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)

Acknowledgement

I would like to express my deepest gratitude to my supervisors Prof. Dr. Andreas Mauthe and Mr. Alexander Rosenbaum for accepting the proposed master thesis topic and provided invaluable assistance throughout the project.
Without my mother's and wife's encouragement and support, I would not have been able to finish this endeavour. I am grateful to my wife for patiently caring for our daughter on her own.
Finally, I'd want to express my gratitude to my beloved father, who passed away during the course. Without him, I couldn't have imagined coming here in a new country and writing the final words of my master's thesis. Your priceless memories and lessons inspired me during this period and will always be with me.

Zusammenfassung

Sprachmodelle haben bei verschiedenen Aufgaben der natürlichen Sprachverarbeitung Spitzenleistungen gezeigt. Jüngste Forschungen haben gezeigt, dass sie Schwächen gegenüber Angriffen haben, bei denen unmerkliches Rauschen im Text dazu führen kann, dass sich die Modelle unerwartet verhalten und ihre 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 Modells zu vergleichen, das mit der herkömmlichen und der vorgeschlagenen Methode feinabgestimmt wurde. Dieser Bericht trägt auch dazu bei, den Umfang der Verbesserungen in Sprachmodellen aufzuzeigen. Das Experiment wurde mit den Sprachmodellen BERT und DistilBERT an zwei Datensätzen durchgeführt. Laut Experiment zeigen die mit dem vorgeschlagenen Ansatz feinabgestimmten Modelle eine Verbesserung von 0$\sim$2% und 20$\sim$30% bei der ursprünglichen Genauigkeit und der Genauigkeit unter Angriffen gegenüber dem konventionellen Ansatz.

Abstract

Language models have shown state-of-the-art performances in various natural language processing tasks. Recent research have shown their weaknesses against adversarial attacks, where imperceptible noise in text can sabotage models to behave unexpectedly and can severely degrade their performance under attacks. 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 that is fine-tuned using conventional and proposed method. This report also contribute in revealing the scope of improvements in language models. The experiment was conducted using BERT and DistilBERT language models on two datasets. As per experiment, the models fine-tuned by proposed approach shows 0$\sim$2% and 20$\sim$30% improvement in original accuracy and accuracy under attacks over conventional approach, respectively.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Problem Statements

The Deep Neural Network (DNN) is the 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 gives it an advantage over other machine learning algorithms [24]. In Natural Language Processing (NLP), recent advancement in DNN language models has led to state-of-the-art transformer-based models like BERT, DistilBERT, RoBERTa, ALBERT etc. [10, 31, 36, 62], which have outperformed in a wide range of NLP tasks and are consequently the most widely adopted models nowadays.

But, studies have also revealed the weakness in DNN models against adversarial attacks [1, 24, 66, 73, 75] which can negatively impact its value and raised concern of trust, safety, and security. Szegedy *et al.* [66] first revealed this vulnerability in computer vision domain. Their study demonstrated that imperceptibly small perturbations in the input image can fool deep neural networks with high probability which they referred as *Adversarial Samples*. Attacks that manipulate a machine learning model with these examples are called *Adversarial attacks* [50]. And, this study received a considerable amount of attention from researchers in all fields.

In real life, people are tend to believe the fact written over social website without verifying it. To find against such campaigns, various DNN based fake news detection approaches are proposed and utilized. However, the attackers could use natural tweets to create adversarial examples to gain political advantage or disrupt society without being detected.

The presence of adversarial attack in the text domain was first confirmed in the study presented by Papernot *et al.* [51]. Their study has shown a small word-level change in sentence structure can sabotage the DNN models. Later, various studies were published detailing different techniques to craft adversarial text samples which are comparatively more than defensive mechanisms.

Furthermore, recent works also revealed that the performance of language models suffers severely under adversarial attacks [17, 34, 45]. Jin *et al.* [28] 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 concerns on the conventional way of fine-tuning language models which is not sufficient when robustness is concerned. At the same time, it opens a scope of work towards the robust-

ness enhancement of language models.
The topic of adversarial attacks is more extensively studied in the computer vision domain than that of NLP [70]. And, in the text domain, the focus is more on approaches related to creating adversarial samples than defense mechanisms. The reason is the generation of adversarial samples in the text domain are comparatively more challenging due to their discrete nature and requirement for maintaining the semantic [34]. Furthermore, there are few studies that aim at improving the robustness of language models, but, mostly revolve around gradient-based methods [26, 43, 77], which may be inspired by seeing its effectiveness in the computer vision domain.

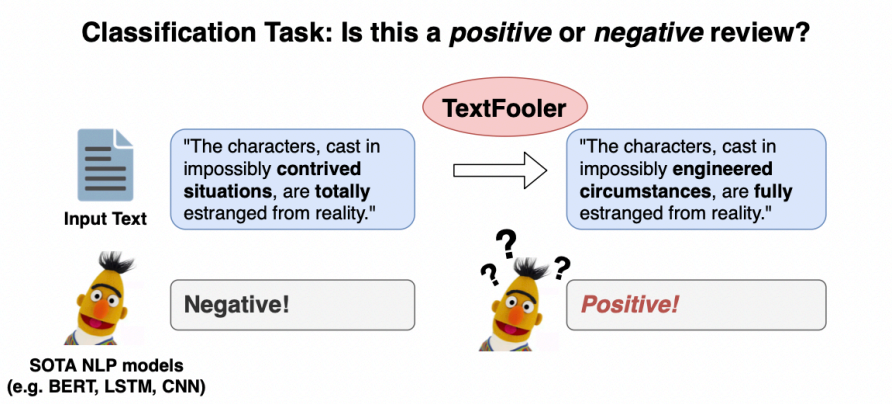

Figure 1.1: Adversarial attack presented by TextFooler [28], where word-level changes in input text influenced the prediction.

Furthermore, most of the text classification tasks such as fake news detection suffer from a lack of labelled data in comparison to unlabelled data. A number of semi-supervised training approaches have been proposed to overcome these challenges, and have shown significant improvements in performance. In 2018, Tarvaian *et al.* [67] proposed a semi-supervised approach that has performed well in the image domain and claimed to provide a robust model. However, its efficiency in the 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 that including noisy data in training samples may result in robust models. Until now, no study has been conducted to investigate the mentioned semi-supervised approach in fine-tuning the language models which includes data augmentation technique with a focus on robustness enhancement.
Therefore, this master thesis study proposes a less complex approach of fine-tuning i.e. mean teacher approach that can lead to a comparatively robust model without compromising in original accuracy. As a next step, a quantitative experiment is conducted to compare the performance of proposed approach with conventional method of fine-tuning. And, language model

capabilities such as back translation and context rewriting to generate prominent adversarial unlabelled samples are also utilized for training purposes. Thus, the proposed study aims to investigate the research question and hypothesis discussed in the next section.

1.2 Research Question and Objectives

The purpose of this research is to study the effect on performance of language models fine-tuned by the proposed semi-supervised mean teacher method which utilizes of prominent adversarial unlabelled data. The adversarial unlabelled data is created using available training data by using the language model capabilities of context rewriting, back-translation, and synonym word identification. Hence, The major research question of this thesis is:

Is the proposed mean-teacher semi-supervised fine-tuning and the use of prominent adversarial unlabelled data an effective method to improve the robustness of language models, specifically concerning accuracy under attack without compromising the original accuracy in contrast to the conventional fine-tuning approach?

It is hypothesized that the proposed means-teacher form of semi-supervised learning and the use of prominent unlabelled adversarial data will produce a robust language model against attack mainly without sacrificing original accuracy.
The robustness of this experiment will be evaluated using metrics such as original accuracy, accuracy under attack, rate of word perturbation, queries required, and attack success rate, which are discussed briefly in figure 5.5. Hence, it is expected that the suggested technique will have greater original accuracy, accuracy under attack, and will need more queries and perturbations to be attacked. The model performance is evaluated using four different attack recipes discussed in the figure 4.2.
This thesis attempts to answer the question by constructing a teacher model using proposed semi-supervised approaches. Then, comparing the performance of both conventional and proposed fine-tuning methods in terms of different evaluation metrics.
Since, proposed approach utilizes data augmentation and exponential moving average (EMA) techniques. It is expected that these technique will play crucial role in robustness enhancement of language models as discussed in section 4.1.
Moving forward, the experiment also attempts to understand the robustness of models by examining the confidence score distribution. A model that does not allow attack recipes to be misclassified with high confidence is a sign of robustness. Furthermore, the transferability of proposed approach is verified by performing similar experiment with two differently pre-trained models i.e. BERT, DistilBERT.

1.3 Thesis Contribution

This proposed study provided an important opportunity to advance the understanding of language models, data augmentation, and various adversarial attacks in the text domain. Moreover, the study contributes to the assessment of language model performance in adverse situations so that mitigation techniques can be developed. By understanding the dynamics of adversarial attack recipes and how to defend against them, language models can become more resilient and the scope for improvement can be more obvious.
The model's resilience is determined by comparing its performance against four attack recipes and several metrics. Additionally, the transferability of the proposed method is demonstrated by doing a comparative experiment with two distinct pre-trained language models, BERT and DistilBERT.
Additionally, the use of adversarial text during training has been shown to improve the robustness of the model [32], but that it is time-consuming. The purpose of this thesis is also to study the performance of language models against attacks that use only basic augmentation techniques, which was a topic of open research until now.
This thesis also contributes to evaluating the effectiveness of an image-domain semi-supervised approach in the text-domain, i.e. the mean teacher approach, by incorporating text-specific noise strategies and investigating the constraints connected with it. Furthermore, this thesis reviews and analyses related work, and highlights the shortcomings.

1.4 Thesis Structure

This master thesis is further divided into 6 chapters i.e. Background, Related work, Proposed Methodology, Experiment Setup, Experiment Results, and Conclusion. To lay the groundwork for understanding the experiment and proposed approach, the subsequent section attempts to provide a foundational understanding of topics such as the evolution of text representation from TF-IDF to contextualized embeddings, transformer-based language models, and the architecture of BERT/DistilBERT models. Subsequently, the background and classification of adversarial attacks are addressed. Following that, there is a detailed description of the proposed strategy and attacking recipes. Afterwards, the experiment conditions, resources, and environment needed to conduct the experiment are discussed followed by the analysis of the observed results. The master thesis concludes with a discussion of limitations, future work, and the conclusion.

2 Background of Language Models and Adversarial Attacks

2.1 Text Representations

In natural language processing (NLP), researchers explore how a computerized system can understand and manipulate natural languages (speech or text) to perform various tasks [9] and aims to gain a human-level understanding of the text [49]. However, there was a major challenge in converting text into numbers or vectors in such a manner that semantic and syntactic information can be restrained. As shown in the 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 [61]. The shape of vectors are equal to the size of vocabulary, thus, this technique suffers severely 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 real-valued number are called word embeddings [2]. The distributed hypothesis is the main principle behind this type of representation [22], which posits that the words of similar contexts tend to have the same meanings. This type of word embedding can be utilized to perform various NLP tasks. Mikolov *et al.* [41] 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 previous and future words given as inputs. At the projection layer, each word's context is averaged as shown in figure 2.2. As the order of the words does not influence the projection of the word hence called bag-of-words.
Word2Vec also uses skip-gram to maximize the classification of a word based on another word in the sentence, as shown in figure 2.2. There are two major concerns with Word2Vec i.e. (1) It only preserves the local information of words, and (2) The semantics of a given word is entirely determined by the surrounding words. Later, Pennington *et al.* [53] proposed word

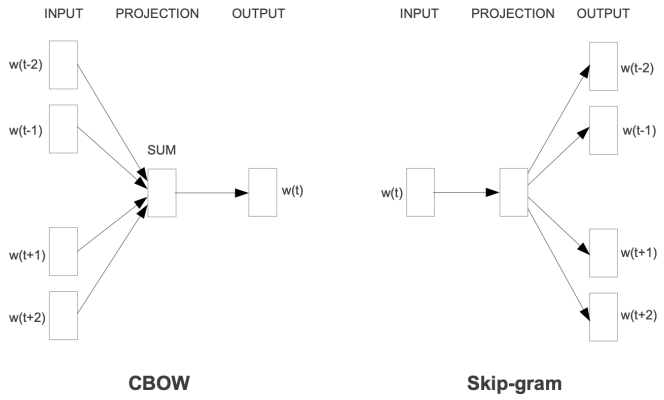

Figure 2.2: The CBOW architecture predicts the current word given past and future words, and the skip-gram architecture predicts the surrounding words given the current words.

embedding called Glove (Global Vectors for Word Representation) which was based on word global information and a big corpus of unlabelled datasets. The idea is to determine how frequently a word pair occurs together by using a co-occurrence matrix and factorising it 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 a log-bilinear regression model where the probability of the next word is determined by the previous word.
Both Word2Vec and Glove word embeddings are static and have a limitation called polysemy, which refers to the fact that the 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 but, however, provided a direction for working on contextualized word embeddings.

2.1.2 Contextualized Embeddings

Peters *et al.* [54] proposed the ELMO(Embedding from Language Model) as a method 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 was introduced by Vaswani *et al.* [68], Generative Pre-Training (GPT) [57], and Bidirectional Encoder Representation from Transformers (BERT) [10] utilized encoders or decoders instead of bidirectional LSTMs to create contextualized embedding. GPT uses decoder architecture where as BERT uses encoder architecture, however, the main concept of is based on ELMO, as shown in figure 2.3.

These approaches introduced the concept of *pre-training*, i.e. learning language representations from the huge corpora of unlabelled data in unsupervised fashion, and *fine-tuning*, i.e. optimizing weights further based on the downstream task. And, also facilitated the approach of transfer learning in NLP to get state-of-the-art performance with minimal training. A brief overview of the transformer model is required to understand the working of language models, so the next section discusses the transformer before language models.

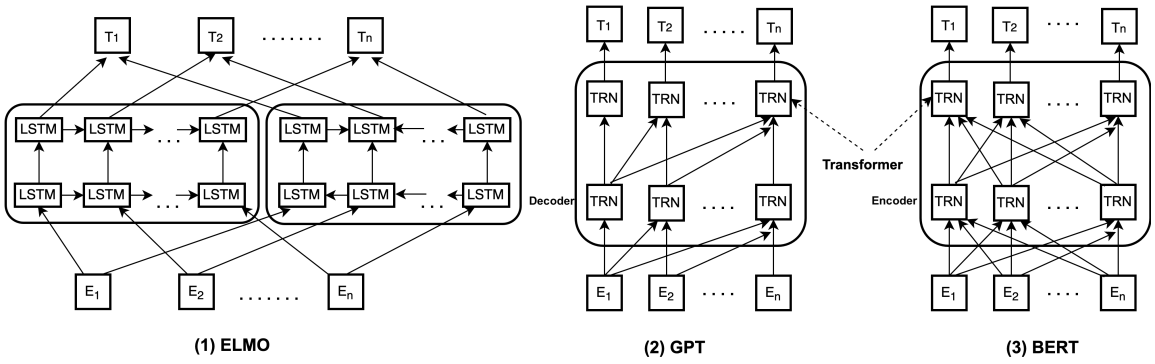

Figure 2.3: Different contextualized embedding models.
Pictorial representation of working of (1) ELMO(Embedding from language model), (2)Generative Pre-training (GPT), and (3) Bidirectional Encoder Representation From Transformers(BERT). TRN represents transformer encoder in BERT and decoder in GPT.

2.2 Transformers

There was a time when NLP tasks were solely based on sequential models like CNN, RNN, LSTM, and BiLSTM 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, Vaswani *et al.* [68] 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 one of the main principles behind the development of break-through models like BERT, GPT, and T5. The figure 2.4 illustrates the architecture of a transformer.

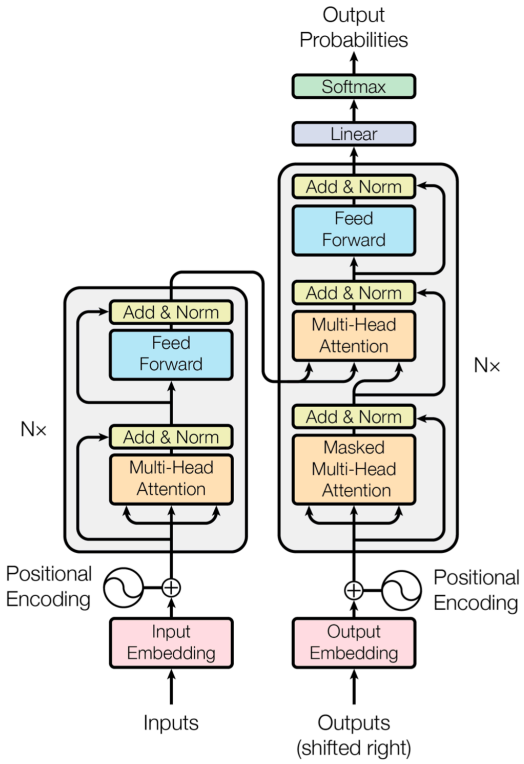

Figure 2.4: Transformer Model Architecture [68].

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 the figure 2.5, which is based on single or multi-headed attention, and the decoder makes predictions based on those contexts vectors. Vaswani *et al.* [68] proposed transformers architecture consists of 6 layers encoders stacked on top of each other and same applies to decoders. Encoder is composed of multi-head attention followed by layer normalization and feed forward networks. However, there is a slight difference in decoder i.e. masked multi-head attention layer is followed by multi-head attention layer as shown in figure 2.4.

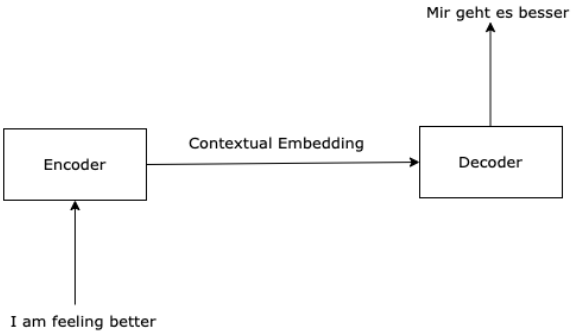

Figure 2.5: Basic example Encoder-Decoder.

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 the encoder learns the pattern of variable length input and generates a fixed shape output. A decoder, on the other hand, takes that fixed shape as input and gives the variable-length as output.

For example, the task is to translate the English sentence "I am good. How are you ?" to German language , given input sequence of tokens "I", "am", "good",...,"?" to encoder which generates the fixed shape representation z_1, z_2, z_3. Using this representation, the decoder outputs the "Mir Geht es gut. Wie Geht es dir ?" in the token format as shown in the figure 2.6.

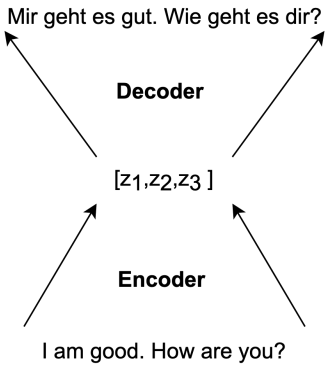

Figure 2.6: Basic working example of encoder-decoder.

Later, several sequence to sequence models utilize this encoder-decoder architecture to perform tasks such as text summarization, question answering, and machine translation. However, the encoder-decoder architecture in the transformer differs significantly.

A transformer encoder block is further divided into three layers as shown in the figure 2.7: multi-head attention, normalization layer, and feed-forward network. The input embedding

component converts the input text tokens into embedding vectors EM of shape d_{model}.

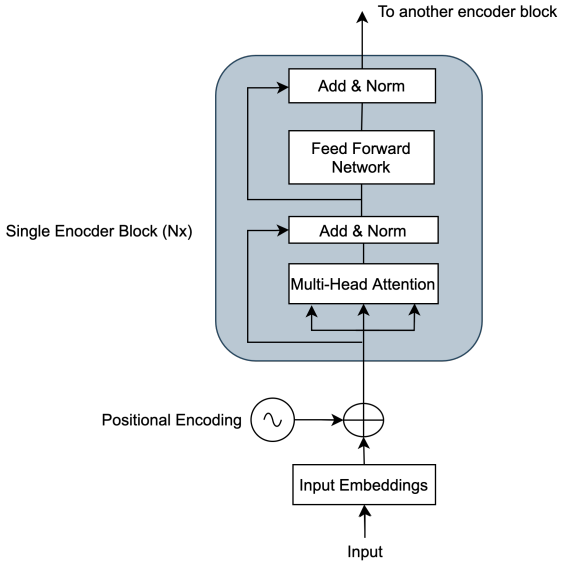

Figure 2.7: Different components of the encoder in the transformer.

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 the word order. The word order in the shape of a position vector will be added to the input embedding before the multi-head attention. Position vector must have the same dimension as word embedding d_{model}.

Two major constraints apply: First, the word embedding information should not severely disturbed and second, word position must be identical. According to Vaswani *et al.* [68], sine and cosine functions of different frequencies are used to form a geometric progression from 2π to $2\pi \cdot 10000$ are used for calculating the position vector (PV) of the words. In other words, the 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 pos & i are position and dimension respectively. Then, adding to the embeddings we get position encoding (PE)

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

2.2.2 Attention Mechanism

The human mind does not process all the available information in the environment, but rather, it focuses on specific information to complete the task and this biological mechanism is called attention. And, the major intention behind achieving the same in machine learning tasks. In 1964, Naradaya *et al.* [48] first introduced the idea of attention in the regression model, proposing a weighted function (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, Luong *et al.* [37] showed that attention mechanisms gained up to 5.0 BLEU over non-attention models in neural machine learning tasks. And, recently, Kardakis *et al.* [29] studied the same mechanism in the text classification task, i.e. sentiment analysis, and reported a 3.5% increase in accuracy.

Attention mechanism in transformer architecture model introduced to reduce computational complexity and facilitate parallel computation [68]. Different types of methods exist to calculate attention mechanisms such as similarity-based [19], dot products [37], scaled dot products [68], additives [4], etc. This report focuses on scaled dot product attention 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, and V is a query, key, and value vector which is created by the 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.

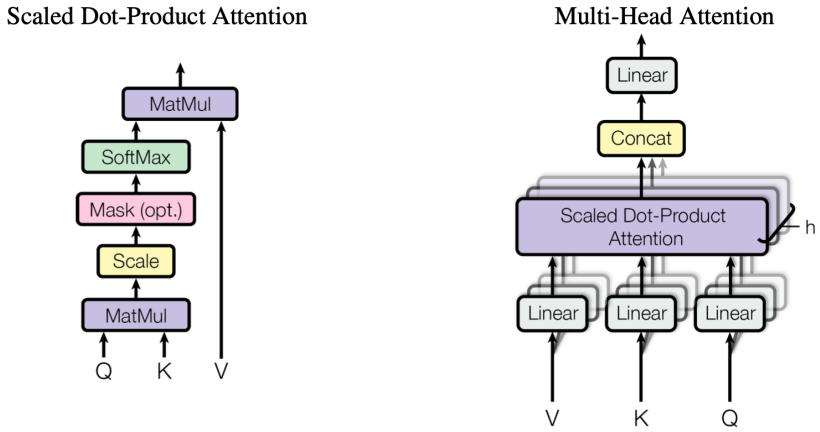

Figure 2.8: Scaled dot product and multi-head attention calculation flow diagram [68].

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, the 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).

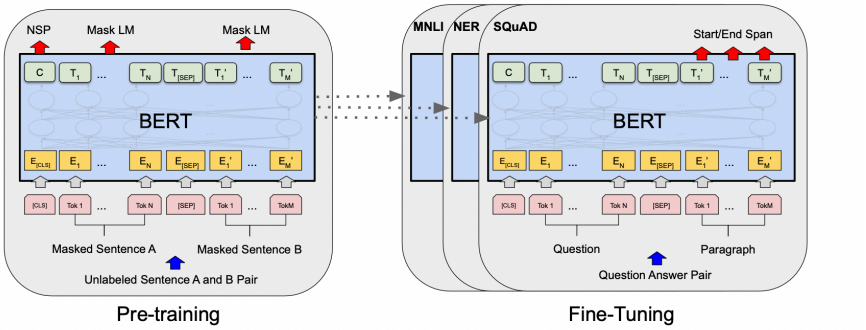

Figure 2.9: Pre-training and fine-tuning method of the BERT model [10]. In pre-training, unsupervised techniques such as next sentence prediction (NSP) and masked word prediction (MWP) are utilized. Pre-trained models are provided to perform further downstream tasks and weights are further optimized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

According to the figure, these language models follow two training processes, (1) *Pre-training* by using huge unlabelled 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 next 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.

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 [68] and ELMo [54] 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 the deep bidirectional representation of large unlabelled text in both rights 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. It's 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 a 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 unlabelled text, however, fine-tuning is still required for specific tasks.

To enable BERT models to handle a variety of downstream tasks, the input representations created by summing three different embeddings (position embeddings, segment embeddings, and token embeddings), as shown in the figure 2.10.

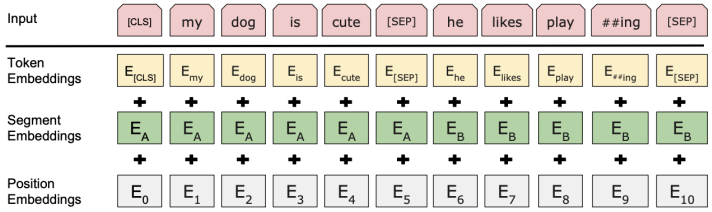

Figure 2.10: BERT model input includes three different embedding information to perform the various downstream tasks[10].

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.* [62]. 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 [23], which can be defined as a process of transferring knowledge from large to small models.

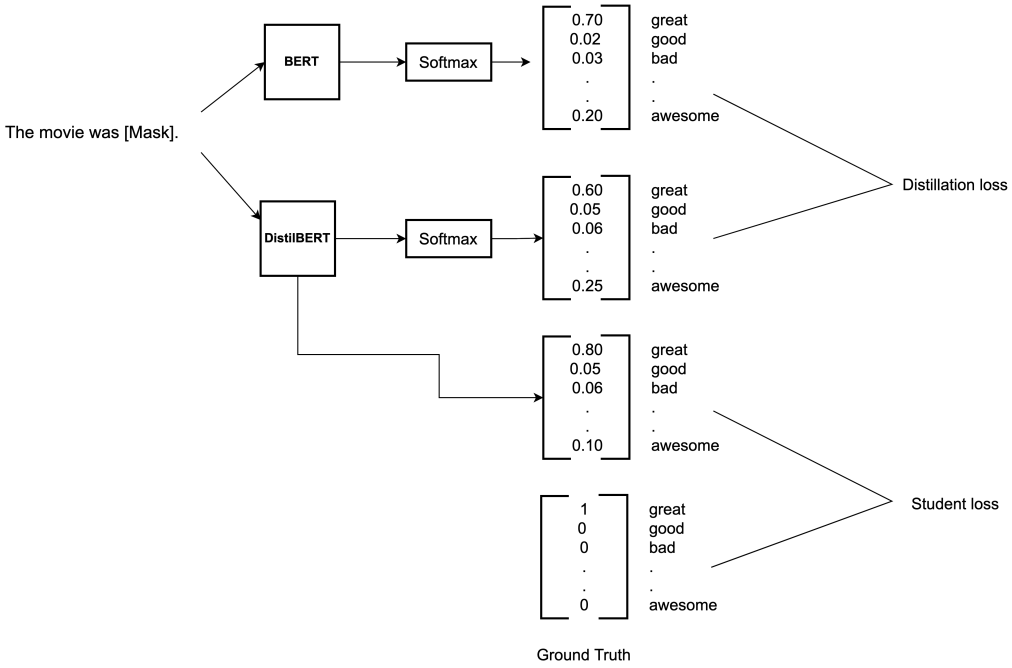

Figure 2.11: Pictorial diagram depicts the knowledge distillation process involved in DistilBERT.

As illustrated in the figure 2.11, a small model is trained to mimic the behaviour of a larger model. In the task of mask word prediction, masked sentences are supplied to both the BERT and DistilBERT models.

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 [62], which has proven it is possible to reduce the size of large language models with minimal compromise.

2.4 Adversarial Attacks

In the year 2013, Szegedy *et al.* [66] proposed study in the image-domain discovered that deep neural networks are vulnerable to perturbations, which can result in unexpected behaviours and such case was referred as adversarial attacks. It was first demonstrated by Papernot *et al.* [51] 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, 16, 17, 34, 59]. Sun *et al.* [65] proposed an 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.* [34] proposed an approach of utilizing the BERT model to generate words for replacements. Firstly, identifying the most important words i.e. the words in a sequence have high influence on the final output logit. Secondly, utilizing BERT model masked language model (MLM) capability to generate a word for replacements. According to their claims, the accuracy under attack is lower than 10% with less than 10% perturbation. However, those attacks have limitation of most of the time not following the semantics of original text.
In text domains, most adversarial attacks consist of two steps: 1) Identification of the most important words, and 2) Replacement them with suitable words. The reason why DNN models are vulnerable to attacks is still an open question, however, Goodfellow *et al.* [18] argued that it is due the linearity in 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 $||\delta|| < \epsilon$.
The 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 examples because humans can easily detect a minute change in 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, the number of queries, word perturbation, and attack success rate are discussed in length in section 5.5.

2.4.2 Types of Adversarial Attacks

Based on recent surveys, adversarial attacks can be classified based on level of knowledge, target, and level of perturbation [24, 70]. If attackers have complete knowledge of the system, then it would be called a white box attack. And, in another case, if the only model output is known, then it would be a black box attack. A white box gradient-based attack was first proposed by Ebrahimi *et al.* [13] to search for adversarial word/character substitutions. And, Jin *et al.* [28] have proposed black box word-level adversarial attacks and shown that BERT models are vulnerable to adversarial samples.

As mentioned in section 2.4.1, when an attacker intentionally attempts to manipulate the model

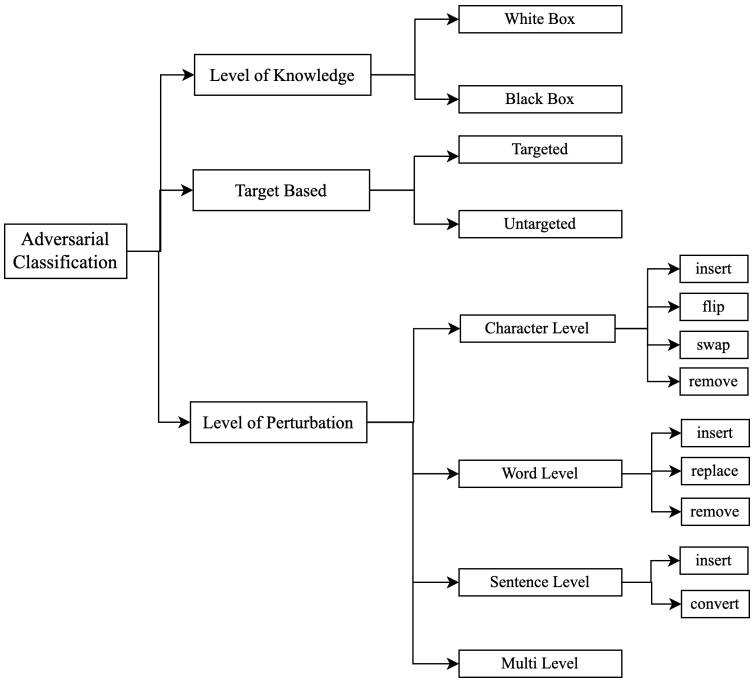

Figure 2.12: Different adversarial attack classification based on different aspects.

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

A word-level of perturbation may result in an attacker inserting, replacing, and removing words. TextDeceptor [63] proposed a 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.* [74] 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.* [14], a visual perturber called VIPER replaces input characters with their visual nearest neighbours in visual embedding space. At the sentence level, attackers try to convert the input sentence using the back-translation or paraphrasing technique and/or insert some sentences into the text. Yankun *et al.* [60] 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 defences is to achieve high test accuracy on both clean and adversarial examples [76]. 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 [70]. According to Goodfellow *et al.* [18], including high-quality adversarial examples in images can improve the robustness and generalization of machine learning models. Similarly, in the 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.* [32], 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.* [43], 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.* [35] proposed adversarial training of the BERT model using the virtual adversarial training (VAT) technique [44] during pre-training. However, their approach is computationally expensive and also based on gradient.

Furthermore, the mean teacher approach has shown comparative performance in the computer visualization domain and the model claimed to be comparatively robust [67]. 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, the proposed approach utilizes the language model capabilities of such as context rewriting and back translation as data augmentation techniques to create prominent adversarial unlabelled data.

Similar approaches have been utilized previously to generate adversarial samples. Siddhant *et al.* [17] proposed approach used BERT's masked language model (MLM) 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 Research on Adversarial Training

In the area of adversarial training and defence mechanisms against attack, various types of techniques are studied and proposed. In spite of this, research in the text-domain remains considerably lower than in the image-domain [70]. Furthermore, few papers have addressed the evaluation of language models against adversarial attacks.
The search for related work is performed on scholarly websites such as google scholar and arxiv. Furthermore, search texts consisted of semi-supervised, adversarial training, robustness, and language models. Moving forward, after getting the relevant paper further research were found based on connected paper service[1]. As per the survey on research paper conducted in this study, adversarial training is divided into three section for better understanding: 1) Gradient-based, 2) Augmentation-based and, and 3) Other. The proposed study is more related to augmentation-based adversarial training.

3.1 Gradient-Based

The objective of gradient-based approach is to optimize the adversarial loss of the model by applying perturbations in the embedding space [18, 26, 35, 43, 77]. But, most of these studies are focused on increasing generalization instead of enhancing and evaluating against attack.
Liu *et al.* [35] proposed a novel adversarial pre-training approach intended to increase the robustness and generalization of large neural language models called ALUM (Adversarial Training for Large Neural Language Models). Both pre-training and fine-tuning can be accomplished using the same approach. Their proposed training approach is mainly based virtual adversarial training [44],where an additional term is added to the training process which maximizes the adversarial loss by applying perturbations to the embedding space and optimizing it. 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.* [77], in which adversarial training is done by Fast Gradient Method (FGM) [43] and ensemble methods, in which multi-BERT model prediction assists in robustness. A proposed approach combines multiple BERT (BERT, SciBERT, RoBERTa, ALBERT, and ELECTRA) and makes an average ensemble for all models to

[1] https://www.connectedpapers.com/

achieve superior performance. Large language models and internal maximization raise concerns about computational cost. Moreover, as mentioned previously, their approach is also completely focused on increasing generalization and performance under attack has not been evaluated.
Wang *et al.* [72] proposed a gradient-based synonym substitution method to quickly generate adversarial samples called Fast Gradient Projection Method (FGPM). The adversarial samples are claimed to be very effective in attacking the models. Including this technique in training have shown significant improvement in the robustness of models and they call this technique Adversarial Training with FGPM enhanced by Logit pairing (ATFL). But, the proposed approach is verified against CNN, LSTM, Bi-LSTM model. And, model's robustness is determined by accuracy under attack as the only metric that cannot provide a complete picture of model robustness.

3.2 Augmentation-Based

As part of a noise-based approach, Si *et al.* [64] proposed robust fine-tuning of language models by augmenting the training dataset and 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. This approach has resulted in improved model performance under attack, but original accuracy has decreased.
Bao *et al.* [5] proposed approach 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 [28], 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 the BERT model, but improvement in accuracy under attack, from 12.4% to 89.2%. In their report, they had not mention any further metrics concerning adversarial attacks, such as the number of queries, word perturbations, and confidence score.
Dong *et al.* [12] proposed an adversarial sparse convex combination (ASCC) method, which models the word substitution attack space as a convex hull and employs regularization to enforce perturbation towards a real substitution. According to their study, introducing the same approach during training results in a robust model only against word substitution attacks. Yet, robustness is evaluated solely on the basis of accuracy and has the possibility of evaluating against a variety of language models.
The Dirichlet Neighborhood Ensemble (DNE) is a technique proposed by Zhuo *et al.* [76] which creates virtual sentences by combining the embedding of the original word in the input sentences

with its synonyms. When training, these virtual sentences can enhance word substitution-based attacks, and the method samples an embedding vector within the convex hull formed by words in and their synonyms to ensure robustness in that region as discussed in previous work. And, this study also has similar focus and shortcomings.

As a defence against character level adversarial attacks, Pruthi *et al.* [56] proposed placing a word recognition model based on RNN based semi-character word recognition model before the downstream classifier to identify unknown and rare words. Both models are trained separately and provide downstream task independent recognition. According to their experiment results, accuracy was restored by approximately 30% only under character-level attacks in BERT model.

3.3 Other Strategies

In another study by Wang *et al.* [71], Synonym Encoding Method (SEM) was proposed as a defensive approach against adversarial attacks. According to this method, encoding the clusters of synonyms to a unique encoding before the input layer of a model can be effective against synonym based adversarial attacks in the context of text classification. According to their findings, the BERT model's accuracy dropped by 0.2% for the IMDB dataset against synonym-based PWWS [59] attack. However, the proposed method is only efficient against synonym based attacks and the model's resilience is measured only by accuracy.

A search-based approach is proposed by Guo *et al.* [20] called RoSearch, where the robust distilled version of the language model is looked at the search space of various student models. As per their claim, the proposed approach can improve the robustness from 7% $\sim$ 18% up to 45.8% $\sim$ 47.8%. This approach is focused to enhance the performance of language models created using knowledge distillation process and has considered only accuracy as a metric for evaluation.

Mozes *et al.* [47] argued and statistically proved in their study that the frequency of replaced words differs from the frequency of adversarial substitutions. And, the effects of substitutions can be mitigated through simple frequency-based transforms and Frequency-Guided Word Substitutions (FGWS) which can be used to detect a given sequence of input. This method has been claimed to identify adversarial samples with 91.4% F_1 score.

According to the study of Han *et al.* [21], the robustness of language models can be enhanced by adding extra bottlenecks layers called adapters within each layer of pre-trained models. During fine-tuning for down streaming tasks, fix the pre-trained layer and train only the adapter layer. According to the study, the attack success rate was reduced. The approach, however, is evaluated against only one metric and attack recipe, i.e. PWWS [59].

According to Dong *et al.* [11], conventional fine-tuning of languages results in the loss of pre-trained weights, thereby failing to retain robust linguistic features. Additionally, they proposed Robust Informative Fine-Tuning (RIFT) as a robust information-theoretic perspective of

fine-tuning which focuses on retaining the features learned in pre-training. According to their experimental results, the BERT model fine-tuned using RIFT yielded a 74.3% accuracy under PWWS attack recipes in contrast to approximately 19.4% accuracy in case of conventional fine-tuning. However, the proposed study considers only accuracy and two attack recipes as signs of robustness and leaves room for further evaluation of recent sophisticated attack recipes.
Wang *et al.* [69] proposed a strategy called Controlled Adversarial Text Generation (CAT-Gen), which uses an encoder-decoder architecture to generate text controlled by attributes present in text data. It is the objective of adversarial text generation strategies to keep the same controlled attribute while attacking a model. Furthermore, they claimed that introducing this adversarial text during training can enhance the models' accuracy. However, this was not the main objective of the research and not evaluated against language models.
Malkiel *et al.* [40] proposed a method to fine-tune language models using a large number of multiverse classifiers, termed Maximal Multiverse Learning (MML). The goal is to fine-tune a large number of multiple classifiers simultaneously and enforce them to be orthogonal to each other. In training, eliminating weaker classifiers and optimizing the models by minimizing two losses, namely, task loss and orthogonal loss, which are in general cumulative losses of all the classifiers. A comparatively robust model was claimed in their paper in terms of generalization, but an evaluation against adversarial attacks remains to be done.
At present, most studies in the text domain are focused on generating adversarial text, and introducing those adversarial texts during training can enhance the model's robustness against the corresponding attack. Furthermore, the term robust is more often used in the context of generalizing rather than resilience towards adversarial attacks or other uncertain scenarios. Hence, it is being suggested the term robust should be taken into account when evaluating against various attack scenarios and generalizations.
Focused research on verification of language models against attacks and defence against them is still in progress. In addition, it is challenging to compare the performances of published work because of no specific standard evaluation framework [45]. Hence, this master thesis is also trying to provide metrics specific to robustness evaluation of language models.

4 Proposed Methodology of Adversarial Fine-Tuning and Attack Recipes

4.1 Mean Teacher Fine-Tuning

The proposed method calls for fine-tuning the language models using the semi-supervised approach for classification downstream task and utilizes the prominent adversarial unlabelled dataset for the same. A pictorial representation of methodology is illustrated in the figure 4.1.

The approach starts with creating adversarial samples for training. As shown in the figure 4.1, after pre-processing, the training dataset is being split into three equal parts and prominent adversarial unlabelled data is created using three different data augmentation strategies i.e. 1) Synonym based, 2) Context-based and, 3) Back-translation based which are discussed in length in section 5.3. Labels of those adversarial datasets are discarded hence called *prominent unlabelled adversarial data*. Then, these generated samples are combined together and shuffled. After creating prominent unlabelled adversarial data, semi-supervised training starts.

The proposed semi-supervised approach is based on the mean teacher approach implemented in the computer vision domain [67]. In the mean teacher model, two identical models are trained with two different strategies called student and teacher models. 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 of taking an average of many models decision, teacher model is a mean of consecutive students models weights hence called *mean teacher*.

As shown in the 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 the teacher model (weights $\hat{\theta}$ and noise η'). The

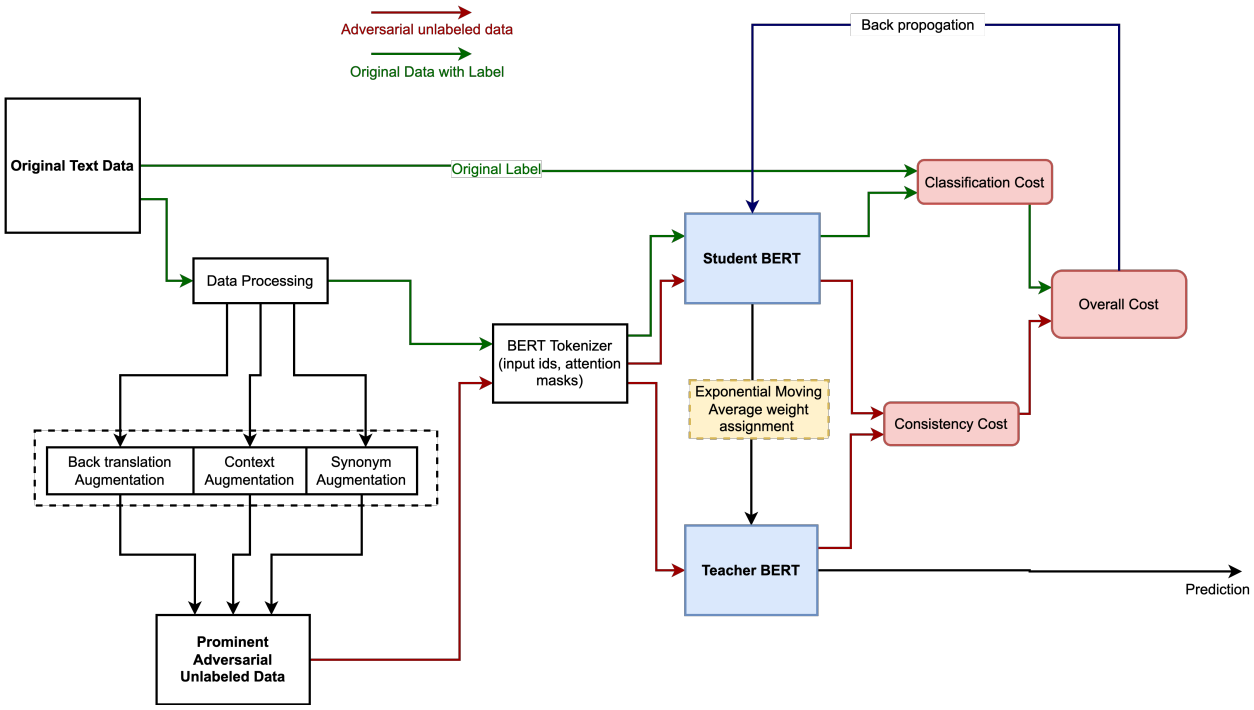

Figure 4.1: Training flow diagram of proposed fine-tuning methodology

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 the student model, the overall cost ($O(\theta)$) is calculated with the 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 the equation 4.4, while assigning weights, the 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 an algorithm 1.

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

Algorithm 1 Mean Teacher Algorithm

Data: train set $(\mathcal{X}, \mathcal{Y})$, Prominent Adversarial unlabelled set $(\mathcal{Z})$
Hyper parameters: *r*, α, *epochs*
Create Model: *student* (θ), *teacher* $(\hat{\theta})$
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

To evaluate the proposed approach, four black box attack recipes that satisfy lexical, grammatical, and semantic constraints have been selected. And, TextAttack python package [46] is used in the experiment to utilize the mentioned attack recipes. The table 4.1 shows an example of perturbation created in this experiment. 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 examples of attack recipes in this experiment.

4.2.1 TextFooler

Jin *et al.* [27] proposed TextFooler, 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. For better understanding, the process is briefly explained in steps:

1. **Word Importance Ranking**: Given a sentence of words, they create a ranking of each word by measuring the change before and after deleting the words, called the importance score. NLTK and spaCy libraries are used to remove words and preserve the grammar of the sentences. 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. Universal Sentence Encoder [7] is used in this study for encoding the sentence into a high-dimensional vector and calculating the cosine similarity between the sentences. Next, replacing candidates with values above the preset threshold value and creating 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 the BERT model under adversarial attacks. In their experiment, the accuracy dropped significantly from 90.9% to 13.6% with perturbed words 6.1, the number of queries to target model 1134, and the average length of IMDB dataset was 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 [25]

4.2.2 TextBugger

The TextBugger system proposed by Li *et al.* [32] 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 token ids, causing machine learning models to behave incorrectly. On the other hand, studies show that similar misspellings can still be perceived by the reader [3, 58]. Both character-level and word-level perturbation are targeted in this attack. Although Li *et al.* [32] 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 the confidence score of a particular sentence by the target model.
2. **Finding Important Words**: The importance score of a word is the difference between the confidence of the target model with a word and without a 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 a word with the 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 [32]

Using the IMDB movie review dataset, the TextBugger model is evaluated against Logistics Regression, CNN, and LSTM models. And, showed 95.2%, 90.5%, and 86.7% success rate with perturbed words of 4.9%, 4.2%, and 6.9%, respectively. But, their study did not assess the effectiveness of the BERT model. TextBugger generates adversarial attacks in much fewer time in contrast to TextFooler because of its sub-linear relationship to text length.

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

Ren *et al.* [59] 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**: The salience of a word is defined as its degree of change in classification probability if the word is set to unknown [33]. Calculating word saliency vectors for each word in a text, and then prioritizing the words based on the degree of change in classification probability after replacement and minimum word saliency.
2. **Replacement Strategy**: They have used WordNet for the search of word synonyms 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 replacing the words to attach the model. According to the result, the accuracy dropped from 84.86% to 2.00% with perturbation 3.38% for the IMDB dataset and Bi-LSTM model, 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[59]

4.2.4 BAE: BERT-Based Adversarial Examples

The BERT masked language model (MLM) was employed by Garg *et al.* [17] 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 [25] and PWWS [59]. 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 the most similar token in decreasing order until the attack is successful and trying all combinations. Figure 4.5 shows a schematic working diagram.

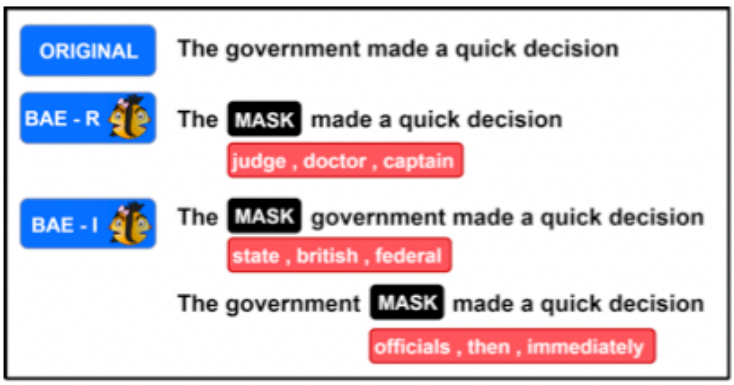

Figure 4.5: Schematic working and example of BAE [17]

5 Experiment Setup

5.1 Dataset

To assess the performance of baseline and proposed model two datasets were selected i.e. 1) Covid-19 fake tweets dataset [52] provided at Codalab competition, and 2) IMDB review binary classification dataset were selected.

The IMDB dataset is a sentiment classification dataset [39] that contains user movie reviews labelled as positive and negative. And, this dataset is majorly 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 that provided 8050 samples to train and test our model. The train and test sizes are shown in table 5.1. Additionally, 6050 dataset is sampled to create augmented unlabelled dataset and the filtered datasets have an average text length of 127. The label distribution in training datasets is completely balanced.

Covid-19 fake tweets dataset is a recent dataset specifically for classifying fake tweets as fake or real. Covid-19 fake tweets dataset sized 8000 and this dataset is fully utilized in the experiment. Contrary to the IMDB dataset, the Covid-19 fake tweets dataset has an average length of 25, as shown in table 5.2. Furthermore, as Covid-19 fake tweets dataset has mostly hashtags and comparatively fewer correct English words. Therefore, the intention of observing the models behaviour under a recent dataset that has comparatively lesser text length and vocabulary size led to selecting this dataset.

Dataset	Train	Test	Aug. unlabelled
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 and removing those words may affect the performance. Therefore, one of the benefits of language

models are their negligible requirement for data cleaning. The pre-processing steps involved in this experiment as follows:

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.

5.2.1 Data Exploration

In the experiment, models were trained with almost equal distributions of labels and the same training data is being used to generate unlabelled augmented data is shown in 5.1. During experiments, various exploration techniques were utilized such as word cloud, word distribution with/without respect to classes in the dataset, and length distribution mentioned in the appendix. However, the most relative information about the dataset i.e. about length is provided in the table 5.2 and figure 5.3.

Dataset	Avg. Length
codalab	25
IMDB	127

Table 5.2: Length details of both the datasets.

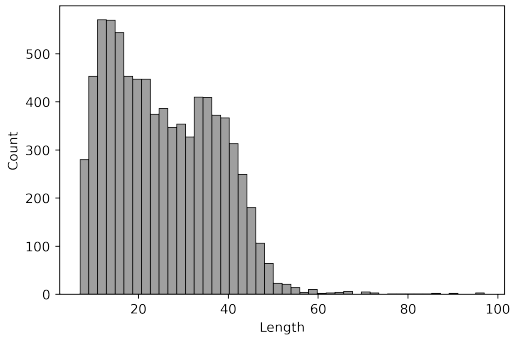

Figure 5.1: Length distribution of Covid-19 fake tweets dataset.

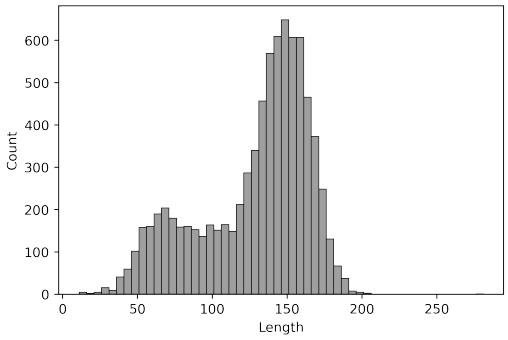

Figure 5.2: Length distribution of IMDB dataset.

Figure 5.3: Length Distribution of Covid-19 fake tweets and IMDB dataset.

The Covid-19 Fake tweets dataset is shorter than the IMDB dataset. The purpose of using two distinct length datasets is to better understand the behaviour of models and attack recipes in relation to the length.

5.3 Data Augmentation

For creating the unlabelled 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 class of the original dataset. Therefore, labels were dropped, hence unlabelled data. 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 [1] to achieve all three augmentation strategies. We have augmented the dataset by removing the label columns from the training dataset and then randomly splitting it into three parts. The first part is for synonym augmentation, the second part is for context-based augmentation, and the last part is for back-translation.
One more idea was to create unlabelled data using different attack recipes, which might provide better results under attack. However, the reason why attack recipes are not used is that the model is created with little or no understanding of how attacks work.
In the case of implementing a specific attack recipe to create unlabelled data, the robustness of the model will be enhanced for specific attack recipes, but the overall evaluation of the model performance can be biased.

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.

WordNet lexical English database [42] is used as a source of synonym augmentation, which

[1] https://github.com/makcedward/nlpaug

includes word definitions, hyponyms, and semantic relationships. Same database is utilized for synonym replacement in this thesis. 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.
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 a time- and memory-consuming. To perform this augmentation, the 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 [38] 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 these 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 the respective proposed models will be referred as MTBERT and MTDistilBERT. For training the models, google colab provided GPU has been utilized as shown in figure 8.1. The source code for the experiment are provided as a github project[2] and complete code written in Goolge colab notebook with proper instruction of reproducing the experiments.

5.4.1 Hyperparameter Details

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

Hyper parameter	Used parameters in this work
Optimizer	Adam
Learning rate	$2e-5$
Classification cost	Binary Cross Entropy
Consistency cost	Mean Squared Error
Epochs	3
Batch Size	4
Loss Ratio (r)	0.5
Alpha (α)	0.999
Dropout	0.2
Max length	100
Threshold (δ)	0.8

Table 5.6: Hyperparameters Details

5.5 Evaluation Metrics

Given test input and its respective label (X,Y) consists of articles $(x_1, x_2, ..., x_n)$ formed by words $(w_1, w_2, ..., w_n)$. The ground truth of test input X is Y i.e. $(y_1, y_2, ..., y_n)$, respectively. The model F is defined as a function which classifies inputs X to its respective label Y i.e. for a given input x_i, $F(x_i) = y_i$ is defined as correct classification and $F(x_i) \neq y_i$ as incorrectly classified as discussed in 2.4.1.
Therefore, the set of such correctly classified samples is C_c. During the experiment, generalization of the model is calculated by using the accuracy of the target model on test samples, referred as original accuracy i.e. the percentage of correct predictions over the entire sample set X as mentioned in the equation 5.1

$$\text{Original Accuracy} = \frac{|C_c|}{|X|} \tag{5.1}$$

[2]https://github.com/bksaini078/Mean-Teacher-BERT-Robustness-Assessment

An attack would be called successful when $F(x'_i) \neq y_i$, given $F(x_i) = y_i$, where x'_i is perturbed input crafted by attack recipe using correctly classified sample x_i by querying the model F q_i times and set of such samples is (s_c). Similarly, the attack is called failed when $F(x'_i) = y_i$ given $F(x_i) = y_i$ and set of such samples are (s_f). The attack recipes skips certain set of samples (s_k) which comes under the scenario where $F(x_i) \neq y_i$, which means model has already failed to classifying inputs correctly.
The robustness of the target models are based on five different metrics:

1. Accuracy under attack
2. Attack success rate
3. Average number of queries
4. Average word perturbation
5. Perturbation score

A target model's accuracy under attack is the percentage of correct predictions given the number of perturbed test samples n_s by attack recipes excluding the skipped attacks (s_k) i.e $n_s = |X| - |s_k|$, as shown in equation 5.2.

$$\text{Accuracy under attack} = \frac{|s_f|}{n_s} \tag{5.2}$$

Compared to the original accuracy, accuracy under attack can provide significant information about the efficiency of the target model and attack recipes. Significant positive difference between original accuracy and accuracy under attack indicate lower robustness against attack. A counter-part of accuracy under attack is the attack success rate, which indicates the effectiveness of attack recipes.

$$\text{Attack success rate} = \frac{|s_c|}{n_s} \tag{5.3}$$

For given input x_k i.e. $(w_{k1}, w_{k2}, .., w_{kn})$, perturbed input x'_k i.e $(w_{k1}, w'_{k2}, .., w'_n)$ is created. A word perturbation P_k is the ratio of words difference between original and perturbed input to the total number of words in x_k. Therefore, the average perturbed word of complete X is defined as :

$$\text{Average word perturbation} = \frac{\sum_{i=0}^{n_s} P_i}{n_s} \tag{5.4}$$

Higher word perturbation indicates less robustness, and this metric is also dependent on the length of the text.
Furthermore, $(q_1, q_2, .., q_n)$ are the number of queries q for inputs $(x_1, x_2, ..., x_n)$ to successfully attack the model F. Therefore average number of queries is number of times attacks recipes sends different perturbed inputs to the target model for successful attacks. And, the average

number of queries is mean of those individual number of queries of all inputs X as mentioned in equation 5.5. A higher number of queries is a sign of robustness.

$$\text{Average number of queries } = \frac{|q|}{n_s} \tag{5.5}$$

Finally, the perturbation score is a confidence score or a predicted probability of the target model F under attack. A 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 tokenizers are used for tokenization in both while training and attacking, i.e. from the huggingface[3] an open-source python package for language models. Evaluation of the models have been performed for the 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 the attacks are basically word level and character level hence, clear the intent of this selection.

The robustness of target models is evaluated against the four attack recipes mentioned in 4.2, hence it is assumed that the threat agent has access to those attack recipes. And, the performance of the 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 in mentioned settings 5.4.1, and the model can perform better or worse in different settings. In terms of transferability, the experiment is conducted on two separate language models that differ in pre-training, it is expected that the approach would perform similarly if additional language models were used.

[3] https://huggingface.co/

6 Experiment Results

6.1 Evaluation of Results

The result presented in tables 6.1 and 6.2 demonstrates that MTBERT has outperformed all the other language models in almost all metrics.

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
	MTBERT	**94.13**	**56.45**	**40.03**	3.55	198.26
	MTDistilBERT	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
	MTBERT	**94.13**	**23.20**	**75.35**	**5.70**	**890.84**
	MTDistilBERT	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
	MTBERT	**94.13**	**35.13**	**62.68**	**28.57**	**449.96**
	MTDistilBERT	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
	MTBERT	**94.13**	**30.92**	**67.16**	**8.04**	**720.77**
	MTDistilBERT	93.17	25.48	72.64	7.54	613.91

Table 6.1: IMDB dataset: MTBERT model has performed well overall followed by MTDistilBERT. Bold and underlined values represent first and second best value respectively.

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
	MTBERT	**95.64**	**77.62**	**18.83**	16.57	**88.64**
	MTDistilBERT	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
	MTBERT	**95.64**	**59.16**	**38.14**	17.93	**236.90**
	MTDistilBERT	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
	MTBERT	**95.64**	**65.61**	**31.39**	23.29	**114.25**
	MTDistilBERT	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
	MTBERT	**95.64**	**54.05**	**43.48**	21.25	**282.66**
	MTDistilBERT	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. Bold and underlined values represent first and second best value respectively.

The 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 6.3, the MTBERT model showed 0∼2% improvement in accuracy over the baseline model considering both datasets and 20∼30% improvement in accuracy under attack. The proposed model also shows the comparatively higher requirement for the number of queries and word perturbations. Furthermore, the drop in accuracy is lowest in proposed approaches than baseline models.

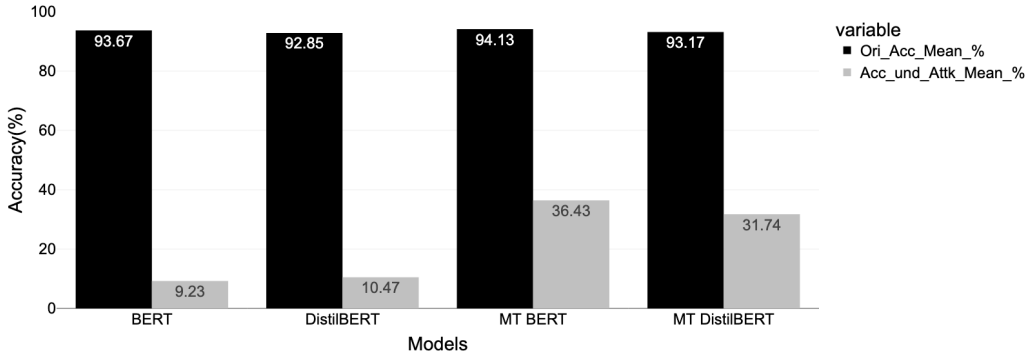

Figure 6.1: IMDB Dataset.

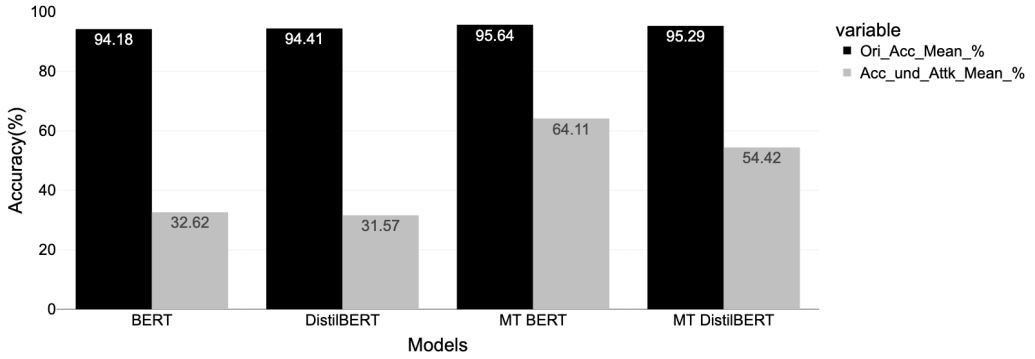

Figure 6.2: Covid-19 fake tweets dataset.

Figure 6.3: Comparative bar plot of original accuracy and accuracy under attack in different datasets. MTBERT showed better performance in both metrics followed by MTDistilBERT. And, the highest drop is observed in baseline models (BERT and DistilBERT).

6.1.1 Number of Queries

In terms of queries, it is observed that overall MTBERT has a high need for the number of queries to be attacked as shown in bar plot 6.4. As shown in the box plot 6.5, the interquartile range of the MTBERT and MTDistilBERT model is comparatively broader than any other model and has also shown incident crossing 2000 queries.

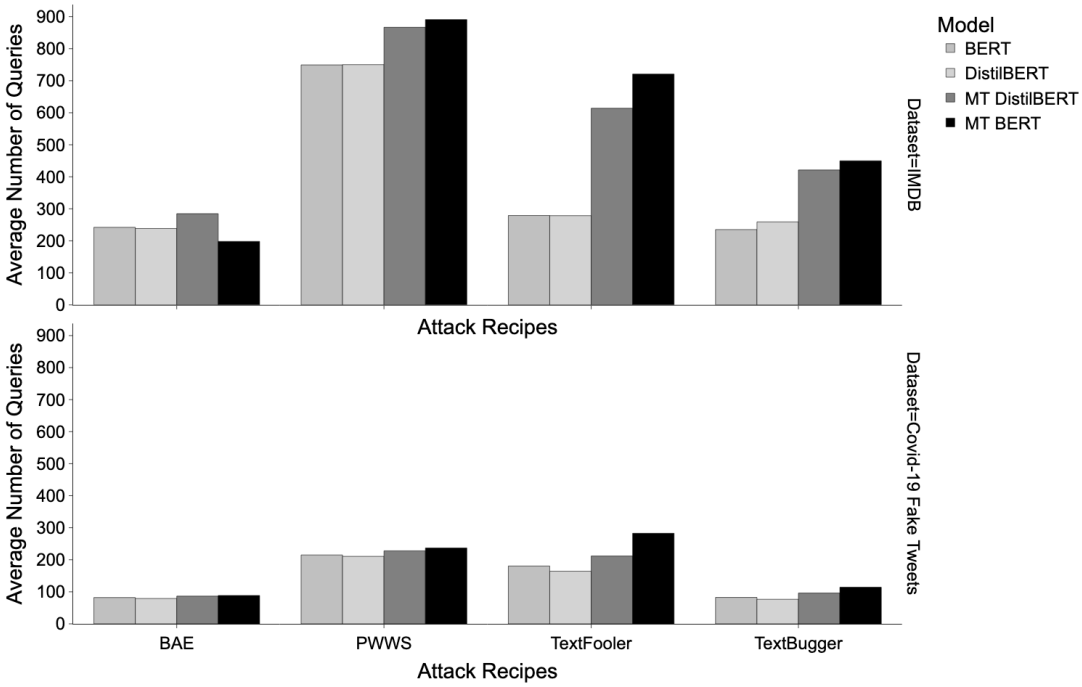

Figure 6.4: Bar plot of the number of queries w.r.t dataset and attack recipes. PWWS has shown the highest overall number of queries requirement followed by TextFooler. And, dependencies w.r.t to text length can also be observed in this plot.

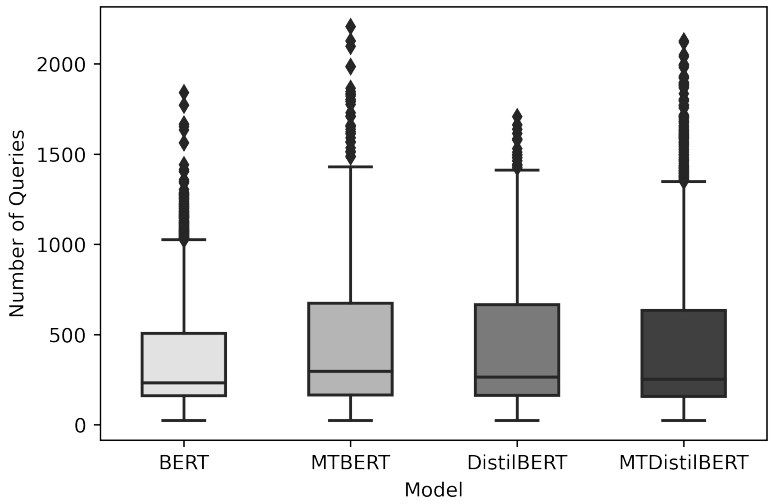

Figure 6.5: Box-plot of the number of queries.
The proposed models are comparatively broader and showed greater values for outliers than baseline models. This particular box plot is based on the IMDB dataset and only successful attacks are considered.

If we compare the performances of attack recipes, BAE attacks have 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 the dataset shown in the figure 6.4. In the TextFooler attack recipe, the difference between proposed and baseline queries also increases with respect to the length of the text which shows its effectiveness depends on text length.

6.1.2 Word Perturbation

To understand the robustness of models with respect to word perturbation, the difference is less significant in between models as shown in the figure 6.6. However, in the Covid-19 fake tweets dataset, the average word perturbation required by proposed models is higher, but, exactly the opposite in the 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 is 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.

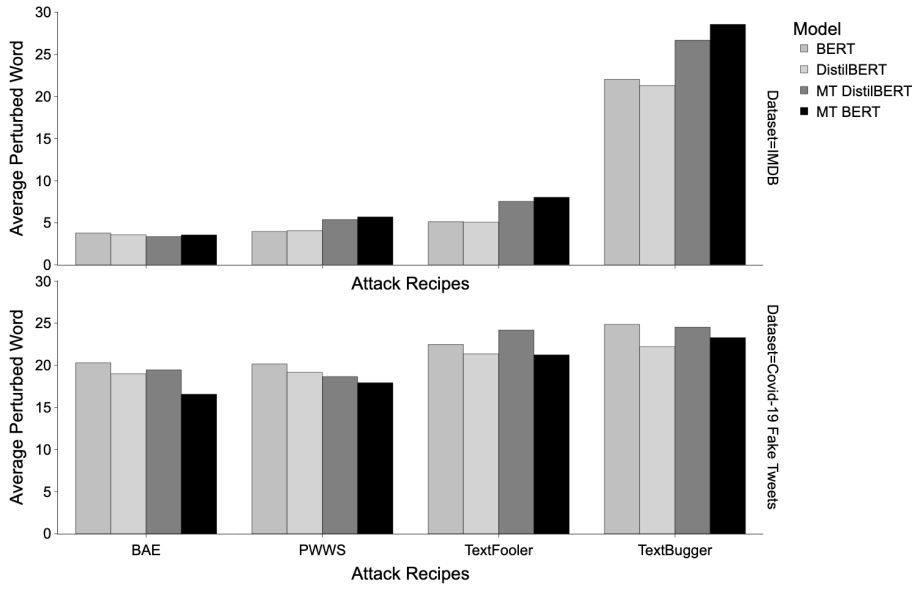

Figure 6.6: Bar plot of word perturbation as per datasets and attack recipes. TextBugger has shown the highest word perturbation in both datasets and also shown less difference between both datasets. However, BAE has lower word perturbation followed by the PWWS attack recipe. Covid-19 fake tweets dataset has shown significantly higher word perturbation requirements.

6.1.3 Perturbation Score

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 change the confidence score of successful attacks. As shown in the figure 6.7, a huge improvement is being observed in the proposed model with respect to the baseline model and the distribution of the proposed model is comparatively more left-shifted than baseline models. The proposed model does not let attack recipes go beyond a particular limit i.e. 0.70, however, baseline models are more uniform and show less resilience. The range of the MTBERT model is 0.50 to 0.55, but for the BERT model, it is 0.50 to 0.66, which is a significant improvement in the model.

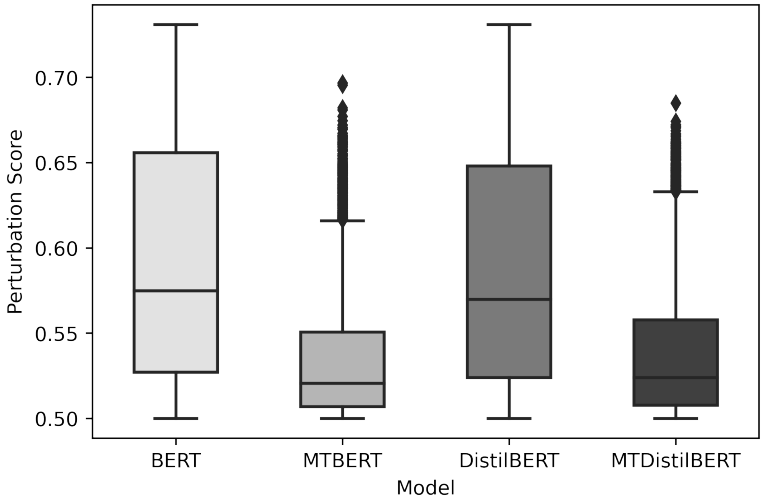

Figure 6.7: Box-plot of perturbation score. MTBERT model has managed attacks confidence score in the leaner interquartile range and does not cross 0.70 in a single instance which is a significant improvement over BERT.

Overall, the model fine-tuned using the proposed approach has shown significant improvements over the conventional way of fine-tuning and MTBERT has shown maximum robustness as compared to other models. In the experiment, it is also observed that compact model i.e. DistilBERT created using knowledge distillation process are performing comparative to its base model i.e. BERT. Furthermore, PWWS and TextFooler appeared to be more successful attacks recipes compared to other attack recipes.

6.2 Discussion on Research Question

The objective of this master thesis was to conduct experiment and investigate the research question:
Is the proposed mean-teacher semi-supervised fine-tuning and the use of prominent adversarial unlabelled data an effective method to improve the robustness of language models, specifically with regard to accuracy under attack without compromising the original accuracy in contrast to the conventional fine-tuning approach?
This experiment hypothesized that the proposed means teacher semi-supervised approach and use of prominent adversarial unlabelled data will produce a comparatively robust language model in terms of metrics such as accuracy under attack, queries required, rate of word perturbation, and confidence score without compromising generalization. And, it was expected that the mean teacher language model would outperform the traditional fine-tuned model, and same has been confirmed throughout the experiment.
According to the findings, the proposed approach has demonstrated notable increase in generalization and a significant improvement in accuracy under attacks. The language model produced using proposed approach requires higher word perturbation and queries to get attacked. Furthermore, the proposed fine-tuned model has shown remarkable improvements in terms of confidence score, i.e., attack recipes are not able to misclassify the prediction with higher confidence.
In the proposed semi-supervised approach, two key techniques are expected to account for observed behaviour: 1) Data Augmentation, and 2) Exponential Moving Average (EMA). However, this research does not attempt to quantify the impact of these strategies on observed behaviour.

6.2.1 Data-Augmentation

During training, data augmentation incorporates numerous additional words as noise, resulting in models learning a bigger word representation and also regularize the model at the same time. This, however, is the most basic explanation and only using noisy data during training does not guarantee improvement. Englesson *et al.* [15] claimed that models trained on noisy data perform worse than models trained on clean data, and that the performance of models trained on noisy data worsens as the noise ratio rises. However, introducing a loss function during training can improve model performance significantly called consistency regularization. Consistency regularization is the process of incorporating a loss function during training to reduce consistency loss and build a more robust model.
Both gradient-based techniques [44] and the proposed methodology exploit the concept of consistency regularization. Furthermore, the suggested strategy also backs up the conclusions of Belinkov *et al.* [6] study, which indicate that include noisy data during training improves

model robustness. This answers the question of whether language models trained on noisy data with a loss function termed consistency loss may increase generalization and robustness. Furthermore, if appropriate data augmentation procedures are implemented in semi-supervised fine-tuning methodology, better predicting and robust language models can be achieved.

6.2.2 Exponential Moving Average

In another case, it is commonly known that ensemble technique outperforms a single machine learning models. Laine et al. [30] exploited this idea and applied it at training level, averaging the prediction of single network over multiple different training epochs and augmentation conditions. Their study found that such type of ensemble technique can also lead to a stronger predictor. Later, Tarvainen *et al.* [67] claimed and demonstrated that instead averaging model prediction, averaging model weights throughout training, can also lead to much better predictor.

Their research was based on the findings of Polyak *et al.* [55], who found that averaging weights over time provides a more accurate model than using final weights directly, and that the teacher model performed better since it was an average of consecutive student models.

This thesis provides experimental evidence supporting all the above mentioned studies, and claiming that the language model created using average weight assignment and data augmentation creates a comparatively better and more robust model when it comes to defending against adversarial attacks.

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. Research also found that while traditionally fine-tuned language models performed admirably in a variety of tasks, they are no different from other DNN models in terms of resilience. Positively, the findings suggest that language models still have room for improvement in terms of generalization and resilience.

7 Conclusion, Limitation, and Future Work

7.1 Conclusion

The purpose of this research was to investigate if the proposed mean teacher strategy to fine-tuning language models resulted in a more resilient language model without sacrificing original accuracy. Experiments were conducted to quantitatively examine and compare performance of the proposed and the conventional fine-tuning approach in terms of generalization and robustness. In addition, this research was also aimed into the efficacy of data augmentation technique which utilizes the capabilities of language models. Moving forward, a review of research studies were conducted in attempt to investigate the cause of the observed behaviour.

Although the mean teacher approach has proven significant performance in the image domain, there was still an open scope to explore how the same strategy behaves in the text-domain. As a result, the intention of this research was also to investigate the behaviour of the mean teacher approach in the text domain, especially with language models. The proposed mean teacher approach in the text domain differs significantly from that in the image domain. Three noise strategies were introduced which use language models capabilities to successfully execute the mean teacher approach.

In this experiment, four attack recipes, data augmentation techniques, and two language models were utilized. As per the result, the proposed approach of fine-tuning improved original accuracy by 0$\sim$2% and accuracy under attack by 20$\sim$30%. Furthermore, it also met higher criteria in terms of word perturbation and queries, which indicates its better robustness in contrast to traditional approach. The mean teacher technique is more resilient to attacks and does not allow attack recipes to sabotage models with high confidence, which is a remarkable improvement in the model and a key finding of this work. Moreover, two different language models are used to demonstrate the transferability of the proposed approach.

According to a study conducted on research paper, consistency regularization and exponential moving average (EMA) weight assignments were the most prominent explanations for the observed behaviour. Furthermore, this research first highlighted that language models are sensitive to adversarial attacks and also, showed scope for development.

This proposed study provided an important opportunity to advance the understanding of language models, data augmentation, and various adversarial attacks in text-domain. This master thesis also contributed by provided a proper framework of metrics in terms of text classification

for evaluating language models and provided direction for future works.

7.2 Limitation

Unlike image data where perturbation can be made imperceptible to humans, achieving even a similar level of imperceptibility is still a challenge in the 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.
As the proposed approach uses two identical models while fine-tuning, therefore it is comparatively more space and time consuming that conventional approach.
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 its nature is uncertain. Furthermore, incorporating the corresponding noisy data can only strengthen the robustness of the system against particular attacks.
There 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.

7.3 Future Work

This approach can 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 model robustness. And in another case, the quantitatively measuring the effectiveness of individual augmentation techniques on robustness can be studied.
As a limitation the proposed approach is comparatively more space and time consuming, hence, in future less space and time complex techniques can be studied to get similar or better result.
In the experiment, training data is used to create augmented prominent unlabelled data, however, huge amounts of unlabelled data could be used instead and 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 technique in single training 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.

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:1901.09069 [cs, stat]* (Jan. 2019). arXiv: 1901.09069 [cs, stat].

[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: *ICLR* (2015).

[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] Xinhsuai Dong et al. "How Should Pre-Trained Language Models Be Fine-Tuned Towards Adversarial Robustness?" In: *arXiv:2112.11668 [cs]* (Dec. 2021). arXiv: 2112.11668 [cs].

[12] Xinshuai Dong et al. "Towards Robustness Against Natural Language Word Substitutions". In: *arXiv:2107.13541 [cs]* (July 2021). arXiv: 2107.13541 [cs].

[13] 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.

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

[15] 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].

[16] 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].

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

[18] 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].

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

[20] Xin Guo et al. "RoSearch: Search for Robust Student Architectures When Distilling Pretrained Language Models". In: *arXiv:2106.03613 [cs]* (June 2021). arXiv: 2106.03613 [cs].

[21] Wenjuan Han, Bo Pang, and Ying Nian Wu. "Robust Transfer Learning with Pretrained Language Models through Adapters". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 854–861. DOI: 10.18653/v1/2021.acl-short.108.

[22] 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.

[23] 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].

[24] Aminul Huq and Mst Tasnim Pervin. "Adversarial Attacks and Defense on Texts: A Survey". In: *arXiv:2005.14108 [cs]* (June 2020). arXiv: 2005.14108 [cs].

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

[26] 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.

[27] Di Jin et al. "Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment". In: *arXiv:1907.11932 [cs]* (Apr. 2020). arXiv: 1907.11932 [cs].

[28] 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.

[29] 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.

[30] Samuli Laine and Timo Aila. "Temporal Ensembling for Semi-Supervised Learning". In: *arXiv:1610.02242 [cs]* (Mar. 2017). arXiv: 1610.02242 [cs].

[31] 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].

[32] 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.

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

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

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

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

[37] 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].

[38] Edward Ma. *Nlpaug*. https://github.com/makcedward/nlpaug. Jan. 2022.

[39] 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.

[40] Itzik Malkiel and Lior Wolf. "MML: Maximal Multiverse Learning for Robust Fine-Tuning of Language Models". In: *arXiv:1911.06182 [cs, stat]* (Nov. 2019). arXiv: 1911.06182 [cs, stat].

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

[42] 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.

[43] 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].

[44] 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].

[45] Milad Moradi and Matthias Samwald. "Evaluating the Robustness of Neural Language Models to Input Perturbations". In: *arXiv:2108.12237 [cs]* (Aug. 2021). arXiv: 2108.12237 [cs].

[46] 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].

[47] Maximilian Mozes et al. "Frequency-Guided Word Substitutions for Detecting Textual Adversarial Examples". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, Apr. 2021, pp. 171–186. DOI: 10.18653/v1/2021.eacl-main.13.

[48] 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.

[49] 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].

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

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

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

[53] 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.

[54] 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.

[55] 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.

[56] Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. "Combating Adversarial Misspellings with Robust Word Recognition". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 5582–5591. DOI: 10.18653/v1/P19-1561.

[57] Alec Radford and Karthik Narasimhan. "Improving Language Understanding by Generative Pre-Training". In: (2018).

[58] 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.

[59] 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.

[60] Yankun Ren et al. "Generating Natural Language Adversarial Examples on a Large Scale with Generative Models". In: *arXiv:2003.10388 [cs, stat]* (Mar. 2020). arXiv: 2003.10388 [cs, stat].

[61] 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.

[62] 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].

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

[64] 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.

[65] 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].

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

[67] 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].

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

[69] Tianlu Wang et al. "CAT-Gen: Improving Robustness in NLP Models via Controlled Adversarial Text Generation". In: *arXiv:2010.02338 [cs]* (Oct. 2020). arXiv: 2010.02338 [cs].

[70] 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].

[71] Xiaosen Wang, Hao Jin, and Kun He. "Natural Language Adversarial Attacks and Defenses in Word Level". In: *arXiv:1909.06723 [cs]* (Apr. 2020). arXiv: 1909.06723 [cs].

[72] Xiaosen Wang et al. "Adversarial Training with Fast Gradient Projection Method against Synonym Substitution Based Text Attacks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.16 (May 2021), pp. 13997–14005. ISSN: 2374-3468.

[73] 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].

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

[75] 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].

[76] 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].

[77] 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 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