UNIVERSITÄT KOBLENZ · LANDAU

IWVI

Assessment of Mean Teacher and Prominent Adversarial Unlabeled Data for Language Classification

Masterarbeit

Master of Science (M.Sc.) im Web and Data Science

vorgelegt von

Bhupender Kumar Saini

[219 100 887]

Koblenz, im January 2021

Erstgutachter:	Prof. Dr. Andreas Mauthe
	(Institut für Wirtschafts- und Verwaltungsinformatik, FG Mauthe)
Zweitgutachter:	Alexander Rosenbaum, M. Sc.
	(Institut für Wirtschafts- und Verwaltungsinformatik, FG Mauthe)

Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde.

Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden. ja ☐ nein ☐

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu. ja ☐ nein ☐

.....

(Ort, Datum) (Unterschrift)

Zusammenfassung

Sprachmodelle haben bei verschiedenen Aufgaben der natürlichen Sprachverarbeitung Spitzenleistungen erbracht. Jüngste Forschungen haben gezeigt, dass diese Modelle Schwächen gegenüber feindlichen Angriffen aufweisen, bei denen unmerkliches Rauschen im Text dazu führen kann, dass sich das Modell unerwartet verhält und seine Leistung bei Angriffen stark beeinträchtigt wird. Darüber hinaus ist die Erforschung von Verteidigungsmechanismen ein vergleichsweise wenig erforschtes Thema im Vergleich zur Generierung prominenter gegnerischer Angriffe. In dieser Masterarbeit wird ein halbüberwachter Ansatz der Feinabstimmung vorgeschlagen, der zu einem robusten Sprachmodell führen kann, ohne die ursprüngliche Genauigkeit zu beeinträchtigen. Es wurde ein Experiment durchgeführt, um die Leistung des mit der konventionellen Methode und der vorgeschlagenen Methode feinabgestimmten Modells zu vergleichen. Das Experiment wurde mit den Sprachmodellen BERT und DistilBERT auf zwei Datensätzen durchgeführt. Das Experiment zeigte, dass der vorgeschlagene Ansatz die ursprüngliche Genauigkeit um 1-2% und die Genauigkeit unter Angriffen um 25-30% verbessert.

Abstract

Language models has shown state-of-the-art performances in various natural language processing task. Recent research has shown their weakness against adversarial attacks, where imperceptible noise in text can lead model to behave unexpectedly and severely degraded their performance under attack. Furthermore, the research towards defensive mechanism is comparatively less studied topic than generating prominent adversarial attacks. In this master thesis, a semi-supervised approach of fine-tuning is proposed which can lead to robust language model without compromising with original accuracy. An experiment was conducted to compare the performance of model fine-tuned using conventional method and proposed method. The experiment was performed using BERT and DistilBERT language models on two datasets. As per experiment, the proposed approach demonstrated 0-2% and 20-30% improvement in original accuracy and accuracy under attacks over conventional method, respectively.

Contents

List of Figures

List of Tables

1 Introduction

The Deep Neural Network(DNN) has been widely adopted technology in real world applications and the most studied topic as well. Its capability of solving complex problems either linearly or non-linearly and ease of computation has proven upper hand to other machine learning algorithm [23]. In natural language processing, recent advancement in DNN language models has led to state-of-the-art transformer-based models like BERT,Roberta, DistlBERT etc. [10, 29, 34, 56] , which has outperformed in wide range of NLP tasks and is consequently the most widely adopted models now a days.

But, studies has also revealed the weakness in DNN models against adversarial attacks [1, 23, 60, 64, 66] which negatively impacted its value and raised concern of trust, safety, and security. This vulnerability was first revealed in computer vision domain in the study published by Szegedy et al. [60]. Their work showed that an imperceptibly small perturbations in the input image can fool deep neural networks with high probability which they referred as *Adversarial Samples* . Attacks that sabotage a machine learning model with adversarial examples are called adversarial attacks *Adversarial attacks* [45]. This study drawn substantial attention from research community in all domains.

The presence of adversarial attack in text-domain was first confirmed in the study presented by Papernot et al. [46]. Their study has shown a small word level change in sentence structure can also sabotage the DNN models in text domain. Later, various studies were later published detailing different techniques to craft adversarial text samples which is comparatively higher than defensive mechanism. Furthermore, recent works also revealed the performance of language models suffers severely under adversarial attacks [15, 32]. Jin et al. [26] has proposed an approach of creating adversarial samples and proved that BERT accuracy drastically drops to less than 10% under adversarial attacks and a pictorial example of attack can be seen in figure 1.1. These studies raised concern on conventional way of fine-tuning language models and not sufficient if robustness is concerned. And, at the same time opens a scope of work towards robustness enhancement of language models.

The topic of adversarial attacks are extensively studied in computer vision domain than that of natural language processing(NLP) [63]. And, in text domain, it is more focused on approaches related to create adversarial samples than defense mechanism. That May be because of generating adversarial samples in text domain are comparatively complex due to their discrete nature and requirement for semantic maintenance [32]. Furthermore, there are few studies that aims to

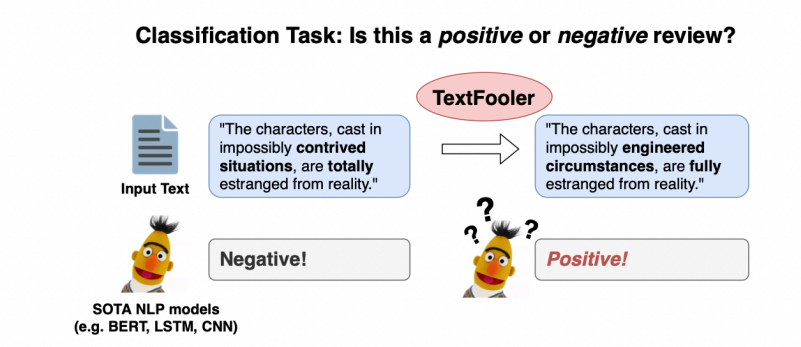

Figure 1.1: Adversarial attack presented by TextFooler [26], where word level changes in input text influenced the prediction.

improve the robustness of language model and mostly revolves around gradient-based methods [25, 40, 68], which are mainly inspired by seeing its effectiveness in computer vision domain.
Furthermore, most of the text classification tasks such as fake news detection suffers from a lack of labeled data in comparison to unlabeled data. A number of semi-supervised training approaches have been proposed to overcome these challenges, and also, shown that these approaches can significantly improves performance. As an example of a semi-supervised approach, mean teacher [61] has performed well in image-domain, however its efficiency in text domain especially with language models hasn't been examined. Implementing such a technique would be quite a challenge because different noise strategies and training approaches are needed to leverage such an approach. In another study, Belinkov et al.' [6] showcased how including noisy data in training samples may result in robust models. Until now, no relevant study has been conducted on the semi-supervised fine-tuning of language models and robustness evaluation as well.
Therefore, this master thesis study proposes a semi-supervised way of fine-tuning language models i.e. BERT, that can lead to a comparatively robust model without compromising with the original accuracy. As a next step, conducting a quantitative experiment to compare the performance of proposed approach with the conventional approach of fine-tuning. Additionally, we utilize language model capabilities, like back translation and context writing, in order to generate prominent adversarial unlabeled samples for training. The main focus of the study' is to answer the research hypothesis i.e. the proposed fine-tuning method will provide a comparatively better model for text classification in terms of generalization and robustness.
The proposed study also contributes in observing performance of the language models under worse conditions so mitigation strategies can be planned. In order to improve the robustness of DNNs, it is necessary to know how adversarial attacks operate and how to defend against them. A robustness assessment is achieved by observing the model's performance against four attack recipes and metrics.

2 Background

This chapter discusses about the contents required to understand the experiment and motivations behind this master thesis. We begin with a discussion of recent advances in text representations, followed by a discussion of transformers architecture, one of the main principle of recent contextualized embedding. Later, fine-tuning of the language model is carried out. An overview of adversarial attacks is provided at the end of section. If the reader already has a basic understanding of the topic, it is recommended to skip this chapter.

2.1 Text Representations (2 pages)

In natural language processing, researchers explore how a computerized system can understand and manipulate natural language (speech or text) to perform various tasks[9] and aims to gain human-level understanding of the text [44]. However, there was a major challenge in converting text into numbers or vectors in such a manner that semantic and syntactic information can be retrained. As shown in figure 2.1, traditionally words were usually represented as a single hot vector in a discrete manner, where each word is shown as a vector of 1 and 0 [55]. The shape of vector is equal to the size of vocabulary which suffers from dimensionality curse and also lacks semantic relationship between words. Later, researchers developed low-dimensional and continuous vector representations of text and discovered word embedding.

Berlin	London	Paris	Amsterdam
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Figure 2.1: An example of one hot encoding, the dimension of vector increases with vocabulary size.

2.1.1 Word Embeddings

Vector representations that map the words and phrases to numbers in a real world vector are called word embeddings [2] . And, the distributed hypothesis is the main principle behind this

type of representations [18], which posits that the words of similar contexts tend to have same meanings. This type of word embedding can be utilized to perform various NLP tasks. Mikolov et al. [38] proposed Word2Vec , "a continuous vector representation of words from very large datasets", where a CBOW (continuous bag of words model) and skip-gram model architectures are utilized to learn the representation. The goal of CBOW is to predict the middle word when given past and future words as inputs. At the projection layer, each word's context is averaged as shown in figure 2.2. As order of the words does not influence projection of the word hence called bag-of-words.
Word2Vec also use skip-gram to maximize classification of a word based on another word present in the sentence, as shown in figure 2.2. The major weaknesses of Word2Vec 1) It only preserve the local information of words, and 2) the semantics of a given word is entirely determined by the surrounding words.

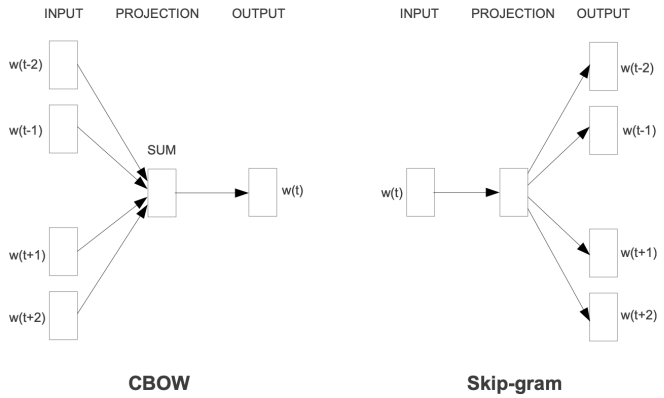

Figure 2.2: The CBOW architecture predicts the current word given past and future words, and the Skip-gram architecture predicts the surrounding words given the current words.

Later, Penning et al. [48] proposed word embedding called Glove (Global Vectors for Word Representation) which was based on word global information and a big corpus of unlabeled datasets. The idea is to determine how frequently a word pair occurs together by using a co-occurrence matrix and factorize this co-occurrence matrix into a lower dimensional space to reduce reconstruction loss. The approach was mainly based on global matrix factorization and local context window methods such as skip-gram, which led to log-bilinear regression model where the probability of next word determined by the previous word.
Word2Vec and Glove both approaches of creating word embeddings are static and have a limitation called polysemy, which refers to the fact that meaning of a word differs across contexts. For example, based on the context, "jaguar" might refer to an animal or to a car brand. These traditional approaches missed these deep details, and provided a direction of working on contextualized word embeddings.

2.1.2 Contextualized Embeddings

Matthew, et al.[49] proposed the ELMO (Embedding from Language Model) as one of the methods to create context-sensitive embedding. The ELMO model uses a bi-directional LSTM to extract contextualized word features to address the problem of polysemy. The model is further optimized by next word prediction using a large corpus of data. A layer's weight can later be used as contextual embedding.
Later, after transformer architecture introduced by Vaswani et al. [62] , Generative Pre-training (GPT) [51], and Bidirectional Encoder Representation From Transformers (BERT) [10] utilized encoders or decoders instead of bidirectional LSTMs to create contextualized embedding. GPT used decoder architecture and BERT used encoder architecture, however, the main concept of was based on ELMO, as shown in figure 2.3.
Both of these approaches introduce the concept of *pre-training*, i.e. learning embedding using large corpora of unlabeled data, and *fine-tuning*, i.e. optimizing weights further based on downstream task. Furthermore, also the approach of transfer learning can be utilized in NLP to get the state-of-the-art performance. A brief overview of the transformer model is required to understand language models work, so the next section discusses the transformer before language models.

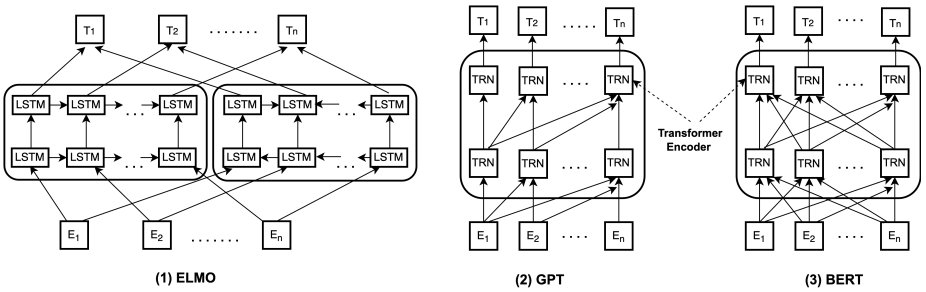

Figure 2.3: Different Contextualized Embeddings model.
Pictorial representation of working of (1) ELMO(Embedding from language model), (2)Generative Pre-training (GPT) , and (3) Bidirectional Encoder Representation From Transformers(BERT). TRN represents transformer encoder.

2.2 Transformers

There was a time when NLP tasks were solely based on sequential models like CNN, RNN, LSTM, and BiLSTM models which had the disadvantage of being computationally expensive, lacking distributing capabilities, and having only satisfactory performance. In order to reduce the amount of sequential computation, in December 2017, Vaswani et al. [62] proposed an attention based architecture called the transformer, which later outperformed the existing state-of-the-art NLP models. A transformer architecture was developed which is exclusively based on a type of attention mechanism called self-attention. Comparatively, this architecture has faster

training times because of its distributed properties, and showed better evaluation results. Later, this transformer architecture became the one of the main principles behind the development of break-through models like BERT, GPT, and T5. Figure 2.4 illustrates the architecture of a transformer.

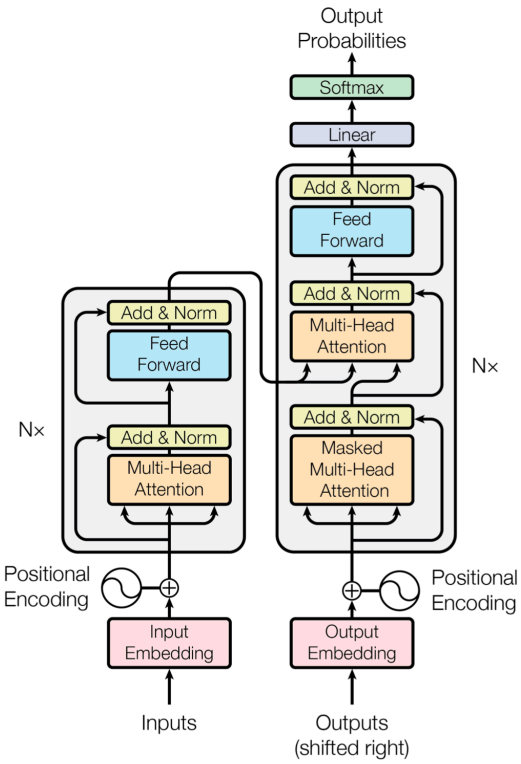

Figure 2.4: Transformer Model Architecture [62].

Transformers are encoder-decoder stacks where the encoder reads inputs and outputs representation as a context vector, often referred to as a "contextualized embedding," as illustrated in figure 2.5, which is based on single or multi-headed attention, and the decoder makes predictions based on those context vectors. Vaswani et al. [62] proposed the transformer model which consist 6 layers of encoders stacked on top of each other and same applies to decoders. Encoder are composed of multi-head attention followed by layer normalization and feed forward networks. However, there is slit difference in decoder i.e. masked multi-head attention layer followed by multi-head attention layer as shown in figure 2.4.

2.2.1 Encoder Architecture

In machine translation, the main difficulty was to translate variable length inputs to another variable length output. As a result, encoder and decoder are proposed as solutions, where encoder learns the pattern of variable length input and outputs a fixed shape output. A decoder,

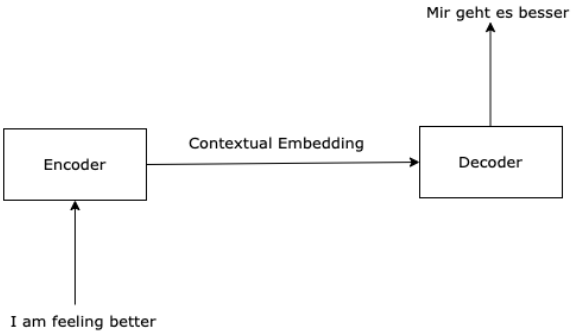

Figure 2.5: Basic example Encoder Decoder.

on the other hand, takes that fixed shape as input and outputs a variable length. For example, the task is to translate english sentence "I am good. How are you ?" to German language , given input sequence of tokens "I", "am", "good",...,"?" to encoder which out the fixed shape representation z_1, z_2, z_3. Using this representation, decoder outputs the "Mir geht es gut. Wie geht es dir ?" in token format as shown in figure 2.6. Later, several sequence to sequence models utilize this encoder-decoder architecture to perform task such as text summarization, question answering, and machine translation. However, the encoder decoder architecture in transformer differs significantly.

A transformer encoder block is further divided into three layers as shown in figure 2.7: multi-

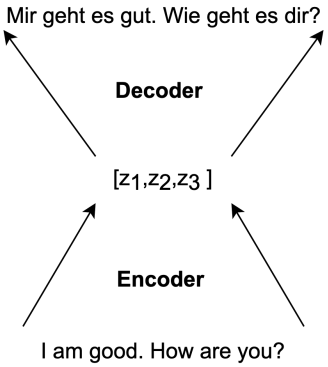

Figure 2.6: Basic working example of encoder decoder.

head attention, layer normalization layer, and feed forward network. The input embedding component converts the input text tokens into embedding vectors EM of shape d_{model}.

Positional Encoding

Sequential models understand sentences and contain information about word position, however, in transformer models, sentences are fed all at once, so a separate mechanism was introduced to determine word order. The word order in the shape of a position vector will be added to

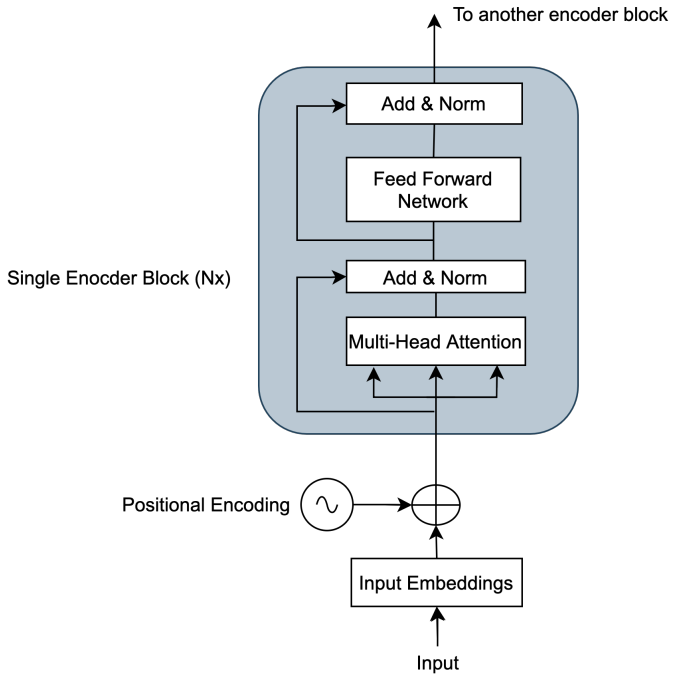

Figure 2.7: Different components of encoder in transformer.

the input embedding before the multi-head attention. Position vector must have the same dimension as word embedding d_{model}.

Two major constraints that applies: First, the word embedding information should not be severe and second, word position must be identical. According to Vaswani et al. [62], sine and cosine functions of different frequencies are used to form geometric progression from 2π to $2\pi \cdot 10000$ used for calculating the position vector,PV of the words. In other words, mentioned function 2.1 generates unique values containing information about the position of words in a sentence.

$$\begin{aligned} PV_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ PV_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2.1}$$

Where $posandi$ is position and dimension respectively. And, after adding to the embeddings we get position encoding (PE)

$$PE_{word} = EM_{word} + PV_{word} \tag{2.2}$$

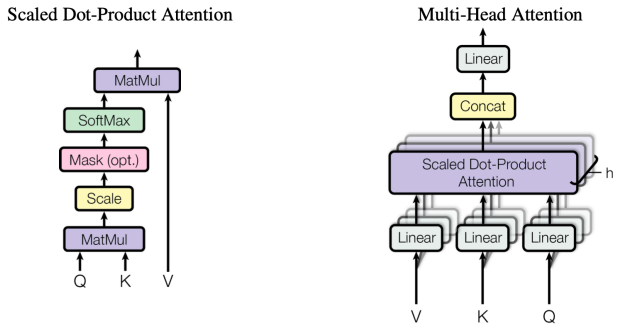

Figure 2.8: Scaled dot product and multi-head attention calculation flow diagram [62].

2.2.2 Attention Mechanism

The human mind does not process all the information available in environment, but rather, it focuses on specific information to complete the task and this biological mechanism is called attention. And, the major motivation behind attention mechanism in various machine learning tasks. In 1964, Naradaya et al. [43] first introduced the idea of attention in regression model, proposing a weighted function $a(x, x_i)$ which encodes the relevance of features. In 2015, Bhadanu et al. [4] proposed encoder-decoders that include attention mechanisms at the decoder level. Later, M.T Luong et al. [35] have shown that attention mechanisms gained up to 5.0 BLEU over non-attention models in neural machine learning tasks. And, recently, Kardakis et al. [27] studied the same mechanism in the text classification task, i.e. sentiment analysis, and reported 3.5 % increase in accuracy.
Attention mechanism in transformer architecture model is introduced to reduce computational complexity and facilitate computation [62]. Different types of methods exist to calculate attention mechanisms such as similarity based [17], dot products [35], scaled dot products [62], additives [4], etc. This report focuses on scaled dot product attention which is used in transformer models. Equation 2.3 and figure 2.8 demonstrate how to calculate the scaled dot product or self-attentions.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.3}$$

Where Q, K,V is query, key and value vector which is created by dot product of different trainable weight matrix w_q, w_k and w_v with input. With this attention mechanism, the relationship between words is more profound, resulting in better performance. In order to capture the different perspectives of the sentence and ensure better accuracy, multiple attention heads are calculated instead of just one. Then, concatenating the result provides a comparatively better attention matrix called multi-head attention and this mechanism exploits the parallelization feature.

Further, the add and norm component is basically a residual connection followed by layer normalization that prevents heavy changes in values during training. Later, feed forward network consists of two dense layers activated by a ReLU.

2.3 Language Models

The language model is a function that learns the word representation from the corpus and provides vectors that can be utilized for further downstream tasks such as machine translation or sentiment analysis. A number of techniques are available to learn a representation, either statistically or by means of neural networks. In recent years, neural network-based language models, such as BERT and DistilBERT, have become the cornerstone of Natural Language Processing (NLP). According to figure , these language models follow two training processes , 1) *Pre-training* by using huge unlabeled text corpora and generating a context representation vector from the model [10] and enables the transfer learning. The pre-training mechanisms of BERT and DistilBERT are discussed in more detail in this section. (2) *Fine-tuning*, user adapts the pre-trained language model to perform specific tasks by adding a feed-forward layer on top of language models as per task and further train the weights with limited data for the target task. The fine-tuning approach can be performed in low-resource environments and have achieved state-of-the-art performance in many popular NLP benchmarks.

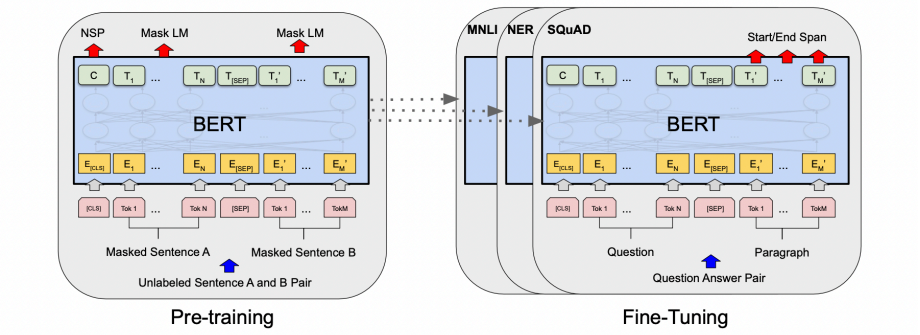

Figure 2.9: Pre-training and fine-tuning method of the BERT model [10]. In pre-training, unsupervised techniques such as next sentence prediction (NSP) and masked word prediction (MWP) are utilized. Pre-trained models are provided to perform further downstream tasks and weights are further optimized during fine-tuning.

2.3.1 BERT(Bidirectional Encoder Representation From Transformers)

BERT (Bidirectional Encoder Representations from Transformers) was proposed by Devlin et al.[10], primarily based on Transformers [62], ULMFit [21], ELMo [49], and the OpenAI transformer [51], but not limited to it. In essence, BERT is a transformer encoder stack that outputs the context representation, also called a pre-trained model. BERT model is pre-trained on deep

bidirectional representation of large unlabeled text in both right and left context, which can be further fine-tuned by adding additional output layers to achieve state-of-the-art results in various NLP tasks like text classification, question answering, language inference, language translation, etc. Its main benefit is simplifying the process of NLP tasks in machine learning, and providing access to contextualized embedding trained on huge amounts of words, which are impossible to collect individually. In addition, it requires high performance computational machines at production scale.
Having access to vast amounts of unlabelled data, BERT uses two learning strategies: Masked Language Model (MLM) and Next Sentence Prediction (NSP). MLM replaces 15% of words with [MASK] tokens, and BERT predicts the masked word based on other words in the sequence. In NSP, BERT models are given two pairs of sentences, with [CLS] as the sentence start and [SEP] as the separation between sentences. The BERT model then predicts whether the next sentence is correct or random. The BERT model is pre-trained on a large amount of unlabeled text, however, fine-tuning is still required for specific tasks.
To enable BERT models to handle a variety of downstream tasks, the input representations are a sum of three different embeddings (position embeddings, segment embeddings, and token embeddings), as shown in figure 2.10.

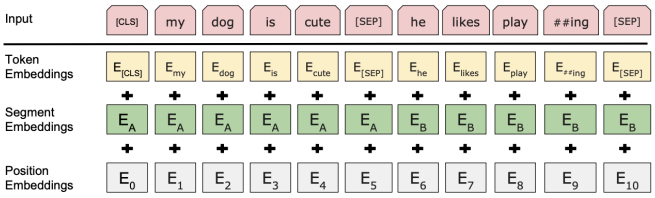

Figure 2.10: BERT model input includes three different embedding information to perform various downstream task[10].

A BERT paper [10] described two BERT models on which they conducted their experiments.

1. $BERT_{BASE}$: 12 Transformers blocks(Encoder, L), 768 Hidden Units(H), Attention Heads(A) 12, Total Parameters 110M.
2. $BERT_{LARGE}$: 24 Transformers blocks(Encoder, L), 1024 Hidden Units(H), Attention Heads A 16, Total Parameters 340M.

In this master thesis experiment, $BERT_{BASE}$ model pre-trained on lower case words is utilized.

2.3.2 DistilBERT- A distilled version of BERT

DistilBERT is a compact version of BERT proposed by Victor et al. [56]. As compared to BERT, the DistilBERT model does not incorporate token embeddings, poolers, and have comparatively half hidden layers. The principle behind this model is based on knowledge distillation [19], which

can be defined as a process of transferring knowledge from large to small models.
As illustrated in figure 2.11, a small model is trained to mimic the behavior of a larger model. In the task of mask word prediction, masked sentences are supplied to both the BERT and DistilBERT models.

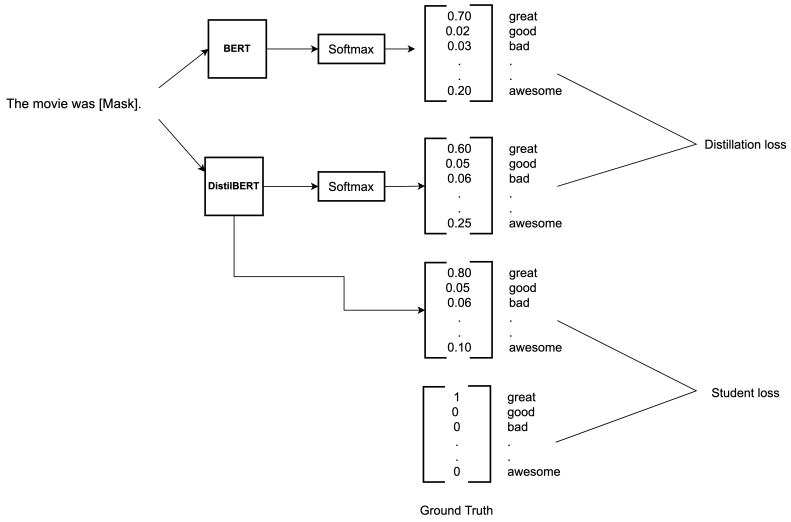

Figure 2.11: Pictorial diagram depicts the knowledge distillation process involved in DistilBERT.

This approach aims to minimize three losses: 1) the loss between BERT and DistilBERT predictions, 2) the loss between DistilBERT predictions and ground truth, and 3) the cosine embedding loss, which measures the distance between the representations learned by BERT and DistilBERT. Cosine embedding loss reduction makes representation more accurate and tries to copy the BERT embeddings. The proposed approach is 40% compact, retaining 97% of its language understanding capabilities and 60% faster [**sanh_DistilBERT_2020**] , which has proven it is possible to reduce the size of large language models with minimal compromise.

2.4 Adversarial Attacks

In year 2013, C. Szegedy et al. [60] study in the image-domain discovered that deep neural networks are vulnerable to perturbations, which can result in unexpected behaviors, referred to as adversarial attacks. It was first demonstrated by [46] that an adversarial example can also lead to an unexpected outcome in the text domain. Later various papers on different attack recipes were published in text domain [3, 8, 14, 15, 32, 53] and not limited to it. Sun et al. [59] proposed approach of generating adversarial misspelling and evaluated the performance of attack recipes against BERT model. Based on their experiment on Stanford Sentiment Treebank (SST) dataset, the BERT model is vulnerable to misspelling attacks and its accuracy decreases by 22.6% under their proposed attack.

In another experiment, Li et al. [32], proposed an approach of utilizing BERT model to generate word for replacements. Firstly, identifying the most important words i.e. the words in sequence have a high influence on the final output logit. Secondly, utilizing BERT model masked language model(MLM) capability to generate word for replacements. According to their claims, the accuracy under attack is lower than 10% with less than 10% perturbation. However, there is a possibility of compromising with semantic constraints.
In text domains, most adversarial attacks consist of two steps: 1) Identification of the most important word, and 2) Replacement of the word with suitable words. The reason behind why DNN models are vulnerable to adversarial attacks is still an open question. The presence of adversarial attacks in DNN models, according to Goodfellow et al. [16], is due to the linearity of DNN models.

2.4.1 Definition

For a given input data and its labels (x, y) and a classifier F function which classify inputs x to its respective class label y i.e. $F(x) = y$. However, adversarial attack techniques introduce small perturbation δ in input data.
Hence, the attack would be an untargeted adversarial attack if $F(x + \delta) \neq y$ and target when $F(x + \delta) = y'$ in constraint of the perturbation δ must be imperceptible to humans which can be defined by a threshold $||\delta|| < \epsilon$.
Most common metrics to determine perturbation δ and define threshold ϵ are cosine similarity, euclidean distance, jaccard coefficient, and word mover distance. However, in the text domain, it is really challenging to create adversarial example because human can easily detect a minute change at character, word or sentence level.
A particular classifier would be called robust against a particular adversarial example $(x + \delta)$ if a classifier F should predict correct class y i.e. $F(x + \delta) = y$. And, various metrics to evaluate model robustness such as accuracy under attack, number of queries, word perturbation and attack success which is discussed in length at section 5.5.

2.4.2 Types of Adversarial attacks

Based on recent survey, adversarial attack can be classified based on level of knowledge, target, and level of perturbation [23, 63]. If attackers have the complete knowledge of the system, then it would be called as white box attack. And, in another case if only mode output is known, then it would be a black box attack. A white box gradient based attack was first proposed by Ebrahimi et al. [11] to search for adversarial word/character substitutions. And, Jin et al. [26] have proposed black box word level adversarial attack and shown that BERT models are vulnerable to adversarial samples.

As mentioned in section 2.4.1, when an attacker intentionally attempts to sabotage a model

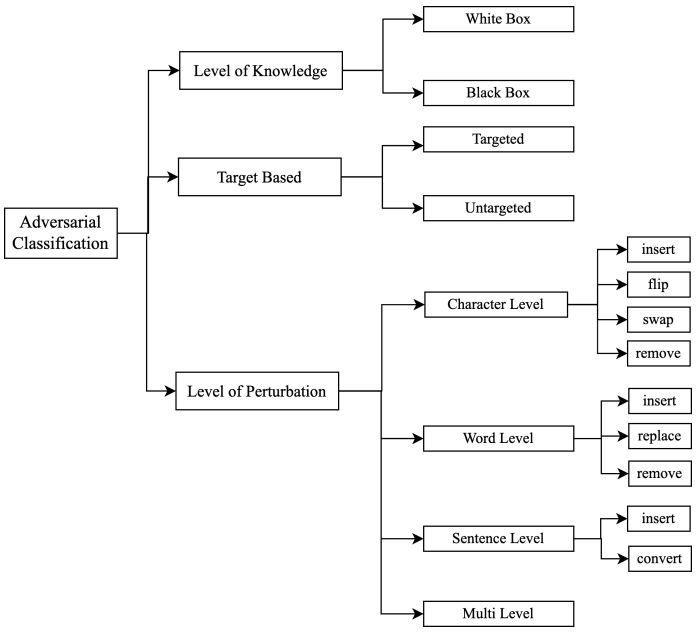

Figure 2.12: Different adversarial attack classification based on different aspect.

based on a specific class classification, it could be classified as targeted attack, whereas if the attacker just has the aim of sabotaging the model without depending on class classification, it would be untargeted attack.

A word level of perturbation may result in an attacker inserting, replacing, and removing words. TextDeceptor [57] proposed text attack approach, where they rank sentences and words, and then replace them with similar words based on a cosine measure of similarity between word vectors, as well as considering the POS (part-of-speech) helps to determine the correct grammar. Furthermore, Yuan et al. [65] proposed a Word-Level Textual Adversarial attack that uses sememe-based word substitution. Using sememe-based word substitution is supposed to be more accurate since the substituted word has probably retained its meaning. In accordance with their claim, their attacks have a 98.70% success rate on the IMDB dataset.

A char-level attack involves inserting, flipping, removing, and swapping operations to create adversarial text attacks. In the proposed approach by Eger et al. [12], a visual perturber called VIPER replaces input characters with their visual nearest neighbours in visual embedding space. In the sentence level, attackers try to convert the input sentence using back translation or parapharsing technique and/or insert some sentences into the text. Yankun et al. [54] developed an approach that generates real-world meaningful text automatically using a variational encoder and decoder model. However, the sentences are often different from the originals.

2.4.3 Adversarial Training

The goal of adversarial defenses is to achieves high test accuracy on both clean and adversarial examples [67]. Two strategies are mainly discussed in the text domain to fight adversarial attacks, the first being proactively detecting adversarial text, and the second being model enhancement by using adversarial training. Detecting adversarial text mainly revolves around detecting unknown words and misspellings, which impose limitations on using only the original corpus vocabulary [63]. According to Goodfellow et al. [16], including high quality adversarial examples in images can improve the robustness and generalization of machine learning models.Similarly in text domain, Belinkov et al. [6] demonstrated in their experiments that mixed noise in text training samples can improve model robustness. In another experiment performed by Li et al. [30], showed that including textbugger attack adversarial samples in training can also improve model performance and robustness against adversarial examples.

A fast gradient method (FSM) approach to text domain training was introduced by Miyato et al. [40], whereby methods generate adversarial examples by adding gradient-based perturbations to input samples with different normalization strategies.

However, research related to adversarial training of language models is few. Liu et al. [33] proposed adversarial training of BERT model using virtual adversarial training(VAT) technique [41] during pre-training. However, proposed approach is computationally expensive and also a gradient based approach.

Furthermore, the mean teacher approach has shown comparative performance in the computer visualization domain and the model claimed to be comparatively robust [61]. And, the performance of this approach in the text-domain, which incorporates language models with adversarial training tactics and evaluation of approach against adversarial attack had open scope of work. In addition, proposed approach utilizes the language model capabilities of such as context rewriting and back translation as data augmentation techniques to create prominent adversarial unlabeled data. Similarity approaches has been utilized previously to generate adversarial sample such as Siddhant et al. [15] proposed approach used BERT"s masked language model(MLM) capability to generate possible adversarial examples. Therefore, one of the major motivations for this experiment is to generate prominent adversarial examples and develop strategies for incorporating them into training.

3 Related Work

In the area of adversarial training and defense mechanisms against attack, a various types of techniques are studied and proposed. Still, the research in text-domain is, however, comparatively less than that in the image-domain [63]. And, very few researches deals in evaluating language models against adversarial attacks recipes and proposing a semi-supervised fine-tuning approach as well. In this section, we discuss work related to adversarial training on the text-domain, using language models and focusing more on increasing accuracy under attack.

Adversarial training are primarily based on two approaches: 1) Gradient-based 2) Data-augmentation-based.

The objective in gradient-based is to optimize the adversarial loss for the model by applying perturbations in the embedding space [16, 25, 33, 40, 68] . However, these studies are mainly focused increasing generalization and not evaluated against attack recipes.

Liu et al. [33] proposed a novel adversarial pre-training approach intended to increase the robustness and generalization of large neural language models. They called it ALUM (Adversarial Training for Large Neural Language Models). Both pre-training and fine-tuning can be accomplished using the same approach. Due to inner maximization and complexity, adversarial training is often quite costly, and verification of proposed approaches under a variety of attacks still needs to be studied.

A more recent approach is proposed by Danqinq et al. [68], in which adversarial training is done by fast gradient methods (FGM) [40] and ensemble methods, in which multi-BERT model prediction assists in robustness. A proposed approach combines multiple BERT (BERT, SciB ERT, RoBERTa, ALBERT, and ELECTRA) and makes an average ensemble for all models to achieve superior performance. Large language models and internal maximization raise concerns about computational cost. Moreover, as mentioned previously, this approach is also completely focused on increasing generalization and performance under attack has not been evaluated.

As part of a noise-based approach, Si et al. [58] proposed robust fine-tuning of language models by augmenting the training dataset as well as performing mix-up augmentation during training, hence the term AMDA (Adversarial and Mixup Data Augmentation). A mix-up augmentation is a linear interpolation of representations and labels pairs of training samples to create different virtual training samples. As per their experiments, the original accuracy for the BERT model is 91.27% and the accuracy under attack is 14.83%; however, BERT with AMDA had 91.10% original accuracy and 31.52% accuracy under attack. Model performance under attack has

improved significantly with this approach, but original accuracy decreased.

One approach proposed by Bao et al. [5] which involves training a language model to classify the input text and also discriminate adversarial samples simultaneously. Their proposed approach also generates adversarial samples using TextFooler [26], applies frequency aware randomization to the adversarial training set, and finally combines it with the original training set. In IMDB data, this approach has shown marginal improvements in original accuracy from 92.4% to 92.8% of BERT model, but improvement in accuracy under attack, from 12.4% to 89.2%. However, further metrics related to adversarial attacks such as number of queries and word perturbation are not mentioned in their report.

Currently, adversarial samples are majorly generated using gradient methods and then included in training to increase the accuracy of original models but verifying robustness against adversarial attacks is still to be known especially considering language model. In addition, because there is no standard evaluation framework, it is challenging to compare the performances of published approaches. However, this master thesis experiment trying to provide information related metrics specific to evaluation of model against adversarial samples will be calculated and showcased, which is discussed in length in section 5.5.

4 Methodology

This chapter discusses about working of proposed approach, followed by attack recipes used to evaluated the model, and research question.

4.1 Proposed Approach

The proposed method calls for fine-tuning the BERT model using semi-supervised approach for classification tasks and utilized the adversarial unlabeled dataset for the same. A pictorial representation of working of proposed methodology can be seen in figure4.1.
The proposed approach uses semi-supervised approach of fine-tuning language model and utilizes the prominent adversarial unlabeled data for training with labeled data. The approach first starts with creating adversarial samples for training. As shown inf figure 4.1, after pre-processing the training dataset is being split into three equals parts and prominent adversarial unlabel data is created on three different data augmentation strategies i.e. 1) Synonym based, 2) Context based and, 3) Back translation based and discussed in length 5.3. Labels of those adversarial dataset is discarded hence called prominent unlabeled adversarial data. Then, these generated samples combined together and shuffled. After, creating prominent unlabeled adversarial data semi-supervised training process starts.
The propose semi-supervised approach is based on the mean teacher approach implemented in computer visualization domain [61]. In the mean teacher model, two identical models are trained with two different strategies called student and teacher model. In which, only student model is trained, however, during training exponential moving weights are assigned to the teacher. The intuition behind this approach can be understood as, instead taking average of many models decision, teacher model is an mean of consecutive students models weights hence mean teacher.
As shown in figure 4.1, two cost function plays important role while back-propagating i.e. classification cost and consistency cost. Classification cost($C(\theta)$) is calculated as binary cross entropy between label predicted by student model and ground truth label y given input x. As, in this master thesis evaluation is performed against binary classification task, hence, binary cross entropy loss is used. However, loss function can be utilized as per task.

$$C(\theta) = -y \log(f(x, \theta)) - (1 - y) \log(1 - f(x, \theta)) \tag{4.1}$$

The consistency cost($J(\theta)$) is the mean squared difference between the predicted outcomes of the student (weights θ and noise η) and teacher model (weights $\hat{\theta}$ and noise η'). The mathematical declaration is as follows.

$$J(\theta) = \mathbb{E}_{x_{adv}}[\|f(x_{adv}, \theta) - f(x_{adv}, \hat{\theta})\|^2] \tag{4.2}$$

While back propagating in student model, the overall cost ($O(\theta)$) is calculated with given formula

$$O(\theta) = rC(\theta) + (1 - r)J(\theta) \tag{4.3}$$

When training, exponential moving average(EMA) weights of the student model are assigned to the teacher model at every step, and the proportion of weights assigned is controlled by parameter alpha(α). As mentioned in equation 4.4, while assigning weights, teacher model holds its previous weights in alpha(α) proportion and $(1 - \alpha)$ portion of student weights. The proposed approach can be seen in form of algorithm 1.

$$\hat{\theta}_t = \alpha\hat{\theta}_{t-1} + (1 - \alpha)\theta_t \tag{4.4}$$

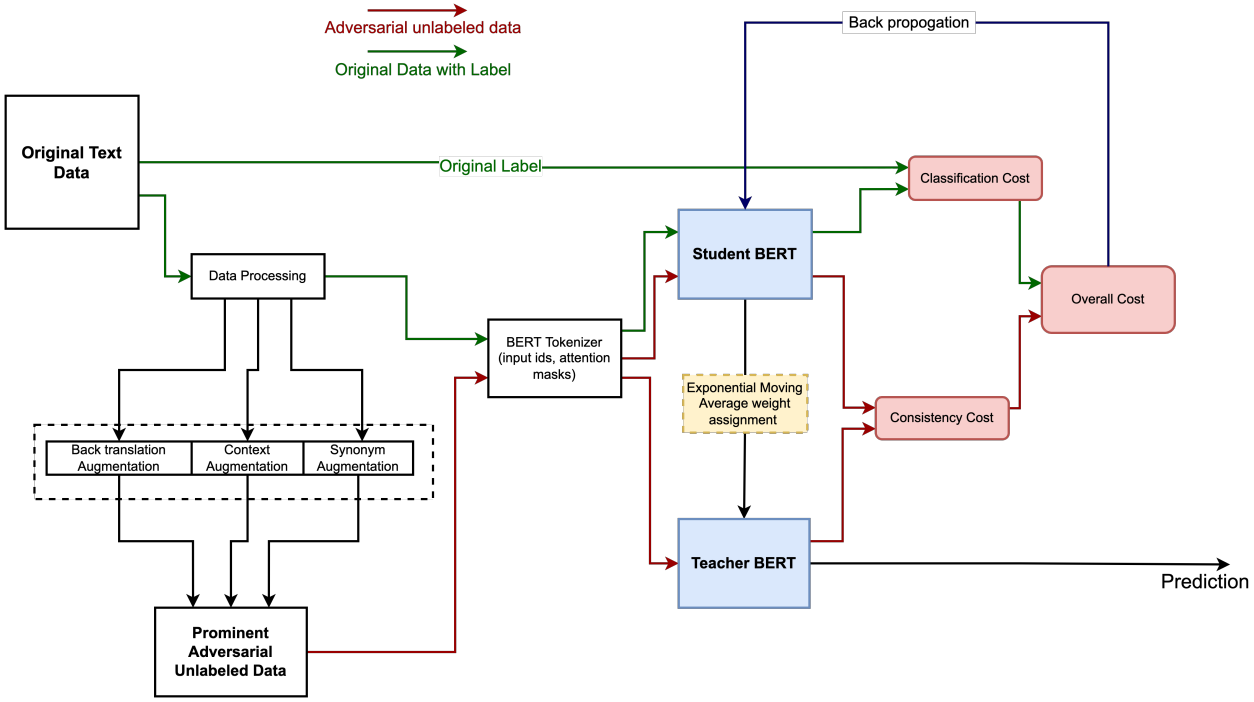

Figure 4.1: Training flow diagram of proposed methodology

Algorithm 1 Mean Teacher Algorithm

Data: train set $(\mathcal{X}, \mathcal{Y})$, Prominent Adversarial Unlabeled set($\mathcal{Z}$)
Hyper parameters: *r*, α,*epochs*
Create Model: *student*(θ), *teacher*($\hat{\theta}$)
Train *teacher* for n epoch
for epochs $= 1$ to N **do**
  while $steps$ **do**
    student(x) $= y$
    Compute Classification cost ($C(\theta)$) =Binary Cross Entropy(y,y')
    student(z) $= y_s$
    teacher(z) $= y_t$
    Compute Consistency cost $J(\theta)$=Mean Squared Error(y_s,y_t)
    Compute Overall cost $O(\theta) = rC(\theta) + (1 - r)J(\theta)$
    Compute *gradients*, $O(\theta)$ w.r.t θ
    Apply *gradients* to θ
    Update Exponential Moving average of θ to $\hat{\theta}$ i.e. $\hat{\theta}_t = \alpha\hat{\theta}_{t-1} + (1 - \alpha)\theta_t$
  end while
end for

4.2 Text Attack Recipes and Tool

In order to evaluate the proposed approach, four black box attack recipes that satisfy lexical, grammatical, and semantic constraints have been selected. In order to evaluate baseline BERT model and proposed Mean Teacher BERT, we have used TextAttack python package[42] that provides attack recipes. Table 4.1 shows an example of perturbation. In this section, we will discuss their attacking principle, working, and characteristics.

Attack Recipe	Original Text	Perturbed Text
TextFooler	absolutely fantastic whatever i say wouldn t do this underrated movie the justice it deserves watch it now fantastic	absolutely sumptuous whatever i say wouldn t do this underrated movie the justice it deserves watch it now fantastic
TextBugger	absolutely fantastic whatever i say wouldn t do this underrated movie the justice it deserves watch it now fantastic	absolutely fa?tastic whatever i say wouldn t do this underrated movie the justice it deserves watch it now fantastic
PWWS	absolutely fantastic whatever i say wouldn t do this underrated movie the justice it deserves watch it now fantastic	absolutely rattling whatever i say wouldn t do this underrated movie the justice it deserves watch it now fantastic
BAE	absolutely fantastic whatever i say wouldn t do this underrated movie the justice it deserves watch it now fantastic	absolutely shit something i say wouldn t do this underrated film the justice it deserves of it now fantastic

Table 4.1: Perturbation example of attack recipes in this experiment.

4.2.1 TextFooler

Di jin et al. proposed Textfooler [24], a simple and effective adversarial attack generation strategy in black box settings which has characteristic of preserving the semantics, and grammar

which they called utility-preserving adversarial examples as illustrated in figure 4.2. In order to better understand this process, we will briefly explain three steps involved:

1. **Word Importance Ranking**: Given a sentence of words, they create a ranking of each word by calculating the change before and after deleting the words, called the importance score. NLTK and spaCy libraries are then used to remove stop words and preserve the grammar of the sentence. To improve the similarity of vectors, antonymy and synonymy are injected into vector space representations. The replacement policy completely depends on three factors: (1) Similar semantic similarity, (2) Fit in the surrounding context, (3) Attack on the model. Using the Universal Sentence Encoder proposed by Cer et al. [7] for encoding the sentence into a high-dimensional vector and calculating the cosine similarity between sentences. Next, select replacement candidates with values above the preset threshold value and create a pool of candidates.
2. **Replacement**: A candidate from a pool of candidates who can change the prediction of a target model is selected if there is an existing candidate with the highest cosine similarity between the original and adversarial sentences. If not, a lower confidence score for the label is chosen.

Using IMDB movie review datasets, TextFooler has assessed the performance of BERT model under adversarial attack. In their experiment, the accuracy dropped significantly from 90.9% to 13.6% with perturbed words 6.1, number of queries to target model 1134, and average length of IMDB dataset 215. Additionally, TextFooler is computationally inexpensive and complexity increases linearly with text length.

Movie Review (Positive (POS) ↔ Negative (NEG))	
Original (Label: NEG)	The characters, cast in impossibly ***contrived situations***, are ***totally*** estranged from reality.
Attack (Label: POS)	The characters, cast in impossibly ***engineered circumstances***, are ***fully*** estranged from reality.
Original (Label: POS)	It cuts to the ***knot*** of what it actually means to face your ***scares***, and to ride the ***overwhelming*** **metaphorical wave** that life wherever it takes you.
Attack (Label: NEG)	It cuts to the ***core*** of what it actually means to face your ***fears***, and to ride the ***big*** **metaphorical wave** that life wherever it takes you.

Figure 4.2: TextFooler example [24]

4.2.2 TextBugger

The TextBugger system proposed by Jinfeng Li et al. [30] uses misspelled words or characters that are visually and semantically similar to the original text. When tokens are misspelled, they become 'Unknown,' which is mapped to unknown tokens id, causing machine learning models to behave incorrectly. On the other hand, studies show that similar misspellings can still be perceived by the reader [3, 52]. Both character-level and word-level perturbation are targeted in this attack. Although Li et al. [30] proposed both white box and black box attack generation strategies, our report focuses on black box attacks. Here are three steps to black box attack

generation:

1. **Finding Important Sentences**: The importance score of individual sentences in an article is determined by confidence score of particular sentence by target model.
2. **Finding Important Words**: The importance score of word is the difference between confidence of target model with word and without word.
3. **Bugs Generation**: In TextBugger, they use five bugs generation strategy (1) **Insert**: Inserting space into words, (2) **Delete**: Deleting random character, (3) **Swap**: Swapping random adjacent character, (4) **Substitute-C**: Substitute character with visually similar characters, and (5) **Substitute-W** : Replacing word with top-k nearest neighbour in context aware word vector space , as shown in figure 4.3

Original	Insert	Delete	Swap	Sub-C	Sub-W
foolish	f oolish	folish	fooilsh	fo0lish	silly
awfully	awfull y	awfuly	awfluly	awfu1ly	terribly
cliches	clich es	clichs	clcihes	c1iches	cliche

Figure 4.3: TextBugger 5 bug generation strategies [30]

Using the IMDB movie review dataset, the TextBugger model is evaluated against LR, CNN, and LSTM and shows 95.2%, 90.5%, and 86.7% accuracy with perturbed words of 4.9%, 4.2%, and 6.9%. This report does not assess the effectiveness against BERT model. Comparatively to TextFooler, TextBugger generates adversarial attacks in much less time thanks to its sub-linear relationship to text length.

4.2.3 Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency (PWWS)

Shhuhuai et al. [53] proposed a method for synonym and named entity (NE) replacement based on the words' saliency and classification probability, as well as a greedy algorithm called Probability Weighted Word Saliency (PWWS). When replacing a word with a synonym, there must be a significant change in classification probability as well as minimum saliency of the word. The approach can be summarized as follows:

1. **Word Selection Strategy**: Salience of a word is defined as its degree of change in classification probability if the word is set to unknown [31]. Calculating word saliency vectors for each word in a text, and then prioritizing the words based on degree of change in classification probability after replacement and minimum word saliency.

2. **Replacement Strategy**: They searched WordNet for the synonyms of the words in order to find the replacement. Also, if the word is a Named Entity(NE), then replacing the NE with another NE of the same type appeared in the opposite class.

Lastly, greedily replace words to make the model change the label. In addition to Word-based CNN [28], LSTM [20], and Char-based CNN [62], this approach has been evaluated against other language models. According to the Bi-LSTM result, the accuracy dropped from 84.86% to 2.00% with perturbation 3.38% for the IMDB dataset, example is shown in figure 4.4. However, the computational and time complexity of the proposed approach is higher than other methods.

Original Prediction	***Adversarial*** Prediction	Perturbed Texts
Positive Confidence = 96.72%	Negative Confidence = 74.78%	Ah man this movie was ***funny*** (***laughable***) as hell, yet strange. I like how they kept the shakespearian language in this movie, it just felt ironic because of how idiotic the movie really was. this movie has got to be one of troma's best movies. highly recommended for some senseless fun!
Negative Confidence = 72.40%	Positive Confidence = 69.03%	The One and the Only! The only really good description of the punk movement in the LA in the early 80's. Also, the definitive documentary about legendary bands like the Black Flag and the X. Mainstream Americans' repugnant views about this film are absolutely ***hilarious*** (***uproarious***)! How can music be SO diversive in a country of supposed liberty...even 20 years after... find out!

Figure 4.4: Example attack of PWWS[53]

4.2.4 BAE: BERT-Based Adversarial Examples

The BERT masked language model (MLM) was employed by Garg et al. [15] to generate adversarial examples. Based on their approach, they first calculate the importance of words by computing the decrease in probability of predicting the correct label after deleting that particular word, similar to Textfooler [24] and PWWS [53]. Using the BERT MLM model, replace a specific word with a MASK token and let the model predict context-specific words. Using the Universal Sentence Encoder [7] and removing words that do not fall into a similar part-of-speech(POS) as the original word, filter the top K tokens using the most similarity score (Threshold 0.8). Substitute top K(50) tokens for the original word, iterate from most similar token in decreasing order until attack is successful and try all combinations. Figure 4.5 shows a schematic working diagram.

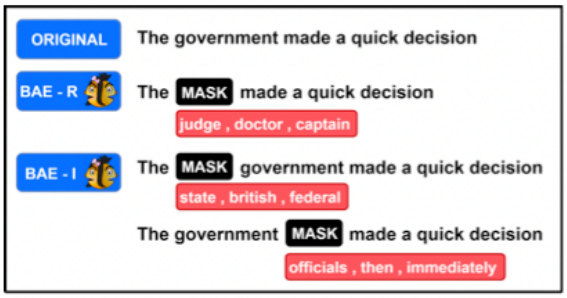

Figure 4.5: Schematic working and example of BAE[15]

4.3 Research Questions

Furthermore, the objective of using this approach is to study the effects on model performance by using comparatively simpler adversarial examples which involves utilizing capabilities of language models such as context re-writing, and back translation.
It is hypothesized that robustness can be achieved by learning more representations and regularizing the model by not allowing model to learn the deep insight of the representation to prevent over-fitting as well. Furthermore, exponential moving averages (EMA) play an essential role in increasing generalization and performance under attack.
The proposed methodology is mainly designed to show how using data augmentation techniques can enhance the performance of language models under attack without compromising on generalization. In other words, semi-supervised fine-tuning can create comparatively robust language models, and language models still have room for improvement.
This thesis attempts to answer the question by constructing a teacher model using proposed semi-supervised approaches and comparing the generalization of both conventional and proposed fine-tuning methods in terms of different evaluation metrics. 1) Averaging weights over time tends to produce a more accurate model than using the final weights directly [50] and teacher model is an average of consecutive student models, therefore, teacher model should be more accurate. 2) Adding data augmentation can enhance model robustness [6]; hence, the proposed model approach should also perform well under attack.
The robustness of this experiment will be evaluated using metrics such as original accuracy, accuracy under attack, average word perturbation, average number of queries, and attack success rate, which are discussed in length in section 5.5. Therefore, the hypothesis would be that the proposed approach would have higher original accuracy, accuracy under attack, and require more queries and perturbations to get attacked. Additionally, the attack success rate is lower. The model performance is evaluated using four different attack recipes discussed in section 4.2. In addition, the experiment also attempts to understand the robustness of the model by examining the confidence score distribution under attack, which can reveal the resilience of

the model. A model that does not allow attack recipes to be classified incorrectly with high confidence is a sign of robustness.

5 Experiment Environment

5.1 Dataset

To assess the performance of the baseline model and the proposed model. Covid-19 fake tweets dataset [47] provided at Codalab competition and IMDB review binary classification dataset were selected. The IMDB dataset is a sentiment classification dataset[37] that contains movie reviews of the user and contains positive and negative ratings, and it is primarily used in classification and adversarial attack papers. Due to computational limitations, we have filtered and sampled only datasets whose length is between 6 and 150 as the augmentation process takes quite a while. This leaves us with 6000 samples to train and test our model. The train and test sizes are shown in table 5.1. Additionally, we have sampled 6000 training labeled samples to create an augmented unlabeled dataset. Filtered datasets have an average length of 100 samples. The label distribution in training datasets is completely balanced. Covid-19 fake tweets dataset is a recent dataset specifically for classifying fake information in tweets relating to COVID-19 with a fake or real label. Covid-19 fake tweets dataset size is 8000 and we have fully utilized this dataset. Selection of this dataset was based on observing the performance of the proposed model in more recent Covid-19 fake tweets datasets. Contrary to the IMDB dataset, the Covid-19 fake tweets dataset has an average length of 25, as shown in table 5.2. We would like to investigate the performance of models under this scenario also because the Covid-19 fake tweets dataset has mostly hashtags and less English words vocabulary.

Dataset	Train	Test	Aug. Unlabeled
codalab (Positive/Negative)	3199/2891	1071/969	6090
IMDB (Fake/Real)	3025/3025	1000/1000	6050

Table 5.1: Train/ Test split details of dataset

5.2 Data Pre-processing and Exploration

Since language models learn the context of sentences, they are least affected by stop words. Therefore, removing those words may affect the performance. Therefore, one of the benefits of the language model is its negligible requirement for data cleaning or no data cleaning at all. Here are the preprocessing steps we performed:

1. HTML tags removal.
2. Digit removal.
3. Lower casing.
4. Punctuation removal.

This particular task was accomplished by using the texthero python library, which offers functions related to data pre-processing and exploration.

Data Exploration

The mentioned models were trained with almost equal distributions of labels in training and test data, and the same training data that was used to generate unlabeled augmented data via the proposed method is shown in 5.1. During experiments, various exploration technique were utilized such as word cloud, word distribution with/without respect to classes in dataset, and length distribution. Those information can be seen in provided github project. However, the most relative information about the dataset i.e. about length is provided in table 5.2 and figure 5.1. The Covid-19 Fake tweets dataset is shorter than the IMDB dataset. The purpose for using two distinct length datasets is to better understand the behavior of models and attack recipes in relation to the length.

Dataset	Avg. Length
codalab	25
IMDB	127

Table 5.2: Length details of both the datasets.

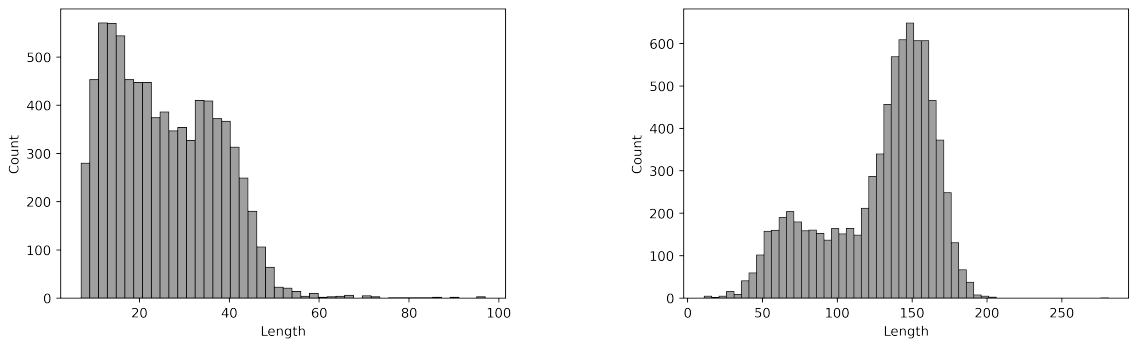

Figure 5.1: Length Distribution of Fake news and IMDB Dataset.

5.3 Data Augmentation

For creating the unlabeled augmented dataset, we have utilized three strategies :

1. Synonym Augmentation
2. Context Based Augmentation
3. Back translation.

While augmenting this dataset, there is a high chance that the information will be changed or completely opposite to the label of the original dataset. Therefore, once we augment the data, we will not use the label of the augmented data, so the augmented dataset will be unlabeled. For synonym changes, various python packages like text attack, nlpaug, and various basic python packages were used during experiments.
In addition, time and computation constraints have led us to choose the nlpaug data augmentation package [36] to achieve all three augmentation strategies. We have augmented the dataset by removing the label columns from the train dataset and then randomly splitting it into three parts. The first part is for synonym augmentation, while the second part is for context based augmentation, and the last part is for back translation.
One more idea was to create unlabeled data using different attack recipes, which might provide better results under attack. However, the reason why attack recipes are not used is because the model is created with little or no understanding of how attacks work.
In the case of implementing a specific attack recipe to create unlabeled data, robustness of the model will be enhanced for specific attack recipes, but overall evaluation of the model performance can be biased.
WordNet lexical English database [39] is used as a source of synonym augmentation, which includes word definitions, hyponyms, and semantic relationships. Same database in our case utilized for synonym replacement. The maximum augmentation (aug_max) parameter controls the level of augmentation. For IMDB and Covid-19 fake tweets datasets, it is set to 50 and 15, respectively. Another parameter called $iter$ is used to create two different copies of the synonym augmentation dataset. Example of generated synonym augmentation can be seen in table 5.3.

Original Text	Augmented Text
look how a true story with a little help of it s friends a welldone and touching script a good directing and a surprising great acting from a bunch of no name actors especially from the yr old jodelle ferland becomes a must seen movie	search how a true story with a lilliputian help of it s friends a welldone and touching script a good directing and a surprising great acting from a bunch of no name actors peculiarly from the yr old jodelle ferland becomes a must seen moving picture
i have to differ from the other comments posted amid sporadic funny moments there are a lot of actors trying too hard to be funny the strain shows i watched this with two friends on another friend s recommendation none of us were thrilled	ace have to disagree from the other comments mail amid sporadic rummy moments there are a circle of actors trying too toilsome to embody funny the strain shows i watched this with 2 friends on another friend s recommendation none of us were thrill

Table 5.3: Sample example of synonym based augmentation from IMDB dataset.

A context-based augmentation replaces the words in the sentence without changing the context. Generally, language models are used to accomplish this particular task, which is quite time- and memory-consuming. To perform this augmentation, DistilBERT language model is used.

Original Text	Augmented Text
i loved this movie and will watch it again original twist to plot of man vs man vs self i think this is kurt russell s best movie his eyes conveyed more than most actors words perhaps there s hope for mankind in spite of government intervention	listeners loved this movie and will watch it again original note to plot of beast vs man vs self all think this is kurt russell bollywood breakout movie his eyes wander more than most actors words perhaps our s hope for mankind knowing spite of no intervention

Table 5.4: Sample example of context based augmentation from IMDB dataset.

During back translation, sentences are converted in different languages and then translated back to the original language. In the same way, Marian's translation framework [36] is used, which is time and memory consuming. As part of our experiment, we are converting sentences into Romance language and back to English. We have used that model to perform this experiments. The Marian translation framework is free, faster, and more efficient.

Original Text	Augmented Text
after seeing the trailer for this movie and finding out spike lee was directing i was excited to hear about this event i wasn t alive at the time i didn t live in new york so i expected more of a history lesson than anything what i got was some interesting acting and about minutes worth of film that actually had anything to do with the son of sam i guess the film wasn t about the son of sam but it was a peek into the summer of label me disappointed	after watching the trailer for this movie and knowing that spike lee was directing, i was thrilled to know about this event that i wasn't alive at the time that i wasn't living in new york, so i expected more than a history lesson that nothing that i could get was something interesting performing and about minutes of film that was really worth filming that had something to do with sam's son, i guess the movie wasn't about sam's son, but it was a look at the summer label i was disappointed

Table 5.5: Sample example of back translation based augmentation from IMDB dataset.

Considering examples of all augmentation techniques, it is evident that word perturbation are quite higher in back translation followed by context-based augmentation.

5.4 Experiment Environment description

The proposed experiment has been performed on two language models i.e. 1) $BERT_{Base}$ uncased version and, 2) DistilBERT uncased version, model architecture is shown in figure 8.2 and 2.11 respectively. During this experiment baseline models will be referred as BERT and DistilBERT, and respective proposed model will be referred as MTBERT and MTDistilBERT. For training the models, google colab provided GPU has been utilized as shown in figure 8.1.

5.4.1 Hyper parameter Details

To train the baseline model and proposed model, we used the hyper parameter values shown in table 5.6. Exploring the performance of the model under different setting is not under the scope of this experiment.

Hyper parameter	Used parameters in this work
Optimizer	Adam
Learning rate	$2\epsilon - 5$
Loss function	Binary Cross Entropy
Epochs	3
Batch Size	4
Loss Ratio	0.5
Alpha	0.99
Dropout	0.2
Max length	100

Table 5.6: Hyper-parameters Details

5.5 Metrics

During the experiment, generalization of the model is calculated by using accuracy of the target model on test samples, here referred as original accuracy i.e. the percentage of correct predictions over the entire sample set. Next, we calculate robustness of the target models based on four different metrics:

1. Accuracy under attack
2. Attack success rate
3. Average number of queries
4. Perturbation Score

A target model's accuracy under attack is the percentage of correct predictions with respect to test samples, i.e. accuracy with respect to crafted test samples. Compared to the original accuracy, accuracy under attack can provide significant information about efficiency of the target model and attack recipes. Significant differences between these metrics indicate lower robustness against attack. A counter-part of accuracy under attack is the attack success rate, which indicates the effectiveness of attack recipes.
A metric other than accuracy is the average perturbed word in percentage, which is average number of words that need to change for successful attack. Higher word perturbation indicates less robustness, and this metric is also dependent on length of the text.
Furthermore, average number of queries i.e. number of times attacks recipes sends perturbed

text sample input to target models for successful attacks. Higher number of queries is a sign robustness.
Finally, the perturbation score is a confidence score or a predicted probability of the target model under attack. Lower average perturbation score of only successful attacks can be a signal of better robustness.

5.6 Threat Model

In this experiment, attack recipes are chosen based on level of knowledge, target, perturbation level, and assumptions. Considering most attacks are done under black box settings, the threat agent is only exposed to the target model output and test samples from the same corpus. Under this black box setting, a threat agent can only query the target model with perturbed inputs and get the corresponding confidence score or prediction. In addition, the threat agent is unaware of the model architecture, parameters, or training data. And, the agent can only query the target model with provided inputs.
As test samples are from the original corpus of the data, it is assumed that present test samples have been taken from different corpora. It is also assumed that attackers have access to similar baseline models. Because, similar tokenizer are used for tokenization in both while training and attacking, i.e. from the huggingface [22] an open source python package for language models. Evaluation of the model has performed for binary classification task, hence the attacks are mainly un-targeted but, because of binary classification it is assumed as targeted classification. And, only word level and multi-level which includes word and char level of word perturbation are considered. As, most of attacks are basically word level and character level hence, clear the intent of this selection.
The robustness of target models is evaluated against four attack recipes mentioned in 4.2, hence it is assumed that threat agent has access to those attack recipes. And, the performance of model could be different if some other attack recipes are utilized.
Furthermore, it is also assumed that the attack test samples are syntactically and semantically similar as per the claimed based on the attack recipes studies, hence, similarity metrics are not included in this experiment.
The model is trained of mentioned settings 5.4.1, and model can perform better or worse in different settings. Moving forward, about the transferability of the proposed approach , as the experiment is performed on two different language models differ in pre-training, it is assumed that the approach would also perform the same if some other language models are utilized.

6 Experiment Result

6.1 Analysis of Result

The result presented in tables 6.1 and 6.2 demonstrates that MTBERT has outperformed all the other language models in all metrics. Moreover, proposed approach models i.e. MTBERT and MTDistilBERT also performed better than the respective baseline models, i.e. BERT and DistilBERT. As shown in figure 6.1, MT BERT model showed 0-2% improvement in accuracy over the baseline model in both datasets and 20-30% improvement in accuracy under attack. The proposed model also shown comparatively higher requirement for number of queries and word perturbations. Furthermore, the drop in accuracy is lowest in proposed approaches than baseline models.

Attack Recipe	Model	Ori. Acc.(%)	Acc. und Attack(%)	Acc. Succ. Rate(%)	Avg. Pert. Word(%)	Avg. No. Queries
BAE	BERT	93.67	33.93	63.77	**3.78**	242.24
	DistilBERT	92.85	33.55	63.87	3.57	238.49
	MT BERT	**94.13**	**56.45**	**40.03**	3.55	198.26
	MT DistilBERT	93.17	53.60	42.47	3.35	**284.94**
PWWS	BERT	93.67	0.60	99.36	3.97	749.33
	DistilBERT	92.85	1.70	98.17	4.07	750.24
	MT BERT	**94.13**	**23.20**	**75.35**	**5.70**	**890.84**
	MT DistilBERT	93.17	17.82	80.88	5.38	866.78
TextBugger	BERT	93.67	2.30	97.54	22.04	235.27
	DistilBERT	92.85	5.57	93.94	21.30	259.16
	MT BERT	**94.13**	**35.13**	**62.68**	**28.57**	**449.96**
	MT DistilBERT	93.17	30.07	67.73	26.68	421.23
TextFooler	BERT	93.67	0.10	99.89	5.14	279.12
	DistilBERT	92.85	1.07	98.85	5.07	278.73
	MT BERT	**94.13**	**30.92**	**67.16**	**8.04**	**720.77**
	MT DistilBERT	93.17	25.48	72.64	7.54	613.91

Table 6.1: IMDB dataset: MTBERT model has performed well overall followed by MTDistilBERT.

Attack Recipe	Model	Ori. Acc.(%)	Acc. und Attack(%)	Acc. Succ. Rate(%)	Avg. Pert. Word(%)	Avg. No. Queries
BAE	BERT	94.17	67.53	28.30	**20.30**	82.10
	DistilBERT	94.41	67.66	28.34	18.99	79.12
	MT BERT	**95.64**	**77.62**	**18.83**	16.57	**88.64**
	MT DistilBERT	95.29	74.04	22.30	19.46	86.47
PWWS	BERT	94.17	24.68	73.80	**20.16**	215.04
	DistilBERT	94.41	25.68	72.80	19.18	210.62
	MT BERT	**95.64**	**59.16**	**38.14**	17.93	**236.90**
	MT DistilBERT	95.29	48.75	48.84	18.66	227.98
TextBugger	BERT	94.17	30.66	66.69	**24.87**	82.55
	DistilBERT	94.41	25.36	73.14	22.21	76.57
	MT BERT	**95.64**	**65.61**	**31.39**	23.29	**114.25**
	MT DistilBERT	95.29	53.99	43.34	24.54	96.14
TextFooler	BERT	94.17	7.62	91.90	22.48	180.55
	DistilBERT	94.41	7.58	91.98	21.37	164.40
	MT BERT	**95.64**	**54.05**	**43.48**	21.25	**282.66**
	MT DistilBERT	95.29	40.90	57.06	**24.19**	211.85

Table 6.2: Covid-19 fake tweets Dataset: MTBERT model has performed well overall followed by MTDistilBERT.

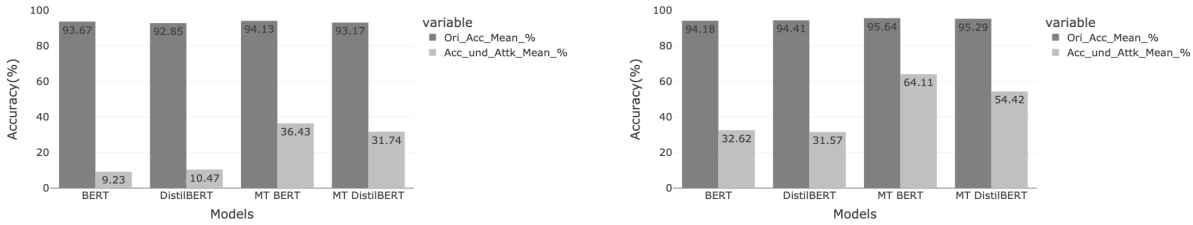

Figure 6.1: Comparative bar plot of original accuracy and accuracy under attack in different dataset. MTBERT showed better performance in both metrics followed by MTDistilBERT. And, highest drop is seen in baseline models (BERT and DistilBERT).

Considering number of queries, it has been observed that overall MTBERT shown highest requirement for number queries to get attacked as per the figure 6.2. As shown in box plot 6.3, the the interquartile range of MTBERT and MTDistilBERT model is comparatively broader than any other models and has also shown incident crossing 2000 number of queries.

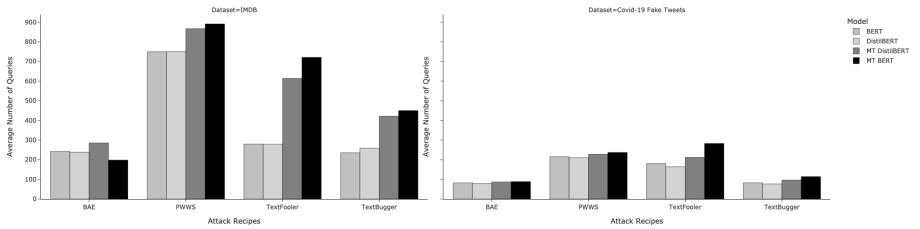

Figure 6.2: Bar plot of number of queries w.r.t dataset and attack recipes. PWWS has shown highest overall number of queries requirement followed by TextFooler. And, dependencies w.r.t to text length can also be observed in this plot.

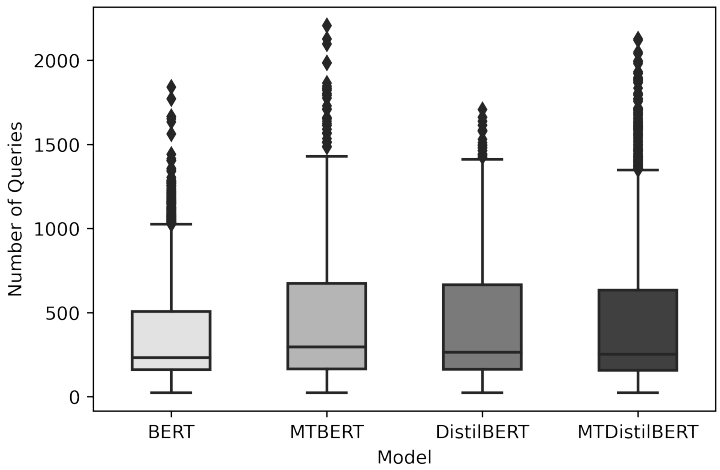

Figure 6.3: Box-plot of number of queries
The Proposed model are comparatively more broader and shown greater values for outliers than baseline models. This particular box plot is based on IMDB dataset and only successful attacks are considered.

If we compare the performances of attack recipes, BAE attacks has shown the lowest requirements for queries, followed by TextBugger, however, it is the least effective in attacking. PWWS requires a significantly higher number of queries followed by TextFooler and is also found to be more effective in attacks. Furthermore, both attack recipes are dependent on the text length, if we observe the considerable differences in queries as per dataset shown in figure 6.2. In TextFooler attack recipe, the difference between proposed and baseline queries also increases with respect to length of the text which shows its effectiveness depends on text length.
To understand the robustness of models with respect to word perturbation, the difference is less significant in between models as shown in figure 6.4. However, in the Covid-19 fake tweets dataset, the average word perturbation required by proposed models is higher, but, exactly opposite in IMDB dataset. It is quite clear that longer text requires comparatively lower word perturbation and also make sense. But, surprisingly TextBugger has shown the same perturbation rate and least affected by text length. However, TextBugger requires a greater number of words perturbed than TextFooler. BAE required the lowest number of perturbations, followed by PWWS.

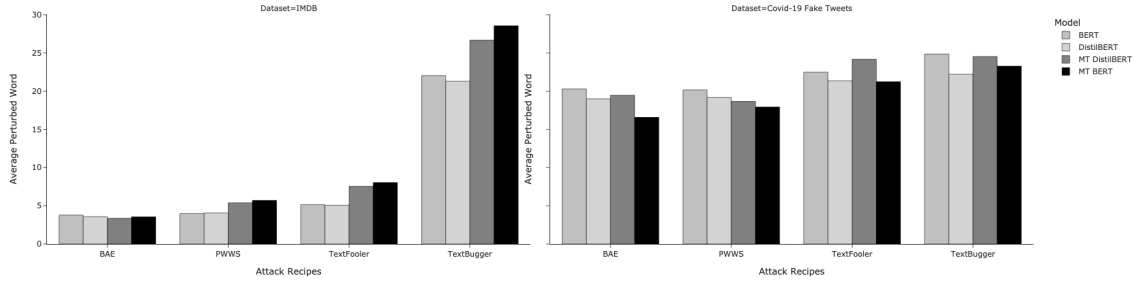

Figure 6.4: Bar plot of Word perturbation as per datasets and attack recipes. TextBugger has shown highest word perturbation in both the datasets and also shown less difference between both dataset. However, BAE has lower word perturbation followed by PWWS attack recipe. Covid-19 fake tweets dataset has shown significantly higher word perturbation requirements.

Intuitively, while running the experiments we logged perturbation score that is predicted probability under attack in other term confidence score. To answer the question, how much proposed and baseline model let the attacker to change the confidence score of successful attacks. As shown in figure 6.5, huge improvement is being observed in proposed model with respect to baseline model and the distribution of proposed model is comparatively more left shifted than baseline models. The proposed model does not let attack recipe go beyond a particular limit i.e. 0.70, however, baseline models are more uniform and shown less resilience. The range of MT BERT model is 0.50 to 0.55, but for BERT model it is 0.50 to 0.66, which is a significant improvement in the model.

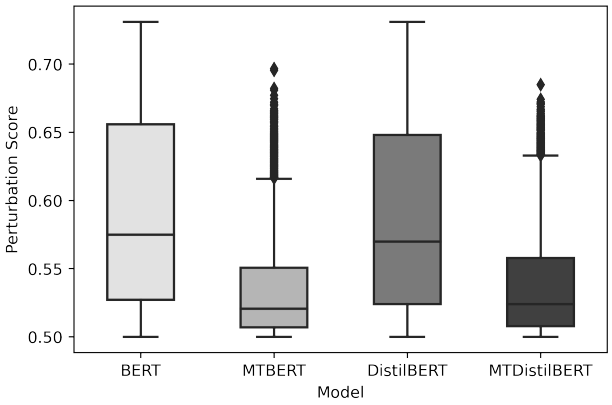

Figure 6.5: Box-plot of perturbation score. MT BERT model has managed attacks confidence score in the leaner interquartile range and does not cross 0.70 in a single instance which is a significant improvement over BERT.

Overall, model fine-tuned using proposed approach has shown significant improvements over conventional way of fine-tuning and MT BERT has shown maximum robustness as compared

to other models. Furthermore, PWWS and TextFooler appeared to be more successful attacks recipes compared to other attack recipes.

6.2 Discussing on Research Questions

According to the results of the experiment and analysis, the baseline model outperformed other models and demonstrated high robustness. In comparison, the proposed technique has shown considerable gains in generalization as well as a much lower attack success rate. In addition, a greater word perturbation rate, a smaller number of queries, and a lower perturbation score are also required.

Which answers to the question that language models built with noisy data and includes a loss function called consistency loss during training can improve generalization and resilience. Also, if appropriate data augmentation procedures are implemented and the semi-supervised methodology of fine-tuning language models, i.e., can exhibit comparable outcomes in the text-domain.

The most typical explanation for the improvement is the inclusion of noisy data in training samples and the corresponding loss function, which plays a critical role in optimizing model weights. Also, protect the model against over-fitting, which prevents the model from gaining a deep understanding of the data. Furthermore, by integrating many new words as noise during training, models also learns a larger word representation.

Erik et al. [13] published a research that supports the recommended strategy. According to their findings, a model trained on noisy data has poorer consistency than one trained on clean data, and the consistency decreases even more as the noise ratio increases. Consistency regularization is the process of integrating a loss function during training in order to decrease consistency loss and generate a more robust model. The notion of consistency regularization is used in both gradient-based approaches [40] and the suggested methodology.

Furthermore, the suggested method supports the findings of the Belinkov et al.[6] study, which claim that incorporating noisy data during training might improve the model's resilience. The experiment's transferability to other language models is partially demonstrated by the fact that it was conducted in two separate models, namely BERT and DistilBERT.

This research also found that while current language models may have performed admirably in a variety of tasks, they are no different from other DNN models in terms of resilience. However, on a more positive note, the findings show that language models still have room for growth in terms of generalization and resilience.

7 Limitation, Future Work and Conclusion

7.1 Limitations

Unlike image data where perturbation can be made imperceptible to humans, achieving even a similar level of imperceptibility is still a challenge in text domain due to its discrete nature. Most studies claimed to manage the same level of syntactic and semantic similarity of the original text, but in reality, the samples could be recognized by a human.
Attacks, data augmentation techniques, and semi-supervised training methodologies established in the image domain cannot be easily applied to the text domain, necessitating a separate effort and commitment to implement and evaluate them.
The suggested technique outperformed traditional models in the experiment; nonetheless, it cannot be said to be resistant to all forms of assaults. No defense plan can handle all forms of attacks since the nature of the attack is uncertain. Furthermore, incorporating the corresponding noisy data can only strengthen the robustness of the system against the assaults.
here are currently no common assessment criteria or frameworks for evaluating different research and strategy techniques in the text-domain, necessitating more investigation. In another case, recent studies have concentrated on improving model generalization rather than examining the robustness of such techniques.

7.2 Future Work

This approach could be evaluated using more recent state-of-the-art language models, and the work could also focus on the hypothesis that extreme pre-training of language models can hurt their robustness. And in another case, the effectiveness of individual augmentation techniques on robustness can be studied.
In the experiment, training data is used to create augmented prominent unlabeled data, however, huge amounts of unlabeled data could be used instead and performance could be evaluated. In another case, including the adversarial samples generated by attack recipes during training can also be evaluated.
The quantitative comparison of gradient-based adversarial training and the proposed fine-tuning approach remains an open question. Moving forward, including both techniques toward robustness can also be evaluated in the future.

In another scenario, more research is needed to build a common robustness verification framework that can be used to compare various methodologies.

7.3 Conclusion

The aim of this study was to do a quantitative comparison of the suggested semi-supervised strategy to fine-tuning language models in terms of generalization and resilience. Also, focus on improving the resilience of the language model. Observing the efficacy of significant unlabeled adversarial data, which can lead to improved resilience. In this experiment, various attack recipes such as PWWS, BAE, TextFooler, and TextBugger, data augmentation techniques such as word synonym, context augmentation, and back translation, and two language models such as BERT and DistilBert were utilized.
the results of the experiment revealed that the proposed strategy of fine-tuning improved initial accuracy by 0-2% and accuracy under assault by 20-30%. Furthermore, it demonstrated greater requirements in terms of word perturbation and number of queries, demonstrating its robustness against traditional fine-tuning methods. In terms of confidence score, the suggested technique is more robust to attacks, which is a substantial improvement in the model.
Furthermore, this research first highlighted that language models are sensitive to adversarial attacks, and then went on to show that language models have a lot of room for development either in terms robustness. The experiment also provides the direction of developing defenses against adversarial attacks, which can be examined in more detail.

Bibliography

[1] Naveed Akhtar and Ajmal Mian. "Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey". In: *arXiv:1801.00553 [cs]* (Feb. 2018). arXiv: 1801.00553 [cs].

[2] Felipe Almeida and Geraldo Xexéo. "Word Embeddings: A Survey". In: *ArXiv* (2019).

[3] Moustafa Alzantot et al. "Generating Natural Language Adversarial Examples". In: *arXiv:1804.07998 [cs]* (Sept. 2018). arXiv: 1804.07998 [cs].

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: (Sept. 2014).

[5] Rongzhou Bao, Jiayi Wang, and Hai Zhao. "Defending Pre-trained Language Models from Adversarial Word Substitution Without Performance Sacrifice". In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, Aug. 2021, pp. 3248–3258. DOI: 10.18653/v1/2021.findings-acl.287.

[6] Yonatan Belinkov and Yonatan Bisk. "Synthetic and Natural Noise Both Break Neural Machine Translation". In: *arXiv:1711.02173 [cs]* (Feb. 2018). arXiv: 1711.02173 [cs].

[7] Daniel Cer et al. "Universal Sentence Encoder". In: *arXiv:1803.11175 [cs]* (Apr. 2018). arXiv: 1803.11175 [cs].

[8] Hongge Chen et al. "Robustness Verification of Tree-based Models". In: *arXiv:1906.03849 [cs, stat]* (Dec. 2019). arXiv: 1906.03849 [cs, stat].

[9] Gobinda G. Chowdhury. "Natural Language Processing". In: *Annual Review of Information Science and Technology* 37.1 (2003), pp. 51–89. ISSN: 1550-8382. DOI: 10.1002/aris.1440370103.

[10] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv:1810.04805 [cs]* (May 2019). arXiv: 1810.04805 [cs].

[11] Javid Ebrahimi et al. "HotFlip: White-Box Adversarial Examples for Text Classification". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 31–36. DOI: 10.18653/v1/P18-2006.

[12] Steffen Eger et al. "Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol-*

ume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 1634–1647. DOI: 10.18653/v1/N19-1165.

[13] Erik Englesson and Hossein Azizpour. "Consistency Regularization Can Improve Robustness to Label Noise". In: *arXiv:2110.01242 [cs, stat]* (Oct. 2021). arXiv: 2110.01242 [cs, stat].

[14] Ji Gao et al. "Black-Box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers". In: *arXiv:1801.04354 [cs]* (May 2018). arXiv: 1801.04354 [cs].

[15] Siddhant Garg and Goutham Ramakrishnan. "BAE: BERT-based Adversarial Examples for Text Classification". In: *arXiv:2004.01970 [cs]* (Oct. 2020). arXiv: 2004.01970 [cs].

[16] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *arXiv:1412.6572 [cs, stat]* (Mar. 2015). arXiv: 1412.6572 [cs, stat].

[17] Alex Graves, Greg Wayne, and Ivo Danihelka. "Neural Turing Machines". In: *arXiv:1410.5401 [cs]* (Dec. 2014). arXiv: 1410.5401 [cs].

[18] Zellig S. Harris. "Distributional Structure". In: WORD 10.2-3 (Aug. 1954), pp. 146–162. ISSN: 0043-7956, 2373-5112. DOI: 10.1080/00437956.1954.11659520.

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the Knowledge in a Neural Network". In: *arXiv:1503.02531 [cs, stat]* (Mar. 2015). arXiv: 1503.02531 [cs, stat].

[20] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[21] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification". In: *arXiv:1801.06146 [cs, stat]* (May 2018). arXiv: 1801.06146 [cs, stat].

[22] *Hugging Face – The AI Community Building the Future.* https://huggingface.co/.

[23] Aminul Huq and Mst Tasnim Pervin. "Adversarial Attacks and Defense on Texts: A Survey". In: (May 2020).

[24] Robin Jia et al. "Certified Robustness to Adversarial Word Substitutions". In: *arXiv:1909.00986 [cs]* (Sept. 2019). arXiv: 1909.00986 [cs].

[25] Haoming Jiang et al. "SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), pp. 2177–2190. DOI: 10.18653/v1/2020.acl-main.197. arXiv: 1911.03437.

[26] Di Jin et al. "Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.05 (Apr. 2020), pp. 8018–8025. ISSN: 2374-3468. DOI: 10.1609/aaai.v34i05.6311.

[27] Spyridon Kardakis et al. "Examining Attention Mechanisms in Deep Learning Models for Sentiment Analysis". In: *Applied Sciences* 11.9 (Jan. 2021), p. 3883. DOI: 10.3390/app11093883.

[28] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *arXiv:1408.5882 [cs]* (Sept. 2014). arXiv: 1408.5882 [cs].

[29] Zhenzhong Lan et al. "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations". In: *arXiv:1909.11942 [cs]* (Feb. 2020). arXiv: 1909.11942 [cs].

[30] Jinfeng Li et al. "TextBugger: Generating Adversarial Text Against Real-world Applications". In: *Proceedings 2019 Network and Distributed System Security Symposium* (2019). DOI: 10.14722/ndss.2019.23138. arXiv: 1812.05271.

[31] Jiwei Li, Will Monroe, and Dan Jurafsky. "Understanding Neural Networks through Representation Erasure". In: *arXiv:1612.08220 [cs]* (Jan. 2017). arXiv: 1612.08220 [cs].

[32] L. Li et al. "BERT-ATTACK: Adversarial Attack Against BERT Using BERT". In: *EMNLP*. 2020. DOI: 10.18653/v1/2020.emnlp-main.500.

[33] Xiaodong Liu et al. "Adversarial Training for Large Neural Language Models". In: *arXiv:2004.08994 [cs]* (Apr. 2020). arXiv: 2004.08994 [cs].

[34] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *arXiv:1907.11692 [cs]* (July 2019). arXiv: 1907.11692 [cs].

[35] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *arXiv:1508.04025 [cs]* (Sept. 2015). arXiv: 1508.04025 [cs].

[36] Edward Ma. *Nlpaug*. Jan. 2022.

[37] Andrew L. Maas et al. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150.

[38] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *arXiv:1301.3781 [cs]* (Sept. 2013). arXiv: 1301.3781 [cs].

[39] George A. Miller. "WordNet: A Lexical Database for English". In: *Communications of the ACM* 38.11 (Nov. 1995), pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748.

[40] Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. "Adversarial Training Methods for Semi-Supervised Text Classification". In: *arXiv:1605.07725 [cs, stat]* (May 2017). arXiv: 1605.07725 [cs, stat].

[41] Takeru Miyato et al. "Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning". In: *arXiv:1704.03976 [cs, stat]* (June 2018). arXiv: 1704.03976 [cs, stat].

[42] John X. Morris et al. "TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP". In: *arXiv:2005.05909 [cs]* (Oct. 2020). arXiv: 2005.05909 [cs].

[43] E. A. Nadaraya. "On Estimating Regression". In: *Theory of Probability & Its Applications* 9.1 (Jan. 1964), pp. 141–142. ISSN: 0040-585X. DOI: 10.1137/1109020.

[44] Usman Naseem et al. "A Comprehensive Survey on Word Representation Models: From Classical to State-Of-The-Art Word Representation Language Models". In: *arXiv:2010.15036 [cs]* (Oct. 2020). arXiv: 2010.15036 [cs].

[45] Maria-Irina Nicolae et al. "Adversarial Robustness Toolbox v1.0.0". In: *arXiv:1807.01069 [cs, stat]* (Nov. 2019). arXiv: 1807.01069 [cs, stat].

[46] Nicolas Papernot et al. "Crafting Adversarial Input Sequences for Recurrent Neural Networks". In: *arXiv:1604.08275 [cs]* (Apr. 2016). arXiv: 1604.08275 [cs].

[47] Parth Patwa et al. "Fighting an Infodemic: COVID-19 Fake News Dataset". In: *arXiv:2011.03327 [cs]* (Mar. 2021). arXiv: 2011.03327 [cs].

[48] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

[49] Matthew E. Peters et al. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202.

[50] B. T. Polyak and A. B. Juditsky. "Acceleration of Stochastic Approximation by Averaging". In: *SIAM Journal on Control and Optimization* 30.4 (July 1992), pp. 838–855. ISSN: 0363-0129. DOI: 10.1137/0330046.

[51] Alec Radford et al. "Improving Language Understanding by Generative Pre-Training". In: (), p. 12.

[52] Graham Rawlinson. "The Significance of Letter Position in Word Recognition". In: *IEEE Aerospace and Electronic Systems Magazine* 22.1 (Jan. 2007), pp. 26–27. ISSN: 1557-959X. DOI: 10.1109/MAES.2007.327521.

[53] Shuhuai Ren et al. "Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 1085–1097. DOI: 10.18653/v1/P19-1103.

[54] Yankun Ren et al. "Generating Natural Language Adversarial Examples on a Large Scale with Generative Models". In: (Mar. 2020).

[55] G. Salton, A. Wong, and C. S. Yang. "A Vector Space Model for Automatic Indexing". In: *Communications of the ACM* 18.11 (Nov. 1975), pp. 613–620. ISSN: 0001-0782. DOI: 10.1145/361219.361220.

[56] Victor Sanh et al. "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter". In: *arXiv:1910.01108 [cs]* (Feb. 2020). arXiv: 1910.01108 [cs].

[57] Sachin Saxena. "TextDecepter: Hard Label Black Box Attack on Text Classifiers". In: *arXiv:2008.06860 [cs]* (Dec. 2020). arXiv: 2008.06860 [cs].

[58] Chenglei Si et al. "Better Robustness by More Coverage: Adversarial and Mixup Data Augmentation for Robust Finetuning". In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, Aug. 2021, pp. 1569–1576. DOI: 10.18653/v1/2021.findings-acl.137.

[59] Lichao Sun et al. "Adv-BERT: BERT Is Not Robust on Misspellings! Generating Nature Adversarial Samples on BERT". In: *arXiv:2003.04985 [cs]* (Feb. 2020). arXiv: 2003.04985 [cs].

[60] Christian Szegedy et al. "Intriguing Properties of Neural Networks". In: *arXiv:1312.6199 [cs]* (Feb. 2014). arXiv: 1312.6199 [cs].

[61] Antti Tarvainen and Harri Valpola. "Mean Teachers Are Better Role Models: Weight-averaged Consistency Targets Improve Semi-Supervised Deep Learning Results". In: *arXiv:1703.01780 [cs, stat]* (Apr. 2018). arXiv: 1703.01780 [cs, stat].

[62] Ashish Vaswani et al. "Attention Is All You Need". In: *arXiv:1706.03762 [cs]* (Dec. 2017). arXiv: 1706.03762 [cs].

[63] Wenqi Wang et al. "Towards a Robust Deep Neural Network in Texts: A Survey". In: *arXiv:1902.07285 [cs]* (Apr. 2021). arXiv: 1902.07285 [cs].

[64] Xiaoyong Yuan et al. "Adversarial Examples: Attacks and Defenses for Deep Learning". In: *arXiv:1712.07107 [cs, stat]* (July 2018). arXiv: 1712.07107 [cs, stat].

[65] Yuan Zang et al. "Word-Level Textual Adversarial Attacking as Combinatorial Optimization". In: (Oct. 2019). DOI: 10.18653/v1/2020.acl-main.540.

[66] Wei Emma Zhang et al. "Adversarial Attacks on Deep Learning Models in Natural Language Processing: A Survey". In: *arXiv:1901.06796 [cs]* (Apr. 2019). arXiv: 1901.06796 [cs].

[67] Yi Zhou et al. "Defense against Adversarial Attacks in NLP via Dirichlet Neighborhood Ensemble". In: *arXiv:2006.11627 [cs]* (June 2020). arXiv: 2006.11627 [cs].

[68] Danqing Zhu et al. "AT-BERT: Adversarial Training BERT for Acronym Identification Winning Solution for SDU@AAAI-21". In: *arXiv:2101.03700 [cs]* (Jan. 2021). arXiv: 2101.03700 [cs].

8 APPENDIX

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 495.44       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   35C    P0    32W / 250W |    375MiB / 16280MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

Figure 8.1: GPU details

Layer (type)	Output Shape	Param #	Connected to
input_ids (InputLayer)	[(None, 100)]	0	[]
attention_mask (InputLayer)	[(None, 100)]	0	[]
tf_bert_model (TFBertModel)	TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 100, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)	109482240	['input_ids[0][0]', 'attention_mask[0][0]']
tf.__operators__.getitem (SlicingOpLambda)	(None, 768)	0	['tf_bert_model[0][0]']
dropout_37 (Dropout)	(None, 768)	0	['tf.__operators__.getitem[0][0]']
dense (Dense)	(None, 64)	49216	['dropout_37[0][0]']
dense_1 (Dense)	(None, 32)	2080	['dense[0][0]']
dense_2 (Dense)	(None, 2)	66	['dense_1[0][0]']

```
Total params: 109,533,602
Trainable params: 109,533,602
Non-trainable params: 0
```

Figure 8.2: Layer overview of BERT model used in the experiment

Layer (type)	Output Shape	Param #	Connected to
input_ids (InputLayer)	[(None, 100)]	0	[]
attention_mask (InputLayer)	[(None, 100)]	0	[]
tf_distil_bert_model (TFDistilBertModel)	TFBaseModelOutput(last_hidden_state=(None, 100, 768), hidden_states=None, attentions=None)	66362880	['input_ids[0][0]', 'attention_mask[0][0]']
tf.__operators__.getitem_1 (SlicingOpLambda)	(None, 768)	0	['tf_distil_bert_model[0][0]']
dropout_57 (Dropout)	(None, 768)	0	['tf.__operators__.getitem_1[0][0]']
dense_3 (Dense)	(None, 64)	49216	['dropout_57[0][0]']
dense_4 (Dense)	(None, 32)	2080	['dense_3[0][0]']
dense_5 (Dense)	(None, 2)	66	['dense_4[0][0]']

Total params: 66,414,242
Trainable params: 66,414,242
Non-trainable params: 0

Figure 8.3: Layer diagram of DistilBERT model used in the experiment