

Yingqi Zhao

AN EXPLORATION OF FINE-TUNING BERT FOR HATE SPEECH ANALYSIS

Faculty of Information Technology and Communication Sciences (ITC)
Master's Thesis in Technology
March 2024

Abstract

Yingqi Zhao: An exploration of fine-tuning BERT for hate speech analysis
Master's Thesis in Technology
Tampere University
Master's Degree Programme in Signal Processing and Machine Learning
March 2024

In the realm of Natural Language Processing (NLP), sentiment classification stands as an essential subject, commonly categorizing text into three classes: positive, negative, and neutral, enabling the statistical analysis of the emotional tendencies in the text. With the development of social networks, the classification requirements for sentiment analysis have gradually become more refined. This thesis draws datasets related to hate speech, fine-tuning BERT as a classification model to categorize the text into five categories to enhance the analysis of hate speech within social media platforms. Simultaneously, to address the substantial parameter size of large language models and the associated high fine-tuning costs, the thesis explores three methods for fine-tuning the BERT model: Fine-tuning BERT with all parameters, Fine-tuning BERT only the classification heads and Fine-tuning BERT with LoRA.

This Thesis empirically validates the feasibility and superiority of designing targeted sentiment analysis models for refined scenarios. Achieving 90% accuracy confirms the outstanding performance of the BERT model in addressing hate speech analysis issues. Furthermore, comparing different fine-tuning methods validates the potential and value of the PEFT method for fine-tuning transformer models.

Keywords: NLP, sentiment analysis, Transformer, BERT, fine tune, PEFT.

The originality of this thesis has been checked using the Turnitin Originality Check service.

Acknowledgements

Since 2022, I have been searching for a suitable master's thesis topic to complete my degree. However, due to the COVID-19 pandemic and my residence in Helsinki, the lack of a network made my application process challenging. During this time, anxiety about graduation and the future made it a difficult period for me. Fortunately, I eventually found suitable supervisors to guide me through completing this thesis.

I am deeply grateful to Konstantinos Stefanidis and Zheyang Zhang for their dedication throughout the process of completing this thesis. From topic selection to review, they provided timely and patient feedback and advice at every stage. Under their guidance, my thinking and work continuously improved, which brought me fulfillment and a sense of accomplishment.

I am also grateful to my family for supporting me in life and finance, allowing me to complete this stage of my study, and giving me the courage to chase my dreams.

Although this thesis may not have high value in the broader research field due to my limitations, it marks a milestone in my learning and research journey. From here, I look forward to further study and hope that one day, I can also help future scholars as my supervisors work.

Contents

1	Introduction	1
2	Review of sentiment analysis, BERT models, PEFT, and related works . .	7
2.1	Sentiment analysis	7
2.2	Evaluation metrics for sentiment analysis	9
2.3	Transformer	10
2.4	BERT	14
2.5	Parameter-Efficient Fine-Tuning (PEFT)	15
2.6	Novelty of the work in this thesis	18
3	Introduction and review of BERT-related components	20
3.1	Embeddings	21
3.2	Normalization	21
3.3	Dropout	24
3.4	BertEncoder	24
3.5	Activation function	25
3.6	Softmax function and AdamW optimizer	27
4	Experiments	29
4.1	Dataset for fine-tuning models	29
4.2	BERT model and related component settings	29
5	Results	31
5.1	Fine-tuning results analysis	31
5.2	Full-parameter fine-tuning BERT and LoRA-BERT application anal- ysis and comparison	35
6	Conclusion and future work	39
	References	46

List of Figures

1.1	Visualization of features in a fully trained CNN model	4
1.2	The overview and workflow of this thesis	6
2.1	The approaches of sentiment analysis	8
2.2	Confusion matrix for binary classification	10
2.3	The multi-head self-attention mechanism	12
2.4	The overall structure of the Transformer	13
2.5	BERT input embedding representation	15
2.6	The development of PEFT methods	17
2.7	The differences between Adapter, Prefix and LoRA in the transformer architecture	18
3.1	Comparison of BatchNorm and LayerNorm	22
3.2	Comparison of BatchNorm and LayerNorm on sequence data	23
3.3	GELUActivation function figure	25
3.4	Tanh function figure	26
4.1	Distribution of different categories of text in the data set	30
5.1	Fine-tuning BERT with all parameters	32
5.2	Fine-tuning BERT only the classification heads diagram	33
5.3	Fine-tuning BERT only the classification head	34
5.4	Fine-tuning BERT with LoRA	35
5.5	Israel-Hamas dataset analysis	36
5.6	Ukraine dataset analysis	37
5.7	Vaccines dataset analysis	37

List of Tables

1.1	General sentiment analysis text example	2
5.1	Comparison of statistical metrics of classification results, the order of numerical values following 'BERT/hBERT/LoRA' corresponding Fine-tuning BERT with all parameters, Fine-tuning BERT only the classification heads and Fine-tuning BERT with LoRA	31

List of abbreviations

NLP	Natural language processing
PEFT	Parameter-Efficient Fine-Tuning.
LoRA	Low-Rank Adaptation of Large Language Models
Prefix	Prefix tuning
BERT	Bidirectional Encoder Representations from Transformers
MPQA	Multi Perspective Question Answering
SVM	support vector machine
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
GRU	Gated recurrent units
LSTM	Long short-term memory
MLP	Multi-Layer Perceptron
GPT	Generative pre-trained transformers
LLaMA	Recurrent Neural Network
VIT	Vision Transformer
CLIP	Contrastive Language-Image Pre-Training
MLM	Masked Language Model
NSP	Next Sentence Prediction
CPU	central processing unit
GPU	graphics processing unit
CNNs	Convolutional neural network
GELU	The Gaussian Error Linear Unit
CDF	cumulative distribution function
LayerNorm	Layer normalization
BatchNorm	Batch normalization

1 Introduction

As a significant branch in artificial intelligence, NLP(Natural language processing) is dedicated to enabling computers to comprehend, interpret, and generate human language, aiming to achieve a communication effect similar to interacting with humans(Chowdhury 2003). Within this overarching objective, sentiment analysis emerges as an indispensable topic, focusing on analyzing emotions and opinions embedded within textual content. In recent years, research on sentiment analysis has continuously evolved, progressing from traditional dictionary methods to machine learning techniques and hybridization of both(Birjali, Kasri, and Beni-Hssane 2021). The problem domain of sentiment analysis has concurrently expanded and subdivided, reflecting the dynamic developments in this field(Wankhade, Rao, and Kulkarni 2022). Traditional sentiment analysis models categorize text into neutral, negative, and positive. Even such a simple classification has found widespread application in the business domain(Wankhade, Rao, and Kulkarni 2022). However, with the evolution of social networks, sentiment analysis tailored for social media texts has garnered attention, particularly in areas such as politics and crime prevention(Chakraborty, Bhattacharyya, and Bag 2020). The traditional simplistic classification methods have gradually become inadequate to meet the growing analytical demands and increasingly nuanced analysis scenarios. Katsarou et al. 2021 proposes a valuable approach to the analysis of hate speech.

As a groundbreaking work in NLP, BERT(Bidirectional Encoder Representations from Transformers Devlin et al. 2019) inherits the self-attention mechanism from the transformer(Vaswani et al. 2017) architecture. It introduces bidirectional processing of text, masked language modeling, and pre-training on combining two sentences. These design choices significantly enhance the model’s generalizability and its ability to capture textual features. Further details about these aspects will be elaborated upon later. However, as a member of the transformer model family, especially with the prevailing notion that larger language models yield better performance, the number of parameters in these models has surged from tens of millions to tens of billions(Kaplan et al. 2020). Consequently, this has increased the hardware requirements for these models, making training, even for fine-tuning, progressively more expensive. In response to this situation, the Parameter-Efficient Fine-Tuning (PEFT) method has emerged as a new research hotspot(Xu et al. 2023).

Traditional approaches to large model applications often involve pretraining a robust feature extractor on massive datasets, followed by fine-tuning the entire large model for specific downstream tasks. In contrast, the PEFT method attempts to train only a tiny proportion of parameters during fine-tuning, aiming to achieve

results close to fine-tuning all parameters. Some classic methods for PEFT include Adapter(Houlsby et al. 2019), LoRA(Low-Rank Adaptation of Large Language Models Hu et al. 2021), and Prefix(Prefix tuning Li and Liang 2021), which will be briefly introduced in the subsequent sections.

Overall, this thesis explores three aspects of the issue.

Firstly, it investigates a method for sentiment analysis in refined scenarios. As previously mentioned, traditional sentiment analysis methods typically start by detecting the basic emotional polarity of text and then extracting the necessary information for real-world applications. For instance, discerning whether the following sentences are neutral statements, negative complaints, or positive expressions of sentiment. The table 1.1 shows several examples of Twitter text from platform X.

Table 1.1 General sentiment analysis text example

Text	Sentiment polarity
I'd have responded, if I were going	neutral
Sooo SAD I will miss you here in San Diego!!	negative
what interview! leave me alone	negative
Journey!? Wow... u just became cooler. hehe... (is that possible!?)	positive
I really really like the song Love Story by Taylor Swift	positive
oh Marly, I'm so sorry!! I hope you find her soon!! <3 <3	neutral

This type of analysis is valuable. For instance:

- Software product maintainers can aggregate and analyze the sentiment polarity of product reviews over time to gather feedback on product versions, thereby gaining insights into product improvement directions(Lin et al. 2022).
- Brands can collect reviews from different regions to explore adapting various product features across different areas, facilitating localized improvements in brand and product offerings(Wankhade, Rao, and Kulkarni 2022).
- In social media, analyzing the sentiment polarity distribution under specific topics, such as individuals or events, can reveal trends in public opinion and sentiment(Chakraborty, Bhattacharyya, and Bag 2020).

However, in reality, issues tend to be specific and contextualized. For example, in war-related topics, discussions often lean towards negativity or neutrality due to casualties and environmental destruction, while positive comments are relatively scarce. Following are several war-related tweets from Lab 2023:

- BREAKING: Minister of Defence of Israel: ""The rules of war have changed, we will cripple Gaza so that it will remember it for the next 50 years.""

- All wars are bankers wars.
- Israel said ””Humanitarian aid to Gaza? No electricity, No water; No fuel will be restored until all Israeli hostages are returned home””

In such scenarios, where a collection of texts from Platform X concerning the Israeli-Palestinian conflict has been gathered, simple sentiment polarity distribution can already be anticipated. At this point, employing traditional sentiment analysis methods for classification may not provide much additional value. Therefore, the need for fine-grained sentiment polarity classification arises.

Katsarou et al. 2021 proposed a new sentiment polarity classification system tailored for topics such as immigration, climate change, and the COVID-19 pandemic. This system categorizes text into five classes: hate speech, neutral, offensive, positive, and sexism, corresponding to sentimental types, including group-based hate speech, neutral statements, verbal attacks, positive text, and sexual harassment. This thesis continues this line of thinking, employing BERT as a sentiment polarity classifier to explore the analysis of hate speech.

The second aspect involves exploring the performance of different methods for fine-tuning BERT. Fine-tuning a model refers to using a pre-trained model, trained on large-scale datasets, and then readjusting its parameters on smaller datasets specific to a particular task. Generally, these pre-trained models are capable of solving certain types of problems or extracting specific data features. By retraining the pre-trained model on a smaller dataset specific to a particular task, the model’s parameters can be adjusted to adapt it to the task at hand better.

Typically, any end-to-end deep learning model system has substantial data requirements. Designing a separate model and its corresponding dataset for each problem can theoretically achieve optimal results. However, the practical cost is evidently prohibitive. Moreover, data for specific tasks is often scarce in real-world production environments. For instance, in developing a medical image diagnosis model, experts typically need to annotate the data, and the availability of such experts is limited, with their work time being precious. On the other hand, recent research on the generalization ability of neural networks has shown that models possess a certain degree of learning ”knowledge” capability. Taking the research on convolutional neural networks (CNNs) in image processing (Zeiler and Fergus 2013) as an example, each layer of the model extracts image features at different scales, showing as figure 1.1, and some features of objects are similar or close. Hence, transfer learning has emerged, which involves leveraging existing model knowledge to address new problems more quickly and cost-effectively (Pan and Yang 2010). Empirical evidence has proven the feasibility of this approach.

Fine-tuning can be considered a component of transfer learning, focusing on

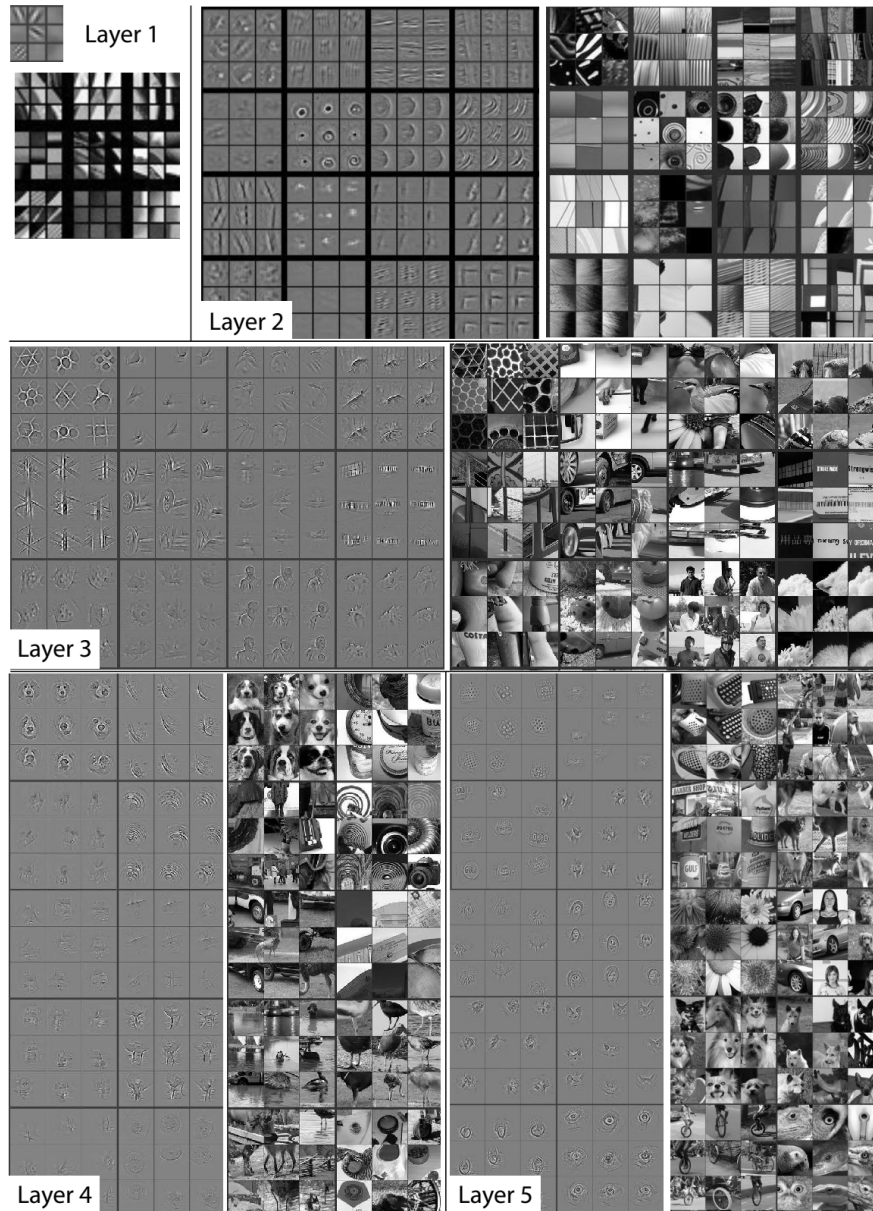


Figure 1.1 Visualization of features in a fully trained CNN model refer from Zeiler and Fergus 2013. Image features at different scales are captured as the network deepens.

adjusting the parameters of a pre-trained model using a smaller dataset specific to a particular task, enabling the model to better adapt to the task at hand (Yosinski et al. 2014). Generally, in classification tasks, the methods for fine-tuning models can be roughly divided into two categories:

The first method is full parameter fine-tuning, where all pre-trained model parameters are retrained on the fine-tuning dataset. This approach often yields good results; however, its drawback lies in retraining all parameters, requiring more time and computational resources. Additionally, overfitting becomes a concern when the

model size is significantly larger than the fine-tuning dataset. Overfitting occurs when the model learns too many details and noise from the training data, making it unable to generalize well to new data (Ying 2019). It is akin to a person mistaking the texture and stains on a cup as its core features; when encountering a new cup without the same texture and stains, this person may not recognize it as a cup.

The second method is fine-tuning only the classification head. The classification head is the part of the model responsible for making classification decisions, and its structure varies widely. It typically includes fully connected layers and activation functions and may also incorporate regularization layers such as dropout. This approach is based on the hypothesis that the main parts of the model have already gained sufficient knowledge (features) about relevant data and only need to classify the data according to the task requirements based on these features. The advantage of this method is that it does not require training all model parameters, thus improving efficiency to some extent. However, its disadvantage is that it may not apply to all models, leading to potentially suboptimal results.

As mentioned earlier, there has been a recent surge in the PEFT method, tailored explicitly for Transformer model structures, with many studies demonstrating its superior performance. Chapter 2.5 will provide more details about PEFT methods. Therefore, this paper conducts fine-tuning on BERT using three methods: full parameter fine-tuning, fine-tuning only the classification head, and fine-tuning with LoRA (a type of PEFT method). This thesis will compare the classification performance of these models on the hate speech dataset to explore the advantages and disadvantages of the three fine-tuning methods. Chapter 5.1 will present these results.

The third aspect involves exploring the practical application of hate speech analysis in real-world scenarios. The best way to assess the effectiveness of a method is to utilize it. Lab 2023 provides highly circulated Twitter texts related to the Israeli-Palestinian conflict, the Russia-Ukraine war, and the COVID-19 vaccine collected from Platform X. This thesis employs the fine-tuned BERT model with promising results to analyze hate speech in these texts. Additionally, it conducts a statistical analysis of the development trends of different sentiment types based on the publication dates of the tweets. Further details are presented in Chapter 5.2. In summary, this thesis explores optimization strategies for sentiment analysis in refined scenarios through hate speech analysis. It validates and investigates the performance of fine-tuning the BERT model in this context. Comparing results from different fine-tuning methods provides practical insights for future model applications. The figure 1.2 shows the overall workflow of this paper.

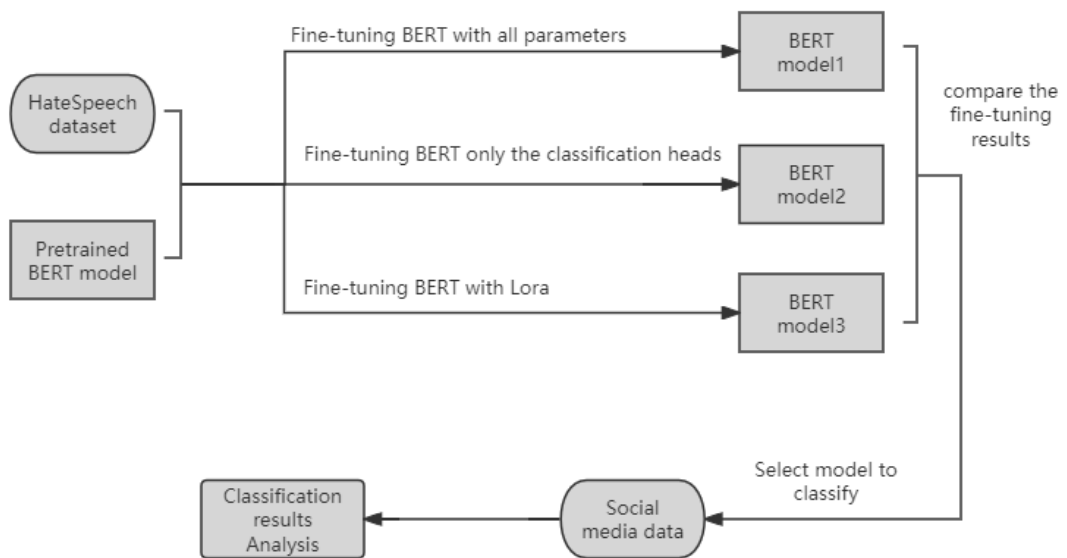


Figure 1.2 The overview and workflow of this thesis. First, this thesis organizes the hate speech data set and the pre-trained BERT model, then compares the performance of the models trained by the three fine-tuning methods, and finally uses the model to classify social media data for hate speech analysis.

2 Review of sentiment analysis, BERT models, PEFT, and related works

2.1 Sentiment analysis

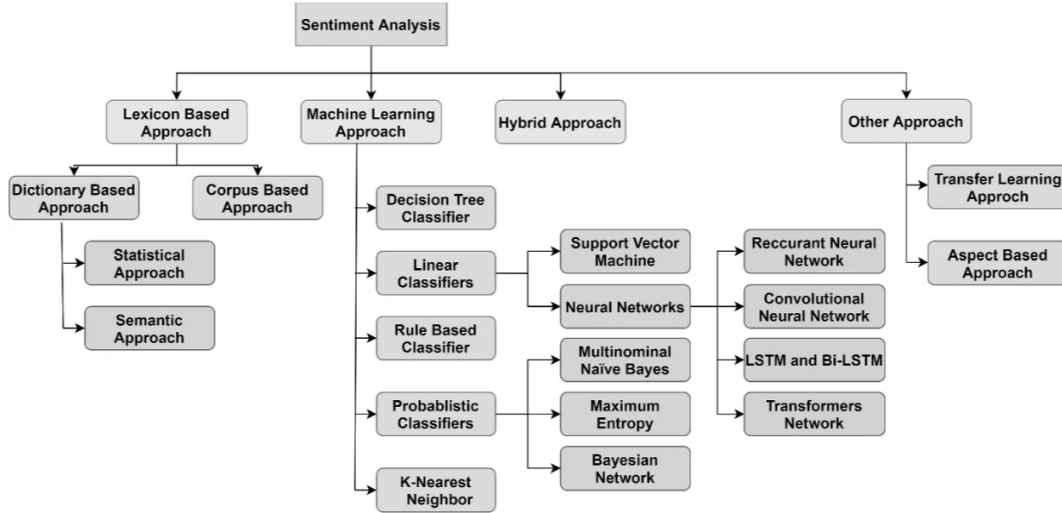
Sentiment analysis (Opinion analysis or Opinion mining) is an important research topic in the field of NLP. It aims to obtain the attitude and views of the text author on specific topics or things by analyzing the emotions contained in the text (Serrano-Guerrero et al. 2015). Sentiment classification is a core research field of sentiment analysis, its main goal is to identify sentiment categories of text. In the early stages of research, sentiment classification usually divides text into positive, negative, and neutral categories (Rana, Nawaz, and Iqbal 2018). It has been widely used in business, such as market research and product evaluation (Wankhade, Rao, and Kulkarni 2022).

With the rapid development of social media, social media has become an essential channel for people to exchange opinions (Kapoor et al. 2018). However, there are also some harms caused by false and harmful information, such as racial remarks, verbal violence, verbal harassment, and conspiracy theories. Therefore, sentiment analysis methods have new application scenarios, such as politics and crime prediction (Gongane, Munot, and Anuse 2022). In this novel context, traditional methods employed for analyzing basic sentiment may evidently prove inadequate to accommodate the more nuanced and specific classification requirements. For example, when judging negative speech, reviewers may need to further determine its type, whether it is offensive or hate speech.

To address this issue, (Katsarou et al. 2021) introduces a new classification method for hate speech analysis. It collected textual data related to the MeToo Movement, Immigration, Climate Change, and Coronavirus topics on the X platform, constructing a dataset. Subsequently, these texts are labeled into five categories - Hate Speech, Offensive, Sexism, Neutral, and Positive. With this dataset, a text classification model can be trained to achieve the purpose of hate speech analysis.

Sentiment analysis methods can be roughly divided into four categories: lexicon based approach, machine learning based approach, hybrid approach and other approach, as figure 2.1.

The lexicon-based approach typically involves constructing a lexicon based on the vocabulary or phrases in the text, where the lexicon contains 'sentiment scores' for each vocabulary term (Jurek, Mulvenna, and Bi 2015). Subsequently, senti-



Approach of sentiment analysis

Figure 2.1 The approaches of sentiment analysis refer from Wankhade, Rao, and Kulkarini 2022

ment analysis is performed by employing statistical methods to aggregate the emotional tendencies of the vocabulary in the text. Dictionaries can be manually created, such as MPQA(Multi Perspective Question Answering) Subjectivity Lexicon(Wilson, Wiebe, and Hoffmann 2005), or constructed using various lexical techniques, such as identifying synonyms and antonyms(Miller et al. 1990). Due to the variability of word meanings in different contexts, there is an improvement known as corpus-based methods. This includes statistical approaches, where the frequency of occurrence of words with the same sentiment tendency is analyzed in context, and semantic-based methods that measure the similarity between words using various techniques. By combining statistical and semantic information, these methods analyze the emotional tendencies of words(Birjali, Kasri, and Beni-Hssane 2021).

Machine learning methods have experienced rapid development over the past few decades(Sarker 2021).

For classical algorithms such as the SVM(support vector machine) algorithm, the approach involves finding a hyperplane that effectively separates data from different categories while maximizing the margin between these categories(Hearst et al. 1998). In 2013, SVM was utilized for sentiment classification(Wankhade, Rao, and Kulkarini 2022). Other algorithms, such as decision trees(Quinlan 1986), random forest algorithms(Breiman 2001), and probability-based methods like Naive Bayes algorithms(Vikramkumar, B, and Trilochan 2014) and Bayesian networks(Heckerman 2008), have also found applications in sentiment analysis(Birjali, Kasri, and Beni-Hssane 2021).

Deep learning(LeCun, Bengio, and G. Hinton 2015) algorithms have gradually

become popular with the rapid development of hardware performance. Built on the Artificial Neural Network (ANN, Agatonovic-Kustrin and Beresford 2000) structure, deep learning divides the model into input, output, and hidden layers. It learns features of input data and outputs a nonlinear function regarding these features. Deep learning enhances and extends feature extraction performance by increasing the number of hidden layers and altering the algorithmic structure of neural network modules.

As RNN (Recurrent Neural Network Sherstinsky 2020) has a natural advantage in handling sequential data, RNN and its variants, such as GRU(Gated recurrent units Chung et al. 2014) and LSTM(Long short-term memory Ting 2010), have been widely applied in sentiment analysis(Wankhade, Rao, and Kulkarni 2022). However, due to the inherent structure of RNN, which requires the output of the computation for the previous sequence node (e.g., a word in a sentence) to be part of the input for the next sequence node (the following word in the sentence), it is constrained to a linear processing flow, handling one word at a time. Capturing connections between contexts often requires storing the computation information of multiple words in memory, resulting in significant memory consumption. The introduction of the Transformer(Vaswani et al. 2017) architecture has largely addressed this issue. As it is highly relevant to the model employed in this thesis, a detailed explanation will be provided in the subsequent chapters.

Indeed, there have been studies proposing hybrid methods. The essence of a hybrid approach lies in combining lexicon-based methods with machine learning methods. This can involve constructing lexicons using machine learning methods or employing lexicons to assist machine learning methods in classification, aiming to achieve enhanced sentiment analysis performance(Birjali, Kasri, and Beni-Hssane 2021). Other methods, such as aspect-based sentiment analysis (ABSA), are mostly used in hotel review and product review applications(Wankhade, Rao, and Kulkarni 2022).

2.2 Evaluation metrics for sentiment analysis

Regarding the performance evaluation metrics for sentiment analysis, since the core of sentiment analysis often lies in sentiment classification, its evaluation metrics are related to those used in classification tasks.

Confusion matrix(Ting 2010), especially in the context of multi-class classification problems, provides a more intuitive representation, and in this thesis, it will be the primary means of presenting results. The confusion matrix for binary classification is shown in figure 2.2, which illustrates the counts of predicted values against actual values. When a predicted sample belongs to a certain class (i.e. classified by the model as a certain class), it is counted as positive, and when it does not belong

Ground truth	Positive	True positive	False negative
	Negative	False positive	True negative
		Positive	Negative
		Prediction	

Figure 2.2 Confusion matrix for binary classification

to that class, it is counted as negative. Similarly, the ground truth represents the true class or label of the samples. By counting the number of samples, the table visually represents the effectiveness of the model's classification. More specifically, when the model predicts a class that matches the actual label of the sample, it is counted as a True Positive. Similarly, other scenarios are counted accordingly, such as False Positive, True Negative, and False Negative.

Confusion matrices can be used to derive other popular classification evaluation metrics, such as precision, recall, and F1 score.

- *Precision:*

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- *Recall:*

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- *F1score:*

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.3 Transformer

As mentioned earlier, in Recurrent Neural Networks, when processing sequential data, it is necessary to sequentially store and compute the states of all hidden nodes in memory to incorporate the correlation between the preceding and succeeding contexts. This sequential processing limits its performance. The Transformer architecture discards the structure of recurrent layers and adopts a multi-head self-attention

module to capture sequence information in parallel. Therefore, when the sequence length is T , the sequence computational complexity on a single layer changes from $O(T)$ to $O(1)$. Combined with the overall architecture of the encoder-decoder, the Transformer has made significant breakthroughs in performance (Vaswani et al. 2017).

Attention is a crucial component in human cognition, allowing individuals to effectively filter out relevant information from complex environments to address specific queries. Research on attention mechanisms inspired by cognitive attention has long been present in machine learning and deep learning domains (Niu, Zhong, and Yu 2021). The Transformer model draws inspiration from previous work and adjusts the focus of information in the context concerning the current problem by learning a weight.

Specifically, a Q, K, V model illustrates the attention mechanism in the Transformer model, showing as figure 2.3. For better comprehension, Q can be regarded as a matrix representing the queries of the questions. By assessing the similarity between Q and K , a weight matrix is obtained. Subsequently, the weights are normalized, and the matrix product of the normalized weights and the V matrix yields the output of attention. The entire process can be viewed as an estimation of the problem's value in the V dimension through the comparison of Q and K based on existing cues from K and V . The original mathematical explanation is as follows.

Given an input sequence X with dimension d_{model} , multi-head attention first projects X into h different spaces for each head i , computing queries (Q), keys (K), and values (V):

$$Q_i = XW_i^Q, \quad K_i = XW_i^K, \quad V_i = XW_i^V$$

where W_i^Q , W_i^K , and W_i^V are parameter matrices of dimensions $d_{\text{model}} \times d_k$, $d_{\text{model}} \times d_k$, and $d_{\text{model}} \times d_v$ respectively, and d_k and d_v represent the dimensions of keys and values.

For each head, the scaled dot-product attention mechanism computes the output as:

$$\text{Attention}(Q_i, K_i, V_i) = \text{softmax} \left(\frac{Q_i K_i^T}{\sqrt{d_k}} \right) V_i$$

They use the dot product as the cosine similarity between Q_i and K_i . Then, apply Softmax to ensure weights are greater than 0 and sum to 1. Since the purpose of Softmax (Goodfellow, Bengio, and Courville 2016) function is to assign high confidence to certain elements (closer to 1) and low confidence to others (closer to 0). This situation will be exacerbated when d_k is large, leading to small gradients during gradient computation, making it challenging for the model to converge. So, they introduce a scaling factor $\sqrt{d_k}$.

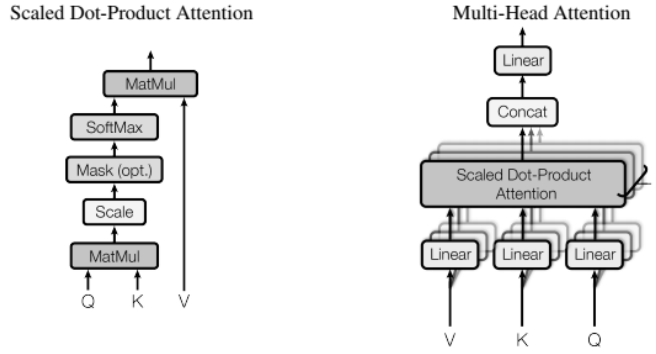


Figure 2.3 The multi-head self-attention mechanism refer from Vaswani et al. 2017. The left side is the attention process of a single head, and the right side is the multi-head attention mechanism.

The self-attention mechanism means that the inputs X are the same. The concept of multi-head attention involves employing parameter matrices multiple times to project Q , K , and V into lower-dimensional spaces, enabling the model to learn different weights. Ultimately, the results are concatenated together, forming the multi-head self-attention mechanism.

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

where $\text{head}_i = \text{Attention}(Q_i, K_i, V_i)$, and W^O is another parameter matrix of dimension $hd_v \times d_{\text{model}}$.

On the other hand, the Transformer adopts an encoder-decoder architecture to address its initial purpose of machine translation. From the diagram 2.4, the encoder and decoder processing structures share many similarities. Both of them require embedding the input text, transforming words into vectors of a unified dimension, which facilitates the processing of text data in models(Almeida and Xexéo 2023). Simultaneously, positional information is encoded to prevent its loss. Subsequently, the positional encoding and the results of attention mechanism processing are transmitted to the subsequent Multi-Layer Perceptron (MLP Murtagh 1991) for spatial transformation of semantic information for each word. The Add and Norm layer conveys the original information, while normalization is applied to enhance the model's efficiency(Ba, Kiros, and G. E. Hinton 2016). The encoder is employed to extract features from the input. At the same time, based on the previously translated output and the following question to be translated, the decoder serves as Q in the attention mechanism. It combines the output gathered by the encoder as K and V for processing. Subsequently, the translation results are an incremental output.

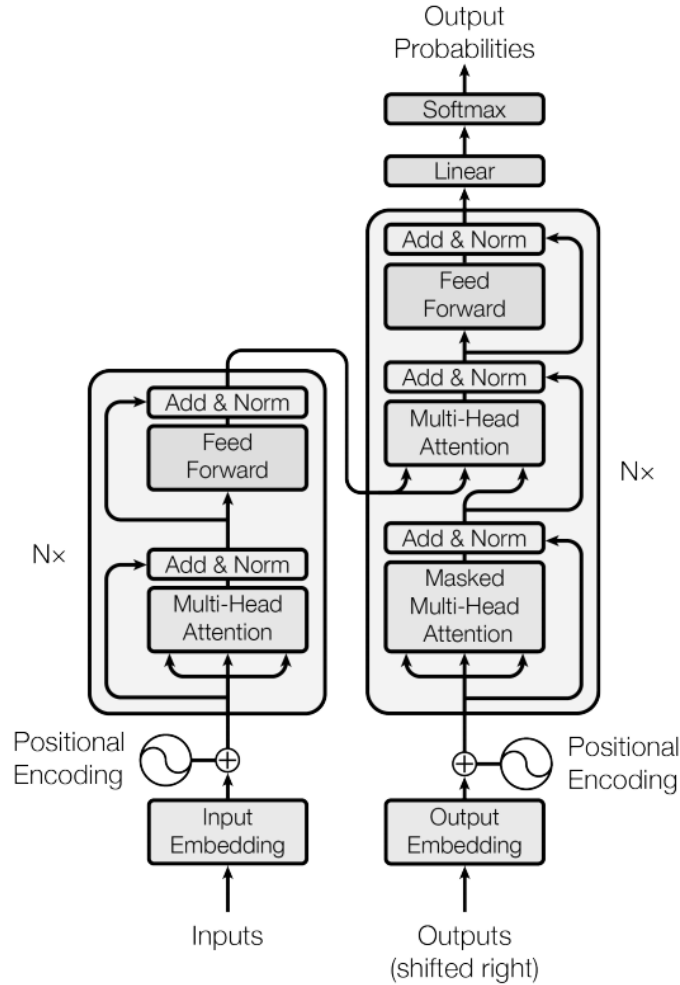


Figure 2.4 The overall structure of the Transformer refer from Vaswani et al. 2017. The part on the left is the structure of the encoder, and the part on the right is the structure of the decoder. The output of the encoder is connected to the decoder through an attention mechanism.

During training, as the source text and target translation are already known, the decoder initially uses a masked multi-head attention module to process the translation. The decoder simulates a non-training state by masking the untranslated positions words. It is achieved by multiplying a minimal number by those positions, making their weights close to 0 in the attention mechanism.

The success of the Transformer model lies not only in its outstanding performance in machine translation, surpassing other approaches at the time, but also in introducing a novel, potent, easily usable and extensible model architecture. (Niu, Zhong, and Yu 2021) From its inception to 2024, research on large language models based on its architecture flourished. Models like BERT (Devlin et al. 2019), built upon its

encoder architecture, have become versatile solutions for various natural language processing tasks. Meanwhile, models like GPT(Generative pre-trained transformers Yenduri et al. 2023) and LLaMA(Large Language Model Meta AI Touvron et al. 2023), based on its decoder architecture, have gained widespread popularity in the market, providing millions of users with the convenience of artificial intelligence. Moreover, the application of the Transformer architecture in computer vision and multimodal domains has become a research hotspot, exemplified by models such as Vision Transformer (ViT Dosovitskiy et al. 2021) and CLIP(Contrastive Language-Image Pre-Training Radford et al. 2021). It is foreseeable that in the coming years, the application and research of Transformer models will continue to hold substantial potential.

2.4 BERT

If the emergence of the Transformer brings a very forward-looking new architecture to NLP, then BERT (Bidirectional Encoder Representations from Transformers, Wankhade, Rao, and Kulkarni 2022) is to prove its performance by significant precision improve and extend its influence to almost all NLP downstream tasks, including tasks such as text classification, named entity recognition, question-answering systems, machine translation, and summarization generation, among others.

BERT adopts the encoder structure of the Transformer. The general methodology involves pretraining a BERT model on a large-scale text dataset and fine-tuning it on different datasets related to downstream tasks to address specific problems. Its bidirectionality is primarily manifested through utilizing the self-attention mechanism from the Transformer. This mechanism allows BERT to focus on different parts of the entire input sequence when processing each position, thereby extracting all contextual information both before and after the processing position.

Regarding the pretraining of BERT, they utilized BooksCorpus (800M words) and English Wikipedia (2,500M words) as training texts. The input sequences were then segmented into fixed-length fragments (embedding). The embedding process consists of a triple structure: token embeddings, segment embeddings, and positional embeddings. These embeddings are trainable, and their summation results in the input embedding, as illustrated in the 2.5.

Then, BERT was trained unsupervised on two tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP), encompassing both word and sentence dimensions.

The MLM task resembles a cloze test. In this task, 15% of the original positions of words in the input text are altered. Specifically, 80% of the words are replaced with a [MASK] token, 10% are replaced with random words to simulate noise, and the remaining 10% are left unchanged to simulate the actual fine-tuning scenario.

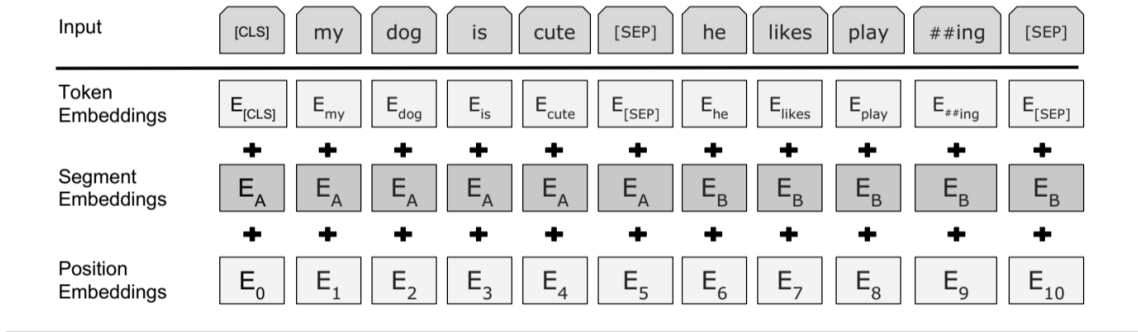


Figure 2.5 BERT input embedding representation refer from Devlin et al. 2019. From top to bottom, it corresponds to word embedding, sentence embedding and position embedding.

The NSP task aims to determine whether two sentences are consecutive in context. They set up equal proportions of data where half of the pairs are consecutive sentences, and the other half are non-consecutive. This task aims to train the model in extracting features at the sentence level.

In the fine-tuning process, the input sentence pairs are treated differently based on the nature of the task. For example, the input sentence pairs are question-passage pairs in question-answering tasks. On the output side, the model generates outputs based on the question task category token. For sentiment analysis, which can be viewed as a text classification problem here, the input pairs are transformed into text- \emptyset pairs, and the [CLS] token is passed to the output side. The remaining steps involve training the pre-trained BERT on the task-specific dataset to complete the fine-tuning.

Although fine-tuning is relatively less resource-intensive than pretraining, (Devlin et al. 2019) resource conservation is crucial in many practical scenarios (Tay et al. 2022). Especially with the trend towards larger and more powerful models underpinning the Transformer family’s recent developments (Kaplan et al. 2020), the number of trainable parameters has skyrocketed from millions to tens or hundreds of billions (GPT-3 175B Brown et al. 2020). The threshold for research and application of large language models has become increasingly elevated. As a result, fine-tuning methods for Transformers have also become an exciting research direction.

2.5 Parameter-Efficient Fine-Tuning (PEFT)

The application pattern of pre-training and fine-tuning models has been widely adopted in deep learning. Models such as BERT and GPT in natural language processing and ResNet (K. He et al. 2015) and ViT (Dosovitskiy et al. 2021) in computer vision have demonstrated the effectiveness of pre-training and fine-tuning in adapting models to specific task features, thereby enhancing performance. As mentioned

earlier, Transformer models are becoming larger in pursuit of ability, accompanied by increased resource consumption(Kaplan et al. 2020). While the model’s scale and the dataset’s size during pre-training impact the model’s capability, can we explore new methods during the fine-tuning phase? If it is possible to fine-tune only a tiny portion of the model parameters and achieve performance close to fine-tuning all parameters, it could save resources — this is the essence of the PEFT method(Xu et al. 2023).

There are various studies on PEFT, shown as figure 2.6. The main research in PEFT methods can be roughly categorized into five types: Adapter Fine-tuning, represented by Adapter; unified fine-tuning where the pre-training data set and the fine-tuning data set are highly correlated; Reparameterized Fine-tuning, represented by LoRA; Spatial Fine-tuning, which involves training only a portion of the model parameters without significantly altering the model structure; and Hybrid Fine-tuning, which combines the methods above(Xu et al. 2023). The following will introduce three classical methods: Adapter(Houlsby et al. 2019), Prefix(Li and Liang 2021) and LoRA(Hu et al. 2021).

The Adapter method primarily involves adding trainable adapter blocks in the frozen transformer layers. Its main structure includes input passing through the Feedforward down-project, going through a non-linear layer, and then being processed through the Feedforward up-project. Given a Transformer model’s hidden layer output H , the Adapter module can be formulated as follows:

- Down Projection: $H_{\text{down}} = HW_{\text{down}} + b_{\text{down}}$, where $W_{\text{down}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{adapter}}}$ and $b_{\text{down}} \in \mathbb{R}^{d_{\text{adapter}}}$.
- Non-linear Activation: $H_{\text{act}} = \text{ReLU}(H_{\text{down}})$.
- Up Projection: $H_{\text{up}} = H_{\text{act}}W_{\text{up}} + b_{\text{up}}$, where $W_{\text{up}} \in \mathbb{R}^{d_{\text{adapter}} \times d_{\text{model}}}$ and $b_{\text{up}} \in \mathbb{R}^{d_{\text{model}}}$.

Prefix Tuning involves using two parts of trainable prefix vectors. One part of it is used to merge with the original K and V in the attention mechanism. Another part is to update feed-forward networks. During fine-tuning, freeze most structures and only train the prefix vectors for each part.

- $head_i = \text{Attention}(Q_i, \text{concat}(P_i^k, K_i), \text{concat}(P_i^v, V_i))$, where $P_i^k, P_i^v \in \mathbb{R}^{d_{\text{model}}}$.

LoRA uses the product of two low-rank matrix $A \in \mathbb{R}^{d_{\text{model}} \times r}$ and $B \in \mathbb{R}^{r \times d_k}$ (r is the rank of the matrix, much smaller than d_{model} and d_k) as ΔW , which approximates the update of the original Q and K parameter matrix W . During fine-tuning, only train the ΔW , and introducing another matrix R can adjust the updated values according to different strategies.

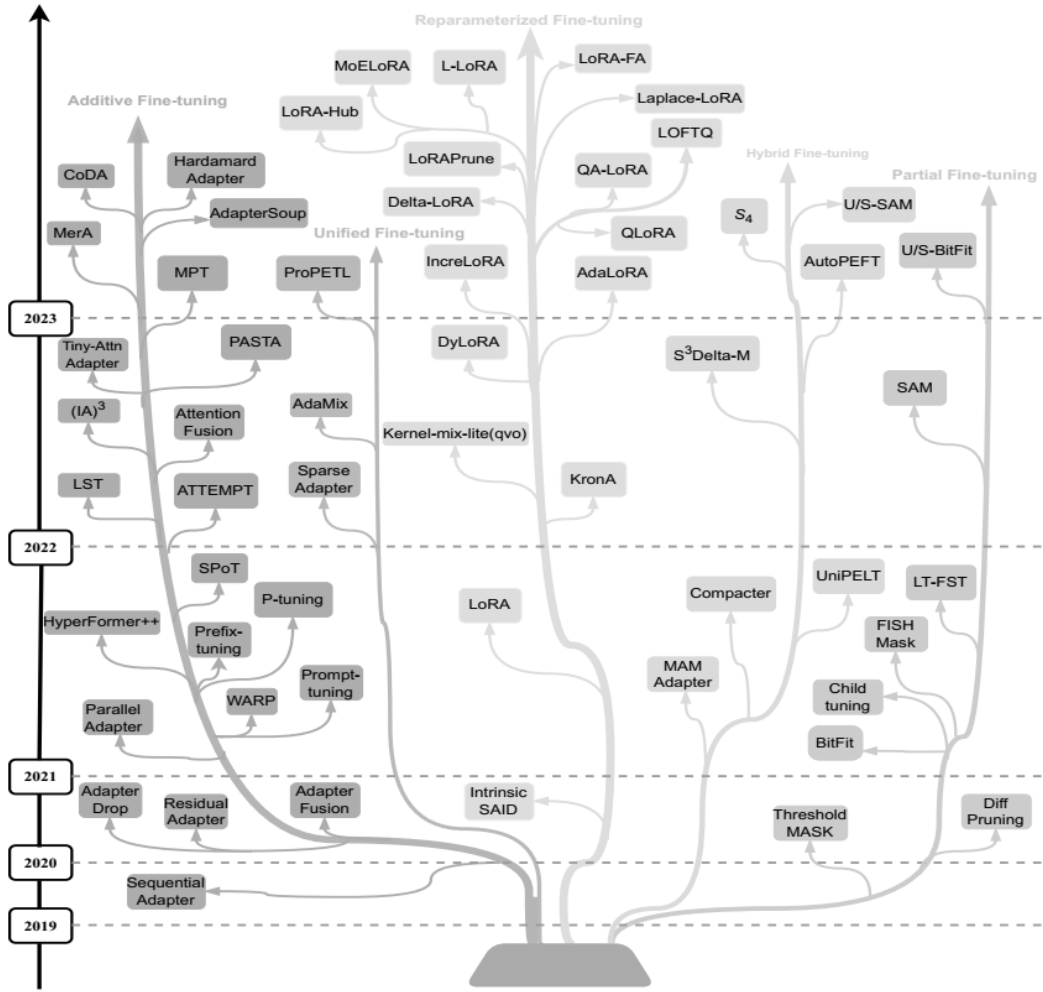


Figure 2.6 The development of PEFT methods refer from Xu et al. 2023

- $W_{\text{updated}} = W + \Delta W$
- $\Delta W = ABR^T$

J. He et al. 2022 provides a comprehensive comparison and summary of recent PEFT methods, figure 2.7 illustrating their differences. It uses h to represent the output of the original attention heads and Δh to define the output update using different PEFT methods. They found that all Δh follow a *projectdown* \leftrightarrow *nonlinear* \leftrightarrow *projectup* structure, and proposed a comprehensive MAM(mix-and-match) Adapter method. The study by Xu et al. 2023 elucidates the potential applications of the PEFT method in multi-task learning, cross-lingual transfer, and backdoor attacks and defence. It compares various experimental results of various PEFT methods, finding that most of these methods significantly enhance parameter efficiency and save memory.

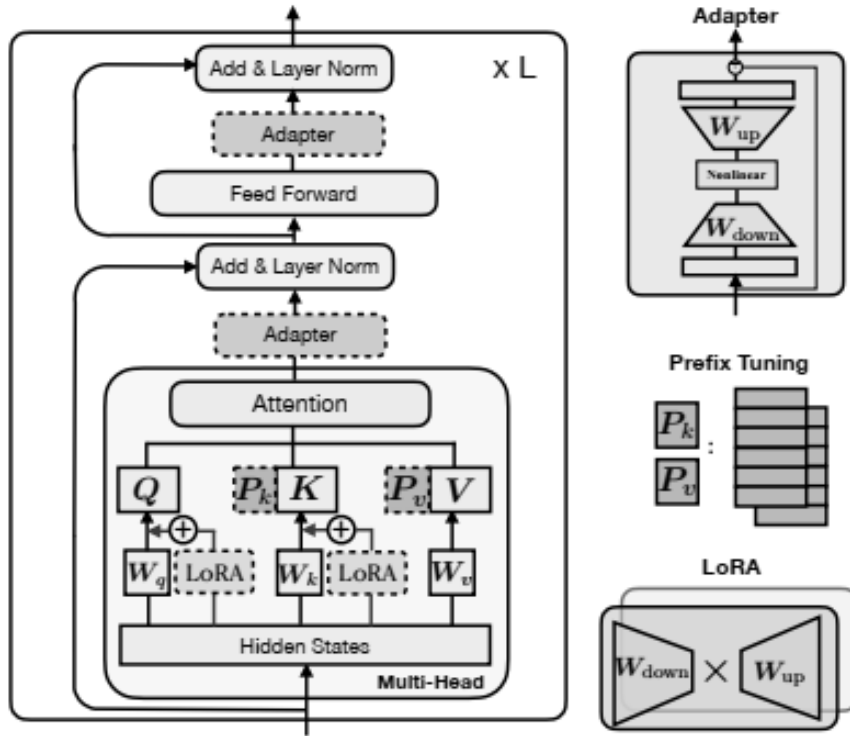


Figure 2.7 Illustration the differences between Adapter, Prefix and LoRA in the transformer architecture refer from J. He et al. 2022

2.6 Novelty of the work in this thesis

As mentioned, most sentiment analysis work aims to analyze text through simple polarity classification followed by statistical methods. For instance, popular datasets like IMDB (Maas et al. 2011) classify text based on a sentiment polarity score, where scores equal to or below four are considered negative, and scores equal to or above seven are considered positive. Various innovative methods continuously improve the accuracy of text classification on IMDB datasets. However, similar attention has not been given to more complex and refined requirements in practical scenarios, such as the application explored in this thesis regarding hate speech analysis. This is partly due to the ongoing development of sentiment analysis research and the diversified nature of refined scenarios, where researchers in different domains may be dispersed.

It is exciting to note that efforts are already exploring applications in this area. As mentioned earlier, Katsarou et al. 2021 is one such example. It proposes new classification methods for hate speech and applies them in practical analyses on social media platforms using ULMFiT (Universal Language Model Fine-tuning, Howard and Ruder 2018) and AWDLSTM (ASGD Weight-Dropped LSTM, Merity, Keskar,

and Socher 2017) models. This thesis continues this idea of classification and analysis approach, introducing innovations in model selection, fine-tuning methods and practical methods. Notably, Vidgen and Derczynski 2020 has collected and maintained a dataset list specifically for hate speech, providing more extensive data for related research endeavors.

Research on fine-tuning methods can generally be categorized into two types: those that are studied as part of transfer learning and those associated with the rise of Transformer models, such as the PEFT method. This thesis briefly summarises previous work in this area and compares different fine-tuning methods based on the BERT model in the context of hate speech analysis. Through practical experiments, various fine-tuning methods are evaluated for their applicability in this scenario, providing valuable insights into their effectiveness.

In summary, building upon prior research, this thesis consolidates findings in hate speech analysis, BERT utilization, comparison of fine-tuning methods, and practical application analysis. It then proceeds to explore these aspects further, aiming to contribute to advancing knowledge in the field.

3 Introduction and review of BERT-related components

The BERT model incorporates several noteworthy components. Taking the 'bert-base-cased' model from Hugging Face as an example, the structure of the BERT model, when printed using the `print(model)` function, is as follows:

```
(bert): BertModel(
  (embeddings): BertEmbeddings(
    (word_embeddings): Embedding(28996, 768, padding_idx=0)
    (position_embeddings): Embedding(512, 768)
    (token_type_embeddings): Embedding(2, 768)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (encoder): BertEncoder(
    (layer): ModuleList(
      (0-11): 12 x BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (intermediate): BertIntermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
      (intermediate_act_fn): GELUActivation()
    )
    (output): BertOutput(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
)
```

)

Program output: The structure of the BERT model

The BERT model comprises three principal structures: BertEmbeddings, BertEncoder, and BertPooler. BertEmbeddings is responsible for input embedding, BertEncoder conducts attention processing on the data, and BertPooler prepares the model output.

3.1 Embeddings

As the BERT section outlines, BERT incorporates three embeddings for the encoded textual input:

- Token embeddings correspond to *word_embeddings* layers, and the embedding vector has a dimensionality of 768 and can embed 28996 different word tokens.
- Positional embeddings correspond to *position_embeddings* layers, 512 means the model can handle sequences up to 512 tokens in length.
- Segment embeddings correspond *token_type_embeddings* layers, indicating which sentence of the sentence pair the token belongs.

3.2 Normalization

Normalization is a data preprocessing technique that scales data proportionally to fit within a specific range, such as 0 to 1 or -1 to 1. This process accelerates the convergence speed of learning algorithms and enhances the model's stability (Bach, Kiros, and G. E. Hinton 2016).

The entire model structure uses LayerNorm (Layer normalization) for normalization instead of BatchNorm (Batch normalization). Here is a brief introduction to the difference between the two. In the case of two-dimensional input, BatchNorm will normalize the features of each column, as 3.1 Given an input batch x , the Batch Normalization transform is defined as:

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad (3.1)$$

where:

- $x^{(k)}$ is the k^{th} feature of the input,
- $\mu_{\mathcal{B}}$ is the mean of the batch,
- $\sigma_{\mathcal{B}}^2$ is the variance of the batch,

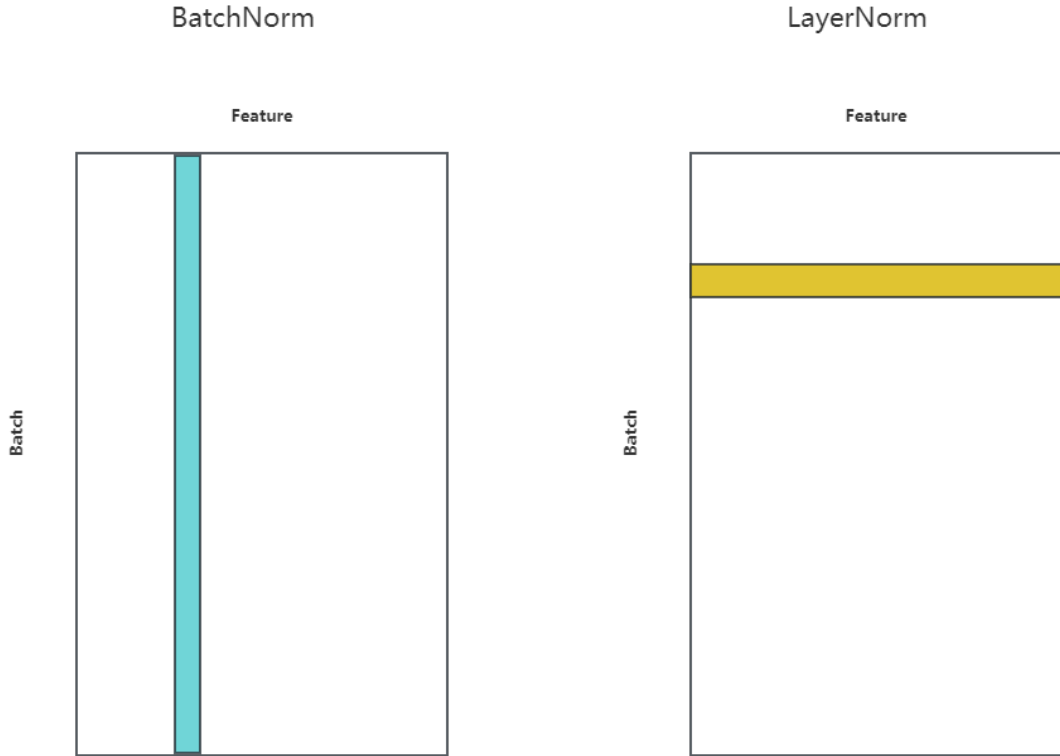


Figure 3.1 BatchNorm (left) will normalize each feature along the Batch dimension, while LayerNorm (right) will normalize each sample along the Feature dimension.

- ϵ is a small constant added for numerical stability, and
- $\hat{x}^{(k)}$ is the normalized k^{th} feature.

After normalization, the features are then scaled and shifted:

$$y^{(k)} = \gamma^{(k)} \hat{x}^{(k)} + \beta^{(k)} \quad (3.2)$$

where:

- $y^{(k)}$ is the output of the Batch Normalization layer for the k^{th} feature,
- $\gamma^{(k)}$ and $\beta^{(k)}$ are parameters learned during training to scale and shift the normalized feature $\hat{x}^{(k)}$ respectively.

Layer normalization normalizes each sample along the feature dimension. Given an input vector $x \in \mathbb{R}^n$ from a layer of the network, the Layer Normalization transform is defined as:

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \quad (3.3)$$

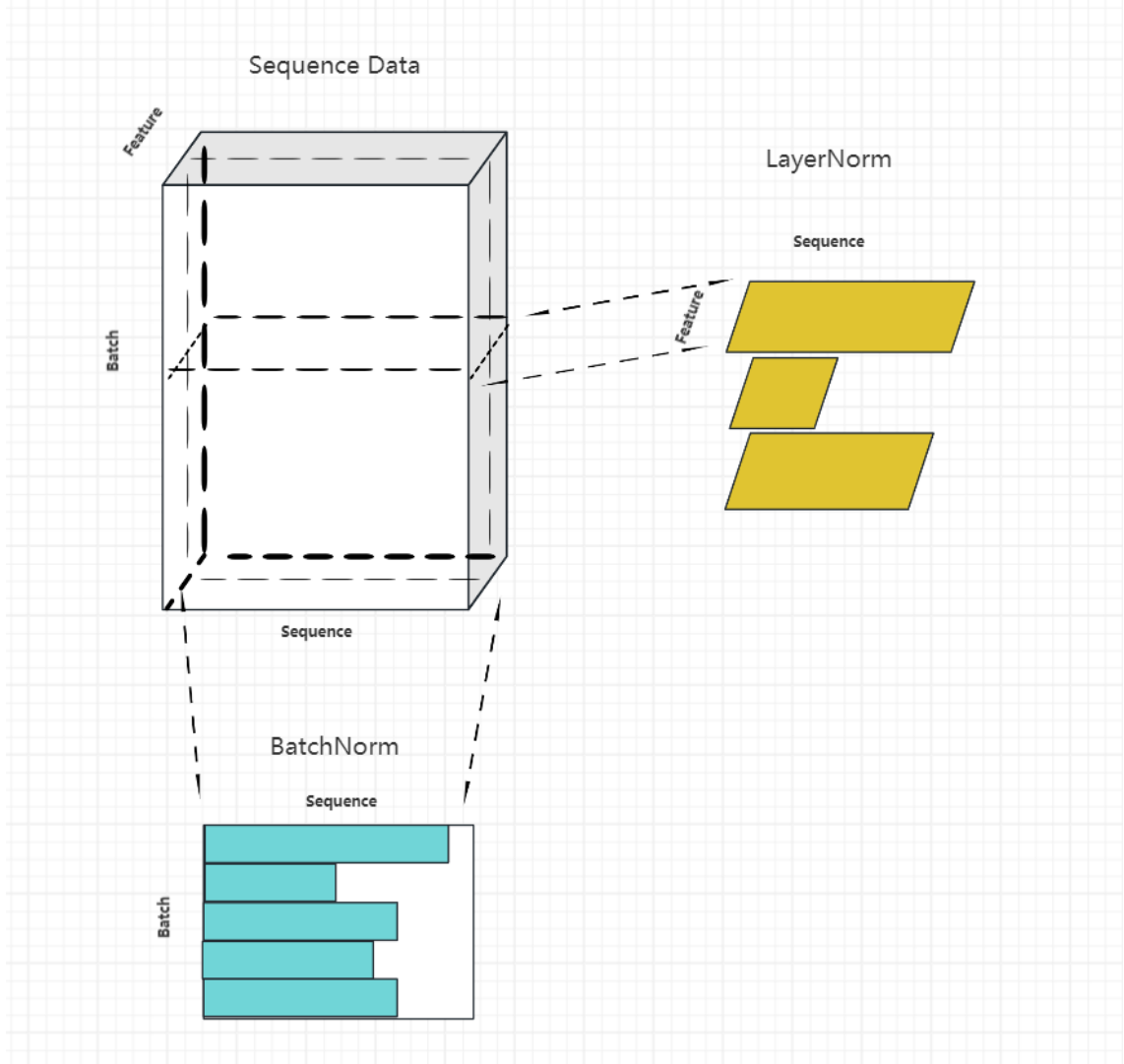


Figure 3.2 The top left picture shows the shape of the sequence data input. The top right picture shows process slicing of LayerNorm. The picture below shows the process slicing of BatchNorm.

where:

- x_i is the i^{th} element of input vector x ,
- $\mu = \frac{1}{n} \sum_{i=1}^n x_i$ is the mean of the elements in the input vector,
- $\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$ is the variance of the elements in the input vector,
- ϵ is a small constant added for numerical stability,
- \hat{x}_i is the normalized i^{th} element of the input vector.

After normalization, the elements are then scaled and shifted:

$$y_i = \gamma \hat{x}_i + \beta \tag{3.4}$$

where:

- y_i is the output of the Layer Normalization for the i^{th} element,
- γ and β are parameters learned during training to scale and shift the normalized element \hat{x}_i respectively.

In the case of sequence data, the input becomes three-dimensional, represented as *batch * sequence * feature*, as shown in the figure 3.2. As sequence lengths can vary, zero-padding is typically applied during encoding. However, using Batch Normalization can still be affected by the differing sequence lengths, leading to significant fluctuations in feature means and variances. Additionally, global means and variances are recorded during inference, which may not perform well with longer sequences if encountered. On the other hand, Layer Normalization is not influenced by varying sequence lengths because it is normalized for each sample.

The parameter *eps* prevents division by zero during normalization. Setting *elementwise_affine = True* indicates that an affine transformation will be applied to each element after normalization. Precisely, each normalized feature will be scaled by a learnable scaling parameter (γ) and shifted by a learnable bias parameter (β). Both parameters have a shape of (768,), matching the last dimension of the input data.

3.3 Dropout

Dropout is a regularization technique primarily used to prevent overfitting by randomly dropping out some neurons during training (Srivastava et al. 2014). With $p = 0.1$, a dropout probability of 0.1 means that each neuron has a 10% probability of being dropped out during forward propagation, thereby not participating in subsequent calculations and backpropagation. The parameter *inplace = False* indicates that the input data is not modified. Dropout can reduce the complexity of the model and prevent it from overly relying on specific local neuron features, thereby enhancing the model's generalization ability. During testing, setting p to 0 allows the entire network to be utilized for prediction.

3.4 BertEncoder

The BertEncoder section comprises 12 attention layers. "Linear" implies that the attention mechanism is processed using a linear layer. A linear layer applies a linear transformation to the input, the most basic neural network structure. When:

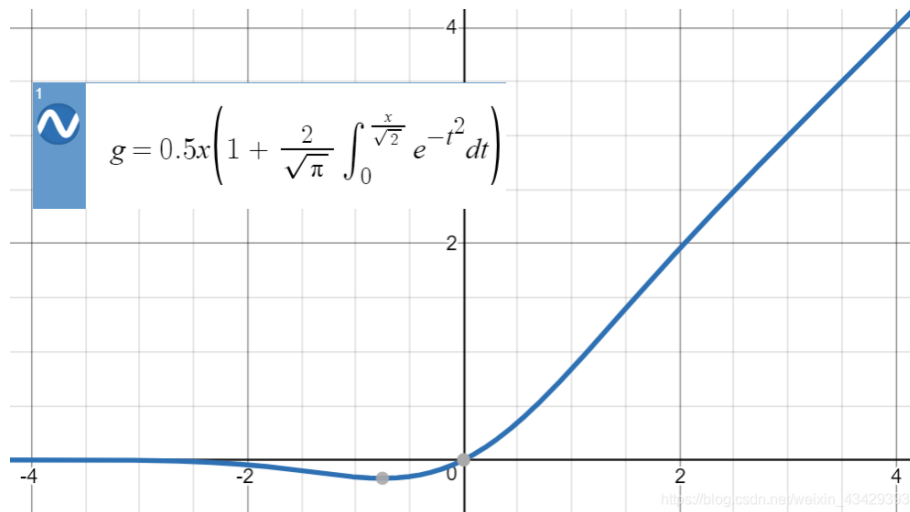


Figure 3.3 GELU Activation function figure

- \mathbf{x} is the input vector,
- \mathbf{W} is the weight matrix,
- \mathbf{b} is the bias vector,
- \mathbf{y} is the output vector.

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \quad (3.5)$$

- $in_features = 768$: This indicates the number or dimensionality of input features. This layer expects each input sample to be a vector of length 768.
- $out_features = 768$: This specifies the number or dimensionality of output features. The layer transforms the input vector into a new output vector of length 768.
- $bias = True$: This parameter indicates whether a bias vector is added after the linear transformation.

3.5 Activation function

The $GELUActivation()$ activation function allows almost all signals to be passed for large positive input values, while only a portion of signals is passed for negative input values, the figure of the function illustrates in figure 3.3 (Hendrycks and Gimpel 2016). Introducing non-linearity and the ability to transmit partial negative signals enhances the model's capability to capture data complexity.

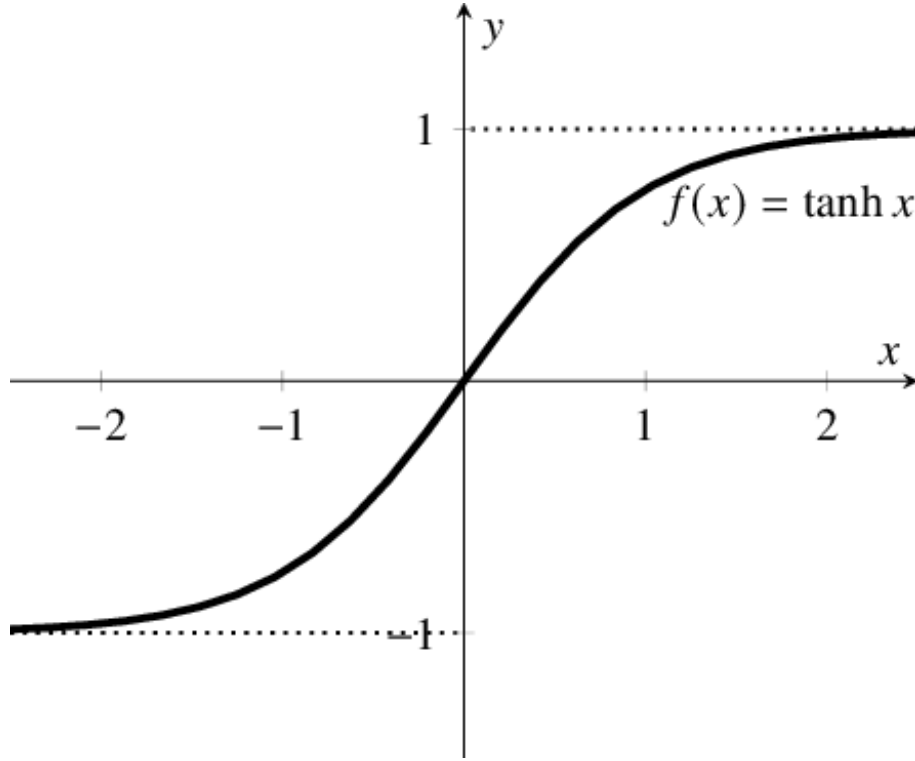


Figure 3.4 Tanh function figure

The Gaussian Error Linear Unit (GELU) activation function is defined as:

$$\text{GELU}(x) = x\Phi(x) \quad (3.6)$$

where $\Phi(x)$ is the cumulative distribution function (CDF) of the standard normal distribution, expressed as:

$$\Phi(x) = \frac{1}{2} \left[1 + \text{erf} \left(\frac{x}{\sqrt{2}} \right) \right] \quad (3.7)$$

The $\text{Tanh}()$ activation function outputs values between -1 and 1, with a mean close to 0, aiding numerical stability during the gradient descent process, function figure on figure 3.4.

The hyperbolic tangent function, or tanh , is defined as:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3.8)$$

Moreover, its derivative lies between 0 and 1, helping to control the issue of gradient explosion.

$$\frac{d}{dx} \tanh(x) = 1 - \tanh^2(x) \quad (3.9)$$

3.6 Softmax function and AdamW optimizer

In addition to the main structure of BERT, addressing classification tasks requires adding a classification function at the output layer. As mentioned earlier in the discussion of attention mechanisms, the *Softmax()* function is a suitable multi-category classification function. It converts outputs into a probability distribution, where the probabilities of the sample belonging to each class range from 0 to 1, with their sum equaling 1. Coupled with the cross-entropy loss function, this enables the model to learn and optimize effectively, making the probability distribution output by the model as close as possible to the real label distribution.

Given a vector of logits z for a multi-class classification problem with K classes, the predicted probability \hat{y}_i for class i using the Softmax function is:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (3.10)$$

The cross-entropy loss for the true label distribution y and the predicted probability distribution \hat{y} is then given by:

$$L(y, \hat{y}) = - \sum_{c=1}^K y_c \log(\hat{y}_c) \quad (3.11)$$

where y_c is an indicator variable equal to 1 if the sample belongs to class c and 0 otherwise.

The *AdamW* optimizer is a method for optimizing model parameters. It separates weight decay and gradient updates, effectively controlling model complexity and preventing overfitting (Loshchilov and Hutter 2017). The update rules for the AdamW optimizer are as follows:

$$\begin{aligned} m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \\ v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \\ \hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot (\hat{m}_t + \lambda \cdot \theta_t) \end{aligned}$$

where:

- t is the current iteration step;
- θ_t is the parameter vector;

- g_t is the gradient at the current iteration step;
- m_t is the momentum estimate vector;
- v_t is the exponential moving average of squared gradients;
- β_1 and β_2 are decay rates for the exponential moving averages;
- η is the learning rate;
- ϵ is a small value to prevent division by zero;
- λ is the weight decay term.

Here, a weight decay term $\lambda \cdot \theta_t$ is introduced, where λ is a hyperparameter controlling the strength of weight decay.

4 Experiments

The device configuration includes a CPU(central processing unit): Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, 16GB RAM, and an NVIDIA GeForce GTX 1060 GPU(graphics processing unit) with 6GB of memory.

Overall, three experiments were conducted to fine-tune BERT and compare the results. The first set involves fine-tuning all parameters. Drawing inspiration from some transfer learning experiments in image classification. Training only the classification head can yield satisfactory results if a model has thoroughly learned the features of image data. The second set fine-tunes only the classification head of the BERT model. The third set employs the LoRA method for fine-tuning.

4.1 Dataset for fine-tuning models

The dataset utilized in this thesis is collected from Katsarou et al. 2021. It comprises tweets from four topics on the X platform: #Coronavirus, #ClimateChange, #Immigrants, and #MeToo. The tweets are categorized into five classes: Hate Speech, Offensive, Sexism, Neutral, and Positive. Due to hardware limitations and to prevent potential biases arising from skewed data, only 30% of neutral texts were sampled for experimentation, ensuring that neutral texts still maintained a majority. The figure 4.1 illustrates the specific distribution. The entire dataset was divided into training, validation, and test sets in a ratio of 8:1:1.

4.2 BERT model and related component settings

The pre-trained BERT model used in the experiment is sourced from the open-source community **HuggingFace**, specifically the **bert-base-cased** model. It comprises 12 transformer blocks, a hidden size 768, 12 self-attention heads, and 110 million parameters, and the model was pre-trained on case English text.

The maximum input sequence length in the experiment is 160, and the AdamW optimizer(Loshchilov and Hutter 2017) is used to prevent overfitting the initial learning rate is $3e-5$. During training, employ accuracy as the metric. The LoRA method is configured with a matrix rank $r=8$, an alpha value of 16 (used for scaling ΔW with $alpha/r$), and a LoRA dropout rate of 1%.

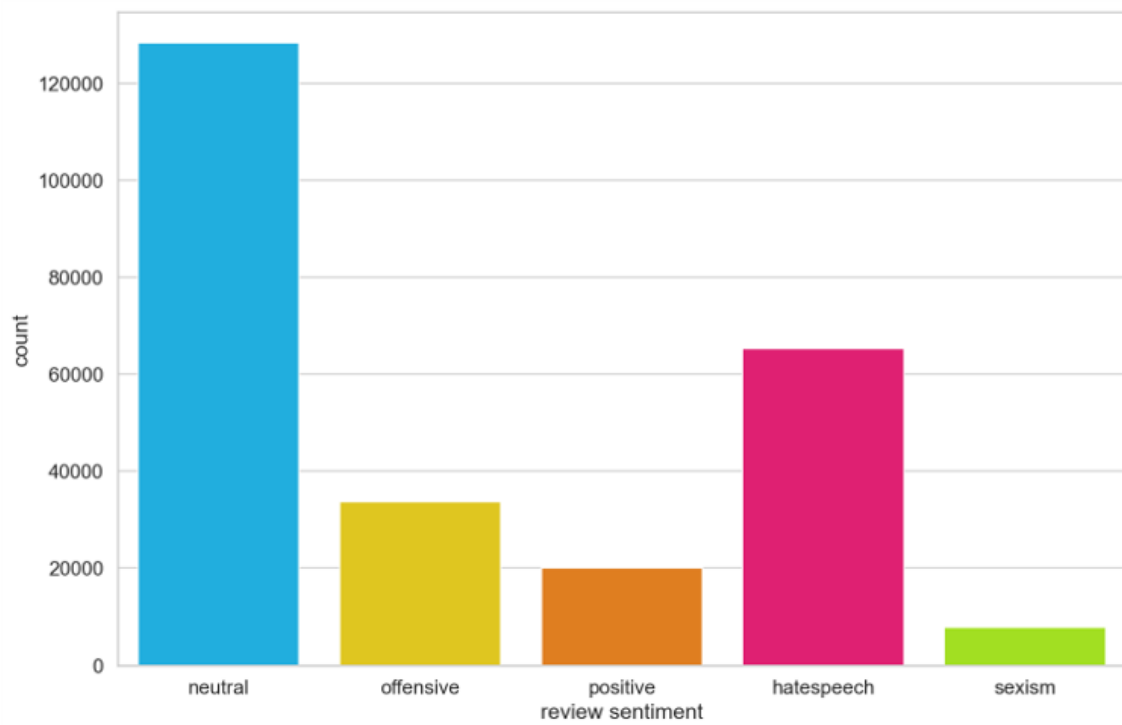


Figure 4.1 Distribution of different categories of text in the data set

5 Results

5.1 Fine-tuning results analysis

This section will show the model’s training and testing results using three fine-tuning methods: Fine-tuning BERT with all parameters, Fine-tuning BERT only the classification heads, and Fine-tuning BERT with LoRA.

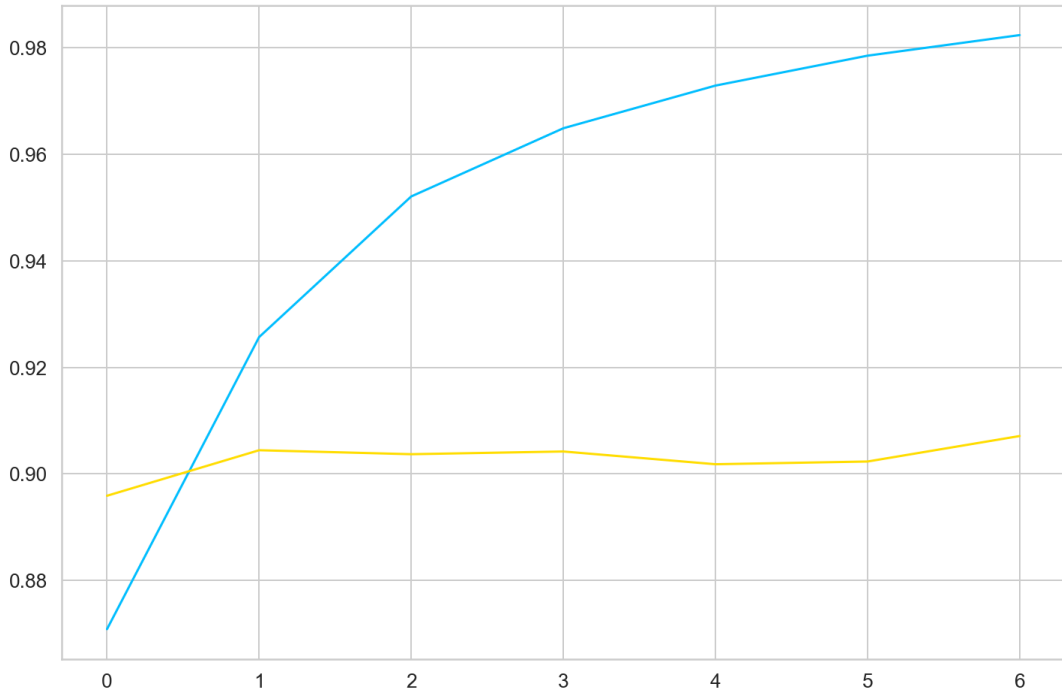
Overall, the table 5.1 presents the metrics for the classification results on the test set, and it compares the results of three models: Fine-tuning BERT with all parameters, Fine-tuning BERT only the classification heads, and Fine-tuning BERT with LoRA, denoted as ‘BERT/hBERT/LoRA’. These results once again demonstrate the superior performance of the BERT model. Moreover, based on the results, the full-parameter fine-tuning method demonstrates overall superiority over the other two approaches, with the LoRA method closely followed. The approach that only fine-tuning the classification head shows poorer results. Although the head-only fine-tuning method achieves greater classification precision in identifying sexism-related texts compared to full-parameter fine-tuning, its recall rate is considerably lower. This suggests that the model remains influenced by the distribution of data samples and has not effectively learned the features of texts related to sexism. Following is the specific result analysis of each model.

Table 5.1 Comparison of statistical metrics of classification results, the order of numerical values following ‘BERT/hBERT/LoRA’ corresponding Fine-tuning BERT with all parameters, Fine-tuning BERT only the classification heads and Fine-tuning BERT with LoRA

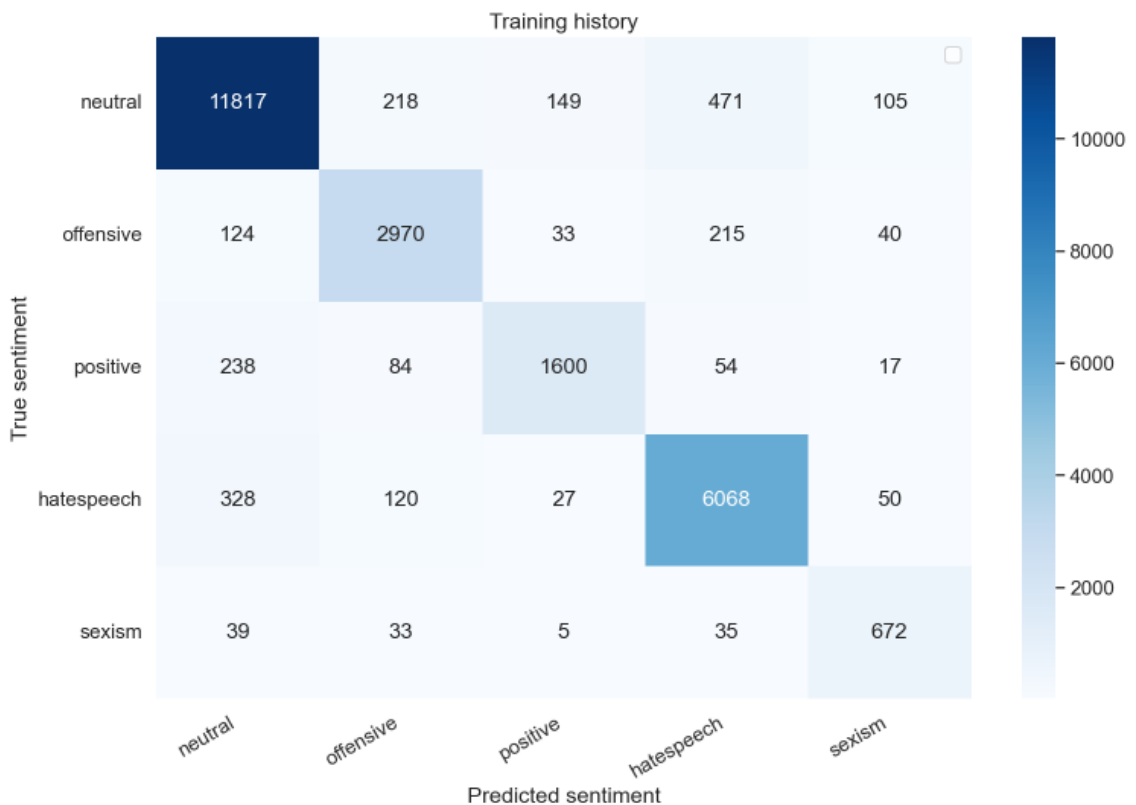
BERT/hBERT/LoRA	precision	recall	f1-score	support
neutral	0.94 /0.67/0.89	0.93 /0.87/0.87	0.93 /0.76/0.88	12760
offensive	0.87 /0.55/0.79	0.88 /0.42/0.78	0.87 /0.47/0.79	3382
positive	0.88 /0.63/0.77	0.80 /0.51/0.77	0.84 /0.57/0.77	1993
hatespeech	0.89 /0.62/0.79	0.92 /0.44/0.85	0.90 /0.51/0.82	6593
sexism	0.76/ 0.88 /0.66	0.86 /0.03/0.68	0.81 /0.06/0.67	784

Fine-tuning BERT with all parameters

BERT was trained for seven epochs, taking 16.5 hours, and the accuracy curves for the training and validation sets are depicted in the figure 5.1(a). It is evident that both the training and validation accuracies started at a high level. The blue line representing the training set shows rapid accuracy growth in the initial five epochs,



(a) The epoche-accuracy curve during Fine-tuning BERT with all parameters, blue line from the training set, yellow line from validation set.



(b) The confusion matrix for the classification results on test set of Fine-tuning BERT with all parameters

Figure 5.1 Fine-tuning BERT with all parameters

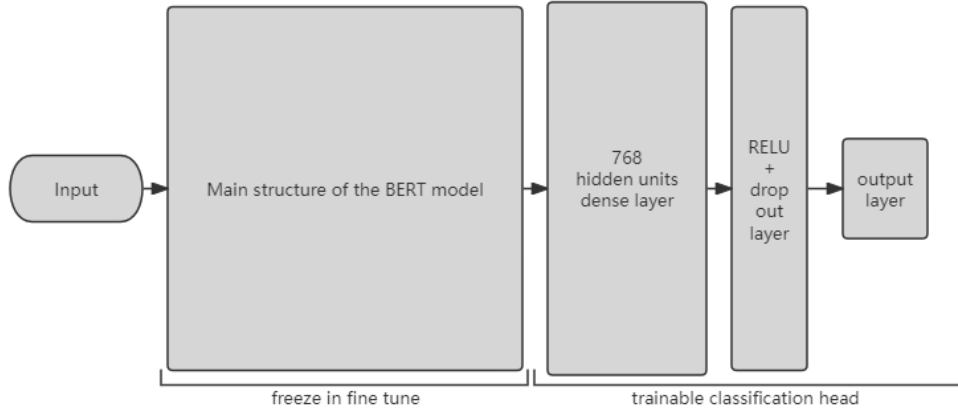


Figure 5.2 *Fine-tuning BERT only the classification heads diagram*

increasing by 10% and then stabilizing. The validation set accuracy fluctuates modestly, hovering around 90%.

According to figure 5.1(a), two straightforward inferences can be drawn. Firstly, the performance of pre-trained BERT is outstanding, and simple fine-tuning yields satisfactory results. Secondly, employing a larger-scale fine-tuning dataset may lead to even better outcomes.

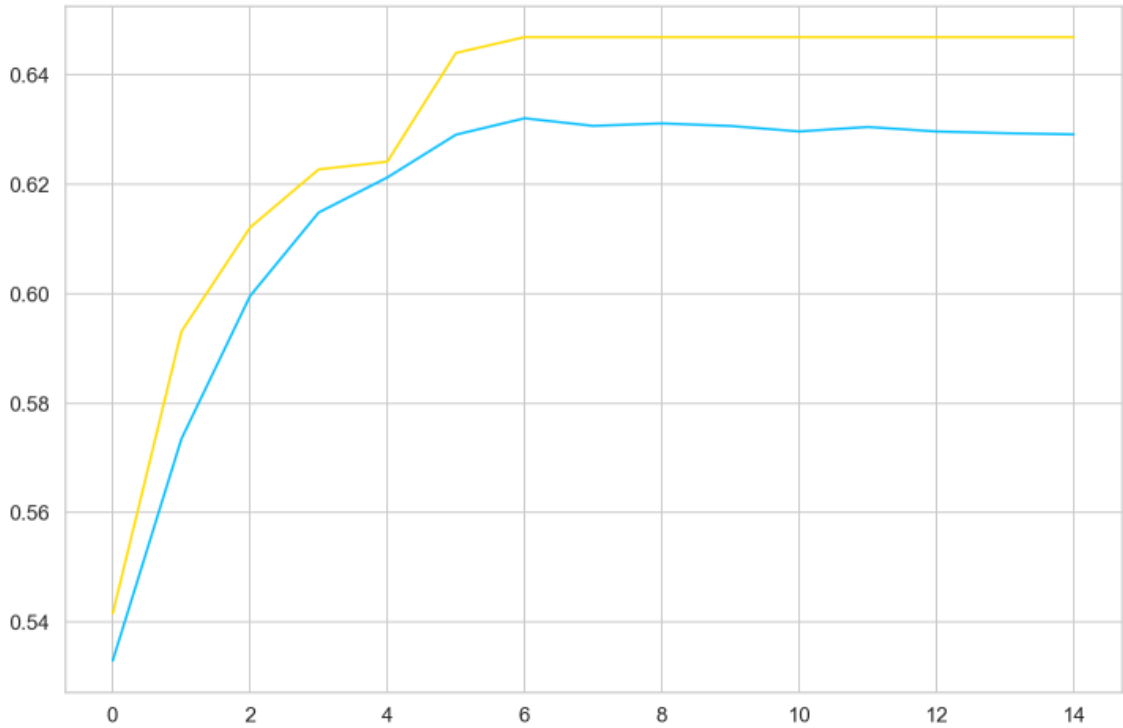
Finally, the fine-tuned model achieved an accuracy of 91% on the test set, and the confusion matrix for the classification results is shown in the figure 5.1(b). The highlighted portions of the confusion matrix are concentrated on the diagonal, indicating excellent classification performance.

Fine-tuning BERT only the classification heads

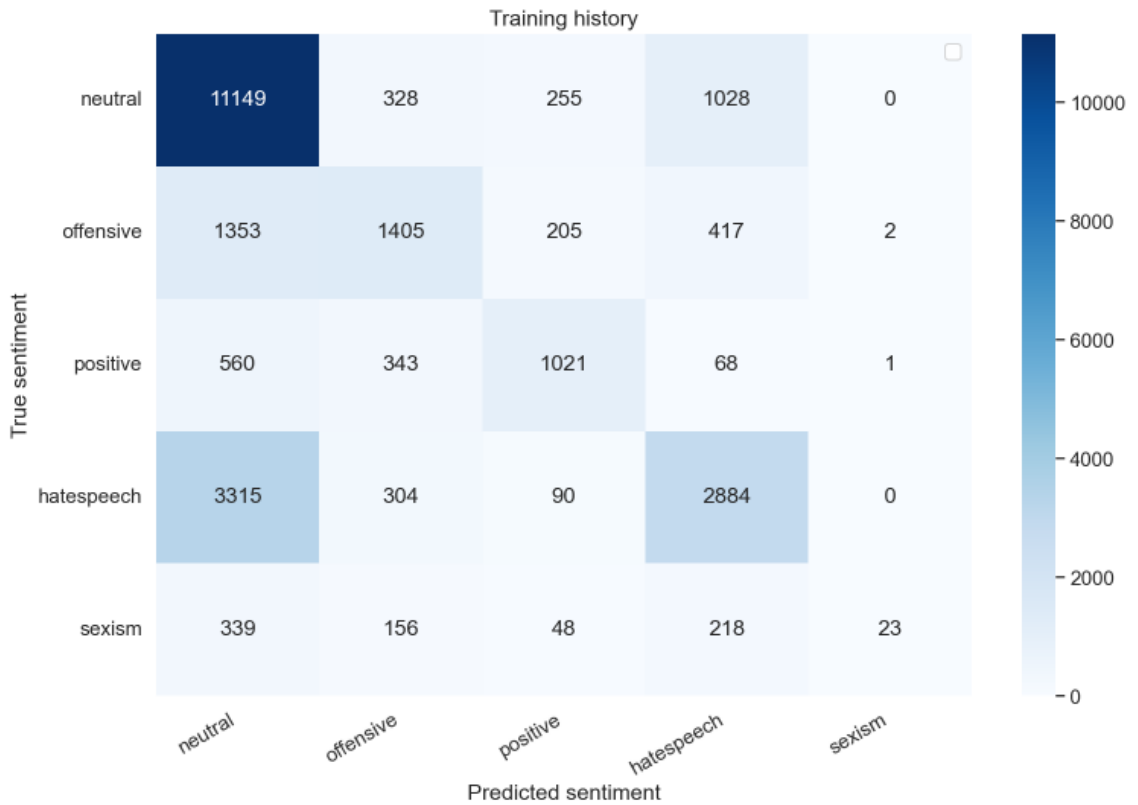
This training session takes around 15 hours for 15 epochs. Since only the classification heads were trained, as illustrated in the figure 5.2, this model only trained 0.55% of all the parameters. From the epoch-accuracy curves (figure 5.3(a)), the growth trends of the two curves are essentially consistent due to the small number of trained parameters. However, the results are not satisfactory; the accuracy on the validation set converges around 63%, and on the test set, the accuracy reaches 65%. The highlighted portions in the confusion matrix, from figure 5.3(b), also appear scattered, indicating a confusing distinction between neutral and hate speech.

Fine-tuning BERT with LoRA

The LoRA method from the HuggingFace community was employed to fine-tune the BERT model, training only 0.14% of the parameters. The training lasted for ten epochs and took 14 hours. Ultimately, this model achieved an accuracy of 84%



(a) The epoche-accuracy curve during Fine-tuning BERT only the classification heads, blue line from the training set, yellow line from validation set.



(b) The confusion matrix for the classification results on test set of Fine-tuning BERT only the classification head.

Figure 5.3 Fine-tuning BERT only the classification head

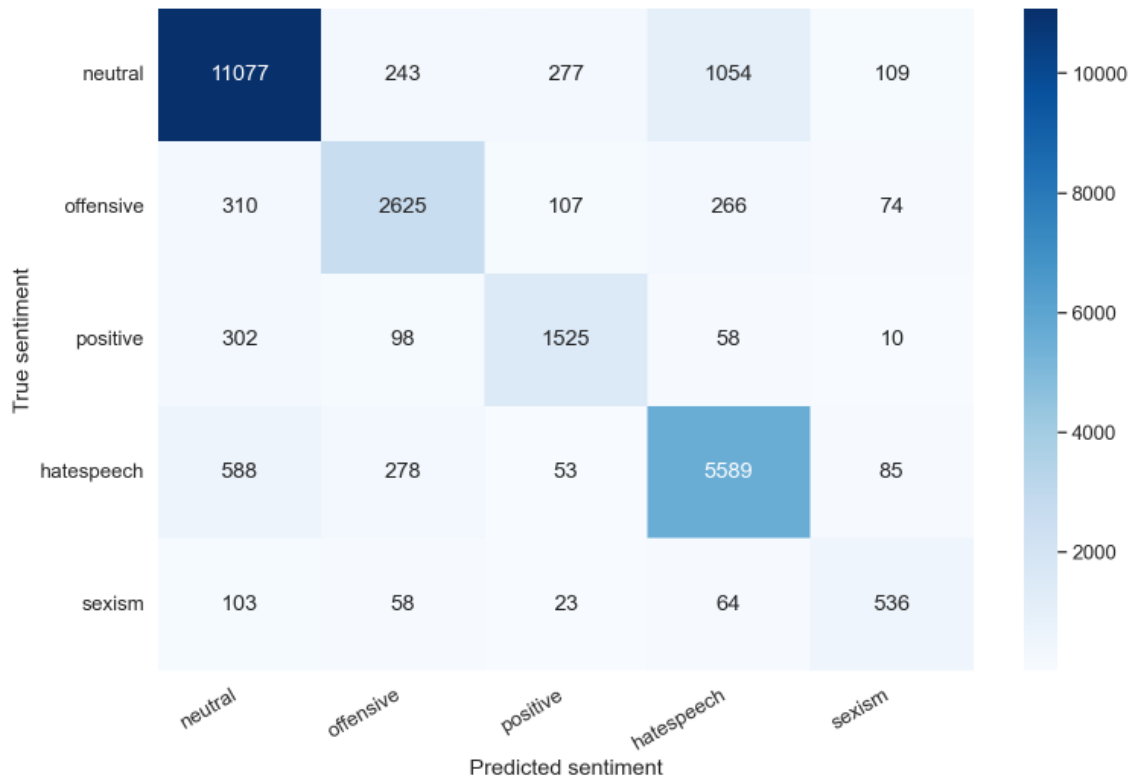
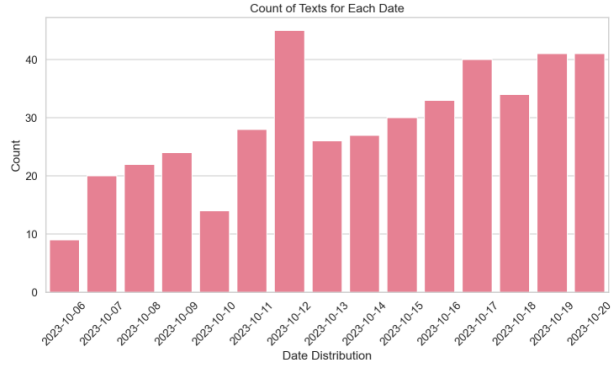


Figure 5.4 The confusion matrix for the classification results on test set of Fine-tuning BERT with LoRA.

on the test set, and the confusion matrix is depicted in the figure 5.4. The classification results are notable, with highlighted portions primarily distributed along the diagonal. However, there is still a tendency to confuse neutral and hate speech. From the metrics, the classification results seem to improve with a larger number of samples for each category, making the overall outcome satisfactory. Nevertheless, this result is surprising considering the proportion of trained parameters, with only a 7% accuracy loss using 0.14% of the parameters.

5.2 Full-parameter fine-tuning BERT and LoRA-BERT application analysis and comparison

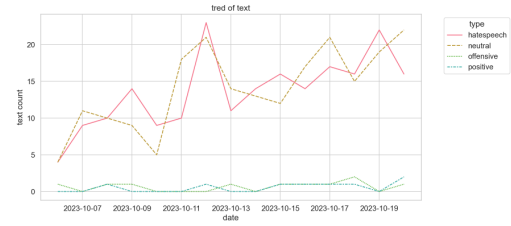
This thesis applied the fully fine-tuned BERT model and the LoRA-Bert model to external datasets to assess the effectiveness of hate speech analysis in real-world scenarios. The dataset is from Lab 2023, publicly available on Harvard Dataverse. The research in Lab 2023 aimed to investigate harmful and misleading posts related to 'Israel-Hamas', 'Ukraine', and 'vaccines' on the X platform from October 7 to October 20, 2023; this includes lexicon-based sentiment scoring for each post text. The provided high-engagement tweets with a timeline serve as suitable research material



(a) Israel-Hamas text data statistics by date



(b) Israel-Hamas dataset classified by Full parameter fine-tuning BERT



(c) Israel-Hamas dataset classified by LoRA BERT

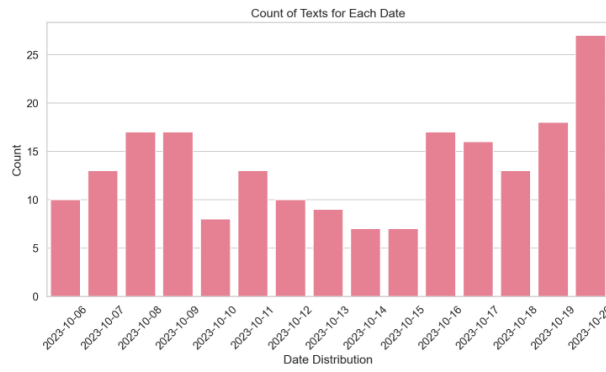
Figure 5.5 Israel-Hamas dataset analysis

for this study’s hate speech sentiment analysis. We can further observe and analyze specific topics’ trends through more refined classification statistics. Subsequently, the application results will be discussed solely from a data analysis perspective.

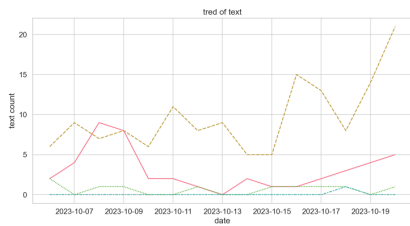
Israel-Hamas

Firstly, from the statistical analysis of the number of unclassified data entries, figure5.5(a), it can be observed that the event’s popularity on the X platform is steadily increasing. Moreover, discussions surged on October 12, 2023. The trends in classification further corroborate this observation. Trend analyses of both models, figure5.5(b) and figure5.5(c), reveal that the text is predominantly concentrated in two categories: neutral and hate speech. The quantity of discussions in both categories continues to rise. The surge in hate speech tweets, particularly on October 12, suggests a negative event for tweet publishers¹. Notably, LoRA-Bert’s inclination to classify text as hate speech is evident, with the red line of hate speech intertwining with the yellow dashed line of neutral texts. In contrast, the red line in the fully fine-tuned BERT model is mainly below the yellow dashed line.

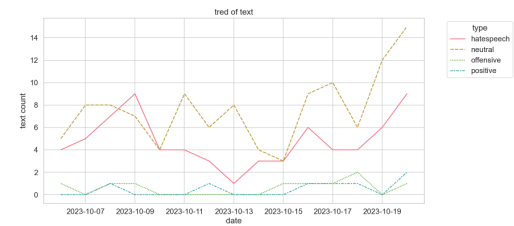
¹The data is cited from Lab 2023, and specific events can be found in the contents of the dataset. The data is used only for model classification and analysis of the distribution of classification results. The analysis presented here and thereafter do not represent any author’s realistic standpoint.



(a) Ukraine text data statistics by date

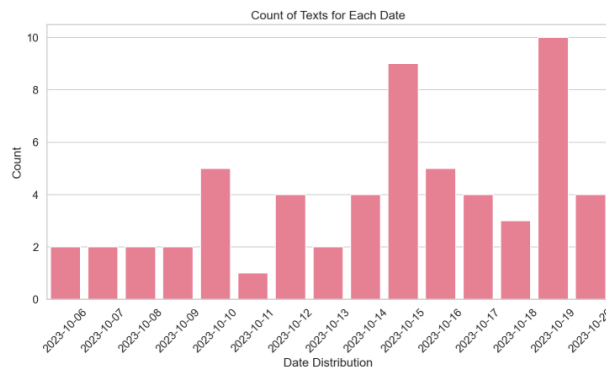


(b) Ukraine dataset classified by Full parameter fine-tuning BERT

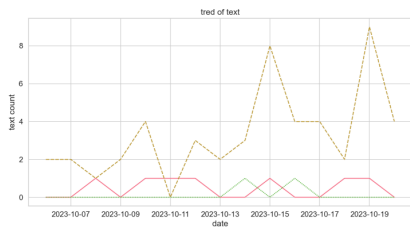


(c) Ukraine dataset classified by LoRA BERT

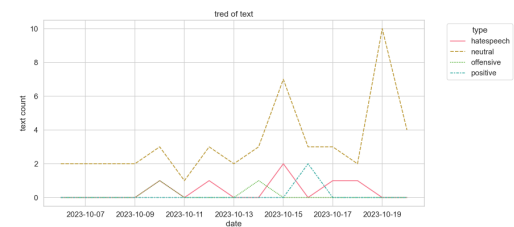
Figure 5.6 Ukraine dataset analysis



(a) Vaccines text data statistics by date



(b) Vaccines dataset classified by Full parameter fine-tuning BERT



(c) Vaccines dataset classified by LoRA BERT

Figure 5.7 Vaccines dataset analysis

Ukraine

By observing figure 5.6(a), discussions on this topic on the X platform seem to concentrate on October 8, October 16, and October 20. The two models exhibit a consistent pattern in the trend analysis of neutral discourse, which constitutes the majority of the dataset. However, regarding the trend in hate speech, the fully fine-tuned BERT model shows a noticeable peak around October 8 (from figure 5.6(b)), followed by a rapid decline and then a stable increase. In contrast, the results from the LoRABERT model display more oscillations and intense variations, as exhibited in figure 5.6(c). Interestingly, on October 13, there was a significant decrease in the proportion of hate speech, accompanied by a slight increase in neutral discussions.

Vaccines

Since the dataset from the data source includes retweeted tweets, this portion of the data spans a longer time frame and is more temporally discrete. Therefore, only data from October 6 to October 20 was extracted for analysis. There are two noticeable peaks on October 15 and October 19 in figure 5.7(a). From a classification perspective, neutral discourse dominates in quantity in both models (figure 5.7(b) and 5.7(c)), while hate speech and other categories exhibit relatively low and fluctuating numbers.

Summary of patterns

The topics covered by these three datasets, including war and disease, tend to be relatively negative or controversial. Considering the timing of these events and the data collection period, combined with the analysis results, some simple patterns can be discerned. At the onset of an adverse event, such as the new round of Palestinian-Israeli conflict behind the Israel-Hamas dataset, there is a continuous increase in user attention on social platforms, accompanied by a more frequent occurrence of hate speech. During relatively slow development in events like the Russo-Ukrainian War, discussions tend to be rational or emotional following news reporting progresses. Moreover, after an event is basically over, discussions tend to be more neutral.

These data-supported analyses and patterns can assist relevant organizations in better anticipating and preventing the harm caused to innocent individuals by harmful dissemination on social media. Furthermore, tailored analytical features can be designed based on specific needs to aid efforts in combating criminal activities and similar endeavors.

6 Conclusion and future work

Overall, this thesis’s hate speech analysis method offers a more refined and flexible macroscopic analysis compared to traditional sentiment classification approaches. It avoids the labor-intensive process of analyzing each item individually and yields more valuable information for text classification.

On the methodological front, this paper delves into the application of fine-tuning BERT in sentiment analysis. The excellent results of fine-tuning demonstrate the robust performance of BERT in hate speech analysis.

Furthermore, this thesis compares three fine-tuning methods: full parameter fine-tuning, fine-tuning only the classification head, and the PEFT approach. The results demonstrate that the full-parameter fine-tuning method, although resource-intensive, still yields the best text classification performance in hate speech analysis, while fine-tuning only the classification head is unsuitable for this scenario. Additionally, the study affirms that considering the PEFT approach for fine-tuning large models is feasible and yields practical results (the results are close to the full parameter fine-tuning method), particularly in resource-constrained scenarios.

In practice and during the writing process, this paper thesis has prompted some reflections on future learning endeavors.

Firstly, concerning the PEFT method, a universally proven optimal approach is currently lacking. However, it is foreseeable that the PEFT method will continue to capture the attention of researchers as transformer models become increasingly prevalent. The outstanding cost-effectiveness it offers undoubtedly positions it as a preferred choice for more practitioners.

Secondly, during the fine-tuning of BERT, it is notable that even when using a tiny proportion of trainable parameters, the time required for the final model to converge remains comparable. On the one hand, this phenomenon is attributed to the fact that using fewer parameters inherently demands more epochs for fine-tuning to convergence. On the other hand, some studies suggest that the structure of BERT itself influences it. Consequently, diverse research has focused on refining BERT, presenting an intriguing avenue for exploration.

Lastly, concerning the dataset, if we liken research and applications in machine learning and deep learning to building a house, data is the foundation. The quality and quantity of data often determine the size and performance of the model. Especially when considering the practical application of machine learning technologies, one often finds that the datasets used in experiments may not meet the requirements of a production environment. Re-annotating a large dataset is a costly work. Various approaches have been developed to address this issue. For instance, models

like BERT employ unsupervised learning for pre-training, followed by designing a smaller fine-tuning dataset for the task. Additionally, semi-supervised and active learning methods are worth continued attention and exploration.

Another point worth noting, which stems from the writing process of the practical section 5.2, is about the inference drawn from the characteristics of the data. Many events and news could be linked to data. While these events and news are not difficult to find, considering that referencing itself may introduce subjective bias, the discussion in the analysis section eschews corroborating the analysis with real events. Furthermore, as a technology user, it serves as a reminder to the author to maintain calm and restraint when using and discussing technology. This is essential to prevent the misuse of technology and ensure that technology serves humanity better.

References

- Agatonovic-Kustrin, S and R Beresford (June 1, 2000). “Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research”. In: *Journal of Pharmaceutical and Biomedical Analysis* 22.5, pp. 717–727. ISSN: 0731-7085. DOI: 10.1016/S0731-7085(99)00272-1. URL: <https://www.sciencedirect.com/science/article/pii/S0731708599002721> (visited on 02/16/2024).
- Almeida, Felipe and Geraldo Xexéo (May 1, 2023). *Word Embeddings: A Survey*. DOI: 10.48550/arXiv.1901.09069. arXiv: 1901.09069[cs,stat]. URL: <http://arxiv.org/abs/1901.09069> (visited on 02/20/2024).
- Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton (July 21, 2016). *Layer Normalization*. DOI: 10.48550/arXiv.1607.06450. arXiv: 1607.06450[cs,stat]. URL: <http://arxiv.org/abs/1607.06450> (visited on 02/20/2024).
- Birjali, Marouane, Mohammed Kasri, and Abderrahim Beni-Hssane (Aug. 17, 2021). “A comprehensive survey on sentiment analysis: Approaches, challenges and trends”. In: *Knowledge-Based Systems* 226, p. 107134. ISSN: 0950-7051. DOI: 10.1016/j.knsys.2021.107134. URL: <https://www.sciencedirect.com/science/article/pii/S095070512100397X> (visited on 01/30/2024).
- Breiman, Leo (Oct. 1, 2001). “Random Forests”. In: *Machine Learning* 45.1, pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. URL: <https://doi.org/10.1023/A:1010933404324> (visited on 02/16/2024).
- Brown, Tom B. et al. (July 22, 2020). *Language Models are Few-Shot Learners*. DOI: 10.48550/arXiv.2005.14165. arXiv: 2005.14165[cs]. URL: <http://arxiv.org/abs/2005.14165> (visited on 02/22/2024).
- Chakraborty, Koyel, Siddhartha Bhattacharyya, and Rajib Bag (Apr. 2020). “A Survey of Sentiment Analysis from Social Media Data”. In: *IEEE Transactions on Computational Social Systems* 7.2. Conference Name: IEEE Transactions on Computational Social Systems, pp. 450–464. ISSN: 2329-924X. DOI: 10.1109/TCSS.2019.2956957. URL: <https://ieeexplore.ieee.org/abstract/document/8951256> (visited on 01/30/2024).
- Chowdhury, Gobinda G. (Jan. 31, 2003). “Natural language processing”. In: *Annual Review of Information Science and Technology* 37.1, pp. 51–89. ISSN: 0066-4200. DOI: 10.1002/aris.1440370103.
- Chung, Junyoung et al. (Dec. 11, 2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. DOI: 10.48550/arXiv.1412.3555. arXiv: 1412.3555[cs]. URL: <http://arxiv.org/abs/1412.3555> (visited on 02/16/2024).

- Devlin, Jacob et al. (May 24, 2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. DOI: 10.48550/arXiv.1810.04805. arXiv: 1810.04805[cs]. URL: <http://arxiv.org/abs/1810.04805> (visited on 01/30/2024).
- Dosovitskiy, Alexey et al. (June 3, 2021). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. DOI: 10.48550/arXiv.2010.11929. arXiv: 2010.11929[cs]. URL: <http://arxiv.org/abs/2010.11929> (visited on 02/21/2024).
- Gongane, Vaishali U., Mousami V. Munot, and Alwin D. Anuse (Sept. 5, 2022). “Detection and moderation of detrimental content on social media platforms: current status and future directions”. In: *Social Network Analysis and Mining* 12.1, p. 129. ISSN: 1869-5469. DOI: 10.1007/s13278-022-00951-3. URL: <https://doi.org/10.1007/s13278-022-00951-3> (visited on 02/13/2024).
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, pp. 180–184.
- He, Junxian et al. (Feb. 2, 2022). *Towards a Unified View of Parameter-Efficient Transfer Learning*. DOI: 10.48550/arXiv.2110.04366. arXiv: 2110.04366[cs]. URL: <http://arxiv.org/abs/2110.04366> (visited on 01/30/2024).
- He, Kaiming et al. (Dec. 10, 2015). *Deep Residual Learning for Image Recognition*. DOI: 10.48550/arXiv.1512.03385. arXiv: 1512.03385[cs]. URL: <http://arxiv.org/abs/1512.03385> (visited on 02/26/2024).
- Hearst, M.A. et al. (1998). “Support vector machines”. In: *IEEE Intelligent Systems and their Applications* 13.4, pp. 18–28. DOI: 10.1109/5254.708428.
- Heckerman, David (2008). “A Tutorial on Learning with Bayesian Networks”. In: *Innovations in Bayesian Networks: Theory and Applications*. Ed. by Dawn E. Holmes and Lakhmi C. Jain. Studies in Computational Intelligence. Berlin, Heidelberg: Springer, pp. 33–82. ISBN: 978-3-540-85066-3. DOI: 10.1007/978-3-540-85066-3_3. URL: https://doi.org/10.1007/978-3-540-85066-3_3 (visited on 02/16/2024).
- Hendrycks, Dan and Kevin Gimpel (2016). “Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units”. In: *CoRR* abs/1606.08415. arXiv: 1606.08415. URL: <http://arxiv.org/abs/1606.08415>.
- Houlsby, Neil et al. (June 13, 2019). *Parameter-Efficient Transfer Learning for NLP*. arXiv: 1902.00751[cs, stat]. URL: <http://arxiv.org/abs/1902.00751> (visited on 01/30/2024).
- Howard, Jeremy and Sebastian Ruder (2018). “Fine-tuned Language Models for Text Classification”. In: *CoRR* abs/1801.06146. arXiv: 1801.06146. URL: <http://arxiv.org/abs/1801.06146>.

- Hu, Edward J. et al. (Oct. 16, 2021). *LoRA: Low-Rank Adaptation of Large Language Models*. DOI: 10.48550/arXiv.2106.09685. arXiv: 2106.09685[cs]. URL: <http://arxiv.org/abs/2106.09685> (visited on 01/30/2024).
- Jurek, Anna, Maurice D. Mulvenna, and Yaxin Bi (Dec. 9, 2015). “Improved lexicon-based sentiment analysis for social media analytics”. In: *Security Informatics* 4.1, p. 9. ISSN: 2190-8532. DOI: 10.1186/s13388-015-0024-x. URL: <https://doi.org/10.1186/s13388-015-0024-x> (visited on 02/14/2024).
- Kaplan, Jared et al. (Jan. 22, 2020). *Scaling Laws for Neural Language Models*. DOI: 10.48550/arXiv.2001.08361. arXiv: 2001.08361[cs,stat]. URL: <http://arxiv.org/abs/2001.08361> (visited on 02/22/2024).
- Kapoor, Kawaljeet Kaur et al. (June 1, 2018). “Advances in Social Media Research: Past, Present and Future”. In: *Information Systems Frontiers* 20.3, pp. 531–558. ISSN: 1572-9419. DOI: 10.1007/s10796-017-9810-y. URL: <https://doi.org/10.1007/s10796-017-9810-y> (visited on 02/13/2024).
- Katsarou, Katerina et al. (Dec. 2021). “Sentiment Polarization in Online Social Networks: The Flow of Hate Speech”. In: *2021 Eighth International Conference on Social Network Analysis, Management and Security (SNAMS)*. 2021 Eighth International Conference on Social Network Analysis, Management and Security (SNAMS), pp. 01–08. DOI: 10.1109/SNAMS53716.2021.9732077. URL: <https://ieeexplore.ieee.org/document/9732077> (visited on 01/30/2024).
- Lab, InfoEpi (Oct. 24, 2023). “Negative and Misleading Posts Driving Critical Discussions on X”. In: *InfoEpi Lab*. Publisher: Information Epidemiology Lab. DOI: 10.7910/DVN/WRAQ4N. URL: <https://infoepi.org/posts/2023/10/24-israel-ukraine-vaccines.html> (visited on 03/02/2024).
- LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (May 2015). “Deep learning”. In: *Nature* 521.7553. Number: 7553 Publisher: Nature Publishing Group, pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: <https://www.nature.com/articles/nature14539> (visited on 02/16/2024).
- Li, Xiang Lisa and Percy Liang (Jan. 1, 2021). *Prefix-Tuning: Optimizing Continuous Prompts for Generation*. arXiv: 2101.00190[cs]. URL: <http://arxiv.org/abs/2101.00190> (visited on 01/30/2024).
- Lin, Bin et al. (Mar. 7, 2022). “Opinion Mining for Software Development: A Systematic Literature Review”. In: *ACM Transactions on Software Engineering and Methodology* 31.3, 38:1–38:41. ISSN: 1049-331X. DOI: 10.1145/3490388. URL: <https://doi.org/10.1145/3490388> (visited on 03/13/2024).
- Loshchilov, Ilya and Frank Hutter (2017). “Fixing Weight Decay Regularization in Adam”. In: *CoRR* abs/1711.05101. arXiv: 1711.05101. URL: <http://arxiv.org/abs/1711.05101>.

- Maas, Andrew L. et al. (June 2011). “Learning Word Vectors for Sentiment Analysis”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 142–150. URL: <http://www.aclweb.org/anthology/P11-1015>.
- Merity, Stephen, Nitish Shirish Keskar, and Richard Socher (2017). “Regularizing and Optimizing LSTM Language Models”. In: *CoRR* abs/1708.02182. arXiv: 1708.02182. URL: <http://arxiv.org/abs/1708.02182>.
- Miller, George A. et al. (Dec. 1, 1990). “Introduction to WordNet: An On-line Lexical Database*”. In: *International Journal of Lexicography* 3.4, pp. 235–244. ISSN: 0950-3846. DOI: 10.1093/ijl/3.4.235. URL: <https://doi.org/10.1093/ijl/3.4.235> (visited on 02/14/2024).
- Murtagh, Fionn (1991). “Multilayer perceptrons for classification and regression”. In: *Neurocomputing* 2.5, pp. 183–197. ISSN: 0925-2312. DOI: [https://doi.org/10.1016/0925-2312\(91\)90023-5](https://doi.org/10.1016/0925-2312(91)90023-5). URL: <https://www.sciencedirect.com/science/article/pii/0925231291900235>.
- Niu, Zhaoyang, Guoqiang Zhong, and Hui Yu (Sept. 10, 2021). “A review on the attention mechanism of deep learning”. In: *Neurocomputing* 452, pp. 48–62. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2021.03.091. URL: <https://www.sciencedirect.com/science/article/pii/S092523122100477X> (visited on 02/21/2024).
- Pan, Sinno Jialin and Qiang Yang (2010). “A Survey on Transfer Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10, pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.
- Quinlan, J. R. (Mar. 1, 1986). “Induction of decision trees”. In: *Machine Learning* 1.1, pp. 81–106. ISSN: 1573-0565. DOI: 10.1007/BF00116251. URL: <https://doi.org/10.1007/BF00116251> (visited on 02/16/2024).
- Radford, Alec et al. (Feb. 26, 2021). *Learning Transferable Visual Models From Natural Language Supervision*. DOI: 10.48550/arXiv.2103.00020. arXiv: 2103.00020[cs]. URL: <http://arxiv.org/abs/2103.00020> (visited on 02/21/2024).
- Rana, Muhammad Rizwan Rashid, Asif Nawaz, and Javed Iqbal (Aug. 1, 2018). “A survey on sentiment classification algorithms, challenges and applications”. In: *Acta Universitatis Sapientiae, Informatica* 10.1, pp. 58–72. URL: <https://sciendo.com/article/10.2478/ausi-2018-0004> (visited on 02/14/2024).
- Sarker, Iqbal H. (Mar. 22, 2021). “Machine Learning: Algorithms, Real-World Applications and Research Directions”. In: *SN Computer Science* 2.3, p. 160. ISSN: 2661-8907. DOI: 10.1007/s42979-021-00592-x. URL: <https://doi.org/10.1007/s42979-021-00592-x> (visited on 02/16/2024).

- Serrano-Guerrero, Jesus et al. (Aug. 1, 2015). “Sentiment analysis: A review and comparative analysis of web services”. In: *Information Sciences* 311, pp. 18–38. ISSN: 0020-0255. DOI: 10.1016/j.ins.2015.03.040. URL: <https://www.sciencedirect.com/science/article/pii/S0020025515002054> (visited on 02/13/2024).
- Sherstinsky, Alex (Mar. 1, 2020). “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network”. In: *Physica D: Non-linear Phenomena* 404, p. 132306. ISSN: 0167-2789. DOI: 10.1016/j.physd.2019.132306. URL: <https://www.sciencedirect.com/science/article/pii/S0167278919305974> (visited on 02/16/2024).
- Srivastava, Nitish et al. (2014). “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56, pp. 1929–1958. URL: <http://jmlr.org/papers/v15/srivastava14a.html>.
- Tay, Yi et al. (Jan. 30, 2022). *Scale Efficiently: Insights from Pre-training and Fine-tuning Transformers*. DOI: 10.48550/arXiv.2109.10686. arXiv: 2109.10686[cs]. URL: <http://arxiv.org/abs/2109.10686> (visited on 02/22/2024).
- Ting, Kai Ming (2010). “Confusion Matrix”. In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, pp. 209–209. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_157. URL: https://doi.org/10.1007/978-0-387-30164-8_157 (visited on 02/16/2024).
- Touvron, Hugo et al. (Feb. 27, 2023). *LLaMA: Open and Efficient Foundation Language Models*. DOI: 10.48550/arXiv.2302.13971. arXiv: 2302.13971[cs]. URL: <http://arxiv.org/abs/2302.13971> (visited on 02/20/2024).
- Vaswani, Ashish et al. (2017). “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc. URL: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 01/30/2024).
- Vidgen, Bertie and Leon Derczynski (Dec. 28, 2020). “Directions in abusive language training data, a systematic review: Garbage in, garbage out”. In: *PLOS ONE* 15.12. Publisher: Public Library of Science, e0243300. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0243300. URL: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0243300> (visited on 03/18/2024).
- Vikramkumar, Vijaykumar B, and Trilochan (Apr. 3, 2014). *Bayes and Naive Bayes Classifier*. DOI: 10.48550/arXiv.1404.0933. arXiv: 1404.0933[cs]. URL: <http://arxiv.org/abs/1404.0933> (visited on 02/16/2024).
- Wankhade, Mayur, Annavarapu Chandra Sekhara Rao, and Chaitanya Kulkarni (Oct. 1, 2022). “A survey on sentiment analysis methods, applications, and challenges”. In: *Artificial Intelligence Review* 55.7, pp. 5731–5780. ISSN: 1573-7462.

DOI: 10.1007/s10462-022-10144-1. URL: <https://doi.org/10.1007/s10462-022-10144-1> (visited on 02/11/2024).

- Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann (Oct. 2005). “Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis”. In: *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. HLT-EMNLP 2005. Ed. by Raymond Mooney et al. Vancouver, British Columbia, Canada: Association for Computational Linguistics, pp. 347–354. URL: <https://aclanthology.org/H05-1044> (visited on 02/14/2024).
- Xu, Lingling et al. (Dec. 19, 2023). *Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment*. DOI: 10.48550/arXiv.2312.12148. arXiv: 2312.12148[cs]. URL: <http://arxiv.org/abs/2312.12148> (visited on 01/30/2024).
- Yenduri, Gokul et al. (May 21, 2023). *Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions*. DOI: 10.48550/arXiv.2305.10435. arXiv: 2305.10435[cs]. URL: <http://arxiv.org/abs/2305.10435> (visited on 02/20/2024).
- Ying, Xue (Feb. 2019). “An Overview of Overfitting and its Solutions”. In: *Journal of Physics: Conference Series* 1168.2, p. 022022. DOI: 10.1088/1742-6596/1168/2/022022. URL: <https://dx.doi.org/10.1088/1742-6596/1168/2/022022>.
- Yosinski, Jason et al. (2014). “How transferable are features in deep neural networks?” In: *CoRR* abs/1411.1792. arXiv: 1411.1792. URL: <http://arxiv.org/abs/1411.1792>.
- Zeiler, Matthew D. and Rob Fergus (2013). “Visualizing and Understanding Convolutional Networks”. In: *CoRR* abs/1311.2901. arXiv: 1311.2901. URL: <http://arxiv.org/abs/1311.2901>.