



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Application of BERTweet for Stance Detection task on Twitter

by

Huang Xudong

Supervisor: Dr Terry Walcott

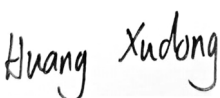
Project submitted as part of the requirements for the
award of MSc Data and Artificial Intelligence

May 2023

DECLARATION

I, Huang Xudong declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

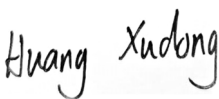
This work has not previously been accepted as part of any other degree submission.

Signed : 

Date : 2023/05/20

FORM OF CONSENT

I Huang Xudong hereby consent that my Project, submitted in candidature for the M.Sc. Data and Artificial Intelligence, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed: 

Date: 2023/05/20

ABSTRACT

With the development of social media, it has become an important research topic to mine the stance information contained in massive microblog data. Twitter has long been one of the most popular microblogging social platforms, with a large number of users and a wide range of topics. Through the mining of Twitter text data, it can provide valuable support or reference for public policy making, public opinion testing and business decision making.

Stance detection is an important part of natural language processing (NLP). Its purpose is to determine the author's stance on a particular object or proposition in the text (Favor, Against, Neither”), and has attracted a lot of attention because of its wide application prospects. Pre-trained language Pre-training language model based on BERT has been proved to be outstanding in this kind of task and has become a common technical means of text stance detection, but the model still has some problems such as performing poorly under adversarial attacks.

This project mainly focuses on the better application of BERTweet in Twitter text stance detection tasks. Attempts are made to improve the robustness of the model by means of adversarial training and mixup data augmentation, the BERTweet model is trained by SemEval-2016 task 6 dataset under the following conditions: 1. BERTweet model (Original); 2. BERTweet model treated with Adversarial training(adv); 3. BERTweet model treated with both adversarial training and mixup data augmentation (adv-mixup); 4. The adversarial training process is incorporated into each mixup training epochs of Bertweet(AMDA).

The results show that the model performance of 5 datasets does not change much due to simple adversarial attack or mixup, but the robustness of the models is improved to some extent in condition 4(AMDA). The reason may be that the fine-tuned pre-training model itself has good adaptability. When the number of adversarial attack datasets is small, the performance of the model does not improve significantly. However, with the increase of the number of cycles and the continuous enrichment of attack samples, the performance of the model can be improved to some extent.

CONTENTS PAGE

DECLARATION.....	2
ABSTRACT	3
List of Tables.....	错误!未定义书签。
List of Figures	错误!未定义书签。
CHAPTER 1: INTRODUCTION	1
1.1 Background.....	1
1.2 Project Aim and Objectives	3
1.3 Report Structure.....	3
CHAPTER 2: LITERATURE REVIEW	5
2.1 Stance detection.....	5
2.2 AI technology related to stance detection	6
2.2.1 Machine learning and deep learning methods	6
2.2.2 Pre-training model	7
2.2.3 Dataset for Twitter text stance detection	9
2.3 Adversarial training	9
2.3.1 Robustness of the pre-training model.....	9
2.3.2 Adversarial attack	10
2.3.3 NLP oriented defense method	12
2.4 Data Augmentation	13
CHAPTER 3: EXPERIMENTAL DESIGN	16
3.1 Experimental purpose	16
3.2 Design the experimental scheme.....	16
3.3 Procedure of experiment implementation	17
3.4 Analysis of experimental results	18
CHAPTER 4: DESIGN AND IMPLEMENTATION.....	20
4.1 Dataset.....	20
4.2 BERT-Base Model.....	21
4.2.1 Model structure of BERT and its variants	21
4.2.2 Network structure.....	23
4.2.3 Calculation process	23
4.2.4 Training tasks.....	24
4.3 Data preprocessing.....	26
4.4 Implementation of adversarial attack	27
4.5 Implementation of Data Augmentation.....	30
4.6 Evaluation Metrics.....	31
CHAPTER5: RESULTS AND DISCUSSION	32
5.1 Experimental results	32
5.1.2 Adversarial training performance indicators	33
5.1.3 The number of adversarial samples.....	34
5.2 Discussion of experimental results.....	36
CHAPTER 6: CONCLUSIONS AND RECOMMENDATIONS.....	44
6.1 Conclusions	44
6.2 Recommendations.....	45

CHAPTER 7: REFLECTION.....	46
APPENDICES.....	52

List of Tables

Tab. 3.1 Information about the experimental environment.....	18
Tab. 5.1 Four main performance indicators of five models trained by different dataset in clean test sets... ..	32
Tab. 5.2 Four main performance indicators of four models trained by different data set in attacked test sets.....	33
Tab. 5.3 Four main performance indicators of four models trained by Climate Change is a Real Concern in attacked test sets.....	36
Tab. 5.4 Accuracy of four models trained by different datasets in attacked test sets.....	37
Tab.5.5 Classification accuracy of each selected model on the original three datasets and the perturbed datasets using different attacking method.....	37
Tab. 5.6 Accuracy of the various models under PWWS and TextFooler attacks.....	38
Tab.5.7 Accuracy of four models trained by different datasets in attacked test sets...39	
Tab.5.8 Accuracy of four models trained by different datasets in clean test sets.....	39

List of Figures

Figure 3. 1 Code flow chart.....	17
Fig. 4.1 SemEval-2016 task 6 dataset (train set)	19
Fig. 4.2 SemEval-2016 task 6 dataset (test set)	20
Fig. 4.3 BERT Model Structure.....	22
Fig. 4.4 PWWS Algorithm.....	30
Fig 5.1 Four adversarial attack performance indicators are verified based on the adversarial performance verification module provided by Openattack.....	34
Fig.5.2 Label distribution of adversarial samples and training set samples.....	35
Fig.5.3 Bar chart of the number of adversarial samples generated and the total number of training set samples in each round of the cyclic AMDA model training process (cc— Climate Change is a Real Concern)	40
Fig.5.4 Bar chart of the number of adversarial samples generated and the total number of training set samples in each round of the cyclic AMDA model training process (la— Legalization of Abortion)	41
Fig.5.5 Robustness of word vectors.....	42
Fig.5.7 Illustration of MixADA.....	43

CHAPTER 1: INTRODUCTION

1.1 Background

With the rapid development and wide use of information technology, the Internet is becoming an indispensable part of people's life. People work, study, socialize and entertain through the Internet, forming a large amount of real-time information, including valuable information for products, services, policies and news. But extracting effective information from these confusing data is a very challenging task. If these data are directly analyzed manually, it will bring huge time and money pressure to the corresponding enterprises or organizations. Therefore, deep processing of natural language generated by the Internet through AI technology has become a hot topic in the field of artificial intelligence.

At present, the application of artificial intelligence in the field of natural language processing has been involved in machine translation, text classification, sentiment analysis, intelligent customer service and other fields. Stance detection is the task of automatically determining from text whether the author of the text is in favor of, against, or neutral towards a proposition or target[1], is widely used in the classification and identification of online text data, such as election or referendum result prediction, information retrieval, rumor detection, fake news detection, etc.

Twitter has always been one of the most popular microblogging social platforms, with a large number of users, a wide range of topics, and a steady stream of large and rich Twitter data, which has proved to be a widely used and real-time information source for stance detection tasks[2]. By mining these massive text data generated by Twitter, it can provide valuable support or reference for public policy making, public opinion testing, business decision-making, etc. Therefore, the stance detection of Twitter text has become an important source of natural language processing (NLP) tasks. In addition, the research on Twitter text stance detection can also provide reference for similar research on other social platforms.

The present AI pre-training model can be well applied to all kinds of NLP tasks. The pre-training model BERT adopts the two-stage method of "pre-training and fine-tuning", which improves the deep learning model's understanding of text semantics to a milestone height. Therefore, BERT and its variant models have become the main technical means for the current text stance detection task[3]. RoBERTa(Robustly Optimized BERT Pretraining Approach) [4] is developed based on BERT model, and has made significant improvements in many aspects, so that the performance is more superior. BERTweet is the first public large-scale language pre-training model for English tweets, which uses the configuration of the BERTbase model and is trained based on the RoBERTa pre-training process[5]. It is helpful to solve the inadaptability of the pre-trained language model on the large-scale traditional text corpus in the text analysis of tweets. These models provide a powerful tool for Twitter text stance detection.

Although the AI technology related to text stance detection has made great progress in recent years, the system threat caused by the defects of AI technology cannot be ignored. Researchers have found that most deep learning models vulnerable to well-designed input samples called adversarial examples[6, 7], That is, models run the risk of being artificially misled into making wrong judgments, which are maliciously designed by adding small perturbations to benign inputs that are imperceptible to the human eye, leading to false predictions by the target model[8]. The instability of model performance brings some risks to stakeholders. In addition, a large pre-training model requires a large amount of data for training. When the amount of data is too small, it is easy to overfit, which will affect the generalization of the model[9].

In deep learning, robustness is an important indicator to evaluate model quality. Robustness refers to the robustness of the model to the input data. A robust model can accurately predict the results even under the circumstances of noise, missing data or other anomalies.

Adversarial training is generally regarded as a training method that can effectively improve the robustness of the model. Its basic principle is to obtain adversarial samples by adding adversarial disturbance to the original input, and then train the adversarial samples to improve the performance of the model. Since natural language text is

discrete, adversarial perturbations are generally added to the embedding layer. In order to maximize the disturbance ability of the adduction sample, the adduction sample is generated by means of gradient rise. In order to avoid excessive disturbance, the gradient is normalized.

In addition, the quality, magnitude, and diversity of data also affect the performance of AI systems. When the number of data samples is small, the trained model is very easy to fall into overfitting. In the field of deep learning, Data augmentation is a commonly used technology. It can generate more training data through a series of transformations on the original data, so as to improve the generalization ability of the model.

Therefore, we can try to further improve the robustness and generalization of the model by means of Adversarial training and Data augmentation, so as to obtain a Twitter text stance detection model with better comprehensive performance for practical needs[10].

1.2 Project Aim and Objectives

In this project, 5 datasets of Twitter SemEval-2016 task 6[1]were selected to train the BERTweet model. In order to study the influence of Adversarial training and Data augmentation techniques on the robustness and generalization ability of the model, the Adversarial training was further implemented on the model, and the mixup method was used to implement Data augmentation.

By comparing and analyzing the performance indexes of the original BERTweet model, the model undergoing Adversarial training, the model undergoing Adversarial training and mixup processing at the same time, the way to further improve the model performance is discussed.

1.3 Report Structure

Structure of the project report:

Chapter 1: INTRODUCTION. Firstly, the research background of the project and the importance and necessity of Twitter text stance detection are introduced. Then, in order to solve the common problems of deep learning model, an idea is proposed to improve the robustness and generalization of the model through Adversarial training and Data

augmentation technology. On this basis, the research content of this project is determined. Finally, the structure of the full text is introduced.

Chapter 2: LITERATURE REVIEW. This chapter mainly introduces the research progress and research status in all aspects related to the project research. The content includes the research progress and current situation of text stance detection, the selection of natural language processing methods and relevant models, the research of Adversarial training and Data augmentation techniques, and the selection of experimental datasets.

Chapter 3: METHODOLOGY AND IMPLEMENTATION. This chapter introduces the research ideas, methods and steps of this project as a whole, as well as the considerations in various aspects of selecting research methods and determining research steps. At the same time, the specific implementation of the experiment is introduced in detail, the construction of four different models and the selection of related parameters, as well as the treatment of experimental results.

Chapter 4: RESULTS AND DISCUSSION. The experimental results of the four models mentioned above are introduced in detail, and the comparative experimental analysis is carried out. The results of the experiment were visualized with charts and graphs.

Chapter 5: CONCLUSIONS AND RECOMMENDATIONS: This chapter summarizes the experimental results of stance detection, gives reasonable suggestions for model optimization, evaluates the results of Adversarial training and Data augmentation, and puts forward prospects for future research.

CHAPTER 6: REFLECTION: Reflect on the implementation of the work of the project, point out the problems in the implementation of the project, and put forward suggestions for the development of similar work in the future.

CHAPTER 2: LITERATURE REVIEW

Text stance detection by AI technology is an important branch of text classification in natural language processing (NLP). At present, the model used for specific text stance detection task is mainly established by training the dataset meeting certain conditions through AI pre-training model. The quality of the text stance detection model is determined by the selection of AI pre-training model, the selection of relevant parameters, the size and quality of dataset and the way of preprocessing. In order to further improve the robustness and generalization ability of the model, it may also be necessary to conduct Adversarial training on the model, or use Data augmentation technology to enhance the data. This chapter mainly focuses on the literature review of the above contents.

2.1 Stance detection

With the rapid development and continuous popularization of Internet technology, the discussion text of social media users on public events and consumer products has increased exponentially. This new source of information could be used to measure people's opinions, stances, and attitudes toward products, events, services, controversial news, and politics. These measurements can play a valuable role in decision-making for companies, policymakers, politicians, and even regular people[11]. Text stance detection is an important and common text classification task in natural language processing. It refers to the automatic judgment of the user's stance on the preset target based on the text published by the user through natural language processing (NLP) technology. In this case, the target mainly includes people, organizations, policies, activities or products.

Early studies on textual stance detection mostly focus on English online forums and political debates. In 2006, Thomas et al. [12] collected the records of speeches and debates in the US Congress in 2005 and marked the support and opposition stance. Subsequently, research on text stance detection for different objectives appeared one after another.

In recent years, the study of stance detection has become a hot topic, with applications in many different fields, such as business intelligence, public opinion analysis, political elections, medical decision-making, and mental health. Its research results can be extended to the research fields of dialogue system, personalized recommendation, rumor verification, social perception, etc., which has important academic research and application value[13].

2. 2 AI technology related to stance detection

The Transformer model is a type of neural network developed by Google. First described in a 2017 paper by Google[14], the Transformer model is one of the newest and most powerful model categories invented to date. They're driving a wave of advances in machine learning that some are calling transformer AI. transformers created using large datasets can make accurate predictions that drive their wider use, generating more data that can be used to create better models. "Self-supervised learning has been made possible by Transformers, and artificial intelligence has skyrocketed," said NVIDIA founder and CEO in his keynote speech at the GTC. This part mainly introduces a series of pre-training models based on Transformer underlying architecture.

2.2.1 Machine learning and deep learning methods

For stance detection task, there are two main types of methods in previous studies: machine learning method based on feature engineering and deep learning method based on neural network. Feature engineering methods rely on artificial feature construction, usually combined with machine learning for stance detection. This method requires manual selection of text features, takes information containing stance features as input, and uses classification model for training and classification.

The method based on deep learning deals with stance detection by constructing neural network structure and introducing various mechanisms. Commonly used neural networks include: Convolutional Neural Networks (CNN), Recurrent Neural networks (CNN), convolutional neural networks (Recurrent Neural Network, RNN, Long-Short Term Memory Networks, etc.[15]. Deep learning aims to extract features from target tasks through deep neural network construction. Such as language text, graphics and other unstructured data, the quality of feature construction often determines the

performance of the model. Traditional machine learning methods need to spend a lot of time and energy on manually selecting features. Deep learning can directly input the initial text features obtained through the word embedding technology into the neural network, and automatically train the corresponding features according to the target constraints by relying on the mutual transmission of information between network nodes. This method can process a large amount of data and shift the focus of research from feature screening to fine-grained semantic feature representation.

2.2.2 Pre-training model

Pre training technology provides an effective solution for large-scale parameter learning in deep neural networks. Its core idea is to pre train deep neural networks on large datasets to obtain model parameters, and then apply these trained models to various specific downstream tasks to avoid training from scratch and reduce the need for labeled data, The results show that the performance of the model has been significantly improved[16].

In essence, the pre-training technology uses the idea of transfer learning[17] to transfer the prior language knowledge learned in the relevant domain dataset to the target domain, so as to improve the learning effect of different target tasks.

In modern natural language processing tasks, the distributed representation of words[18] obtained by unsupervised training with large-scale corpus Widely used, including Word2Vec (Word to Vector) [19] As a typical representative, the method realizes the representation of words with dense vectors in low dimensional conditions, which reduces the computational cost and improves the representation accuracy.

However, this method only carries out a single global representation of words and extracts superficial textual representations, but ignores their context, so it cannot effectively represent the syntactic and semantic features of words in different contexts. ELMo (Embeddings from Language Model)[20] implements context-based word embeddings for the language model using the bidirectional Long - and Short-Term Memory (LSTM) network, and significantly improves the model's performance in downstream tasks.

Later, Google introduces Attention mechanism in Transformer model[14]. Based on Attention, target words and source text are unified and Self-Attention structure is proposed. Moreover, by combining the superimposed Self-Attention structure with the Multi-Head Attention mechanism, it achieves the simultaneous acquisition of context information, solves the long-term dependence problem, and possesses the key capability of parallel computing, which proves to a certain extent that increasing the size of model parameters can improve the model effect. This architecture is therefore widely used by subsequent pre-training models.

Transformer solves the long-term dependency problem and implements parallel computing while being able to obtain more semantic information. Transformer injects powerful power into the underlying architecture of the pre-training model, elevating the text feature extraction to a new level, and also points out a new way for the development of the future pre-training model. Between BERT (Bidirectional Encoder Representations from Transformers) [21] and GPT (Generative Pre-Training)[22] The appearance of the deep pretraining model laid an important foundation, and since then, the NLP mission is increasingly inseparable from the pretraining technology.

In 2018, Google Corporation introduced the BERT (Bidirectional Encoder representation from Transformers) model based on transformer model [8]. It consists of multiple bidirectional encoder layers stacked to form a fully connected neural network structure. It adopts the two-stage idea of "pre-training + fine-tuning" and covers the information of speech class, syntax and location when embedding. It can learn the general expression of the language through bidirectional pre-training on the unmarked text, which can capture the contextual information in a real sense. By fine-tuning these general representations to specific downstream tasks, such as text classification, named entity recognition, sentence contrast, etc., the deep learning model's ability to understand text semantics has reached a milestone.

RoBERTa(Robustly Optimized BERT Pretraining Approach)[23] is an improved version based on BERT, which is a significant improvement of BERT on multiple levels. Compared with BERT model, RoBERTa has a larger number of model parameters and more training data in terms of model scale, computational power and data, and has many improvements in training methods.

BERTweet model is the first public large-scale language pre-training model for English tweets[5]. This model uses BERTbase model configuration and is trained based on RoBERTa pre-training process, so the inadaptability of the pre-trained language model on the large-scale traditional text corpus to the text style of Twitter language is solved.

2.2.3 Dataset for Twitter text stance detection

SemEval 2016 English Twitter stance dataset is a widely used benchmark dataset for target stance detection. It is the International Workshop on Semantic Evaluation (semEval), a dataset used for public evaluation tasks[24]. This dataset is an English stance detection dataset constructed by Mohammad et al. based on English Twitter. Task 6 contains two sub-tasks: Sub-task A is a supervised classification task, which uses labeled training data to detect users' stance on five given targets; Subtask B is a weakly supervised categorization task that uses large amounts of unlabeled training data to detect the Twitter text's stance on the "Donald Trump" target.

FNC—I Dataset . In 2017, Dean et al.[25] proposed to take text stance detection as the first subtask of fake news detection task (fake-news-challenge I, FNC—I). The data used in this task was extracted from the dataset published by literature []. It is divided into four categories: support, opposition, need to discuss or have nothing to do with the title, which belongs to the stance detection at the text level.

At present, apart from the datasets published in the above evaluation, there are also some datasets established and published by scholars themselves.

2. 3 Adversarial training

2.3.1 Robustness of the pre-training model

Because of its linearity, deep neural networks are prone to attack by adversarial inputs[26]. Although pre-training models perform well on different NLP tasks, most of them adopt the structure of deep neural network, so there is a problem of poor robustness. Adversarial attack is widely used in Computer Vision as a means to test the robustness of the model. But text is different from images, in that a single word can change the meaning of an entire sentence. The research on text adversarial attack started

slowly. Jin et al.[27] attacked many existing models with adversarial text and found that their performance was sharply reduced. Therefore, improving the robustness of pre-training model is an urgent problem to be solved in the future.

The Adversarial training is a key means to improve the robustness of the pre-training model, which can be trained by using the adversarial samples. Goodfellow[28] and Madry et al.[29] conducted research on text Adversarial training methods. Zhu et al.[30] also proposed an Adversarial training algorithm, FreeLB (Free Large Batch), on the basis of previous studies. Robustness is one of the important issues facing the current pre-training model, and more methods will appear to enhance the robustness of the model in the future[31].

2.3.2 Adversarial attack

With the development of the research on adversary attack, a series of studies have divided the papers of NLP's attack into four granular attacks: character-level attack, word-level attack, sentence-level attack and mixed-level attack. In addition to granularity, attackers 'mastery of target information divides text attacks into three categories: white, gray, and black. Operator selection for fine-grained text attacks can be divided into gradient-based attacks and importance-based attacks.

(1) Character level attack

The character level attack mainly adds disturbances to the characters in the original input, and the common methods include character deletion, modification, addition, replacement, etc. the quality of the counter text generated by the character level disturbance is poor, which destroys its syntax[32], so this kind of attack method is less applied.

In the text classification task, Gao et al.[33] proposed an algorithm Deepwordbug, which can generate hostile text sequences and mislead the deep learning model by taking advantage of the security vulnerabilities in the NLP model. This algorithm can effectively generate micro text perturbations in black box settings, thus forcing the NLP model to misclassify text input. The author also designed a new scoring strategy for the Deepwordbug algorithm, to find the most important words to modify, so that the NLP

model makes wrong predictions. The Deepwordbug algorithm applies simple character level conversion to the highest ranked words to minimize the editing distance of disturbance and change the original classification. The results show that Deepwordbug reduces the prediction accuracy of the current latest NLP model, and reduces the performance of the model by generating adversarial sequence text.

While the success rate of character-level attacks is high, it destroys the grammar and the structure of the original input. However, such attacks are easily defended[34].

(2) Word level attack

Unlike character level attacks, word level attacks perturb words in the original input. The main method is word replacement, including word replacement methods based on word vector similarity synonyms, semaphores, language model scores. Word level attacks have better performance in resisting text quality and effectiveness control and attack success rate. In addition, using language models for control, the smoothness and fluency of the generated adversarial text can also be guaranteed.

Among the word level attack methods, typical white box attacks such as the shared word replacement attack and Textfooler attack need to obtain the detailed information of the target model; While common black box attacks, such as word substitution attacks based on semantic primitives, use the mobility of the text to attack. Compared to character- and sentence-level textual adversarial attacks, word-level attack can generate higher-quality adversarial examples, especially in a black-box setting. Danish Pruthi et al.[34] propose a novel attack method, the main idea of which is to fully utilize the adversarial examples generated by the local model and transfer part of the attack to the local model to complete ahead of time, thereby reducing costs related to attacking the target model.

(3) Sentence level attack

The sentence level attack regards the original input of the whole sentence as the object of disturbance, aiming to generate a counter text with the same semantics as the original input (at least the real label of the current task does not change), but makes the threatened model judge the change. Common sentence level attack methods include re decoding after encoding, adding irrelevant sentences, paraphrasing, etc. the following will introduce the sentence level attack methods.

Wang et al.[35] proposed a controlled adversarial text generation model, which generates adversarial text through controllable attributes unrelated to known task tags given the input text. Compared with many existing adversarial text generation methods, this method can generate more diverse and smooth adversarial text, and it has strong robustness to model retraining and different model architectures. Therefore, the generated combat text can be used for combat training to improve the model, but how the model implicitly extracts attributes needs to be explored. Therefore, Li et al.[36] proposed a more efficient combat text generation method, BERT attack, by using a pre trained mask language model to produce high-quality combat text, the author contrasts BERT with the model finely adjusted in the downstream task and other DNNs, so as to mislead the prediction results of the target model. This method is superior to the most advanced attack strategy in terms of success rate and interference percentage. At the same time, the generated combat text is smooth and semantically unchanged, but in terms of interference rate, attack success rate There is still much room for improvement in grammatical correctness and semantic consistency.

2.3.3 NLP oriented defense method

The basic purpose of generating adversarial texts for NLP models is to use these adversarial texts to enhance the robustness of the models, so as to prevent malicious adversarial texts from interfering with model predictions. This section mainly introduces text defense methods in the NLP field, which are divided into Adversarial training, privacy protection defense and detection defense. Because of the endless emergence of attack methods, the defense work against texts has become more important and difficult, at present, the defense research around NLP still needs further exploration.

(1) Defense based on Adversarial training

Adversarial training is an important way to enhance the robustness of neural networks. In the process of Adversarial training, some adversarial texts with small perturbations will be mixed in the training samples, and then the neural network will adapt to such perturbations, so as to be robust to adversarial texts. In the NLP field, Miyato et al.[37] introduced adversarial and virtual Adversarial training into text classification, and by

adding regular terms to the model, so that the output distribution of a sample is the same as the output distribution after perturbation. This method has a good defense effect in text classification tasks, and can use a large number of unlabeled samples to regularize the model through Adversarial training to improve the effect. In addition, in order to overcome the distribution difference problem, the framework obtains Adversarial sentences by adding some small perturbations to each word in the sentence and generates a character level word representation for the words in the sentence.

(2) Defense based on adversarial detection

In addition to enhancing the robustness of the training process and the privacy security of text data through adversary training and privacy protection technology, the defense against NLP models can also achieve resistance to model attacks caused by synonyms or spelling errors by detecting adversary texts.

Gao et al.[38] used Python automatic correction toolkit v0.3.0 to detect malicious input. Python automatic correction tool improved the prediction accuracy of the model from an average of 13.34% to an average of 74.17%, Belinkov et al.[39] used Google's spell checker to correct spelling errors caused by hostile methods. Google's spell checker improved the bilingual evaluation understudy (Bleu) score of the French corpus from 16.7 to 21.4, and the Bleu score of the German corpus from 18.6 to 25.0, Li et al.[40] used context aware spell-checking operations to achieve similar effects. However, the experimental results show that this detection method is effective for character level modifications, but partially effective for word level attacks, which may be caused by differences in modification strategies, Spell checking methods are also not applicable to adversarial texts based on other languages.

2.4 Data Augmentation

As a large deep learning model, pre-training model often requires a large amount of data input into the model for training to prevent model overfitting. As the tweet dataset needs to focus on the topic, in most cases, there may be a lack of sufficient samples for the training of tweet text classification. Therefore, Data augmentation is a relatively effective optimization method for the tweet text classification task.

Data augmentation was initially used in computer vision, and then natural language processing was introduced. Data augmentation is mainly used to improve the diversity of training data to help the model obtain better generalization. According to the diversity of the enhanced data, Data augmentation of text samples can be divided into three categories: Data augmentation based on interpretation, Data augmentation based on noise and Data augmentation based on sampling.

A few of the more mainstream approaches deserve attention. The most convenient method is undoubtedly Easy data Augmentation Techniques, developed by Jason Wei et al.[41] in 2019. This is a Data augmentation based on four simple text operations such as Synonym replacement, Random Insertion, Random Swap, and Random Deletion to prevent overfitting and improve text generalization ability. However, such Data augmentation may destroy the syntactic structure and smoothness of the original text to some extent.

The pre-training model and its fine tuning can also be considered as a Data augmentation method, among which the most classic is the fine-tuning BERT model by randomly masking some words in the marked sentences, and then being filled by conditional bert. The large-scale language model can also be used as an enhancement tool in some cases because of the rich language knowledge acquired through pre-training.

Mixup is a less common approach to Data augmentation. It was first used in computer vision to reduce the possibility of overfitting by mixing two different images into a model after mathematical calculation. Then Guo et al.[42] were inspired by this and proposed the Mixup method for natural language processing. Studies have shown that Mixup's approach is simple but effective for Data augmentation.

To solve the problem that the number of adversarial examples for text enhancement is still very insufficient due to the exponential growth of the attack search space, Si et al.[10] propose a simple and effective method to cover a much larger proportion of the attack search space, called Adversarial and Mixup Data Augmentation (AMDA). AMDA linearly interpolates the representations of pairs of training samples to form new virtual samples, which are more abundant and diverse than the discrete text

adversarial examples in conventional ADA. In text classification experiments of BERT and RoBERTa, AMDA achieves significant robustness gains under two strong adversarial attacks and alleviates the performance degradation of ADA on the clean data.

CHAPTER 3: EXPERIMENTAL DESIGN

3.1 Experimental purpose

This project focuses on the issue of Twitter text stance detection, aiming to establish a model that can better realize the task of Twitter text detection through AI technology, and try to further optimize the performance of the model.

3.2 Design the experimental scheme

For the above purposes, the main work involved in the experiment is as follows:

1. Selection of AI natural language processing model.

According to the existing literature analysis, deep learning technology provides a powerful tool for natural language processing, and transformers models often perform well in various text classification tasks. BERTweet is trained based on RoBERTa's pre-training process in transformers models series. Is a public large-scale language pre-training model specially used for English tweets, with superior performance and adapted to the characteristics of tweets, so it is the preferred model for this project. RoBERTa is a general-purpose Transformer models without domain-specific training, show how vanilla Transformer-based models perform on the stance detection task.

2. Selection of model test dataset.

SemEval 2016 is a data set used by the International Workshop on Semantic Evaluation (semEval) for public evaluation tasks, and it is also a data set commonly used in stance detection evaluation. For this project, SemEval 2016 task 6 are selected as the data sets for the experiment.

3. According to existing studies, the robustness of deep learning models is often not ideal. It is easy to be affected by malicious attacks, resulting in misclassification of samples. According to the relevant literature, Adversarial training is an effective method to improve the robustness of the model. In order to further optimize the performance of the model, the approach to improve the robustness of the model is discussed through Adversarial training.

4. In view of the small size of tweet dataset and limited samples, Mixup Data augmentation technology is selected to explore ways to improve model generalization ability.

5. In view of the good performance of Adversarial and Mixup Data Augmentation (AMDA) methods reported in the literature[10], this paper draws on the practice of the literature and uses AMDA as a working condition to verify the effectiveness of this method in this project, and explore various possibilities to maximize the robustness of the model.

Based on the above situation, determine the four conditions of the experiment. The model is trained under the following working conditions:

1. BERTweet model (Original);
2. Adversarial training(adv);
3. adversarial training and mixup data augmentation (adv-mixup);
4. The adversarial training process is incorporated into each mixup training epochs of Bertweet (AMDA).

3.3 Procedure of experiment implementation

1. Write the program code according to the experimental purpose, including input, execution, output, etc., with the python program JupyterNoteBook, the code flow is shown in Figure 1, and the code see Appendix 1 for details.

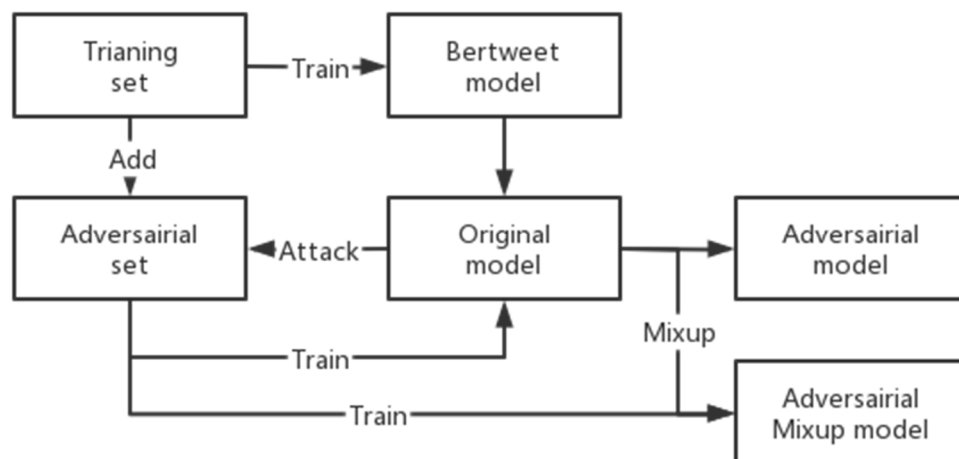


Figure 3. 1 Code flow chart

2. Build hardware and software conditions and run related programs.

Tab. 3.1 Information about the experimental environment

Environment	Equipment disposition
Hardware environment	Amazon Sagemaker Studio Lab: CPU: AMD x86_64 GPU: Tesla T4 16G Supercomputing Center, Fuzhou University: NVIDIA Titan-V-GPU, 16GB
Software environment	Amazon Sagemaker Studio Lab: Operating system: Linux-4.14.313-235.533.amzn2.x86_64-x86_64-with-glibc2.31 CUDA: 11.10.3.66 Cudnn: 8.5.0.96 Supercomputing Center, Fuzhou University (for AMDA): Slurm scheduling system, vasp python IDE: Jupyter Notebook Develop frameworks and libraries: pytorch、huggingface、numpy、pandas、nltk、Openattack

3. Model hyperparameters

epochs = 5

n_gpu = 0

train_batch_size = 16

weight_decay = 0.0

learning_rate = 3e-5

adam_epsilon = 1e-8

warmup_steps = 0

num_labels = 3

mix_layers_set = [7,9,12]

alpha = 8.0

eval_batch_size = 64

4. Output experimental results.

3.4 Analysis of experimental results

1. Compare the experimental results under several different working conditions, and analyze the robustness performance of the model under different working conditions by comparing the relevant parameters of the experimental results.

2. The performance of model generalization ability under different working conditions was analyzed by comparing experimental results with relevant parameters.
3. Through comprehensive evaluation and analysis, determine the optimal specific scheme of Twitter text stance detection model construction, and further put forward suggestions to optimize the model performance.

CHAPTER 4: DESIGN AND IMPLEMENTATION

According to the experimental scheme, the following five specific aspects related to the experiment will be introduced in detail, including: dataset, stance detection model, adversarial attack method, Data augmentation method, as well as the specific situation of the experimental results evaluation and analysis method.

4.1 Dataset

There are currently six major publicly available tweet datasets on the web. They are WT-WT, SE16, M-T, RE17, RE19, Encryption. Among them, SE16 can be regarded as the most classic dataset for stance detection. It attracted a large number of researchers into the field of stance detection after its publication in 2016. In addition, in terms of the coverage area of the dataset, SE16 is the largest coverage area of the six datasets. In terms of the total amount of data, there are 4,162 samples, which is a relatively small amount of data. Overall, SE16 is a fairly standard multi-topic tweet stance detection dataset.

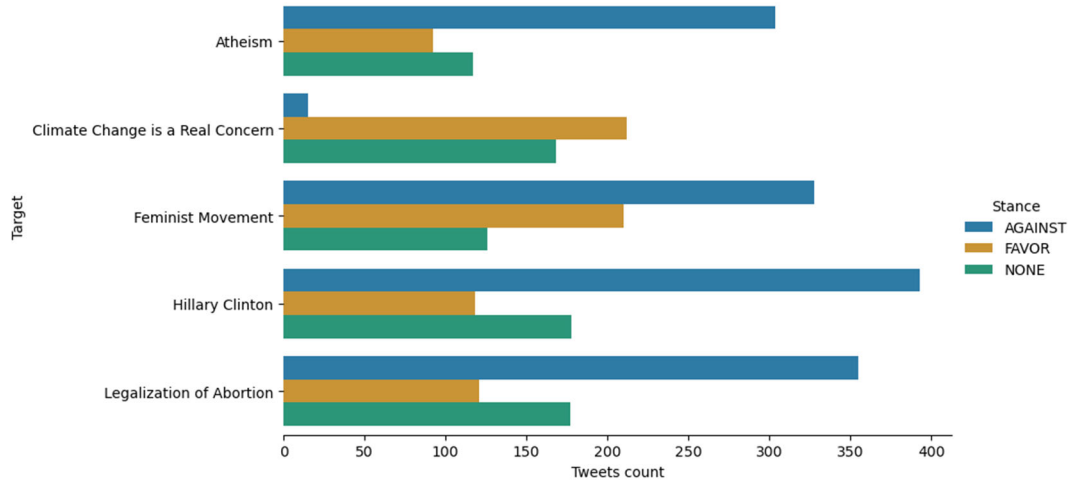


Fig. 4.1 SemEval-2016 task 6 dataset (5 train sets)

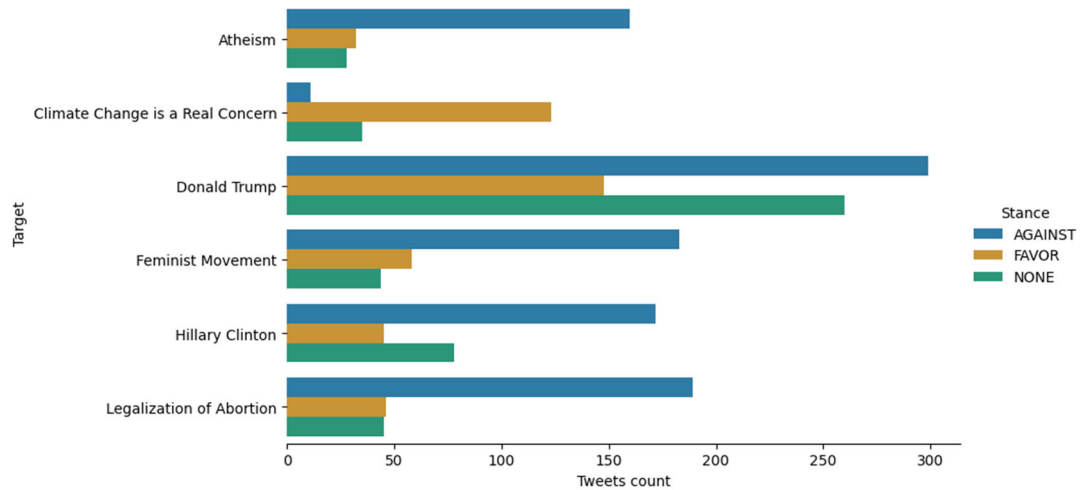


Fig. 4.2 SemEval-2016 task 6 dataset (6 test sets)

There was one Donald Trump question in the test set. This topic was for task b of unsupervised training and had no effect on task a. In general, the text distribution of the five topics in Task a was similar to that in the training set.

4.2 BERT-Base Model

4.2.1 Model structure of BERT and its variants

BERT is a pre-training language proposed by Google[3]. The proposed method has greatly promoted the progress of NLP technology. At the same time, it has gradually formed a new language processing process: pre-training using large-scale and high-quality corpus of multi-human languages, building a language representation model suitable for a variety of NLP tasks, and fine-tuning specific tasks on this basis to achieve wide application. The structure of BERT model is shown in Figure 2.

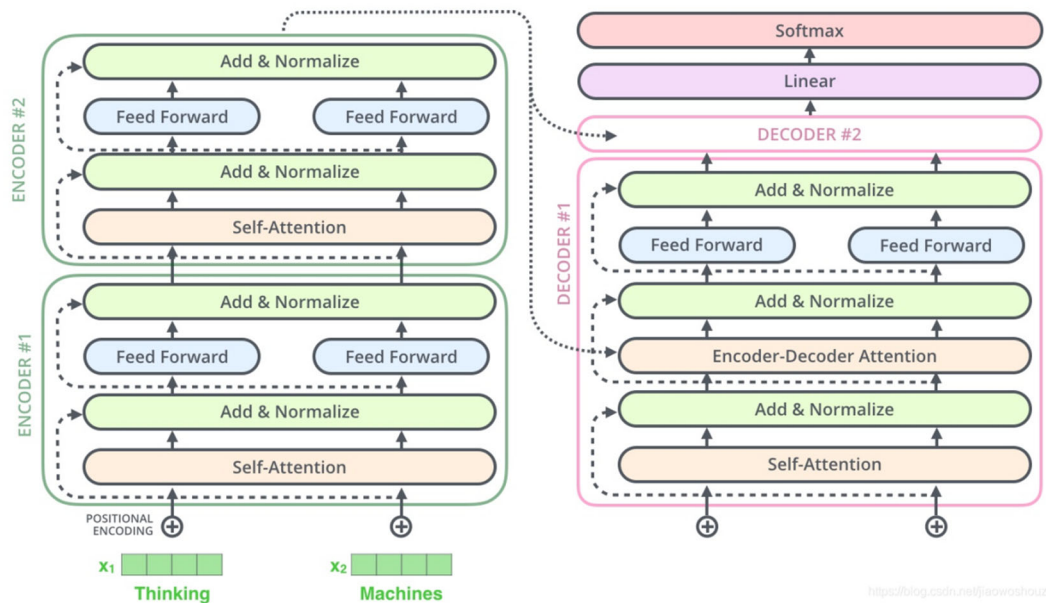


Fig. 4.3 BERT Model Structure[3]

Since the BERT model is built based on transformer, its model input is also different from other models. In BERT, the input vectors are summed by three different embedding methods, which are:

1. Wordpiece embedding: vector representation of the word itself. Wordpiece refers to dividing a word into a limited group of common sub word units, which achieves a good balance between the effectiveness of the word and the flexibility of the character.

2. Position embedding: encode the word position information into a feature vector. Due to the lack of time-series neural networks such as RNN or LSTM in the BERT structure, the position information of the sequence cannot be obtained, so a position embedding needs to be built.

3. Segment embedding: vector representation used to distinguish two sentences.

In order to make the BERT model flexibly adapt to various downstream tasks related to text analysis, it is necessary to generate a continuous and fixed length text stream (starting with the classification mark [CLS] and separated by [SEP] at the end of the sentence) by splicing a single sentence in the document, and randomly mask the tokens in the sentence as input in a certain proportion.

4.2.2 Network structure

Although BERT is based on transformer, it only uses the encoder part of transformer, and its overall framework is stacked by the encoder of multi-layer transformer. The function of the encoder is to obtain the semantic features of the text, such as the context features and the syntactic dependencies, which is attributed to the fact that it is composed of N multi head self-attention layers and feedforward neural network layers. For the input $x = \{X_1, X_2, \dots, X_i\}$ (sentence sequence, X_i represents the i th word), the output of each sub layer in the encoder is 1. By summing the sub layers adjacent to the normalized connection, the output word embedding vector list is transferred to the next layer.

In this encoder, the main function of each attention is to re encode the target word through the correlation between the target word and all the words in the sentence. Therefore, the calculation of each attention includes three steps: calculating the degree of correlation between words, normalizing the degree of correlation, and obtaining the code of the target word by weighted summation of the degree of correlation and the codes of all words.

The self-attention layer is created to solve the problem that network structures such as RNN and LSTM, which are often used to process serialized data, cannot speed up parallel computation in GPU. In addition, the self-attention model can also solve the problem of long-distance dependence in the traditional RNN model. Through attention, the distance between two words that are far away can be reduced to 1 to directly calculate the relevance of words.

4.2.3 Calculation process

For each token encoding, the semantic information of the received sequence x is divided into the following steps:

1. Perform matrix transformation on X to obtain Q, K, V
2. Calculate self-attention function.
3. Project Q, K, V into H subspace through independent linear transformation matrix $W_1, \dots, W_h$
4. The feature vector Z is generated by stitching h different attention results.

The formula is expressed as follows:

$$Z(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^0$$

Where, q is the output of the upper sub attention layer, and K and V are the attention vector sets output by the encoder.

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Feedforward neural network layer: to enhance the expression ability of transformer, calculate the nonlinear hierarchical characteristics:

$$\text{FN}(Z) = \max(0, ZW_1 + b_1)W_2 + b_2$$

Through this layer, the output vector Z in the formula is changed nonlinearly and the vector length remains unchanged.

Summation and normalization layer: calculate the standardized output of each layer of data:

$$\mu = \frac{1}{d} \sum_{i=1}^d x_i, \sigma = \sqrt{\frac{1}{d} \sum_{i=1}^d (x_i - \mu)^2 + \varepsilon}$$

$$x_i' = f\left(g \frac{x_i - \mu}{\sigma} + b\right)$$

Where, f is a nonlinear function, g and b are model parameters. The short cut idea in the residual network is adopted to add the input and output x_i and output x_i' of each layer to realize the effective superposition of the network structure and make the result smoother.

4.2.4 Training tasks

1.Masked language model

Some words are masked randomly, and then the word is predicted by context. In BERT, 15% of wordpiece tokens will be randomly masked, 80% of these 15% tokens will be

replaced by [mask], 10% will be replaced by a random word, and 10% will remain the same. This design makes the model have the ability to capture the context relationship, and can be conducive to token level tasks such as sequence annotation.

2.next sentence prediction

For 50% of the sentences in the corpus, select the corresponding next sentence to form the upper and lower sentences together as the positive sample; The remaining 50% of the sentences randomly select a sentence other than the next sentence to form the upper and lower sentences as the negative sample. This setting is beneficial to sense level tasks.

3.Fine-tuning test stage

The BERT series model includes a general language representation model and a process of fine tuning the model for various NLP tasks. In the last stage, based on the multilingual universal language representation model obtained by pre-training, the word vector of each word in the text fused with the context semantics can be dynamically learned. In the fine-tuning trial stage, it is necessary to train on the new corpus according to the specific NLP task, and refine the network structure (update the parameters of the model) to obtain the final model.

Taking a text two classification problem as an example, we represent the input text as a sequence of [CLS] + (masked) tokens after preprocessing. First, each word (including [CLS]) is output as a word vector with a length of 768 through the pre-training model, and a full connection layer is added outside the model. After dot multiplication of the embedding matrix and the output vector, it is converted into the dimension of the word, and then the probability of classifying the sentence sequence is calculated through the SoftMax layer, Thus, a sentence text classifier is constructed.

The fine-tuning of multi classification problems is similar to the above method. The SoftMax function is used to judge the probability of each category, and the 768-dimensional word vector is output as an n-dimensional real vector ($n = \text{Ili}$ is the number

of classification labels, and the sum of all elements in the vector is 1). The calculation formula is as follows:

$$P_i = \frac{e^{V \cdot C_i}}{\sum_{j=1}^{|L|} e^{V \cdot C_j}}$$

Where, V is the weight matrix (including the pre-training model parameters and the new task specific parameters, and the weights are adjusted according to the training data), C is the [CLS] output of the multi label classification, and the inner product represents the scores under different classifications. Use SoftMax to calculate the conditional probability P_L , and select the category with the highest probability as the classification label.

BERT model can be said to be the groundbreaking model of pre-trained transformers model. The latter several improved models are mostly based on BERT model. One of the models modified on the basis of bert model to improve the TWEET sample data is BERTweet model, which is built on the basis of Bert-base and gives the model better robustness with the pre-training way of Roberta model.

The pre-trained text base in BERTweet ontology is a text base formed after processing the collected Twitter text. In this paper, Cardiff University uses five lexical databases specifically for training on five topics of Semal-2016.

4.3 Data preprocessing

In natural language processing, the processing of original text samples is a very important step. In the early text classification based on statement text feature engineering, text preprocessing largely depended on the professional knowledge of the researcher in the language field. The result of the experiment directly depends on whether the features constructed by the linguistic knowledge are well differentiated.

With the wide application of deep learning, sentence feature extraction can be accomplished by the model to a large extent. Therefore, how to convert text data into digital data that can be directly used by the model has become the main research method. Eventually, for most pre-trained models from one-hot to embedding, most text data preprocessing can be solved through the tokenizer, a highly integrated module.

The Tokenizer is designed to break sentences into small chunks and then generate a thesaurus. The size of the glossary and the length of tokens are the most critical factors, which usually determine the learning ability and computing time of the model. So the

tokenizer method of word segmentation is the most important thing to mention when discussing tokenizer.

There are four common word segmentation methods: Byte-Pair Encoding, WordPiece, Unigram, SentencePiece. wordpiece is a subword segmentation algorithm used in BERT model, which is very similar to BPE algorithm. WordPiece first initializes the vocabulary to contain each character in the training data and gradually learns a given number of merge rules. In contrast to BPE, WordPiece does not select the most frequent symbol pair, but rather a symbol pair that maximizes the possibility of training data once it is added to the vocabulary.

However, these methods are difficult to use directly for Tweet text, because most of the initialization vocabulary contains training data that is often standardized text information, whereas the text of a tweet contains a lot of non-standard text content. So BERTweet's tokenizer module includes a standardized function. In addition, BERTweet's tokenizer module also uses emoji packs to translate emotional ICONS into text strings, and normalizes tweets by converting user mentions and web/url links into special tokens @USER and HTTPURL, respectively. The white box standardization of text required by bert model to train tweet text is simplified.

4.4 Implementation of adversarial attack

Because the deep neural network mentioned above is vulnerable to attack whether it is applied to image processing or natural language processing. Therefore, in order to effectively prevent the deep neural network from failure in the case of relatively simple attacks. Adversarial defense has become a hot research direction of researchers. Compared with the field of image processing, the Adversarial defense means in natural language processing can often improve the robustness of NLP models more effectively, that is, to some extent, improve the performance of models in test sets. Therefore, in recent NLP competition tasks, adversarial defense has become a major means to obtain high scores. Adversarial defense methods mainly include Adversarial training, privacy protection defense and detection defense. Adversarial training is considered as one of the most effective defense methods among these methods. The basic idea of this method is to use the adversarial text obtained by the attack method and the normal text to mix them, and then use them as a new training dataset to fine tune the target model. Basic idea of Adversarial training min max formula:

$$\min_{\theta} E_{(x,y) \sim D} [\max_{r_{adv} \in S} L(\theta, x + r_{adv}, y)]$$

That is, it is necessary to find a set of counter samples in the sample space within a certain disturbance range that maximizes loss. This counter sample is composed of the original sample and the disturbance term. And the function indicates that in the face of the sample set with such data distribution, the expected loss of the model on the counter sample set should be minimized by updating the model parameters[43].

As far as the current Adversarial training means are concerned. Most of them obtain new false text samples by applying limited perturbation to the original text samples, which are provided to the discriminator for recognition. There are generally two mainstream methods to add restrictions to the disturbance. One is that the disturbance added to the text cannot be recognized by people to a certain extent, and the other is to replace the content of the original text sample through Adversarial word conversion and similar methods to form new synonymous sentences or text approximate sentences.

Although the gradient-based Adversarial training method is very convenient and effective to improve the generalization and robustness of the text model, it is not very resistant to design attacks. Especially with the rapid development of the means of adversarial attack, many of the relatively effective methods of adversarial defense have lost their defensive ability. Therefore, in order to solve this situation, researchers will introduce adversarial defense to the adversarial attack, which can be used to train the model to resist the means of adversarial attack.

The core of this method is to generate adversarial samples whose text format is reasonable enough to resemble the meaning of the original training samples. The most effective of these methods is adversarial word replacement. Researchers have divided two directions on how to apply resistance samples to training models. One is to improve the model to improve its defense against a specific attack, the other is to add resistance samples to the training set and turn them into a means of Data augmentation.

It can be seen from the above that the data distribution in task a of the dataset is rather extreme, and even for 'climate change is a real concern', it has almost no objection samples. In other words, Data augmentation may be a better way to enhance the generalization of SemEval 2016 task A's dataset.

However, ADA has two main limitations for NLP model. First of all, NLP is difficult to create effective enhanced text samples because of the discreteness of text data and the different characteristics that its samples are more easily recognized by people than images. In addition, the search space is too large for text adversarial attacks. Therefore, it is relatively difficult to generate enough adversarial samples.

On the other hand, generating more reasonable adversarial samples that can be used as training set samples to input model training needs to be similar to training set samples in semantics and syntax. That is, the adversarial sample generator needs to train the semantics and grammar of the original samples more effectively. And adversarial attack is often a good choice for generators. According to the above, adversarial attacks are generally divided into character level attacks, word level attacks, sentence level attacks and mixed level attacks. At present, adversarial word substitution (AWS) is one of the most effective methods to attack high-level neural models. That is, AWS method is the adversarial sample generator that can best fit the semantics and grammar of the original sample.

The most classic of these methods is probability weighted word saliency (PWWS). The PWWS method mainly focuses on two aspects, one is word substitution and the other is the importance of the corresponding word in the sentence. The main strategy of word replacement is to obtain the replacement word from the synonym table which is most likely to make the model judgment wrong. The strategy of obtaining the replacement object word is based on the significance of the word in the target sentence. After prioritizing the words according to importance, you iterate through replacing the synonyms of each word until you change the final category label.

Algorithm 1 PWWS Algorithm

Input: Sample text $\mathbf{x}^{(0)}$ before iteration;
Input: Length of sample text $\mathbf{x}^{(0)}$: $n = |\mathbf{x}^{(0)}|$;
Input: Classifier F ;
Output: Adversarial example $\mathbf{x}^{(i)}$

- 1: **for all** $i = 1$ to n **do**
- 2: Compute word saliency $S(\mathbf{x}^{(0)}, w_i)$
- 3: Get a synonym set $\mathbb{L}_i$ for w_i
- 4: **if** w_i is an NE **then** $\mathbb{L}_i = \mathbb{L}_i \cup \{\text{NE}_{adv}\}$
- 5: **end if**
- 6: **if** $\mathbb{L}_i = \emptyset$ **then** continue
- 7: **end if**
- 8: $w_i^* = R(w_i, \mathbb{L}_i)$;
- 9: **end for**
- 10: Reorder w_i such that
- 11: $H(\mathbf{x}, \mathbf{x}_1^*, w_1) > \dots > H(\mathbf{x}, \mathbf{x}_n^*, w_n)$
- 12: **for all** $i = 1$ to n **do**
- 13: Replace w_i in $\mathbf{x}^{(i-1)}$ with w_i^* to craft $\mathbf{x}^{(i)}$
- 14: **if** $F(\mathbf{x}^{(i)}) \neq F(\mathbf{x}^{(0)})$ **then** break
- 15: **end if**
- 16: **end for**

Fig. 4.4 PWWS Algorithm[44]

4.5 Implementation of Data Augmentation

Data augmentation is a commonly used method to expand data when training samples are insufficient. The methods of Data augmentation include EDA, text generation, Adversarial generation, etc. The pre-training language model adopted in this paper can be regarded as Data augmentation for a specific language to some extent. Ada-style Adversarial training also has Data augmentation effects. Therefore, in terms of Data augmentation means, if the Data augmentation method of applying disturbance to the data is adopted, its effect may be weakened to some extent due to the influence of the above circumstances.

On the other hand, it can be seen from the characteristics of the dataset that the text dataset of semal-2016 is very unbalanced. There is a large difference in the data amount of three viewpoints in the dataset of each topic, so the characteristics of such dataset should be taken into account when selecting the Data augmentation method.

According to the analysis from Bohan Li et al.[45], compared with other Data augmentation methods, mixup produces more diverse and fluid data than interpretative and noise-based methods. Mixup is the only one of all learnable methods that implements Data augmentation during model training. Thus, mixup can output cross-labels and discrete emplacements from enhanced data. Based on the above characteristics, this paper considers mixup as a Data augmentation method.

4.6 Evaluation Metrics

In the classification task, the most commonly used four indicators to evaluate the performance of the model : Accuracy, Precision, Recall and F1-score.

The four evaluation indicators are calculated by the following four basic parameters of the classification task: True Positive (TP) , False Positive (FP) , True Negative (TN) , False Negative (FN) .

Accuracy is the proportion of all the samples predicted correctly to the total number of samples:

$$Accuracy = \frac{\text{Correct classifications}}{\text{All classifications}} = \frac{TP + TN}{(TP + TN + FP + FN)}$$

Precision is the percentage of the predicted positive samples that are actually positive:

$$Precision = \frac{TP}{TP + FP}$$

Recall indicates how many positive samples in the sample have been predicted correctly:

$$Recall = \frac{TP}{TP + FN}$$

F1-score is a composite index based on Recall and Precision: (β is usually 1)

$$F = \frac{(\beta^2 + 1) * Precision * Recall}{\beta^2 * (Precision + Recall)} (\beta \geq 0)$$

In the case of multi-label classification, for Precision and Recall, each class needs to calculate its own Precision and Recall separately, which also means that the overall Precision and Recall need a certain method for comprehensive calculation.

At present, there are three common methods: 1. Macro-average method 2. Weighted-average method and 3. Micro-average method. Due to the serious imbalance in the data set, the Weighted-average method is chosen in this project.

CHAPTER5: RESULTS AND DISCUSSION

5.1 Experimental results

The experimental results of models trained in a clean test set are shown in Tab. 5.1, The experimental results of models trained in an attacked test sets are shown in Tab. 5.2.

The first column in the table is the abbreviation symbol of the data set, corresponding to the data set name as follows:

cc— Climate Change is a Real Concern; la— Legalization of Abortion

hc— Hillary Clinton; aa—Atheism sets; fm— Feminist Movement

The first row in the table corresponds to four different working conditions in the experimental scheme.

Four performance indicators of each model were output based on the experimental results: Accuracy, Precision, Recall and F1-score. For specific meanings, see Chapter 4.6 of this paper.

Tab. 5.1 Four main performance indicators of five models trained by different dataset in clean test sets

data set	indicator	Original	adv	adv-mix	ADAM
cc-clean data	accuracy	0.85	0.85	0.85	0.82
	f1	0.82	0.82	0.81	0.78
	precision	0.80	0.80	0.79	0.76
	recall	0.85	0.85	0.85	0.82
la-clean data	accuracy	0.66	0.66	0.66	0.7
	f1	0.61	0.61	0.61	0.62
	precision	0.56	0.56	0.56	0.57
	recall	0.66	0.66	0.66	0.7
hc-clean data	accuracy	0.69	0.69	0.69	0.71
	f1	0.65	0.65	0.65	0.65
	precision	0.73	0.73	0.73	0.6
	recall	0.69	0.69	0.69	0.71
aa-clean data	accuracy	0.77	0.77	0.77	0.75
	f1	0.77	0.77	0.77	0.69
	precision	0.78	0.78	0.78	0.64
	recall	0.77	0.77	0.77	0.75
fm-clean data	accuracy	0.56	0.56	0.56	
	f1	0.58	0.58	0.58	
	precision	0.67	0.67	0.67	
	recall	0.56	0.56	0.56	

Tab. 5.2 Four main performance indicators of four models trained by different data set in attacked test sets

data set	indicator	Original	adv	adv-mix	ADAM
cc-attacked data	accuracy	0.65	0.65	0.84	0.73
	f1	0.66	0.66	0.81	0.71
	precision	0.71	0.71	0.79	0.68
	recall	0.65	0.65	0.84	0.73
la-attacked data	accuracy	0.60	0.60	0.60	0.65
	f1	0.57	0.57	0.57	0.57
	precision	0.55	0.55	0.55	0.55
	recall	0.60	0.60	0.60	0.65
hc-attacked data	accuracy	0.62	0.62	0.62	0.67
	f1	0.59	0.59	0.59	0.62
	precision	0.67	0.67	0.67	0.57
	recall	0.62	0.62	0.62	0.67
aa-attacked data	accuracy	0.63	0.63	0.63	0.72
	f1	0.64	0.64	0.64	0.65
	precision	0.68	0.68	0.68	0.61
	recall	0.63	0.63	0.63	0.72
fm-attacked data	accuracy	0.46	0.46	0.46	
	f1	0.47	0.47	0.47	
	precision	0.54	0.54	0.54	
	recall	0.46	0.46	0.46	

5.1.2 Adversarial training performance indicators

Figure 5 shows the adversarial data index of the model trained under three different working conditions. The three working conditions refer to the first three working conditions in the experimental scheme.

The four counter performance indexes are:

Successful Instances: The number of samples in which adversarial attacks make the model's judgment wrong

Attack Success Rate: The number of samples that the model failed due to the judgment attack accounted for the total number of samples that the model successfully predicted

Avg. Levenshtein Edit Distance: The average distance between the changed word vector and the replaced word vector that the adversarial attack sample successfully misleads the model.

Avg. Word Modif. Rate: The average percentage of modifiable words per sentence.

As can be seen from Figure 5.1, there are obvious differences in the accuracy of each data set. However, the attack success rate is generally less than the probability of the

model prediction error. It shows that the effect of adversarial attack is not significant under several experimental conditions.

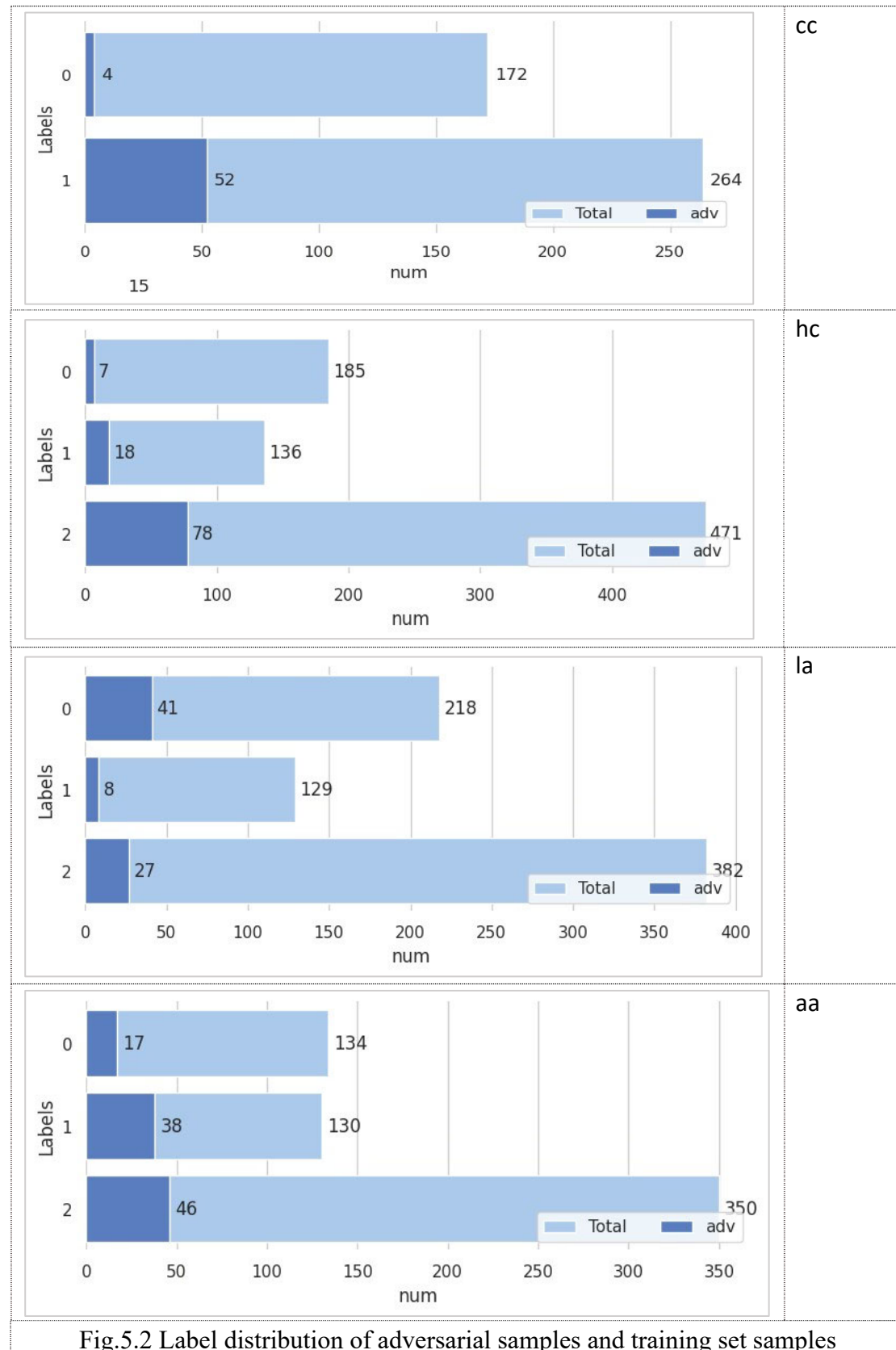


Fig 5.1 Four adversarial attack performance indicators are verified based on the adversarial performance verification module provided by Openattack

5.1.3 The number of adversarial samples

Figure 5.2 shows the bar chart of the number and total number of Adversarial training samples generated by the original model after adversarial attacks. The bar chart, with labels as Y-axis and number of samples as X-axis, shows the balance of the data set

when the samples are generated by Adversarial training. This chart is directly generated by Jupyter notebook.



5.2 Discussion of experimental results

As can be seen from the above data presentation, in most cases, Adversarial training has no effect on the performance of the model in the pure test set, which is a phenomenon that is highly inconsistent with the expectation based on the literature review above. In other words, the adversarial samples generated by adversarial attacks did not lead to the deviation of the data tendency of the samples to some extent.

According to the results of adversarial attack attacking training sets (Fig. 5.1), adversarial samples did not make the model immune to adversarial attack effectively. The adversarial defense based on PWWS is basically not effective in this model. However, according to the adversarial attack indicator square graph in the results (Fig. 5.1(a,b)), adversarial attacks can indeed produce adversarial samples and cause the model to make incorrect judgments in predicting the correct samples.

The model training process was first checked for errors. The adversarial defense in this article is based on the standard adversarial defense example from Openattack. If an error occurs during the training process, all three models of each data set should have the same output. However, there are some changes in the adv-mixup model of the Climate Change is a Real Concern data set. Therefore, the adversarial attack is applied to the test set of Climate Change is a Real Concern, and the existing three models are used to predict it. The results are shown in Table 5.3.

Tab. 5.3 Four main performance indicators of four models trained by “Climate Change is a Real Concern” in attacked test sets

	Original-B	ADV-B	ADV-MIX-B
accuracy	0.65	0.65	0.84
f1	0.66	0.66	0.81
precision	0.71	0.71	0.79
recall	0.65	0.65	0.84

It can be seen that the adv-mix combinatorial optimization model obtained a good accuracy score in the test set mixed with the adversarial samples. It can even be said

that the adv-mix model is almost immune to adversarial attacks. The accuracy of the AD mix model fell by only 1%, compared with a fifth for each of the first two models. This is a data that doesn't look normal. The reasons for this need to be further explored.

Tab. 5.4 Accuracy of four models trained by different datasets in attacked test sets

	Original-B	ADV-B	ADV-MIX-B	ADAM
la adv	0.60	0.60	0.60	0.65
hc adv	0.62	0.62	0.62	0.67
cc adv	0.65	0.65	0.84	0.73
aa adv	0.63	0.63	0.63	0.72

Compared with the adversarial analysis of the training set, more studies tend to focus on the adversarial attack of the test set as an important indicator to evaluate the robustness of a model. It is not difficult to find that the model has a large performance degradation when predicting the test sets after adversarial attack. However, if these declines are compared with the performance declines of several models subjected to adversarial attacks in the PWWS article, it can be known that the accuracy of the model in this paper is limited when facing the adversarial attack samples of PWWS. From the PWWS examples (Tab.5.5), each model basically has an accuracy attenuation of more than 40%, while the model in this paper (Tab.5.2) does not exceed 40% most of the time. It can be seen that the model itself has a certain degree of robustness. There are three possible sources of this robustness: 1. Excellent pre-training models; 2. Complex textual data; and 3. Small amount of text data. This means that, to a certain extent, Adversarial training may not improve the robustness of the model.

Tab.5.5 Classification accuracy of each selected model on the original three datasets and the perturbed datasets using different attacking methods.[44]

Dataset	Model	Original	Random	Gradient	TiWO	WS	PWWS
IMDB	word-CNN	86.55%	45.36%	37.43%	10.00%	9.64%	5.50%
	Bi-dir LSTM	84.86%	37.79%	14.57%	3.57%	3.93%	2.00%
AG's News	char-CNN	89.70%	67.80%	72.14%	58.50%	62.45%	6.30%
	word-CNN	90.56%	74.13%	73.63%	60.70%	59.70%	56.72%
Yahoo!	LSTM	92.00%	74.50%	73.80%	62.50%	62.50%	3.00%
Answers	word-CNN	96.01%	82.09%	80.10%	69.15%	66.67%	57.71%

By combining the four indicators of adversarial attack in Figure 5.1, it can be found that the success rate of adversarial attack in each data set is basically consistent with the predicted error rate of data set. In other words, the success rate of adversarial sample attacks generated by adversarial attacks on the training set is lower than the failure rate of model prediction, which is not enough to produce a sufficient number of samples to change the model's prediction tendency.

That is, adversarial attacks lack the ability to generate successful samples on these models. Therefore, a hypothesis can be proposed that due to the complexity of tweet text and the small number of words in a single text sample, PWWS adversative attack is difficult to find highly significant words in its text that can stabilize the misleading model.

Then, combined with the relevant data of similar models in the existing literature, further analysis is carried out. Chenglei Si et al. [37] proposed an adversarial defense method based on the combination of Adversarial training and Mixup in 2021, namely Adversarial and Mixup Data Augmentation (ADAM). This approach greatly optimizes the robustness of the model without losing predictive performance on the clean test set. In addition, the paper also provides some data related to the performance of some adversarial attacks and Mixup.

Tab. 5.6 Accuracy of the various models under PWWS and TextFooler attacks.

	SST-2				IMDB			
	PWWS		TextFooler		PWWS		TextFooler	
	Original	Adversarial	Original	Adversarial	Original	Adversarial	Original	Adversarial
BERT	91.27	14.83 (20.88%)	91.27	2.97 (16.21%)	97.75	24.18 (24.10%)	97.75	1.64 (10.18%)
+ADA	90.12	27.18 (24.46%)	90.50	9.01 (18.32%)	96.93	25.82 (34.53%)	96.93	3.07 (11.81%)
+TMix	91.82	21.20 (19.36%)	91.82	3.51 (16.39%)	97.13	43.24 (32.51%)	97.13	0.00 (12.06%)
+SMix	91.82	22.52 (20.47%)	91.82	4.61 (16.76%)	97.13	31.97 (23.74%)	97.13	2.66 (12.39%)
+AMDA-TMix	91.54	38.82 (23.73%)	91.93	<u>13.23</u> (19.66%)	97.34	<u>51.02</u> (36.76%)	96.72	<u>4.51</u> (17.23%)
+AMDA-SMix	91.10	<u>31.52</u> (24.11%)	92.15	17.35 (18.64%)	96.72	60.86 (27.79%)	96.72	17.42 (13.85%)
RoBERTa	94.62	28.39 (23.06%)	94.62	5.44 (18.51%)	97.54	28.07 (37.48%)	97.54	6.35 (12.61%)
+ADA	94.07	25.26 (27.07%)	92.75	9.67 (19.71%)	97.54	24.80 (49.36%)	96.93	12.50 (14.39%)
+TMix	94.18	30.04 (23.19%)	94.18	11.04 (17.69%)	97.54	44.06 (39.33%)	97.54	21.11 (14.01%)
+SMix	93.96	31.52 (22.86%)	93.96	8.29 (17.80%)	97.34	41.39 (34.90%)	97.34	22.34 (11.96%)
+AMDA-TMix	93.90	<u>36.74</u> (26.02%)	93.03	<u>13.78</u> (20.15%)	98.57	<u>50.41</u> (59.68%)	97.13	51.84 (16.62%)
+AMDA-SMix	93.96	41.85 (27.17%)	93.47	16.80 (21.88%)	97.54	55.12 (45.30%)	97.54	<u>49.18</u> (15.52%)

Tab. 5.6 shows that the Mixup method itself can improve the robustness of the model to a certain extent. The combination of AMDA and Mixup can reduce the performance

loss of the model when predicting the clean test set, and greatly improve the robustness of the model.

So, in this paper, the AMDA method is introduced as a supplementary method, and the model accuracy data are as follows Tab.5.6 and Tab.5.7

Tab.5.7 Accuracy of four models trained by different datasets in attacked test sets

	Original-B	ADV-B	ADV-MIX-B	ADAM
cc-attacked data	0.65	0.65	0.84	0.73
la-attacked data	0.60	0.60	0.60	0.65
hc-attacked data	0.62	0.62	0.62	0.67
aa-attacked data	0.63	0.63	0.63	0.72
fm-clean data	0.46	0.46	0.46	

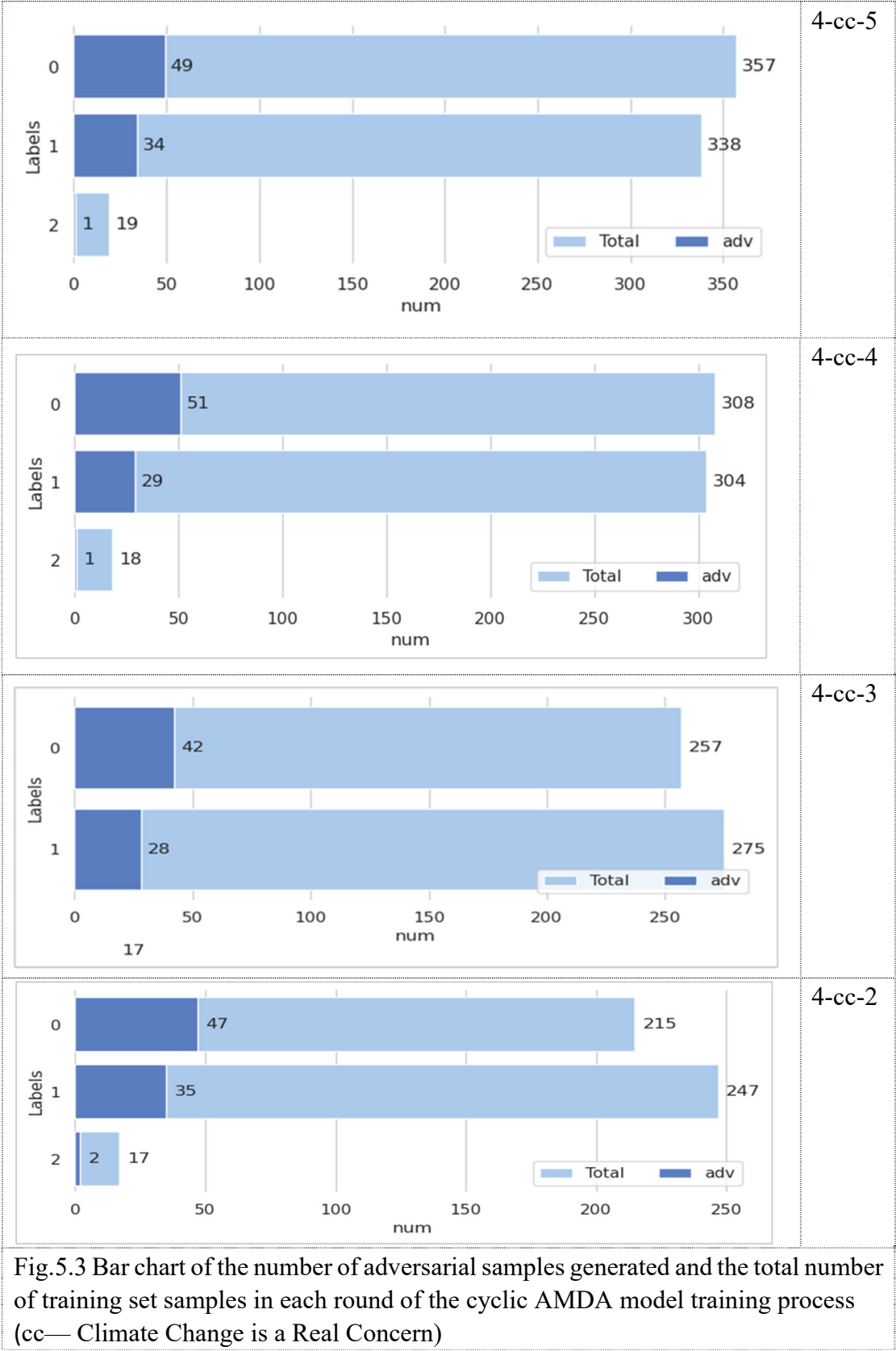
Tab.5.8 Accuracy of four models trained by different datasets in clean test sets

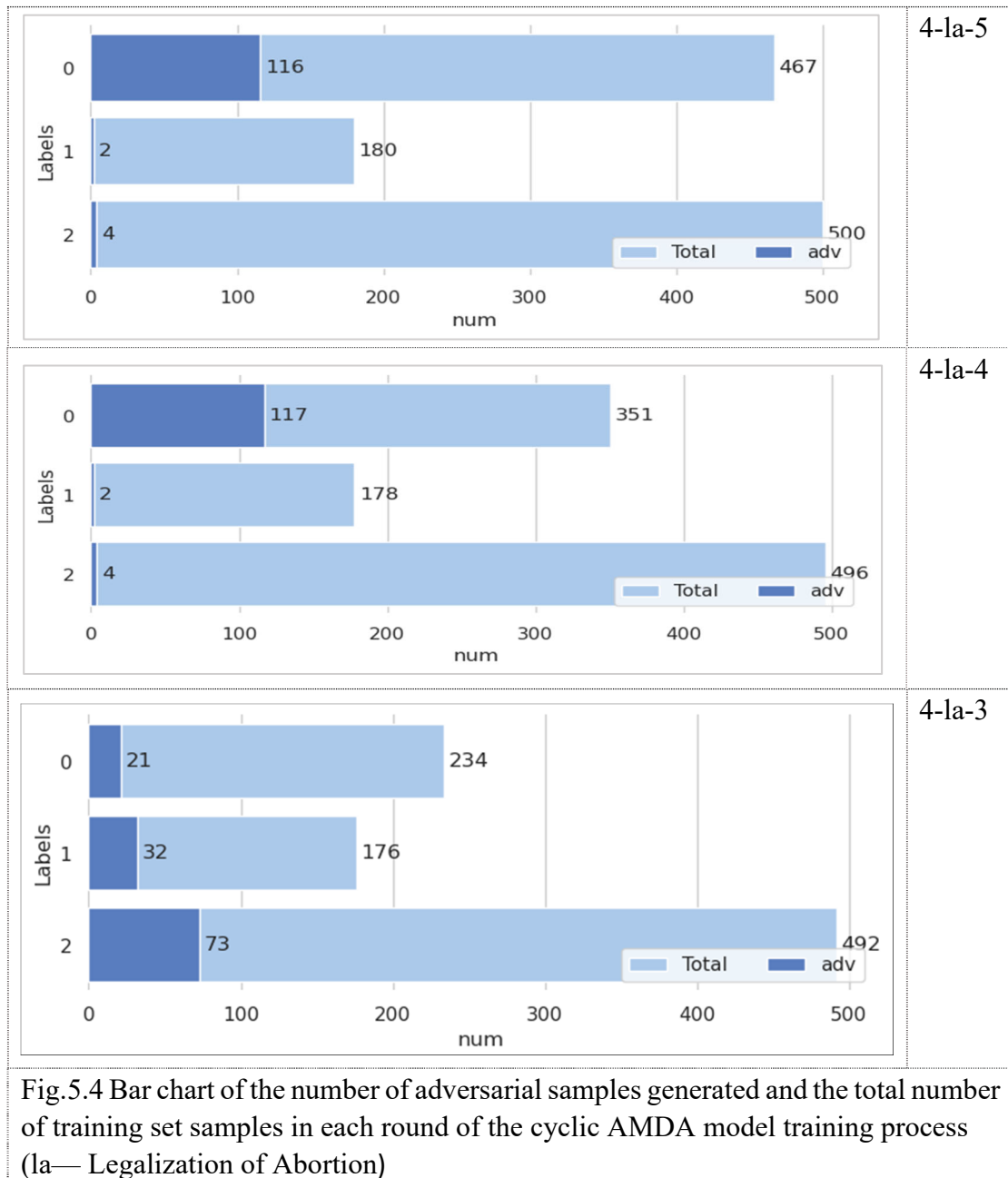
	Original-B	ADV-B	ADV-MIX-B	ADAM
cc-clean data	0.85	0.85	0.85	0.82
la-clean data	0.66	0.66	0.66	0.7
hc-clean data	0.69	0.69	0.69	0.71
aa-clean data	0.77	0.77	0.77	0.75
fm-clean data	0.56	0.56	0.56	

Compared with the prediction performance of the three existing models on the clean test set, it can be seen that the prediction accuracy of the AMDA model on the clean test set of the cc data set and the aa data set decreases, and the prediction accuracy on the other data sets increases. After introducing the adversarial attack samples into the test set, it can be seen that the performance of the AMDA model is better than the previous three models in most data sets. So on the model of AMDA, Adversarial training plays a role in the tweet text.

In order to analyze why the AMDA model can provide effective robustness optimization for tweet text. In this paper, the adversarial sample generation process in the loop process of a part of the AMDA model is taken form Jupyter notebook. (Fig.5.4, Fig. 5.6). The graph is the number of adversarial samples output by each training epochs of the iterative AMDA model and the overall number of training samples. It can be seen that each training epochs, the adversarial attack module will output a certain amount of adversarial samples. This process is equivalent to

expanding the adversarial defense sample set of the model. The regional breadth of adversarial attacks is expanded.





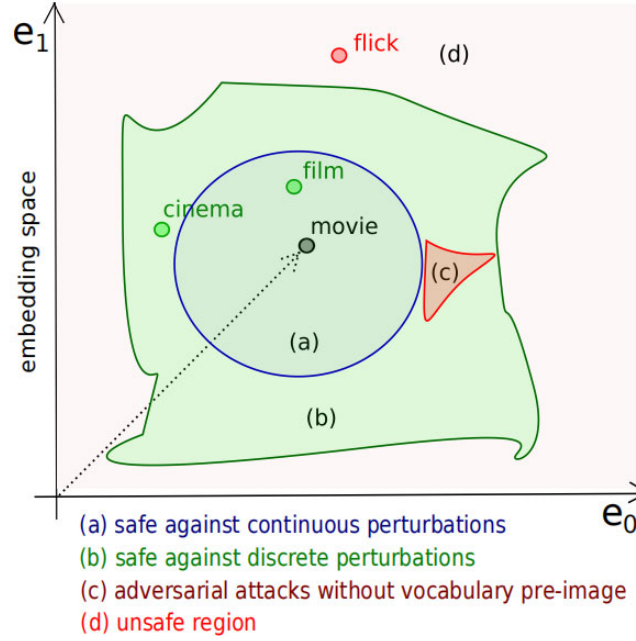


Fig.5.5 Robustness of word vectors [46]

Emanuele La Malfa et al.[46] published an explanatory study on robustness to text data in 2022. One explanation for the robustness of word vectors is proposed. That is, the robustness of a model to a single word vector is a range on a vector space. The successful adversarial attack samples are in the discrete robust region which is not covered by the existing dictionary. The adversarial defense approach is to introduce such replacement words into the lexicon of the model in order to fill the gaps in the discrete robust region as much as possible. Therefore, the failure of the first three models to effectively provide adversarial defense may be due to the fact that the adversarial attack does not attack the effective attack area, but the adversarial samples generated by the natural judgment error of the model when the model judges the sentence again. However, the ADMA model successfully attacked some areas by expanding the attack range.

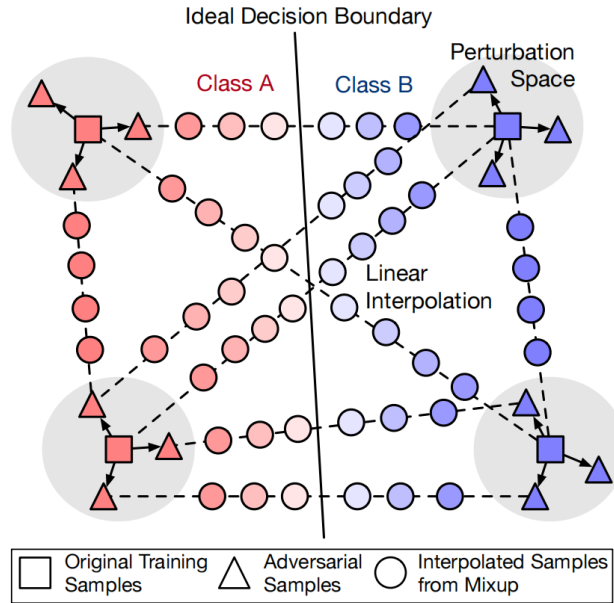


Fig.5.7 Illustration of MixADA[10]

On the other hand, the mixup method fills in the connections between adversarial samples and ordinary samples, so that the model has a decision boundary that is replaced by a robust one. This also expands the scope of adversarial attacks to a certain extent, so that adversarial attacks can attack more vulnerable areas of the model.

CHAPTER 6: CONCLUSIONS AND RECOMMENDATIONS

In this project, BERTweet is applied to the stance detection task in Twitter. 5 datasets of Twitter SemEval-2016 task 6 were selected to train the BERTweet model. In order to study the influence of Adversarial training and Data augmentation techniques on the performance of the model, the Adversarial training was further implemented on the model, and the mixup method was used to implement Data augmentation.

By comparing and analyzing the performance indexes of the original BERTweet model, the Adversarial training model, the Adversarial training & mixup model and the AMDA model, the way to further improve the model performance is discussed. This project aims to explore ways to improve model performance in Twitter stance detection through experiments, and give reasonable suggestions for model optimization.

6.1 Conclusions

1. In the training of the BERTweet model under 4 working conditions and 5 data sets, the accuracy range of the model is between 0.46 and 0.85, which varies according to the specific data sets.
2. Based on the experimental results of this project, in most cases, it seems that, in the three working conditions (Adversarial training, mixup, Adversarial training & mixup) almost has no effect on the performance of the model of the pure test set, which is inconsistent with the expectation based on the literature review above.

By combining the four main indicators of adversarial attacks of each model, it can be found that the success rate of adversarial attacks on the training set generated by adversarial attacks is lower than the failure rate predicted by the model, which is not enough to generate a sufficient number of samples to change the prediction tendency of the model.

Through comparison and analysis with relevant works in other existing literatures, it is found that degree of accuracy decline in the model is less than 40% in most cases, which is far lower than the value in relevant literatures. It is likely that the model itself already

has some robustness. There are three possible sources of this robustness: 1. Excellent pre-job training mode; 2. Complex text data; And 3. A small amount of text data. This means that, to a certain extent, Adversarial training may not improve the robustness of the model.

3. The results shows that the model in the fourth condition, That is, ADMA can improve the robustness of the model to a certain extent.

After comparative analysis of the experimental results, it is found that in each training period, the counter attack module will output a certain number of counter samples. This procedure is equivalent to the adversarial defense sample set of the extended model. Expanded the area range of confrontational attacks. With the increase of the number of cycles and the enrichment of attack samples, the performance of the model tends to be further improved.

6.2 Recommendations

This project conducted a preliminary literature survey and experimental research on the role of BERTweet model in the task of Twitter text stance detection. Due to the limitations of conditions, SemEval 2016 task6 Dataset was used to study the model instead of the real data set. Although the model's performance in all aspects was discussed comprehensively, it was not deep enough. Due to limitations such as the number of model samples, the experimental results in Adversarial training are not significant, and there is a certain gap between them and the expectation. The underlying reasons need to be further studied.

In addition, the data obtained from the experiment, there are individual abnormal points. However, due to time constraints, it is not possible to conduct repeated experiments on this abnormal performance, and the specific reasons remain to be further explored.

CHAPTER 7: REFLECTION

Finally, my Master's Project (ACFA7003) has been completed. Looking back on the whole process I experienced in the completion of the project, I have many feelings. Now I reflect on the following:

1. About topic selection: Follow your interests. Although I graduated from mechanical engineering, I am very interested in computer engineering and AI. Through the early learning, I found that the Transformers series models of AI deep learning are very powerful. My hometown is in the countryside of Fujian, China, which is known far and wide as the "land of fruit". As a whole, life is not easy for villagers there. rural technology is still weak. They cannot predict the market situation and user demand in time, so in the fruit selection, planting and sales are often passive. I believe that AI can change the world, but also can change the lives of villagers, so that they work twice the result with half the effort.

Therefore, I would like to use this opportunity to practice a useful project, that is, to extract useful information about crop demand and sales through AI stance detection technology and mass data from public social media platforms, so as to guide farmers in agricultural production projects. However, due to my first contact with this kind of large-scale AI model and my lack of experience in Chinese text stance detection data processing, it is difficult to realize it within the specified time of the project. Therefore, I finally set my goal to use AI pre-training model and open dataset to detect the stance of English Twitter text. I hope that the implementation of this project will accumulate experience and lay a certain technical and practical foundation for my later realization of my own ideas.

2. About the difficulties encountered in the project development: Interest is the best teacher. To be honest, the difficulties encountered during the project were beyond my imagination. Because of the COVID-19 pandemic, I can't go to the campus to study, I can't study with teachers and classmates, and I can't use the various resources of the university to carry out my projects. Although the teachers were very concerned about

me, it was the first time for me to undertake such a large project, and I was not familiar with related models, hardware, etc., so I met great challenges. Among all the challenges, the first thing that has sustained me so far is interest. I deeply understand the meaning of the saying "interest is the best teacher". To learn a command, I often stay up late at night reading instructions, browsing professional websites, or trial and error until I finally solve it. In order to get hardware support, I kept contacting relevant teachers, consulting various websites, seeking help from nearby universities' research institutes, supercomputing centers, etc., until my code finally ran smoothly. If it wasn't for interest, I don't think I would have gotten to where I am today.

3. About experience: To solve the real project needs comprehensive ability, to learn to overcome all difficulties. I think I got a lot out of this project. On the one hand, it has exercised my comprehensive ability to manage a project. I soon found that what I had to manage was not only technology, although technology itself was already facing a lot of challenges, I also attached great importance to progress, because with the progress of the project, I found that the reminders given in the handbook of the project were actually God-like predictions! "The project can be a deceptive experience. As the submission deadline is so far away when you start, there is a danger of wasting time in the early stages of the process." "Always allow as much time as possible for the final stages. Final amendments, organizing, testing, debugging, word processing and doing a thorough job on the proof reading all take longer than you imagine. Something will always go wrong to thwart your schedule!" And so it was! In addition to technology and progress, I will learn to actively strive for resources in all aspects, solve the problem of distance learning, solve the inconvenient problem of epidemic containment, solve the problem of literature database search, solve the GPU problem of code running, solve the problem of frequent network disconnection..... The most important thing is to actively seek solutions and seek help when encountering difficulties. Therefore, this project made me realize that comprehensive ability is needed to complete a practical task within a specified time. Only when we solve all the difficult problems one by one can we complete it.

4. As for the lesson-there should always be a plan for possible future risks. As mentioned above, I have been busy solving all kinds of problems throughout the progress of the project, and every time I expect this to be the last one, but it is impossible.

Later, I realized that not being able to actively foresee possible difficulties is a kind of passive behavior. Only by mastering systematic thinking, strengthening the planning of work, learning to foresee difficulties, preparing in advance, and responding to emergencies at any time can we remain invincible. This is a lesson that I will continue to study hard next.

5. Reflection on the technical problems of the project. One is the insufficient estimation of the running time of the model, and the other is that the running results of the model are not consistent with the results reported in the literature, especially the Adversarial training is not so effective in improving the performance of the model, because of the time problem, the specific reasons cannot be found in a short time, and further research needs to be carried out outside this project.

In short, I benefited a lot from the opportunity to complete such a project. I would like to take this opportunity to thank and pay tribute to all the teachers who have given me great help.

References:

- [1]. Mohammad, S., et al. Semeval-2016 task 6: Detecting stance in tweets. in Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016). 2016.
- [2]. Abdelrazeq, A., et al., Sentiment analysis of social media for evaluating universities. Automation, Communication and Cybernetics in Science and Engineering 2015/2016, 2016: p. 233-251.
- [3]. Devlin, J., et al., Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
- [4]. Liu, Y., et al., Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.
- [5]. Nguyen, D.Q., T. Vu and A.T. Nguyen, BERTweet: A pre-trained language model for English Tweets. arXiv preprint arXiv:2005.10200, 2020.
- [6]. Alturayef, N., H. Luqman and M. Ahmed, A systematic review of machine learning techniques for stance detection and its applications. Neural Computing and Applications, 2023. 35(7): p. 5113-5144.
- [7]. TONGX, W.B., Survey on Adversarial Sample of Deep Learning Towards Natural Language Processing. Computer Science, 2021. 48(1): p. 258-267.
- [8]. Zhang, Y., et al., Textual Adversarial Attacking with Limited Queries. Electronics, 2021. 10(21): p. 2671.
- [9]. Liu, Y., et al., Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.
- [10]. Si, C., 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. 2021.
- [11]. Alturayef, N., H. Luqman and M. Ahmed, A systematic review of machine learning techniques for stance detection and its applications. Neural Computing and Applications, 2023. 35(7): p. 5113-5144.
- [12]. Thomas, M., B. Pang and L. Lee, Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. arXiv preprint cs/0607062, 2006.
- [13]. Li, Y., Y. Sun and W. Jing, Summary of text stance detection. Journal of Computer Research and Development, 2021. 58(11): p. 2538-2557.
- [14]. Vaswani, A., et al., Attention is all you need. Advances in neural information processing systems, 2017. 30.
- [15]. Pennington, J., R. Socher and C.D. Manning. Glove: Global vectors for word representation. in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.
- [16]. Mathew, L. and V.R. Bindu. A review of natural language processing techniques for sentiment analysis using pre-trained models. in 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC). 2020: IEEE.
- [17]. Pan, S.J. and Q. Yang, A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 2010. 22(10): p. 1345-1359.
- [18]. Pennington, J., R. Socher and C.D. Manning. Glove: Global vectors for word representation. in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.
- [19]. Mikolov, T., et al., Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.

- [20]. Burstein, J., C. Doran and T. Solorio. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 2019.
- [21]. Devlin, J., et al., Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
- [22]. Radford, A., et al., Improving language understanding by generative pre-training. 2018.
- [23]. Liu, Y., et al., Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.
- [24]. Zhang, Z. and M. Lan. ECNU at SemEval 2016 task 6: relevant or not? Supportive or not? A two-step learning system for automatic detecting stance in tweets. in Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016). 2016.
- [25]. Pomerleau, D. and D. Rao, Fake news challenge stage 1 (FNC-I): Stance detection. 2017.
- [26]. Goodfellow, I.J., J. Shlens and C. Szegedy, Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014.
- [27]. Jin, D., 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. 2020.
- [28]. Miyato, T., A.M. Dai and I. Goodfellow, Adversarial training methods for semi-supervised text classification. arXiv preprint arXiv:1605.07725, 2016.
- [29]. Madry, A., et al., Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083, 2017.
- [30]. Zhu, C., et al., Freelib: Enhanced adversarial training for natural language understanding. arXiv preprint arXiv:1909.11764, 2019.
- [31]. LIU, R., X. YE and Z. YUE, Review of pre-trained models for natural language processing tasks. Journal of Computer Applications, 2021. 41(5): p. 1236.
- [32]. Papernot, N., et al. Crafting adversarial input sequences for recurrent neural networks. in MILCOM 2016-2016 IEEE Military Communications Conference. 2016: IEEE.
- [33]. Haibin, Z., et al., Survey of adversarial attack, defense and robustness analysis for natural language processing. Journal of Computer Research and Development, 2021. 58(8): p. 1727-1750.
- [34]. Pruthi, D., B. Dhingra and Z.C. Lipton, Combating adversarial misspellings with robust word recognition. arXiv preprint arXiv:1905.11268, 2019.
- [35]. Wang, T., et al., Cat-gen: Improving robustness in nlp models via controlled adversarial text generation. arXiv preprint arXiv:2010.02338, 2020.
- [36]. Li, J., et al., Textbugger: Generating adversarial text against real-world applications. arXiv preprint arXiv:1812.05271, 2018.
- [37]. Miyato, T., A.M. Dai and I. Goodfellow, Adversarial training methods for semi-supervised text classification. arXiv preprint arXiv:1605.07725, 2016.
- [38]. Gao, J., et al. Black-box generation of adversarial text sequences to evade deep learning classifiers. in 2018 IEEE Security and Privacy Workshops (SPW). 2018: IEEE.
- [39]. Belinkov, Y. and Y. Bisk, Synthetic and natural noise both break neural machine translation. arXiv preprint arXiv:1711.02173, 2017.
- [40]. Li, Y., T. Baldwin and T. Cohn, Towards robust and privacy-preserving text representations. arXiv preprint arXiv:1805.06093, 2018.

- [41]. Wei, J. and K. Zou, Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint arXiv:1901.11196, 2019.
- [42]. Guo, H., Y. Mao and R. Zhang, Augmenting data with mixup for sentence classification: An empirical study. arXiv preprint arXiv:1905.08941, 2019.
- [43]. Liu, X., et al., Multi-task deep neural networks for natural language understanding. arXiv preprint arXiv:1901.11504, 2019.
- [44]. Ren, S., 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. 2019.
- [45]. Li, B., Y. Hou and W. Che, Data augmentation approaches in natural language processing: A survey. AI Open, 2022. 3: p. 71-90.
- [46]. La Malfa, E. and M. Kwiatkowska. The king is naked: on the notion of robustness for natural language processing. in Proceedings of the AAAI Conference on Artificial Intelligence. 2022.

APPENDICES

The program source code of this project will be submitted through the attachment upload channel at the same time.