sentiment Encoder method to verify the semantic similarity of the crafted document.

We have defined four different variants of the proposed framework based on the keyword extractor, namely Rake or Yake methods, and the importance score strategy, namely keyword extractor's scoring or target model's scoring. We evaluated the performance of these four variants through adversarial accuracy, change rate, and the number of queries. The analytical and experimental investigations showed that the variant using Rake keyword extractor and target model scoring has the best performance. This variant successfully attacked two state-of-the-art DNNs models trained for text classification by reducing their accuracy to lower than six on average on four benchmark datasets. To show the strength of the proposed method, we designed an

experiment to compare our outcomes with one of the state-of-the-art adversarial attacks. Figure 4.5 - Figure 4.8 compare the proposed framework's adversarial accuracy and change rate with the baseline. The proposed adversarial attack's results are comparable with the baseline. However, we argue that our attack can effectively decrease the number of queries in all the experiment's situations.

## 5.2 Future Work

The outcomes of this thesis can be further studied in multiple directions. By developing EQFooler, we are first building a framework that can efficiently attack any textual DNN model. We list a few directions for future work that can be considered future works in the following.

- EQFooler has been applied in text classification tasks, and the experimental results showed that it completely fooled the targeted model. It is beneficial to the community if EQFooler is applied in other NLP tasks such as Machine Translation(MT), Text entitlement, Text generation, and image captioning, and considering different target models.

- Keyword extraction methods helped EQFooler generate adversarial examples semi-independently of the target model. Studying the generation of totally independent adversarial examples with universal adversarial perturbation methods is an open research direction.

- EQFooler is a language-independent model that can be applied to various languages if an embedding language model is available. Analyzing the performance of EQFooler on other languages could be considered for future work.

- It would also be interesting to explore the application of proposed framework's to the BERT models.

# Bibliography


[1] Abdullah Al-Dujaili, Alex Huang, Erik Hemberg, and Una-May O'Reilly, *Adversarial deep learning for robust detection of binary encoded malware*, 2018 IEEE Security and Privacy Workshops (SPW), IEEE, 2018, pp. 76–82.

[2] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang, *Generating natural language adversarial examples*, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (Brussels, Belgium), Association for Computational Linguistics, October-November 2018, pp. 2890–2896.

[3] Edward J Anderson and Michael C Ferris, *Genetic algorithms for combinatorial optimization: the assemble line balancing problem*, ORSA Journal on Computing **6** (1994), no. 2, 161–173.

[4] Yonatan Belinkov and Yonatan Bisk, *Synthetic and natural noise both break neural machine translation*, arXiv preprint arXiv:1711.02173 (2017).

[5] Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu, *Advances in optimizing recurrent networks*, 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 8624–8628.

[6] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin, *A neural probabilistic language model*, The journal of machine learning research **3** (2003), 1137–1155.

[7] Floria n Boudin, *Unsupervised keyphrase extraction with multipartite graphs*, 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT), 2018, pp. 667–672.

[8] Adrien Bougouin, Florian Boudin, and Béatrice Daille, *Topicrank: Graph-based topic ranking for keyphrase extraction*, International joint conference on natural language processing (IJCNLP), 2013, pp. 543–551.

[9] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt, *Yake! keyword extraction from single documents using multiple local features*, Information Sciences **509** (2020), 257–289.

[10] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil, *Universal sentence encoder*, CoRR **abs/1803.11175** (2018).

[11] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen, *Enhanced lstm for natural language inference*, arXiv preprint arXiv:1609.06038 (2016).

[12] Ronan Collobert and Jason Weston, *A unified architecture for natural language processing: Deep neural networks with multitask learning*, Proceedings of the 25th international conference on Machine learning, 2008, pp. 160–167.

[13] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa, *Natural language processing (almost) from scratch*, Journal of machine learning research **12** (2011), no. ARTICLE, 2493–2537.

[14] George E Dahl, Tara N Sainath, and Geoffrey E Hinton, *Improving deep neural networks for lvcsr using rectified linear units and dropout*, 2013 IEEE international conference on acoustics, speech and signal processing, IEEE, 2013, pp. 8609–8613.

[15] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell, *Long-term recurrent convolutional networks for visual recognition and description*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 2625–2634.

[16] Javid Ebrahimi, Daniel Lowd, and Dejing Dou, *On adversarial examples for character-level neural machine translation*, arXiv preprint arXiv:1806.09030 (2018).

[17] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou, *Hotflip: White-box adversarial examples for text classification*, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 01 2018, pp. 31–36.

[18] Corina Florescu and Cornelia Caragea, *Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents*, Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, pp. 1105–1115.

[19] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi, *Black-box generation of adversarial text sequences to evade deep learning classifiers*, 2018 IEEE Security and Privacy Workshops (SPW), IEEE, 2018, pp. 50–56.

[20] Yoav Goldberg, *Neural network methods for natural language processing*, Synthesis lectures on human language technologies **10** (2017), no. 1, 1–309.

[21] Christoph Goller and Andreas Kuchler, *Learning task-dependent distributed representations by backpropagation through structure*, Proceedings of International Conference on Neural Networks (ICNN'96), vol. 1, IEEE, 1996, pp. 347–352.

[22] Zhitao Gong, Wenlu Wang, Bo Li, Dawn Song, and Wei-Shinn Ku, *Adversarial texts with gradient methods*, arXiv preprint arXiv:1801.07175 (2018).

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT press, 2016.

[24] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, *Explaining and harnessing adversarial examples*, arXiv preprint arXiv:1412.6572 (2014).

[25] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, *Speech recognition with deep recurrent neural networks*, 2013 IEEE international conference on acoustics, speech and signal processing, Ieee, 2013, pp. 6645–6649.

[26] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel, *Adversarial examples for malware detection*, European symposium on research in computer security, Springer, 2017, pp. 62–79.

[27] Geoffrey E Hinton, *A practical guide to training restricted boltzmann machines*, Neural networks: Tricks of the trade, Springer, 2012, pp. 599–619.

[28] Sepp Hochreiter and Jürgen Schmidhuber, *Long short-term memory*, Neural computation **9** (1997), no. 8, 1735–1780.

[29] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin, *Black-box adversarial attacks with limited queries and information*, International Conference on Machine Learning, PMLR, 2018, pp. 2137–2146.

[30] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Mądry, *Adversarial examples are not bugs, they are features*, Curran Associates Inc., Red Hook, NY, USA, 2019.

[31] Alexey Grigorevich Ivakhnenko, *Polynomial theory of complex systems*, IEEE transactions on Systems, Man, and Cybernetics (1971), no. 4, 364–378.

[32] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer, *Adversarial example generation with syntactically controlled paraphrase networks*, arXiv preprint arXiv:1804.06059 (2018).

[33] Robin Jia and Percy Liang, *Adversarial examples for evaluating reading comprehension systems*, arXiv preprint arXiv:1707.07328 (2017).

[34] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits, *Is bert really robust? a strong baseline for natural language attack on text classification and entailment*, Proceedings of the AAAI Conference on Artificial Intelligence **34** (2020), no. 05, 8018–8025.

[35] Meenu Mary John, Helena Holmström Olsson, and Jan Bosch, *Towards mlops: A framework and maturity model*, 2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE, 2021, pp. 1–8.

[36] Karen Sparck Jones, *A statistical interpretation of term specificity and its application in retrieval*, Journal of documentation (1972).

[37] Yoon Kim, *Convolutional neural networks for sentence classification*, arXiv preprint arXiv:1408.5882 (2014).

[38] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al., *Adversarial examples in the physical world*, 2016.

[39] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang, *Textbugger: Generating adversarial text against real-world applications*, 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019, The Internet Society, 2019.

[40] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi, *Deep text classification can be fooled*, arXiv preprint arXiv:1704.08006 (2017).

[41] Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun, *Automatic keyphrase extraction via topic decomposition*, Proceedings of the 2010 conference on empirical methods in natural language processing, 2010, pp. 366–376.

[42] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu, *Towards deep learning models resistant to adversarial attacks*, arXiv preprint arXiv:1706.06083 (2017).

[43] Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi, *Generating natural language attacks in a hard label black box setting*, Proceedings of the 35th AAAI Conference on Artificial Intelligence, 2021.

[44] Rada Mihalcea and Paul Tarau, *Textrank: Bringing order into text*, Proceedings of the 2004 conference on empirical methods in natural language processing, 2004, pp. 404–411.

[45] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, *Efficient estimation of word representations in vector space*, arXiv preprint arXiv:1301.3781 (2013).

[46] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, *Distributed representations of words and phrases and their compositionality*, Advances in neural information processing systems, 2013, pp. 3111–3119.

[47] Pasquale Minervini and Sebastian Riedel, *Adversarially regularising neural nli models to integrate logical background knowledge*, arXiv preprint arXiv:1808.08609 (2018).

[48] Takeru Miyato, Andrew M Dai, and Ian Goodfellow, *Adversarial training methods for semi-supervised text classification*, arXiv preprint arXiv:1605.07725 (2016).

[49] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard, *Universal adversarial perturbations*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1765–1773.

[50] Nikola Mrksic, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Lina Maria Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young, *Counter-fitting word vectors to linguistic constraints*, CoRR **abs/1603.00892** (2016).

[51] Bo Pang and Lillian Lee, *Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales*, Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, 2005, pp. 115–124.

[52] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami, *Practical black-box attacks against machine learning*, Proceedings of the 2017 ACM on Asia conference on computer and communications security, 2017, pp. 506–519.

[53] Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang, *Crafting adversarial input sequences for recurrent neural networks*, MILCOM 2016-2016 IEEE Military Communications Conference, IEEE, 2016, pp. 49–54.

[54] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami, *Distillation as a defense to adversarial perturbations against deep neu-*

*ral networks*, 2016 IEEE symposium on security and privacy (SP), IEEE, 2016, pp. 582–597.

[55] Jeffrey Pennington, Richard Socher, and Christopher D Manning, *Glove: Global vectors for word representation*, Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[56] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom, *Reasoning about entailment with neural attention*, arXiv preprint arXiv:1509.06664 (2015).

[57] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley, *Automatic keyword extraction from individual documents*, Text mining: applications and theory **1** (2010), 1–20.

[58] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, *Learning representations by back-propagating errors*, nature **323** (1986), no. 6088, 533–536.

[59] Suranjana Samanta and Sameep Mehta, *Towards crafting text adversarial samples*, arXiv preprint arXiv:1707.02812 (2017).

[60] Motoki Sato, Jun Suzuki, Hiroyuki Shindo, and Yuji Matsumoto, *Interpretable adversarial perturbation in input embedding space for text*, arXiv preprint arXiv:1805.02917 (2018).

[61] Blaž Škrlj, Andraž Repar, and Senja Pollak, *Rakun: Rank-based keyword extraction via unsupervised learning and meta vertex aggregation*, Statistical Language and Speech Processing (Cham) (Carlos Martín-Vide, Matthew Purver, and Senja Pollak, eds.), Springer International Publishing, 2019, pp. 311–323.

[62] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai, *One pixel attack for fooling deep neural networks*, IEEE Transactions on Evolutionary Computation **23** (2019), no. 5, 828–841.

[63] Mengying Sun, Fengyi Tang, Jinfeng Yi, Fei Wang, and Jiayu Zhou, *Identify susceptible locations in medical records via adversarial attacks on deep predictive models*, Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 793–801.

[64] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus, *Intriguing properties of neural networks*, 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (Yoshua Bengio and Yann LeCun, eds.), 2014.

[65] Kai Sheng Tai, Richard Socher, and Christopher D Manning, *Improved semantic representations from tree-structured long short-term memory networks*, arXiv preprint arXiv:1503.00075 (2015).

[66] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel, *The space of transferable adversarial examples*, arXiv preprint arXiv:1704.03453 (2017).

[67] André Viebke, Suejb Memeti, Sabri Pllana, and Ajith Abraham, *Chaos: a parallelization scheme for training convolutional neural networks on intel xeon phi*, The Journal of Supercomputing **75** (2019), no. 1, 197–227.

[68] Eric Wallace, Mitchell Stern, and Dawn Song, *Imitation attacks and defenses for black-box machine translation systems*, Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 5531–5546.

[69] Xiaojun Wan and Jianguo Xiao, *CollabRank: Towards a collaborative approach to single-document keyphrase extraction*, Proceedings of the 22nd International

Conference on Computational Linguistics (Coling 2008) (Manchester, UK), Coling 2008 Organizing Committee, August 2008, pp. 969–976.

[70] ———, *Single document keyphrase extraction using neighborhood knowledge.*, AAAI, vol. 8, 2008, pp. 855–860.

[71] Shuohang Wang and Jing Jiang, *Machine comprehension using match-lstm and answer pointer*, arXiv preprint arXiv:1608.07905 (2016).

[72] Wenqi Wang, Benxiao Tang, Run Wang, Lina Wang, and Aoshuang Ye, *A survey on adversarial attacks and defenses in text*, CoRR **abs/1902.07285** (2019).

[73] Wenqi Wang, Run Wang, Lina Wang, Zhibo Wang, and Aoshuang Ye, *Towards a robust deep neural network in texts: A survey*, arXiv preprint arXiv:1902.07285 (2019).

[74] Yu Emma Wang, Gu-Yeon Wei, and David Brooks, *Benchmarking tpu, gpu, and cpu platforms for deep learning*, arXiv preprint arXiv:1907.10701 (2019).

[75] Adina Williams, Nikita Nangia, and Samuel Bowman, *A broad-coverage challenge corpus for sentence understanding through inference*, Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018, pp. 1112–1122.

[76] Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning, *Kea: Practical automated keyphrase extraction*, Design and Usability of Digital Libraries: Case Studies in the Asia Pacific, IGI global, 2005, pp. 129–152.

[77] Yi Wu, David Bamman, and Stuart Russell, *Adversarial training for relation extraction*, Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 1778–1783.

[78] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al., *Google's neural machine translation system: Bridging the gap between human and machine translation*, arXiv preprint arXiv:1609.08144 (2016).

[79] Yang You, Aydın Buluç, and James Demmel, *Scaling deep learning on gpu and knights landing clusters*, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2017, pp. 1–12.

[80] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li, *Adversarial attacks on deep-learning models in natural language processing: A survey*, ACM Transactions on Intelligent Systems and Technology (TIST) **11** (2020), no. 3, 1–41.

[81] Xiang Zhang, Junbo Zhao, and Yann LeCun, *Character-level convolutional networks for text classification*, Advances in neural information processing systems **28** (2015), 649–657.

[82] Ye Zhang and Byron Wallace, *A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification*, arXiv preprint arXiv:1510.03820 (2015).

[83] Zhengli Zhao, Dheeru Dua, and Sameer Singh, *Generating natural adversarial examples*, arXiv preprint arXiv:1710.11342 (2017).

# Vita

**Candidate's full name**: Mohammad Mehdi Yadollahi

**University attended** (with dates and degrees obtained):
University of New Brunswick, Master of Computer Science (May 2019 - Dec 2021).
University of Tehran, Master of Computer Engineering, Software Engineering (Sep 2013 - June 2016).
University of Zanjan, Bachelor of Computer Engineering, Software Engineering (Jan 2004 - June 2008).

**Publications**:
Mohammad Mehdi Yadollahi, Arash Habibi Lashkari and Ali A. Ghorbani. *"Towards Query-efficient Black-box Adversarial Attack on Text Classification Models"*, In 2021 The 18th International Conference on Privacy, Security and Trust, December 2021.

Mohammad Mehdi Yadollahi, Farzaneh Shoeleh, Sajjad Dadkhah and Ali A. Ghorbani. *"Robust Black-box Watermarking for Deep Neural Network using Inverse Document Frequency"*, In 2021 The 6th IEEE Cyber Science and Technology Congress, October 2021.

Sajjad Dadkhah,Farzaneh Shoeleh, Mohammad Mehdi Yadollahi, Xichen Zhang and Ali A. Ghorbani. *"A real-time hostile activities analyses and detection system."*, In Applied Soft Computing(2021), 107175, 2021.

**Conference Presentations:**
Mohammad Mehdi Yadollahi, Farzaneh Shoeleh, Sajjad Dadkhah and Ali A. Ghorbani. *"Robust Black-box Watermarking for Deep Neural Network using Inverse Document Frequency"*, In 2021 The 6th IEEE Cyber Science and Technology Congress, October 2021.