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

Masterarbeit

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

vorgelegt von

Bhupender Kumar Saini

[219 100 887]

Koblenz, im January 2021

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

Eidesstattliche Erklärung

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

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

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

. . . .

(Ort, Datum) (Unterschrift)

Zusammenfassung

Die Zusammenfassung Ihrer Thesis.

Abstract

The Abstract of your thesis.

Contents

List of Figures

List of Tables

1 Introduction

Deep Neural Network(DNN) has been widely adopted in real world applications and study in every domain and research field. DNN got upper hand in contrast to any other machine learning algorithm because of its ease of computation at large scale and capability of solving complex problem either linear or non-linear problem [22]. In natural language processing tasks, DNN models also has shown significant advancement in solving various tasks such as text to speech, fake news detection, reviews comment classification and so on. Furthermore, recent works [9, 28, 33, 50] in NLP has led to state-of-the-art transformer based language model such as BERT, which has been successful across a wide variety of NLP tasks and is consequently the most widely adopted language model. However, studies[1, 22, 54, 59, 61] have found weaknesses in DNN against adversarial attacks which exposed alarm about the weakness of DNN system against attacks which can reduces its value and raise question *"Can we trust ML models?'* .
Introducing a small perturbations in the input image can fool state-of-art deep neural network with high probability and these misclassified samples were called as *Adversarial Samples* [54]. Sabotaging machine learning model using adversarial examples are called as *Adversarial attacks* [41]. In computer vision domain, Szegedy et al. [54], first revealed this vulnerability to research community, which has drawn substantial attention from the research community. And, Papernot et. al. [42], first revealed that small change in sentence can also sabotage the model prediction. Later, the various studies published describing various different techniques to craft such adversarial samples in text domain, an example is shown in figure 1.1.
Furthermore, the studies related to adversarial attack is more extensively studied in the domain of computer vision than that of natural language processing [58]. In that, more rigorously concentrated on approaches generating adversarial samples then defense mechanism. The main challenges in crafting adversarial attacks in text domain is its discrete in nature and constraint of maintaining semantics [31].

Specifically, BERT performance also degrade under adversarial attacks and has shown accuracy lower than 10% under adversarial attack [15, 31]. A less sophisticated spelling error can lead to bad performing machine learning model [53]. Unfortunately, the studies that work towards improving the language model robustness are few and revolves around gradient based method which has shown significant improvements in image-domain as well as majorly studied topic. Furthermore, for specific downstream task user majorly relies on conventional way of fine-tuning and significantly lesser studies which discuss about fine-tuning approach which lead to compar-

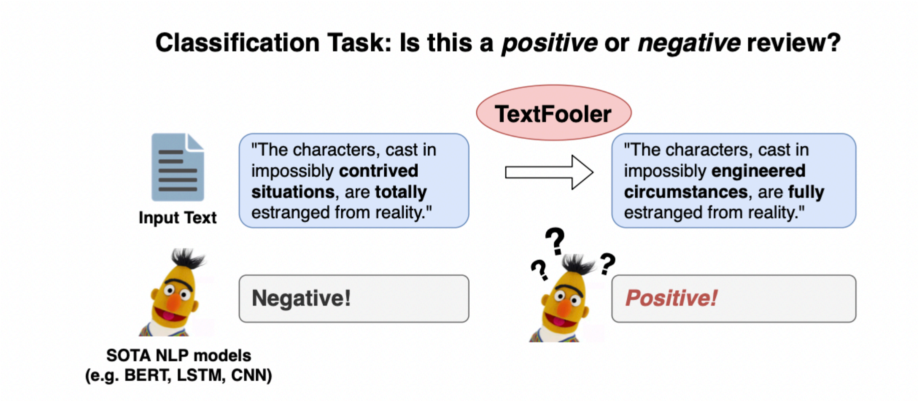

Classification Task: Is this a *positive* or *negative* review?
TextFooler
Input Text
"The characters, cast in impossibly **contrived situations**, are **totally** estranged from reality."
"The characters, cast in impossibly **engineered circumstances**, are **fully** estranged from reality."
Negative!
Positive!
SOTA NLP models
(e.g. BERT, LSTM, CNN)

Figure 1.1: Adversarial attack presented by TextFooler [25], small change in input text influenced the prediction.

atively robust model.

In another case, text classification task such as fake news detection, suffers from limited labeled data as compared to unlabeled data. Many semi-supervised approach has been proposed to overcome such challenges and has shown that semi-supervised way of training has shown significant improvements in performance with respect to generalization and robustness. One of such semi-supervised approach such as mean teacher [55] performed well in image-domain, but the efficiency of this approach has not been studied in text-domain especially with language models. The challenge of implementing such technique would be completely different from image domain because of different noise strategies and techniques are required to take a leverage from such approach. Furthermore, Belinkov et al. [5] study discuss how including noisy data in training sample can lead to robust models, but such technique is not studied with respect to language model and evaluated.

Hence, this master thesis study proposed a novel BERT model fine-tuning approach which can create result in comparatively robust model without compromising with original accuracy. Moving forward, conducting quantitative comparative experiment to observe the performance proposed and conventional way model. The proposed training methodology is inspired by a semi-supervised approach called Mean Teacher proposed by Tarvarian et. al [55]. And, utilizing the capabilities of language model like back translation, context writing to create synthetic unlabeled data in generating prominent adversarial unlabeled samples for training. The focus of the study is to answer the research question, the proposed approach of fine-tuning will lead to comparatively better model with respect to generalization and robustness.

The proposed study contribute in observing the performance of language models under worse situation so mitigation strategies can be planned. Hence, it is significant to know how adversarial attacks conduct and what measures can defend against them to make DNNs more robust. Till now, no relevant study has been conducted which discuss the robustness of language model and semi-supervised fine tuning of language models. Evaluation of the robustness is determined

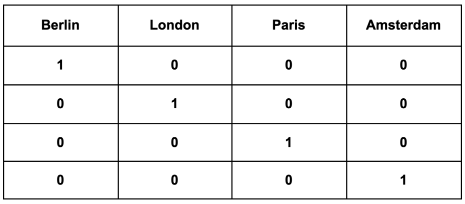
by observing the performance of model against four attack recipes and metrics.
The master thesis report first discusses about relevant background about evolution of text representation from TF-IDF to contextualized embeddings, transformer based language models, and explain the model architecture of two langauge models i.e. BERT and DistilBERT. Moving forward, background and classification of adversarial attacks are discussed. Then, complete working of proposed approach and attack recipes are as been explained. In Experiment chapter, details about experiments environment, and result has been discussed. Finally, master thesis is concluded with limitations, challenges, and final conclusion.

2 Background

In this section, basic terminology and concepts related to study is discussed to get the glimpse experiments and motivation. While writing the background, reader have initial understanding on text representation and language models is assumed. The section is starts with discuss on recent advancements in text representations followed by transformers architecture, one of the main principle behind recent contextualized embedding. And, later language model with fine tuning technique is elaborated. In the end of the section, adversarial attacks and their types are discussed. In case, reader already have understanding on the topic, it is recommended to skip the section.

2.1 Text Representations (2 pages)

Natural language processing is an area of research and application to explores how a computerized system can understand and manipulate natural language (speech or text) to perform various tasks[8]. The goal of natural language processing is to gain human level understanding of the text [40] and language model is a statistical model of language usage [2]. But, one major challenges was to convert text into numbers or vectors such that semantic and syntactic information is contained. Initially, words if often represented as one hot vector in discrete manner where word is represented as vector of 1 and 0 as shown in figure 2.1 and shape of the vector is equal to size of vocabulary. Hence, faces a limitation called dimensional curse and semantic relationship in words were missing. Later research focused towards learning low-dimensional and continuous vector representation of text and lead to 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

Word embedding is fixed length vector representations for words [2], where words or phrases are mapped to vectors of real numbers. The principle behind word embedding is based on distributed hypothesis [18], which states that words occurring in similar contexts tend to have similar semantics. These word embedding later can be used to perform various NLP tasks. Mikolov et. al [35] 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. In CBOW, model goal is to predict the middle word when given past and future words as input, at projection layer context of each words are averaged as shown in figure 2.2. As order of the words does not influence the projection of the word hence called bag-of-words. In skip gram, it tries to maximize classification of a word based on another word in the same sentences as shown in figure 2.2. One major disadvantage of Word2Vec is it only rely on location information. The semantic learn for given word is completely based on surrounding words.

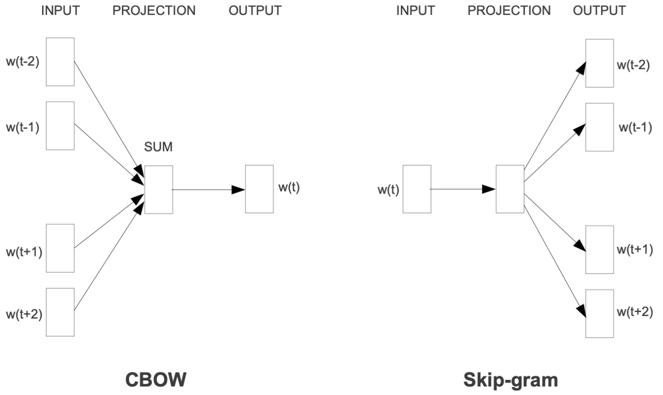


Figure 2.2: The CBOW architecture predicts the current word given past and future words, and Skip gram architecture predicts the surrounding words given the current words.

Penning et. al [43] proposed word embedding called Glove (Global Vectors for Word Representation), which relies on global information of the word and utilize large corpus of unlabeled data. The idea behind word embedding is based on co-occurrence matrix where how often a particular word pair occurs together. And, then factorize this occurrence matrix to lower dimensional space with an aim of minimum reconstruction loss. It was mainly based on global matrix factorization and local context window method such as skip-gram, which resulted in log-bilinear regression model where probability of the next word is calculated based on the previous words. These traditional approach embedding are static and has the limitation of context independence called polysemy i.e. the meaning of word can vary across different context. For example, word *'jaguar'* can be treated as animal or car brand based on sentence. Capturing those deep details were missing in traditional approaches and hence, recent research are mainly focused on creating contextualized word embedding.

2.1.2 Contextualized Embeddings

ELMO(Embedding from language model) proposed by Matthew et al.[44] , was one research towards that direction of creating context sensitive embedding. In ELMO, bi-directional LSTM model which help in extracting the contextualized word features to address the issue of polysemy. And, model is optimized by next word prediction using large corpus of data. The layers weight later can be utilized as contextualized embedding. After the introduction of transformer architecture [56] , Generative Pre-training (GPT) [46] and Bidirectional Encoder Representation From Transformers(BERT) [9] introduced utilized encoders or decoders in place of Bi-directional LSTM to create contextualized embedding. GPT utilized decoder architecture and BERT utilized encoder architecture, however the main principle of creating is inspired by ELMO, a pictorial representation is shown in figure 2.3. These two methods also introduced the concept of pre-training i.e. learning embedding using large corpus of unlabeled data and fine-tuning i.e. optimizing the weights further as per task. To understand the working of these language, brief overview of transformer model is required and hence, next section is about transformer before language models.

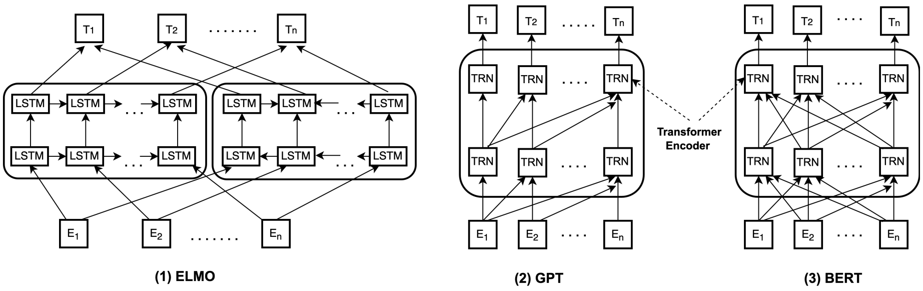


Figure 2.3: Different Contextualized Embeddings model.
Pictorial representation of working of (1) ELMO(Embedding from language model), (2)Generative Pre-training (GPT) , and (3) Bidirectional Encoder Representation From Transformers(BERT). TRN represents transformer encoder.

2.2 Transformers

Previously, NLP tasks were solely depends on sequential model like CNN, RNN, LSTM and BiLSTM models which has disadvantage of computational expensive, lack of distributing capabilities, and only satisfactory performance. In December 2017, Vaswani et. al [56], proposed a simple architecture is based attention mechanism called the transformer architecture which outperformed the existing state-of-art NLP models shown in figure 2.4 . And, The goal to reduce sequential computation leads to the transformer architecture which is solely based on a special type of attention mechanism called *self attention*. This proposed model architecture comparatively can be trained faster because of its distributed properties, and showed better evaluation result. This transformer model is a revolutionized and game changer architecture for Natural Language Understanding(NLU) and Natural Language Processing (NLP). Moving forward, became one of the main principle behind recent break through and state-of-art language models like BERT, GPT, and T5. A pictorial representation transformer architecture is shown in figure 2.4

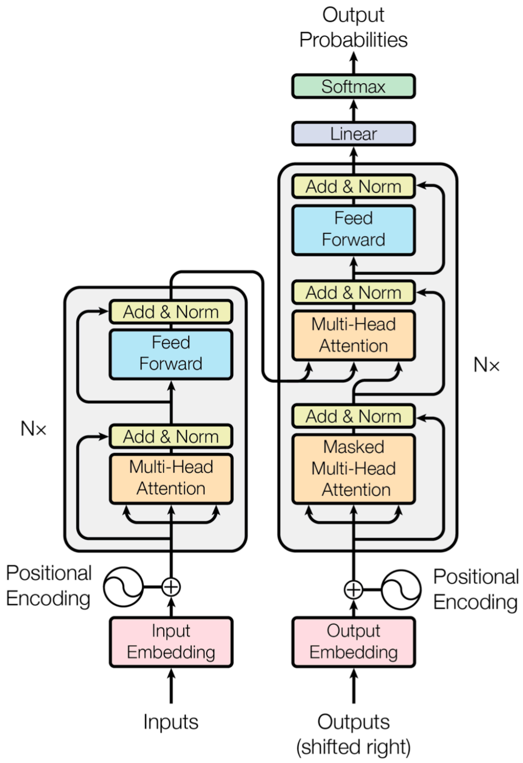


Figure 2.4: Transformer Model Architecture [56].

Transformer is an encoder and decoder stack where the encoder reads the inputs and outputs a representation as a context vector also called as *contextualized embedding* as shown in figure 2.5, based on single-head attention or multi-head attention, and the decoder makes predictions based on those context vectors. In proposed architecture by Vaswani et. al [56], transformer model is a composed of 6 layers of encoder stacked on each other and same applies to decoders. Each encoder is composed of multi-head attentions followed by layer normalization and Feed forward network, and the only difference in decoder is before multi-head attentions it has masked multi-head attentions layer.

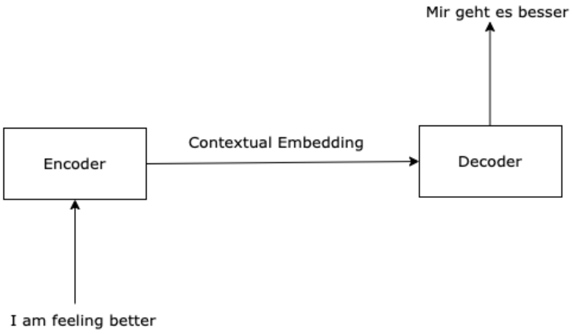


Figure 2.5: Transformer Encoder Decoder.

2.2.1 Encoder Architecture

The main problem in machine translation task was to translate with variable lengths input to another variable length output. Hence, two components called encoder and decoder proposed as solution, where encoder learns the pattern of variable length input and output a fixed shape output. On the other hand, decoder takes that fixed shape as input and output a variable length output. For example, the task is to translate english sentence "I am good. How are you ?" to German language , given input sequence of tokens "I", "am", "good",...,"?" to encoder which out the fixed shape representation z_1, z_2, z_3. Using this representation, decoder outputs the "Mir geht es gut. Wie geht es dir ?" in token format as shown in figure 2.6

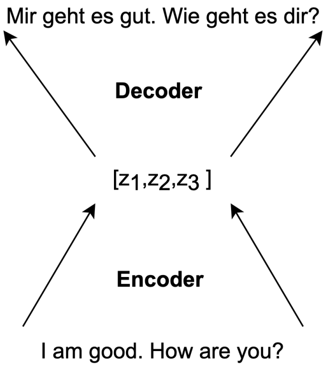


Figure 2.6: Basic encoder decoder example.

This encoder decoder model were utilized by many sequential to sequential models for task such as text summarisation, question answering and machine translation where inputs are sequentially. However, there is significant difference in encoder decoder architecture in transformer model. As shown in figure 2.7, encoder block in transformer is further consist of three sublayer: multi-head attention, layer normalisation layer and feed forward network. In this case, input embedding component convert the input text tokens into embedding vectors EM of shape d_{model}.

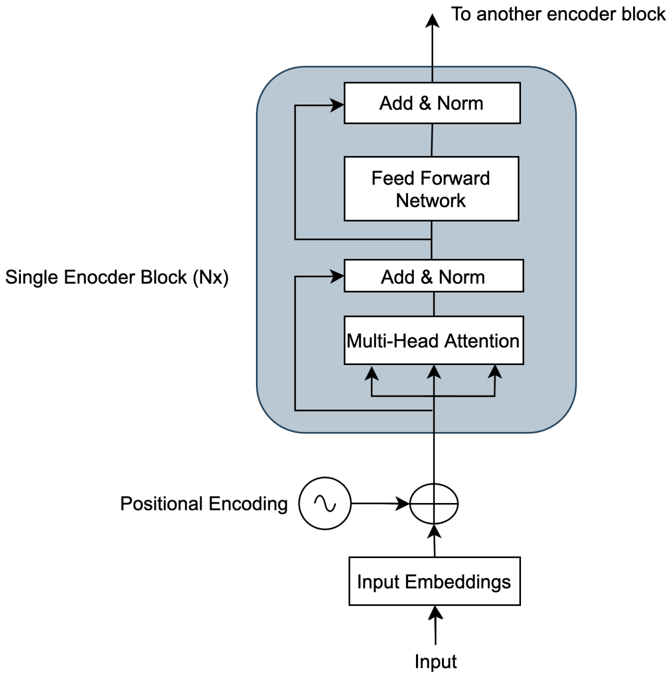


Figure 2.7: Different layer of encoder stack of transformer model.

Positional Encoding

Sequential model understand sentence and contain information of word position, but, in transformer model sentences are fed at once, so a separate mechanism introduced to determine the word order called positional encoding technique. Moving forward, instead of processing this position information separately, those word order in form of position vector are added to input embedding before multi-head attention layer. The dimension of position vector must be same as word embedding d_{model}. There are two major constraints, One, the word embedding information must not severely affected by the position vector and Second, each word position information should be identical. In Vaswani et. al [56] proposed transformer, sine and cosine function of different frequencies are used which form geometric progression from 2π to $2\pi \cdot 10000$ to calculate position vector,PV of the word. In other words, mentioned function 2.1 generates unique values which contain information about the position of word 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 is position and dimension respectively. And, after adding to the embeddings we get position encoding (PE)

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

2.2.2 Attention Mechanism

Human mind do no process all the available information from environment, rather, it focus on specific part of relevant information to complete the task. This biological mechanism is called attention and motivation behind attention mechanism in various machine learning tasks. And, this technique has shown significant improvements in performance of models.The idea of attention was first introduced in regression model by Naradaya et.al in 1964 [39], where they have proposed a better approach where weighted function $a(x, x_i)$ which encodes the relevance of feature as per prediction. In 2015, Bhadanu et al. [4] proposed an encoder-decoder approach with attention mechanism integrated at decoder. And, M.T Luong et. al [34] has shown that attentions mechanism gained up to 5.0 BLEU over non-attention models in neural machine learning task. Spyridon et. al [26] has examined the same mechanism in text classification task i.e sentiment analysis and observed 3.5 % improvement in accuracy.

The motivation behind using attention mechanism in transformer architecture model is computational complexity per layer and to introduce parallelized computation [56].There are different types of technique to calculate attention mechanism similarity based [17], dot product [34], scaled dot product [56], additive[4], etc. The focus of this report is scaled dot product attention used in transformer model. The scaled dot product or self-attentions is calculated as shown in equation 2.3 and figure 2.8

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

Where Q, K,V is query, key and value vector which is created by dot product of different weight matrix w_q, w_k and w_v with input. These which weights are initialized randomly initially and learned during training. This attention mechanism provide deeper relationship between words and result in better performance. But, for better representation and relationship between words, instead of just calculating single attention head, multiple attention heads are calculated to capture the different perspective of the sentence. Then, concatenating the result will provide comparatively better attention matrix. This mechanism is called multi-head attentions which utilizes the feature of parallelization.

So, in order to make sure that our results are accurate, instead of computing a single attention matrix, we will compute multiple attention matrices and then concatenate their results. The idea behind using multi-head attention is that instead of using a single attention head, if we

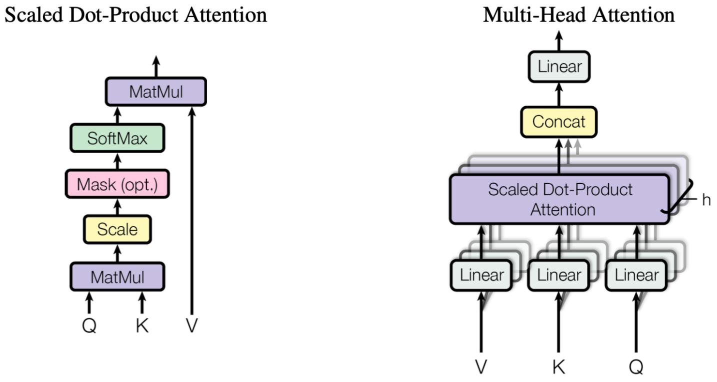


Figure 2.8: Scaled dot product and multi head attention calculation flow diagram [56].

use multiple attention heads, then our attention matrix will be more accurate.
Moving forward, the add and norm component is basically a residual connection followed by layer normalization which prevent faster training and preventing the heavy change in values during training. And, the feedforward network consists of two dense layers with ReLU activations. The parameters of the feedforward network are the same over the different positions of the sentence and different over the encoder blocks.

2.3 Language Models

The langauge model is function which learn the word representation from the text corpus and provide vectors which can further be utilized for downstream task such as machine translation, sentiment analysis etc. Their are various techniques to learn the representation either statistically or using neural network. Recently, neural network based language models such as BERT and DistilBERT became cornerstone of Natural language processing(NLP). These language models are trained in two ways as shown in figure , 1) Pre-training which involves learning the representation from huge unlabeled text corpus and model outputs the context representation vector [9] and enables the transfer learning. In this section, pre-training mechanism of BERT and DistilBERT is discuses further in this section. 2) Fine-tuning, user adapt the pre-trained language model to perform specific tasks by adding top feed forward layer to language models based on task. Then, continue to train the new model with the limited data of the target task. Such a fine-tuning approach can performed in low-resource and achieved state-of-the-art performance in many popular NLP benchmarks.

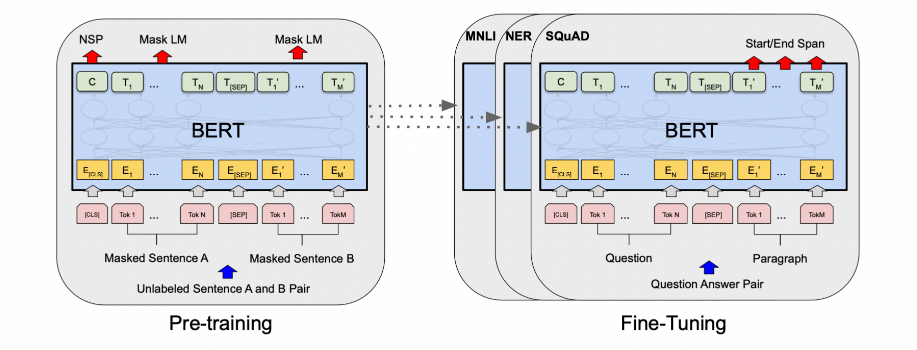


Figure 2.9: Pre-training and fine-tuning method of BERT model [9]. Pre-training learned in unsupervised fashion such as next sentence prediction(NSP) and masking. Pre-trained models are provided to perform further downstream task. During fine-tuning, all weights are further optimized based on available limited labeled data.

2.3.1 BERT(Bidirectional Encoder Representation From Transformers)

BERT(Bidirectional Encoder Representations from Transformers) was proposed by Devlin et. al[9], mainly based on Transformers [56],but not limited to it. BERT is basically a transformer encoder stack that outputs the representation context also called pre-trained model. BERT model is pre-trained on deep bidirectional representation of large unlabeled text in both right and left context, which can be trained further called fine-tuning by ending additional output layers to get state-of-art result in various NLP tasks like text classification, question answering, language inference, language translation, and so on. The main advantage of BERT based is simplifying the process of NLP tasks in machine learning and open access to contextualized embedding trained huge amount of words which is quite impossible at individually. Unfortunately, it is highly computational intensive which makes it costly at production scale and demands high performance computational machine.

BERT(Bidirectional Encoder Representations from Transformers) was proposed by Devlin et. al[9], mainly based on Transformers [56], ULMFit [20], ELMo [44], and the OpenAI transformer [46] but not limited to it. The transformer is an encoder and decoder in which the encoder reads the inputs and outputs a representation as a context vector, based on single-head attention or multi-head attention, and the decoder makes predictions based on those context vectors. However, BERT is the only Transformer Encoder stack that outputs the representation context. Moreover, unlike OpenAI transformers, which read data from left to right or right to left, BERT reads complete sequences at a time, making it bidirectional.

For training the large amount of unlabelled data, the main challenge is the lack of a label or a goal, so BERT uses two different learning strategies called Masked Language Model(MLM) and Next Sentence Prediction(NSP).In MLM, 15% of words are replaced with [MASK] tokens, and the BERT model predicts the masked word based on other words in the sequence.

In NSP, BERT model is given two pairs of sentences with [CLS] as the sentence start and [SEP] as the separation between sentences. Then, BERT model predicts whether the next sentence is the correct one or random. BERT model is pre-trained on a large amount of unlabeled text, but still, fine-tuning is required for specific tasks.
To make BERT models handle variety of downstream task, the input representations is sum of three different embeddings i.e. position embeddings, segment embeddings, and token embeddings as shown in figure 2.10.

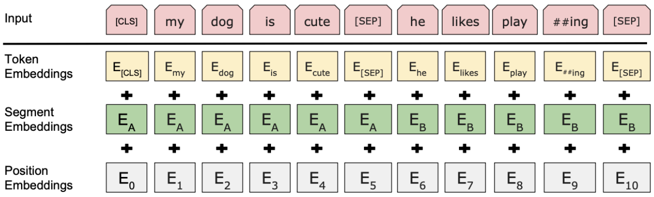


Figure 2.10: BERT model input includes three different embedding information to perform various downstream task[9].

A BERT paper [9] 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.

2.3.2 DistilBERT- A distilled version of BERT

DistilBERT, is compact version of BERT proposed by Victor et. al [50],technique and is comparable to BERT. The DistilBERT model doesn't have token type embeddings, pooler, and only have half of the layers as compare to BERT and leverage the principle of knowledge distillation [19] . In this technique, a smaller model is trained to mimic the behavior of larger model as shown in figure 2.11. For example, in the task of mask word prediction, masked sentence are provided to both BERT and DistilBERT model.

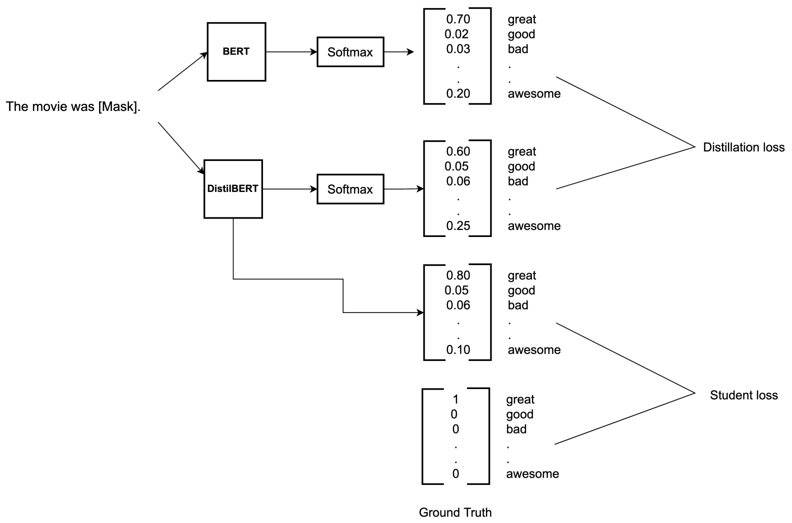


Figure 2.11: Pictorial diagram of DistilBERT working.
The knowledge distillation process of DistilBERT model.

The approach tries to minimize three losses 1) the loss between BERT and DistilBERT prediction, 2) loss between DistilBERT prediction and ground truth, and 3) the cosine embedding loss which is basically distance measure between representation learned by BERT and DistilBERT. Minimizing cosine embedding loss 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 [50] which has proved that possibility of reducing the size of large language model with minimum compromise.

2.4 Adversarial Attacks

In year 2013, C. Szegedy et al. study in image domain revealed that deep neural network are susceptible to small perturbation that can lead to unexpected model behaviour called adversarial attacks. Later, this phenomenon gained lot of attention in image domain and many research paper published on it. In the text domain, the adversarial example can be generated which can lead to unexpected outcome was first showcased by [42]. And, later various paper on different attack recipes were published in text domain [3, 7, 14, 15, 31, 48] and not limited to it. Sun et. all [53] proposed research on generating adversarial misspelling and observing the performance of BERT. It was found that the BERT model is prone to misspelling attacks and accuracy drops by 22.6 % on Stanford Sentiment Treebank(SST) dataset.

One of the related research paper proposed by Li et al.[31], in their proposed approach, the BERT model is used to generate word replacements for the target word. First, they identify the most important words of the BERT model i.e. the words in sequence have a high significance influence on the final output logit.

Then, by using another BERT model, they try to replace these words with the target word by utilizing its MLM capability. As per their claim, the average after attack accuracy was lower than 10% and perturb percentage was less than 10%. However, during the process of generation, there are chances of compromising with semantic constraints.
Most of the adversarial attacks in text domain is mainly two steps process 1) Identification of most important word and 2) Strategies to replace the word with suitable words. Goodfellow et al. [16] proposed the linearity hypothesis and claimed that the existence of adversarial examples was the linear behavior of DNNs in high-dimensional space.

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

2.4.2 Types of Adversarial attacks

Based on recent survey on adversarial attacks in text domain [22, 58], the adversarial attack can be classified based on level of knowledge, target based, and level of perturbation based as shown in figure. In level of knowledge, if attackers have complete knowledge of the system then it would be called white box attack and in case of only model output is known then it would be black box attack. Ebrahimi et al. [11] are the first to propose a whitebox gradient based attack to search for adversarial word/character based substitution. Jin et al. [25] proposed black box word level advesarial attack examples generations and showed that BERT model are vulnerable to those adversarial samples. In target based attack, attacker intentionally tries that model classify a specific class it would be called targeted attack and in case attacker just have aim to sabotage the model and does not depend on class then it would be untargeted attack as mentioned in section 2.4.1. In case word level of perturbation, attackers tries to

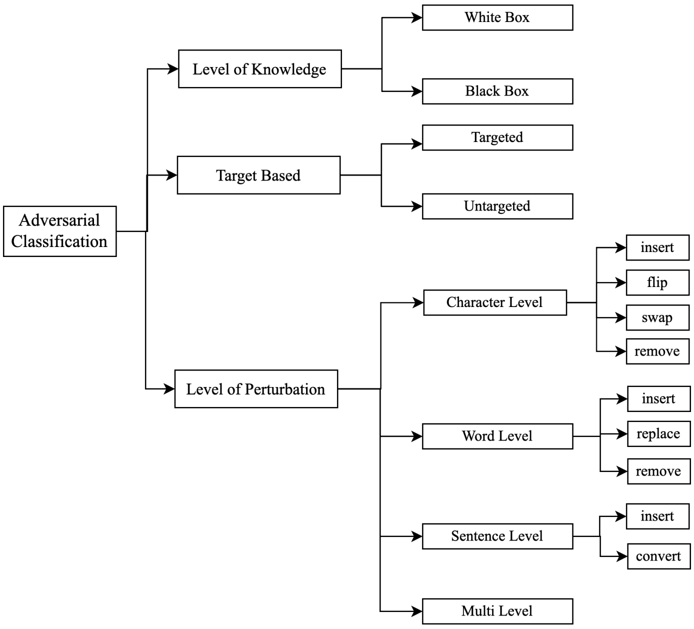


Figure 2.12: Different adversarial attack classification based on different aspect.

insert, replace, and remove the word. TextDeceptor [51] proposed a text attack approach, first they rank sentences and words and then replace them with similar words based on cosine similarity between word vectors, also considering POS (part-of-speech), which helps them to get grammatical correctness. Word level Textual Adversarial attack proposed by Yuan et. al [60], they are using sememe based word substitution. Using sememe-based word substitution is supposed to be more accurate since the substituted word has probably retained the same meaning. As per their claim, their attacks are notably having 98.70% success rate on IMDB dataset.

In char level, attackers tries to perform insert, flip, remove and swap operation to create adversarial text attacks. Eger et al. [12], proposed approach introduces visual perturber called VIPER that randomly replaces characters in the input with their visual nearest neighbours in visual embeddings space.

In sentence level, attackers tries to convert is via back translation or paraphrasing technique and insert some sentences in the text. Yankun et. al[49] proposed an approach that generates real-world meaningful text automatically using a variational encoder and decoder model, however, the sentences are sometimes completely different than the original.

2.4.3 Adversarial Training

The goal of adversarial defenses is to learn a model that is capable of achieving high test accuracy on both clean and adversarial examples [63].There are two strategies in text domain to fight against adversarial attack, one, proactively detecting adversarial text and second, model enhancement using adversarial training. In detecting adversarial text, mainly focussed on detecting unknown words and misspellings which create limitation of completely relies on original corpus vocabulary [58]. Goodfellow et al. [16] showed that robustness and generalization of machine learning models can be improved by including high quality adversarial example in training data.

Belinkov et al. [5] showed in their experiments that including mixed noises in training data can improve model robustness. In the experiments, performed by Li et al. [29], showed that textbugger attack adversarial training can also improve model performance and robustness against adversarial examples.

In another case, fast gradient method (FSM) approach training in text domain proposed by Miyato et al. [36] was introduced where methods generate adversarial examples by adding gradient-based perturbation to the input samples with different normalization strategies. Research related to adversarial training of language model is few. Liu et al. [32]. BERT model requires considerable computational power to perform virtual adversarial training [37] during pre-training.

Similarly, the mean teacher approach has shown comparative performance in computer visualization domain and claimed comparatively robust [55]. But, the performance of this approach in text - domain which includes language model and also verification of against adversarial attack was an open scope. Additional, the proposed approach utilizes the capabilities of language models such as context rewriting and back translation as data augmentation technique to create prominent adversarial unlabelled data was also an open scope which is discussed in length in section 5.3. Same approach utilized in generating adversarial examples, such as Siddhant et. al[15] proposed BERT's masked language models to generate alternate words for masked tokens, possible adversarial examples are derived. Hence, one of the major motivation behind this experiment.

3 Related Work

The proposed mechanism of adversarial training and verifying against different attack recipes was not an open scope of exploration. Various research has contributed towards adversarial training and defense mechanism against attach. However, respective research in text-domain is comparatively lower than in image-domain. The related work mentioned in this section is basically the result based on search on adversarial training on text-domain, based on language model, and focused more on increasing accuracy under attack. Mainly, the defense mechanism which deals in the adversarial training are mainly based on two approaches 1) Gradient based 2) Data Augmentation based.

In gradient based [16, 24, 32, 36, 64] , where the objective is to regularize the model by applying perturbations in the embedding space that maximize the adversarial loss. This approach also utilizes different noise strategies as perturbations to increase the adversarial loss and regularize model accordingly. The proposed researches on adversarial training are more focused on decreasing generalization error than increasing accuracy under attacks. And, studying performances under attacks still require a open scope of work. Based on the same approach, Liu et al. [32] first proposed the adversarial pre-training of language models which can increase both generalization and robustness and called the approach ALUM(**A**dversarial Training for large neural **L**ang**u**age **M**odel). The same approach is suitable for pre-training and fine-tuning both. However, often adversarial training is rather expensive due to inner maximization as well as complex and verification of proposed approach under various attack are yet to be studied.

A more recent approach is proposed by Danqinq et al . [64], where adversarial training is done by fast gradient methods(FGM) [36] and ensemble methods where multi-BERT model prediction aids in achieving robustness. Proposed approach uses multiple BERT(BERT, SciBERT, RoBERTa, ALBERT and ELECTRA) and make average ensemble for all the models to achieve superior performance . The use of large language model and internal maximization raise concerns on computational cost. Furthermore, as mentioned before this approach is also completely focused on increasing generalization and performance under attack is yet to find.

In noise based approach, Si et al. [52] proposed robust fine tuning of language models where augmenting the training dataset and perform mix-up augmentation during training hence called AMDA(Adversarial and Mixup Data Augmentation). Mixup augmentation is actually linear interpolation of the representations and labels pairs of training samples to create different virtual training samples. This approach has shown significant improvement in model performance un-

der attack but comparatively lesser original accuracy. In case of IMDB data, BERT model with original accuracy 91.27 % and accuracy under attack 14.83 % ,however, BERT with AMDA has shown original accuracy 91.10 % and accuracy under attack 31.52 % . The proposed approaches has shown very less or no improvement over generalization.
In an another approach, Bao et al. proposed a a multi-task learning framework, where language model is trained to classify the input text and also discriminate adversarial samples at the same time. Their proposed approach also generate adversarial samples using TextFooler [25] then apply frequency aware randomization of adversarial training set and finally combined to original training set. In IMDB data, this approach has shown marginal improvement in original accuracy from 92.4% to 92.8 % of BERT model but shown significant improvement in accuracy under attack i.e. 12.4 % to 89.2% . However, number of queries and word perturbation required to attack the model is required to further evaluate the proposed approach.
Previous works are mainly based on generating adversarial samples using gradient method and including them during training in order to increase model original accuracy. But, And, verifying robustness against adversarial attack is still open scope for research. Furthermore, because of no standard evaluation framework it is quite impractical to compare the performance of the approaches. In this master thesis, we are trying to calculate all required metrics for complete evaluation of the model under attacks. And, no approach in text domain that has used the proposed approach to increase above marginal increase in generalization as well as under attack performance.

4 Proposed Methodology

The proposed method calls for fine-tuning the BERT model using the mean-teacher approach for classification tasks and the adversarial unlabeled dataset for training as shown in figure 4.1. In the mean teacher model, two identical models are trained with two different strategies called student and teacher model. In which, only student model is trained, however, during training exponential moving weights are assigned to the teacher. Instead of taking average of many models decision in ensemble technique, teacher model is an average of consecutive students models hence mean teacher. As shown in Figure 4.1, two cost function plays important role while back-propagating i.e. classification cost and consistency cost. Classification cost($C(\theta)$) is calculated as binary cross entropy between label predicted by student model and original label. Consistency cost($J(\theta)$) is mean squared difference between the predicted outputs of student (weights θ and noise η) and teacher model (weights $\hat{\theta}$ and noise η'). The mathematical declaration is as follows.

$$J(\theta) = \mathbb{E}_{x_{adv}}[\|f(x_{adv}, \theta) - f(x_{adv}, \hat{\theta})\|^2] \tag{4.1}$$

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

$$O(\theta) = rC(\theta) + (1 - r)J(\theta) \tag{4.2}$$

During training, exponential moving average(EMA) weights of the student model are assigned to the teacher model at every steps and the proportion of weights assigned is controlled by parameter alpha(α). As mentioned in equation 4.3, while assigning weights, teacher model holds its previous weights in alpha(α) proportion and $(1 - \alpha)$ portion of student weights. The proposed approach can be seen in form of algorithm 1.

$$\hat{\theta}_t = \alpha\hat{\theta}_{t-1} + (1 - \alpha)\theta_t \tag{4.3}$$

Training data is utilized to create prominent unlabeled adversarial samples and three different data augmentation techniques are utilized 1) Synonym based, 2) Context based and, 3) Back translation based and discussed in length 5.3. The label of those adversarial dataset is discarded hence called prominent unlabeled adversarial data.

Erik et. al. [13]study revealed that model trained on noisy data exhibits a lower consistency

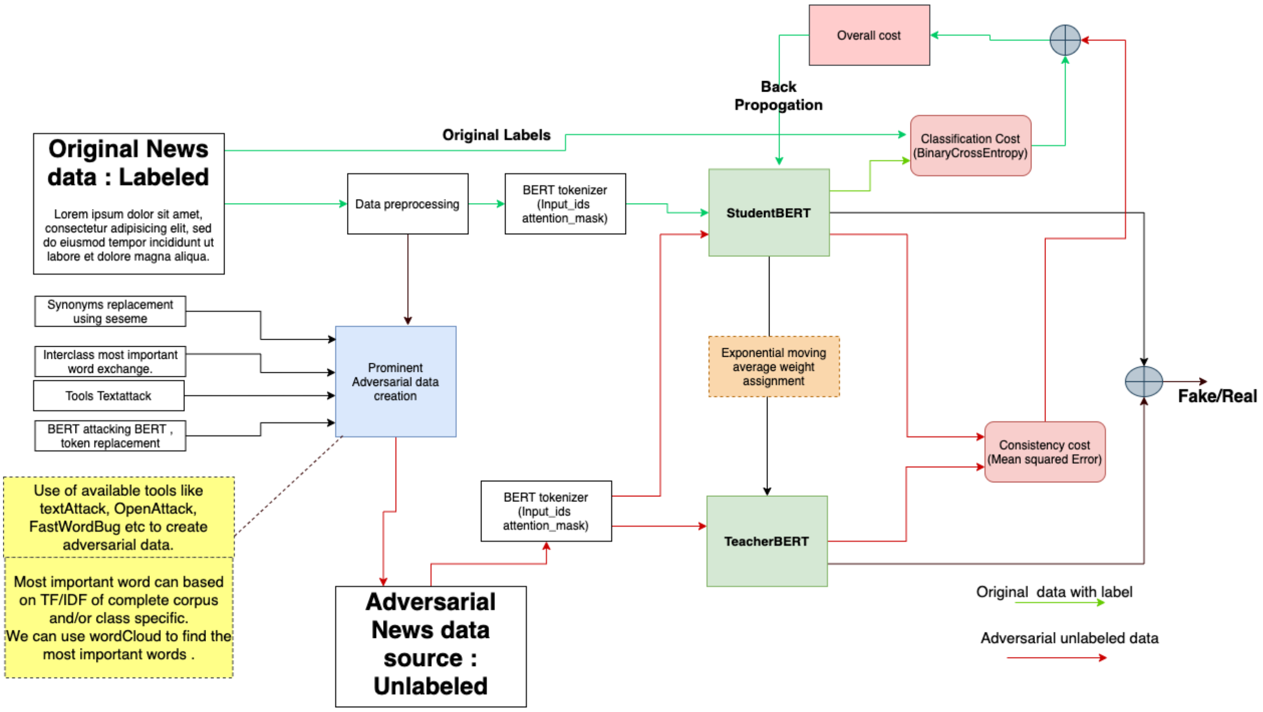

Overall cost
Back Propogation
Original Labels
Classification Cost (BinaryCrossEntropy)
Original News data : Labeled
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Data preprocessing
BERT tokenizer (Input_ids attention_mask)
StudentBERT
Synonyms replacement using seseme
Interclass most important word exchange.
Tools Textattack
BERT attacking BERT , token replacement
Prominent Adversarial data creation
Exponential moving average weight assignment
Fake/Real
Consistency cost (Mean squared Error)
Use of available tools like textAttack, OpenAttack, FastWordBug etc to create adversarial data.
Most important word can based on TF/IDF of complete corpus and/or class specific. We can use wordCloud to find the most important words .
BERT tokenizer (Input_ids attention_mask)
TeacherBERT
Adversarial News data source : Unlabeled
Original data with label
Adversarial unlabeled data

Figure 4.1: Proposed methodology

Algorithm 1 Mean Teacher Algorithm

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

that trained on clean data and reduces further if we increase the ratio of noise. However, including loss function during training which tries to reduce this consistency loss and can create comparatively robust model, hence called consistency regularization. The gradient based methods [36] and the proposed approach both exploit the principle of consistency regularization

[55]. However, in text domain the later approach has not been studied hence, the inspiration behind the study. Moreover, the motivation behind using the approach is to study the model performance by using comparatively simpler adversarial example and observing the impact of model in utilizing the capabilities of language models such as context writing and back translation. It is hypothesized that robustness can be achieved by learning more representations and regularize model by not letting to learn the deep insight of the representation to avoid over fitting as well. Furthermore, exponential moving average (EMA) plays crucial role in increasing generalization and performance under attack.

4.1 Text Attack Recipes and Tool

To evaluate the proposed approach, four black box attack recipes has been selected which satisfy lexical, grammatical, and semantic constraints. To utilize all the attack recipes to evaluate baseline BERT model and proposed Mean Teacher BERT, we have utilized TextAttack python package[38] for attack recipes. And, perturbation example can be seen in table 4.1. In this section, we will discuss about their attacking principle, working and characteristics.

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

Table 4.1: Perturbation example of attack recipes.

4.1.1 TextFooler

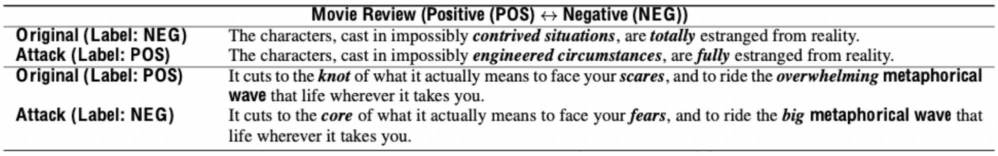
Di jin et. al. proposed Textfooler [23], a simple and effective adversarial attack generation strategy in black box settings which has characteristic of preserving the semantics, and grammar which they called utility-preserving adversarial examples as shown in figure 4.2. For better understanding, we briefly explain the three steps process of generating adversarial attacks below:

1. **Word Importance Ranking**: Given sentence of words, they create ranking of each word by calculating change before and after deleting the words called importance score. Followed by filtering out stop words using NLTK and spaCy libraries just to preserve the grammar of the sentence. which is basically injects antonymy and synonymy into vector space representations to improve vectors capability of semantic similarity. The

replacement policy completely depends on three constraint (1) Similar semantic similarity, (2) Fit within the surrounding context , (3) attacks the model. To calculate the similarity, using Universal Sentence Encoder proposed by Cer et al. [6], for encoding the sentence into high dimensional vector and calculating the cosine similarity between sentences. Then, selecting replacement candidates which has value above preset threshold value and create a pool of candidate.

2. **Replacement**: Among pool of candidates, if there already exist any candidate that can alter the prediction of target model then candidate with highest cosine similarity score between original and adversarial sentence is selecting. Otherwise, lower confidence score of label is selected.

TextFooler, has accessed the performance of BERT model under adversarial attack using IMDB movie review dataset . As per their experiment, the accuracy significantly dropped from 90.9% to 13.6 % with perturbed words 6.1 , number of queries sent to target model 1134 and average length of the IMDB dataset 215. Furthermore, TextFooler is computationally inexpensive and complexity increases linearly with respect to text length. *TextFooler, a strong black-box attack baseline for text classification models. However, the adversarial examples generated by TextFooler solely account for the token level similarity via word embedding, and not the overall sentence semantics. This can lead to out-of-context and unnaturally complex replacements (see Table 3), which are*



	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 [23]

4.1.2 TextBugger

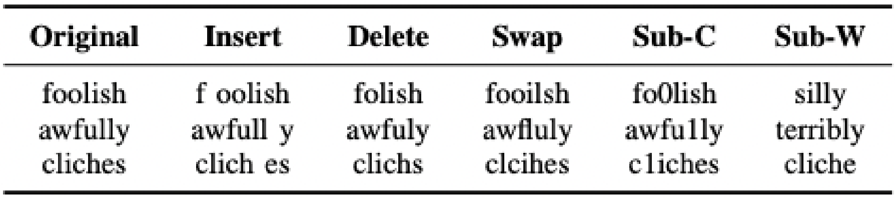
TextBugger proposed by Jinfeng Li et. al. [29], is based on misspelling of words or characters which are visually and semantically similar to the original text for human being. A simple misspelling can lead token to 'Unknown' which is mapped to unknown tokens id can also force machine learning model to behave incorrectly. On the other hand, studies shows that similar misspelling can still be perceptibly or inferred by the reader [3, 47] . This attack is focused on both character-level and word-level perturbation. Jinfeng Li et. al has proposed both white box and black box attack generation strategies, however, our report is focused on black box attack. Black box attack generation strategies is briefly discussed in Three steps:

1. **Finding Important Sentences**: The importance score of individual sentences in an

article is determined by confidence score of particular sentence by target model.

2. **Finding Important Words**: The importance score of word is the difference between confidence of target model with word and without word.

3. **Bugs Generation**: In TextBugger, they use five bugs generation strategy (1) **Insert**: Inserting space into words, (2) **Delete**: Deleting random character, (3) **Swap**: Swapping random adjacent character, (4) **Substitute-C**: Substitute character with visually similar characters, and (5) **Substitute-W** : Replacing word with top-k nearest neighbour in context aware word vector space , as shown in figure 4.3



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

Figure 4.3: TextBugger 5 bug generation strategies [29]

TextBugger model is evaluated against LR, CNN, and LSTM using IMDB movie review dataset, and have shown 95.2%, 90.5% and 86.7% respectively, with perturbed word 4.9%, 4.2% and 6.9 % respectively. However, observing the effectiveness against BERT model is still not evaluated and is explored in this report. Unlike TextFooler, TextBugger computational complexity is sub-linear to text length and can generate adversarial attacks in comparatively less time.

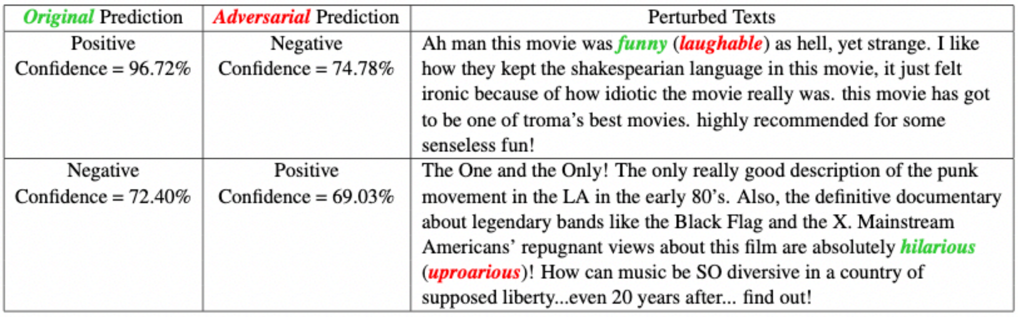
4.1.3 Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency (PWWS)

Shuhuai et al.[48] proposed method of synonym and named entity (NE) replacement method which is determined by the words saliency and the classification probability, and proposed a greedy algorithm called probability weighted word saliency(PWWS). To replace the word with the synonym is decided by significant change in classification probability and at the same time have minimum word saliency. Their approach is mainly can be explained in two steps as follows:

1. **Word Selection Strategy**: Word saliency defined as degree of change in classification probability of the model if the word is set to unknown [30]. Calculating word saliency vector of each word in a text, and prioritizing the words according to degree of change in classification probability after replacement as well as minimum word saliency of that words.

2. **Replacement Strategy**: To find the substitution, they used WordNet to find the synonym of the words. And, if word is Named Entity(NE), then replacing the NE with similar type NE appeared in opposite class.

Finally, greedily iterate through words replacement to make model change the label. This approach has been evaluated with Word based CNN [27], Bi-directional LSTM model, LSTM and Char-based CNN[62], however, still have open scope for evaluating against language models. Considering Bi-LSTM result , the accuracy dropped from 84.86 % to 2.00% with perturbation 3.38% for IMDB dataset, example is shown in figure 4.4. However, the computational and time complexity of the proposed approach is comparatively higher than other discussed strategies.



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[48]

4.1.4 BAE: BERT-Based Adversarial Examples

Garg et al.[15] proposed a novel black box approach to generating adversarial examples by utilizing BERT masked language model(MLM). According to their proposed approach, first they calculate words importance by computing the decrease in probability of predicting the correct label after deleting that particular word, similar to Textfooler [23] and PWWS [48]. And, using pre-trained BERT MLM model, where a particular word is replaced with MASK token and let the MLM model predict the context specific words. Then, filter top K tokens based on most similarity score(Threshold 0.8) using Universal Sentence Encoder [6] and removing words that doesn't fall into similar part-of-speech(POS) as the original word. Now, replacing the original word with top K(50) tokens, iterate from most similar token in decreasing order until attack is successful and trying all combination. A schematic working diagram is shown in the figure 4.5.

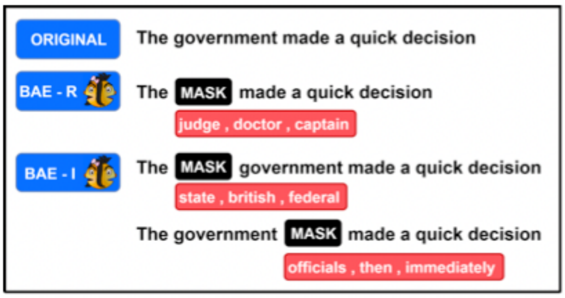

ORIGINAL
The government made a quick decision
BAE - R
The MASK made a quick decision
judge , doctor , captain
BAE - I
The MASK government made a quick decision
state , british , federal
The government MASK made a quick decision
officials , then , immediately

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

4.2 Research Questions

The main motivation behind the proposed methodology is to reveal using various data augmentation technique can enhance the performance of language model under attack without compromising on generalization. In other word, semi-supervised fine tuning of language models can create comparatively robust language models and language models still have scope of improvements. Precisely, the master thesis is trying to answer by creating teacher model using proposed semi-supervised approach and comparatively study the performance of conventional and proposed way of fine tuning in terms of generalization and under attack accuracy. The reason behind this hypothesis lies mainly in two concept, 1) averaging models weights over training steps tends to produce a more accurate model than using the final weights directly [45] and teacher model is an average of consecutive student models thus teacher model should perform better. 2) The inclusion of data augmentation can enhance the robustness of the model [5] , hence the proposed approach should perform well in under attack too.

The robustness in this experiment would be measured using metrics like original accuracy, accuracy under attack, average word perturbation , average number of queries, and attack success rate which is discussed in length in 5.5. Hence, the hypothesis would be the proposed approach comparatively should have higher original accuracy, accuracy under attack and require higher number of queries and perturbation requirement to get attacked as well. And, comparatively lower attack success rate. Four different attack recipes are used to evaluate the model performance which is discussed in 4.1. Furthermore, the experiment also tries to understand the robustness of model by studying the distribution of confidence score prediction under attack can reveal the resilience of model towards attacks. If model not allowing attack recipes to wrongly classify with higher confidence is a sign of robustness as well.

5 Experiment

5.1 Dataset

For assessing the performance of baseline model and proposed model. We have selected two datasets, one is Covid-19 fake news dataset provided at Codalab competition and IMDB review binary classification dataset. IMDB dataset is sentiment classification dataset which contains movie reviews of the user and having two classes positive and negative and most generally used in classification and adversarial attack research papers. This dataset has 50,000 labelled but due to computational limitation, we have filtered and sampled only dataset whose length is in between 6 to 150 as augmentation process takes quite longer time which leads 6000 samples to train and evaluate our model. The train and test size is show in table 5.1. And, we have sampled 6000 training labeled samples to create augmented unlabeled dataset. Average length of the filtered dataset is 100. The label distribution is completely balanced in training dataset.

Covid-19 Fake news dataset is recent dataset specific for fake news detection tweets related to COVID 19 with label as fake and real. Covid-19 fake news dataset size is 8000 and we have completely utilized this dataset. Observing the performance of proposed model in more recent fake news dataset is motivation behind selecting the dataset. In contrast to IMDB dataset, the average length of Covid-19 fake news dataset is 25 as mentioned in table 5.2.

Covid-19 fake news dataset has mostly hashtags and less English words vocabulary which might create challenge for those recipes, we would like to investigate the performance of models under this scenario too. The label distribution is almost balanced as compared to IMDB dataset which is completely balanced. The intention is to observe the effect of label distribution.

Dataset	Train	Test	Unlabeled	Aug. Unlabeled
codalab (Positive/Negative)	3199/2891	1071/969	xxxx	xxxx
IMDB (Fake/Real)	xxxx/yyyy	xxxx/yyyy	xxxx	xxxx

Table 5.1: Train/ Test split details of dataset

5.2 Data Pre-processing and Exploration

As language models are based on learning the context of the sentences hence least affected by stop words and removing those words might affect the performance. Hence, the one of the benefit of language model is its negligible requirement of data cleaning or no data cleaning. In our case we have performed following data pre-processing steps:

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

For achieving the particular task, we have utilized texthero python library which provide function related to data pre-processing and exploration.

Data Exploration

For training the mentioned models, we selected almost equal distribution of label in training data and test data, and same training data we have utilized to create unlabeled augmented data for training via proposed method as shown in 5.1.

Dataset	Avg. Length
codalab	xxxx
IMDB	xxxx

Table 5.2: Train/ Test split details of dataset

5.3 Data Augmentation

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

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

The reason behind calling this dataset unlabeled augmented dataset is while augmenting the dataset there are high chance that the information is changed or completely opposite in contrast to label. Therefore, once we augment the data , we will not be using the label of augmented data hence unlabeled augmented dataset. During experiments, various python packages like text attack , nlpaug, and various basic python packages for synonym changes were tried. But,

in the end , considering the time and computation constraint, we have selected nlpaug data augmentation package to achieve all three augmentation strategies and can be accessed in following link. To augment the dataset, we have taken train dataset with dropping the label columns , then we randomly split this dataset into three part. One part for synonym augmentation, second part for context based augmentation, and remaining for back translation.
There was one more idea to create unlabeled data using different attack recipes, which might can give better result under attack. But, the motivation behind not using attack recipes because of assumption that model is created with no or little understanding of nature of attacks. If we have implemented a particular attack recipe to create unlabeled data then, it might increase the robustness for specific attack recipes but overall evaluation of model performance would be biased and we want it completely random.
For synonym augmentation, wordNet lexical English database is used as augmentation source which consist of word definitions, hyponyms, and semantic relationship. Same database in our case utilized for synonym replacement. Parameter maximum augmentation(aug_max) is used to control the level of augmentation. It is set to 50 and 15 for IMDB and Covid-19 fake news dataset respectively. One more parameter called *iter* is utilized to create two different copies of synonym augmentation dataset.
TODO: Image of synonym augmentation.
Context based augmentation is based on replacing the words the words in the sentence without changing the context. Generally, language models are used to achieve this particular task, hence it is quite time and memory expensive. Therefore, DistilBert language model is utilized to perform this augmentation.
Back translation is the process of converting sentences in different languages and then translating back to original language. Like, context based augmentation, Marian translation framework is utilized , hence time and memory expensive. In our experiment, we are converting sentence into Romance language and back to English. We have used CITE model to perform this experiments. Marian translation framework is comparatively an free, faster and efficient .
TODO: Image for Back translation.

5.4 Experiment Environment description (1-2 paragraph)

To successfully perform experiments, Google colab notebook with GPU is utilized to perform the experiments. TODO: GPU details need to mention or image.

5.4.1 Hyper parameter Details

As shown in table 5.3, to train the baseline model and proposed model, we have used these parameter values. However, observing the performance with different settings is not the current focus and open for future task.

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

Table 5.3: Hyper-parameters Details

5.5 Metrics

In the experiment, the generalization of the model is calculated using accuracy of the target model on original test samples, here it would be referred as original accuracy. Original accuracy is percentage of correct prediction over complete test samples.

Then, robustness of the target models is calculated using 4 different metrics 1) Accuracy under attack, 2) Average word perturbation 3) Average number of queries and 4) Perturbation Score. Accuracy under attack is the percentage of correct prediction over test samples i.e accuracy of the target model against crafted test samples. Comparison of original accuracy and accuracy under attack can provide significant information about efficiency of the target model and attack recipes as well. If more significant difference between both metrics signals the lower robustness against attack. The attack success rate is also counter part of accuracy under attack which reveals the success of attack recipes.

Another metric apart from accuracy, the average perturbed word in percentage is noted which is average number of words need to change in order to get successful attack. Generally, higher word perturbation signals lower robustness and this metric also depends on text length.

Additonally, average number of queries is the average of number of times attacks recipes send perturbed text samples to the target models for successful attack. More the average number of queries leads higher robustness against attack.

Moving forward, perturbation score which is confidence score or predicted probability of the target models under attack. According to hypothesis, the only for successful attacks, lower the mean signals higher robustness .

5.6 Threat Model

In this experiment, the attack recipes are selected based of level of knowledge, target, level of perturbation, and assumptions. As most of the attack are basically done under black box setting, the attacks recipes are only exposed to the target model output and test samples from same corpus. Under this black box setting, the attacker can only query the target model with perturbed inputs and getting the corresponding confidence score or predictions . And, the attacker is not aware of the model architecture, parameters, or training data. It can only query the target model with supplied inputs. As test samples is from original corpus of the data, hence that the present test samples has extracted from different corpus is an assumed. It is also assumed that attacker has the access to similar baseline model. As the tokenizer used while training i.e. from huggingface [21] open source python package for language models, similar tokenizer is initialised from same package.

Evaluation of the model is only performed in binary classification task, hence the attacks are mainly un-targeted but, because of binary classification it is assumed as targeted classification. And, only word level and multi-level which includes word and char level of word perturbation are considered. As, most of attacks are basically word level and character level hence, clear the intent of this selection.

The robustness of target models is evaluated using four attack recipes mentioned in4.1, hence it is assumed that attacker has access to those attack recipes and the performance of model could be different if some other attack recipes are utilized. Furthermore, it is also assumed that the attack test samples are syntactically and semantically similar as per the claimed based on the attack recipes paper , hence, similarity metrics are not included in the experiment. The model is trained of mentioned settings 5.4.1, and model can perform better or worse in different settings. Moving forward, about the transferability of the proposed approach , as the experiment is performed on two different language models differ in pre-training, it is assumed that the approach would also perform the same if some other language models are utilized.

6 Result Analysis

According to experiment result of both the datasets shown in table 6.1 and 6.2, the proposed approach i.e. MTBERT has outperformed all the other language models considering all the metrics. If comparing proposed versus baseline model, the MTBERT and MTDistilBERT model has also performed better than respective baseline models i.e. BERT and DistilBERT. MT BERT model has shown 1-2% improvement in original accuracy than baseline model in both the datasets as shown figure 6.1. Proposed model has shown 25-30% improvement in accuracy under attack at the same instance require considerably higher number of queries and word perturbation and in other words, has shown lowest drop in original accuracy. In opposite, BERT and DistilBERT has shown lowest performance considerably require less number of queries and overall word perturbation.

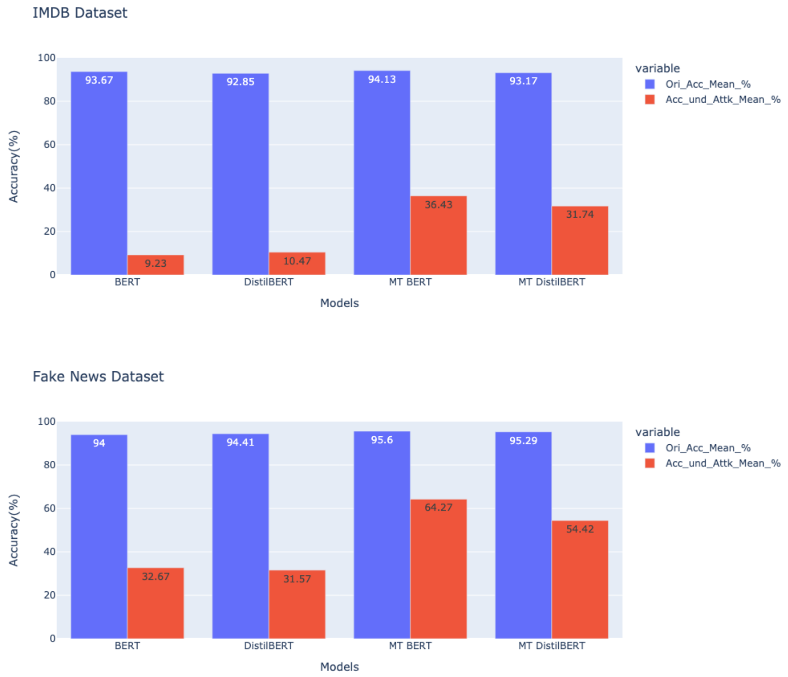


Figure 6.1: Original accuracy and Accuracy under attack comparison plot. MTBERT has shown better performance in both metrics followed by MT DistilBERT. Baseline models (BERT and DistilBERT) has shown highest drop in accuracy when under attack , however, proposed models have shown significant robustness against attack.

Attack Recipe	Model	Acc. und Attack(%)	Acc. Succ. Rate(%)	Avg. Pert. Word(%)	Avg. No. Queries	Ori. Acc.(%)
BAE	BERT	33.93	63.77	3.78	242.24	93.67
	DistilBERT	33.25	64.18	3.56	238.20	92.80
	MT BERT	56.45	40.03	3.55	198.26	94.13
	MT DistilBERT	53.50	42.51	3.31	285.70	93.05
PWWS	BERT	0.60	99.36	3.97	749.33	93.67
	DistilBERT	1.70	98.17	3.98	750.12	92.80
	MT BERT	23.20	75.35	5.70	890.84	94.13
	MT DistilBERT	17.55	81.14	5.37	867.65	93.05
TextBugger	BERT	2.30	97.54	22.04	235.27	93.67
	DistilBERT	5.45	94.03	20.80	258.47	92.80
	MT BERT	35.13	62.68	28.57	449.96	94.13
	MT DistilBERT	30.05	67.70	26.66	420.39	93.05
TextFooler	BERT	0.10	99.89	5.14	279.12	93.67
	DistilBERT	1.07	98.85	5.07	278.73	92.85
	MT BERT	30.92	67.16	8.04	720.77	94.13
	MT DistilBERT	25.48	72.64	7.54	613.91	93.17

Table 6.1: IMDB datasets experiment result. MTBERT model has performed well overall and shown comparatively better robustness towards attack.

Attack Recipe	Model	Acc. und Attack(%)	Acc. Succ. Rate(%)	Avg. Pert. Word(%)	Avg. No. Queries	Ori. Acc.(%)
BAE	BERT	67.40	28.30	20.43	82.32	94.00
	DistilBERT	67.53	28.46	18.86	79.08	94.40
	MT BERT	77.60	18.83	16.54	88.56	95.60
	MT DistilBERT	74.00	22.24	19.46	86.40	95.17
PWWS	BERT	24.50	73.94	20.23	214.95	94.00
	DistilBERT	25.40	73.09	19.28	210.40	94.40
	MT BERT	59.47	37.80	17.90	236.99	95.60
	MT DistilBERT	48.87	48.65	18.70	227.90	95.17
TextBugger	BERT	31.27	65.74	24.77	82.49	94.00
	DistilBERT	25.57	72.92	22.54	76.61	94.40
	MT BERT	65.87	31.10	23.45	114.34	95.60
	MT DistilBERT	54.23	43.01	24.57	96.05	95.17
TextFooler	BERT	7.53	91.98	22.52	180.88	94.00
	DistilBERT	7.47	92.09	21.40	164.27	94.40
	MT BERT	54.13	43.38	21.24	283.06	95.60
	MT DistilBERT	42.27	55.57	24.45	198.96	95.17

Table 6.2: Fake news dataset: MTBERT model has performed well overall and shown comparatively better robustness towards attack.

Now, if we discuss about the performance of model considering individual metrics metrics. MTBERT require comparatively higher number of queries to get attacked. As shown in figure 6.2. BAE attacks has shown lowest requirements for queries followed by TextBugger, but also found out to be least effective in attacking. Overall, PWWS need significantly higher number of queries followed by TextFooler and also found out to be effective in attacks. Furthermore, both attack recipes has shown positive dependencies on text length,if we observer the considerable difference in queries for both the datasets 6.2. However, BAE and TextBugger shown least difference with respect to text length. In another point, considering only TextFooler attack recipe, the difference between proposed and baseline models queries also increases with respect text length. In fake news dataset, TextFooler number of queries difference between BERT and MTBERT is less but doubled in IMDB datasets. As shown in table 6.3, the proposed model has mean value VALUE and VALUE compared to V1 and V2 a difference of and have broader distribution than baseline model. It is evident that model created by proposed approach required higher number of queries, and the difference is clearly visible in IMDB datasets.

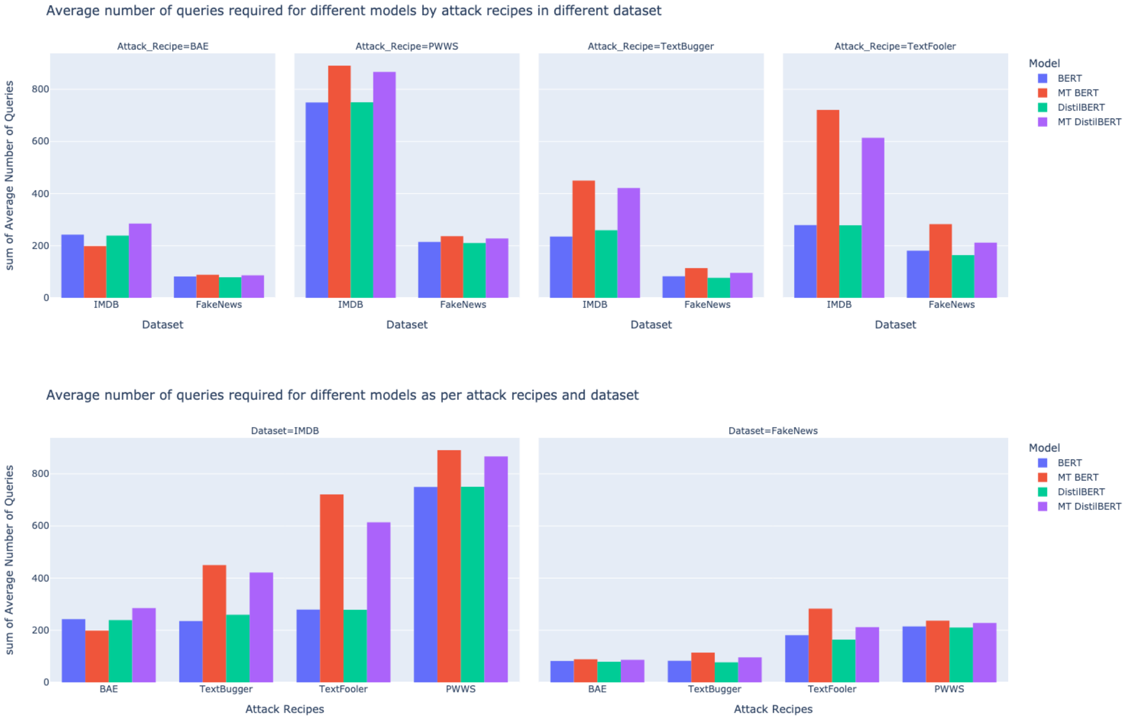


Figure 6.2: Number of queries with respect to datasets and attack recipes.
PWWS has shown highest overall number of queries requirement followed by TextFooler.
And, we can observe how number of queries increases with respect to text length.

	Dataset	model	count	mean	std	min	25%	50%	75%	max
0		BERT	7,005.00	160.17	114.93	10.00	75.00	130.00	214.00	885.00
1	FakeNews	DistilBERT	12,096.00	138.74	100.89	10.00	63.00	110.00	188.00	945.00
2		MTBERT	3,143.00	138.50	115.72	13.00	58.00	99.00	181.00	708.00
3		MTDistilBERT	7,373.00	142.82	122.53	10.00	56.00	103.00	188.00	1,075.00
4		BERT	16,101.00	361.27	282.52	24.00	161.00	232.00	507.00	1,841.00
5	IMDB	DistilBERT	16,608.00	410.52	309.74	24.00	164.00	264.00	667.00	1,708.00
6		MTBERT	10,316.00	434.36	334.49	24.00	165.00	297.00	674.00	2,207.00
7		MTDistilBERT	12,287.00	411.52	340.87	24.00	158.00	253.00	634.50	2,128.00

Table 6.3: Queries descriptive statistics of successful attack. Proposed models require higher number of queries in contrast to baseline model.

To understand the robustness of models with respect to word perturbation, less significant difference is observed. In fake news dataset, the average word perturbation required by proposed models is comparatively higher , however, for IMDB dataset it is opposite. The difference in word perturbation in both dataset is higher, and IMDB dataset has shown higher words perturbation in contrast of Fake news dataset except TextBugger which has shown almost same perturbation rate.TextBugger require higher number of word perturbation followed by TextFooler. On the other hand, BAE required lowest number of perturbation followed by PWWS.

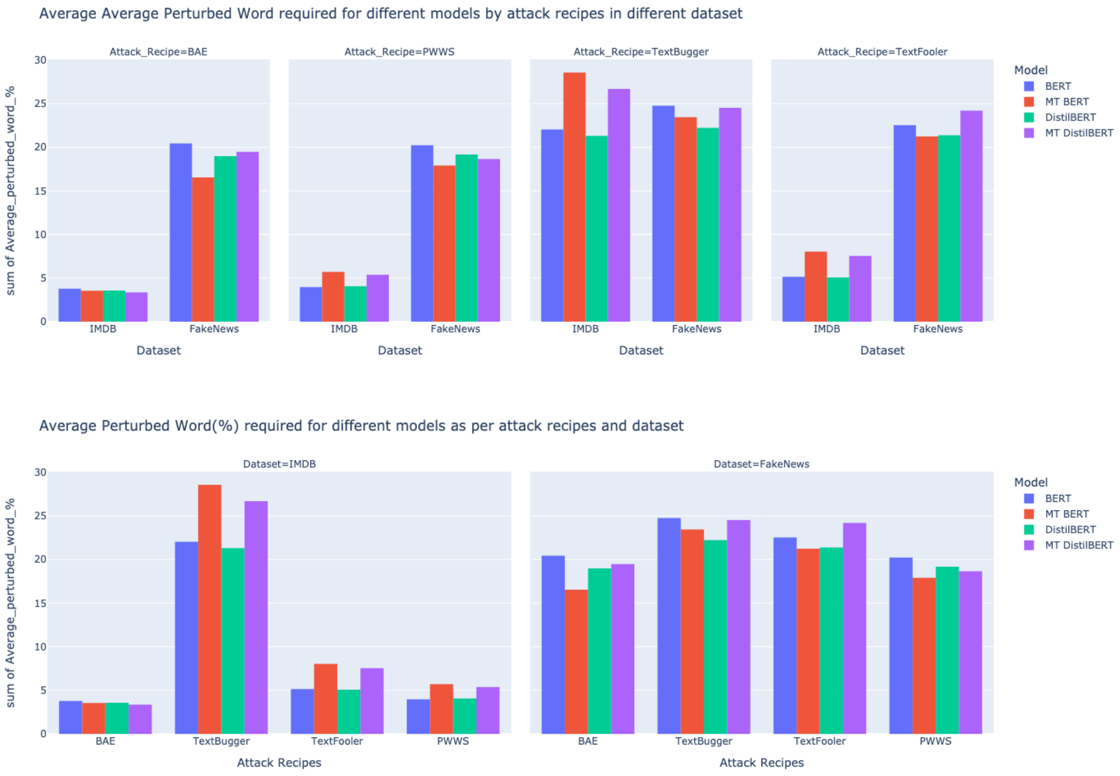


Figure 6.3: Word perturbation as per datasets and attack recipes
TextBugger has shown highest word perturbation in both the datasets and also shown less difference between both dataset. However, BAE has lower word perturbation followed by PWWS attack recipe. Except TextBugger huge difference of word perturbation between datasets is observed. Fake news dataset has shown significantly higher word perturbation requirements.

Intuitively, while running the experiments we logged perturbation score that is predicted probability under attack in other term confidence score. To answer the question, how much proposed and baseline model let the attacker to change the confidence score of successful attacks. As shown in figure 6.4, the distribution of proposed model is comparatively more left shifted than baseline models. The proposed model does not let attack recipe go beyond a particular limit i.e. 0.70 6.4, however, baseline models are more uniform and shown less resilience. The mean and range of MT BERT model is 0.53 and 0.50 to 0.55 , but for BERT model 0.59 and 0.50 to 0.66, which is a significant improvement in the model.

In case of attack recipes, MT BERT has shown highest robustness towards all attack recipes.

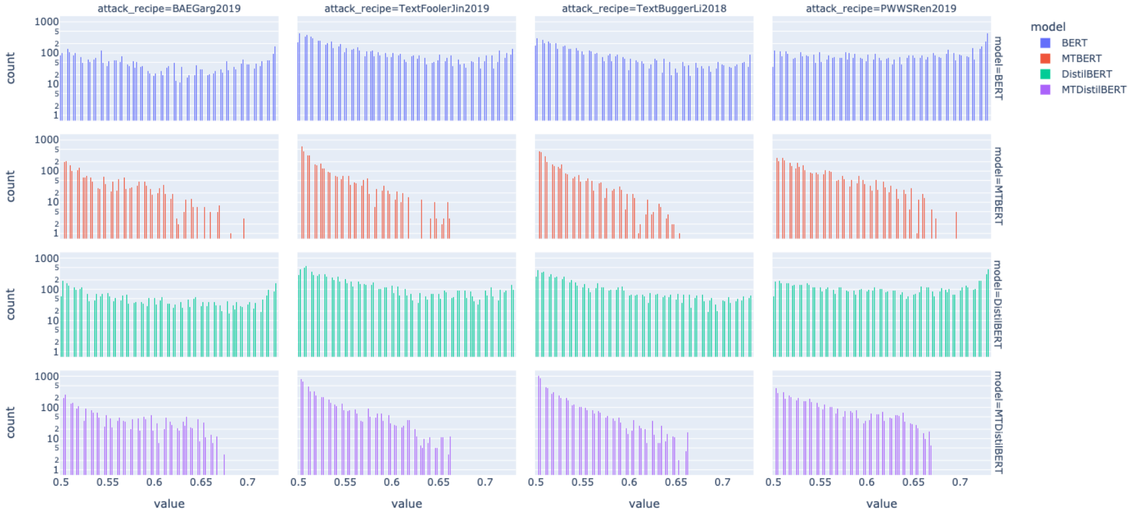


Figure 6.4: Perturbation Score Distribution
The Proposed model are comparatively more left shifted and collected in range of 0.50 to 0.55 i.e 10 % improvement compared to baseline model.

	model	count	mean	std	min	25%	50%	75%	max
0	BERT	38,507.00	0.59	0.07	0.50	0.53	0.57	0.66	0.73
1	DistilBERT	47,222.00	0.59	0.07	0.50	0.52	0.57	0.64	0.73
2	MTBERT	22,262.00	0.53	0.04	0.50	0.51	0.52	0.55	0.70
3	MTDistilBERT	32,012.00	0.54	0.04	0.50	0.51	0.52	0.56	0.67

Table 6.4: Perturbation score table
Perturbation score descriptive statistics of successful attacks combining both dataset. Proposed model has shown significant improvement over baseline model. And, mostly in limited in range 0.53-0.70 and 75 % of attacks score is under 0.55 a sign of robustness.

However, especially highest in BAE attack recipes and lowest in PWWS and TextFooler(still higher than other models. PWWS and TextFooler are more successful in attacking the models considering both dataset. However, MT BERT has shown highest robustness compared to other mentioned models.

6.1 Discussing on Research Questions

As per the experiment result and analysis , the baseline model has outperformed considering all metrics and attack recipes, and shown significant robustness against attacks. The proposed approach comparatively has shown notable improvement in generalization and also significantly lower attack success rate. Additionally, require comparatively higher word perturbation rate ,

number of queries and perturbation score is also limited in lower range. Which answer to the question that the language model trained with noisy data and including loss function called consistency loss while training could result in better generalization and robustness than fine-tuned in conventional way as it is achieved by gradient based method. The mean teacher as proposed by [55] in image- domain can show similar result in text-domain , if suitable noisy data specific to text is included. In the end, opens the scope of improvements in language models either during pre-training or fine tuning.
Answering to the question of why proposed approach works, is because of including only noisy data in training samples would not actually lead to lower generalization, but at the same time if we include respective loss function which optimizes weights according to both classification and consistency loss can help model avoiding over-fitting and wont let model to learn deep insight of data. And, at the same time let model learn broad level of representation by including many noisy word synonym. Which also backs points made by on consistency regularization via noisy data [5, 13]. As the experiment performed in two different models, i.e. BERT and DistilBERT also proves its transferability to other language models.

7 Limitation, Future Work and Conclusion

7.1 Limitations

One of the major challenges in generating text samples are, unlike image domain the perturbation can be made impreceptible to human but in case of text domain achieving similar level of imperceptibility is still a open challenge due text data discrete nature. Most of the researches claimed to manage similar level of syntactic and semantic of the original text, but, in reality, those attack samples can be recognizable to human.

The attacks, data augmentation technique, and semi-supervised training approaches implemented in image-domain can not be directly utilized in text domain, hence, it still require separate effort and contribution to implement it into the text-domain and observer their performance. In the experiment, the proposed approach has shown comparatively better performance than conventional model, but, does not claim to be robust to all types of attacks. Till now, no defense strategy can handle all different types of attacks as the nature of attack is not known. And, including respective noisy data can improvement robustness against its corresponding attacks.

Till now in security domain in text-domain, no standard evaluation metrics and framework present to compare different researches and strategies which demand focused research on this particular domain. In another case, the recent researches are more focused on increasing model generalization, than study the of those approaches in models robustness are not studied.

7.2 Future Work

The evaluation of the proposed approach could be performed to evaluate more recent state-of-art language models, and the work could also concentrate on hypothesis that extreme pre-training of language model can hurts robustness of the model. Furthermore, effectiveness of individual augmentation techniques on robustness can be studied in future.

In the experiment, training data is utilized to create augmented prominent unlabeled data, however, huge amount of unlabeled data can be utilized instead of augmentation and performance can be evaluated. In another case, including respective adversarial text generated by attack recipes can evaluated in future. Quantitative comparison of gradient based adversarial training and proposed training approach still has open scope of work.

7.3 Conclusion

The motivation of this experiment is quantitative comparative analysis of generalization and robustness proposed of semi-supervised approach of fine-tuning language model and focused on increasing the robustness of language model. Furthermore, including prominent unlabeled adversarial data can lead to better robustness. For experiment, attack recipes i.e. PWWS, BAE, TextFooler, and TextBugger, data augmentation i.e. word synonym, context augmentation, and back translation, and two language models i.e. BERT and DistilBert are utilized.

In conclusion, the result of experiment has shown that proposed approach has shown 1-2 % improvement in original accuracy and 25-30 % improvement in accuracy under attack. Moreover, shown higher requirement for word perturbation and number of queries which proves its robustness against conventional way of fine-tuning. Furthermore, this study first revealed that language models are vulnerable to adversarial attacks and moving forward, also shown that there is an open scope of improvement in language models. The experiment, also provided a direction of working towards defense against adversarial attacks which is discussed in future work.

Bibliography

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

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

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

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

[5] 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]`.

[6] Daniel Cer et al. “Universal Sentence Encoder”. In: *arXiv:1803.11175 [cs]* (Apr. 2018). arXiv: `1803.11175 [cs]`.

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

[8] 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`.

[9] 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]`.

[10] Chunning Du et al. “Adversarial and Domain-Aware BERT for Cross-Domain Sentiment Analysis”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 4019–4028. DOI: `10.18653/v1/2020.acl-main.370`.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[24] 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`.

[25] 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`.

[26] 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`.

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

[28] 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]`.

[29] 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`.

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

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

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

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

[34] 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]`.

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

[36] 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]`.

[37] 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]`.

[38] 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]`.

[39] 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`.

[40] 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]`.

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

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

[43] 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.

[44] 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.

[45] 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.

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

[47] 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.

[48] 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.

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

[50] 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].

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

[52] 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.

[53] 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].

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

[55] 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]`.

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

[57] Boxin Wang et al. "InfoBERT: Improving Robustness of Language Models from An Information Theoretic Perspective". In: *International Conference on Learning Representations*. Sept. 2020.

[58] 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]`.

[59] 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]`.

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

[61] 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]`.

[62] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-Level Convolutional Networks for Text Classification". In: *arXiv:1509.01626 [cs]* (Apr. 2016). arXiv: `1509.01626 [cs]`.

[63] 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]`.

[64] 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]`.