Toxic Text Classification by Fine-tuning Pre-trained BERT

Hankun Wang
hankunw@sfu.ca

Hanjie Liu
hanjiel@sfu.ca

Abstract

Using deep neural network to classify toxic texts from scratch suffers from vanishing gradient and expensive training. Fine-tuning pre-trained language model based on transformer mechanism can efficiently solve these problems. This paper proposes a toxic text classification model designed to categorize text into six independent toxic comment labels. Our model takes text comments as input and output the probability vector. The classification model we construct can separately determine whether an input text belongs to each of six categories. By accurately classifying toxic words, it will be helpful for communities to maintain healthy and safe user communication. Our model utilize pre-trained BERT Sequence Classification model as baseline and perform further fine-tuning to improve model accuracy. We will leverage dataset from Kaggle competition and perform pre-processing on it. Our main goal is to increase model performance on the Area Under the Curve of the Receiver Operating Characteristic (AUC-ROC) evaluation metric. We did ablation study on different fine-tuning strategies. The evaluation results indicate that performing fine-tuning on all layers of Bert had the best performance on our task.

1 Introduction

In today's digital age, where communication primarily occurs through online platforms, the detection of toxic content poses significant challenges to maintaining healthy online communities. Toxic text, characterized by its harmful, abusive, or offensive nature, not only undermines the quality of discourse but also fosters an environment of negativity and hostility (Lodge, 1998). Consequently, there has been a growing need for effective mechanisms to identify and mitigate toxic text in online spaces.

Toxic text comments classification is a classic natural language processing(NLP) task. Research on toxic text classification involves the development and implementation of algorithms to automatically detect and categorize harmful or offensive language in digital communication channels. By analyzing linguistic features and contextual cues using methods like CNN and LSTM (Poojitha et al., 2023). Some other research studies demonstrate high-quality categorization using deep neural networks as well. (Khan, 2023). However, Deep neural network like RNN, LSTM and CNN always face the challenge of vanishing gradient. In contrast, too shallow network may encounter problem of under-fitting, making the network training become expensive. Our model will be built based on pre-triained BERT (Devlin et al., 2018), which involves mechanism of transformer. Unlike traditional recurrent or convolutional neural networks, the transformer architecture relies solely on self-attention mechanisms without any recurrence or convolution(Vaswani et al., 2017). This attention mechanism allows each token to attend to every other token in the sequence, regardless of their position. This means that information can flow more directly from distant tokens to each other, mitigating the vanishing gradient problem associated with long sequences.

The goal of this paper is to enhance the accuracy of identifying toxic text by fine-tuning the pretrained BERT model. Utilizing a dataset sourced from Kaggle(cjadams, 2017), we categorize toxic words into various labels. Our model will determine whether a text input belongs to each of labels, making the main task being a multi-label binary classification. Our model is an end-to-end toxic classifier which takes a text comment as input and output a probability vector. We will be proposing a targeted fine-tuning approach encompassing appropriate training strategies, fine-tuning strategies, data augmentation techniques, and hyper-parameter optimization to enhance the ultimate performance of our model on the test dataset. We will employ the AUC-ROC and classification accuracy as our evaluation metric to assess the performance of our model(Fawcett, 2006) through ablation study. AUC-ROC encompasses both a quantitative measure(AUC score) and a graphical representation(ROC curve) in classification tasks, making it well-suited for our project.

2 Related Works

Our study centers on fine-tuning a pre-trained BERT model to develop a toxic comments classifier. We will delve into the most relevant aspects within this context and direct readers to recent surveys for a broader understanding.

2.1 Toxic Text Classification

In the related research field, the work of (Wulczyn et al., 2017). integrates crowdsourcing with machine learning to assess a large amount of personal attacks. The project "Perspective," launched in collaboration between Google and Jigsaw, employs machine learning algorithms to detect harmful content, threats, and abusive speech online(Hosseini et al., 2017). Some studies have adopted convolutional neural networks (CNNs) for text classification of online content, a process that does not rely on any syntactic or semantic prior knowledge(Georgakopoulos et al., 2018). Additionally, model based on deep learning neural network like LSTM was proposed to determine a word's toxicity (Poojitha et al., 2023). With the advent of Transformer technology, fine-tuning pre-trained large models for various downstream tasks has become a mainstream approach, and our project is an example of implementing SOTA toxic comment classification by fine-tuning large language models.

2.2 Prompt-based Inference

Automatic toxic text classification faces challenges because language models like CNN and LSTM only catch the surface meaning of a word and convert it as the feature weights to classify its toxicity. Whereas, large language models(LLMs) aim to capture the complex structure and semantics of language, enabling them to perform various tasks such as text generation, translation, summarization, and sentiment analysis (Minaee et al., 2024). Some researches discussed about applying LLMs to detect toxic words . They proposed a model to classify toxic texts with zero-shot prompt-based inference(Wang and Chang, 2022). They employed prompt engineering (Ye et al., 2024) and did some fine-tuning on large language models, making the model well-performed on classifying implicit toxicity of a word.

2.3 Natural Language Processing

Several significant contributions have been made to the field of natural language processing, leveraging advanced language models like BERT and its derivatives. The BERT model, introduced in the paper of (Devlin et al., 2018), has transformed natural language processing by effectively capturing contextual language nuances through pre-training on extensive datasets. Expanding on BERT's achievements, researchers have delved into strategies to enhance its efficiency and scalability. For example, MiniLM, proposed by (Wang et al., 2020), offers a streamlined version of BERT aimed at reducing computational resources while maintaining competitive performance. Similarly, DistilBERT, introduced by (Sanh et al., 2019), tackles computational constraints by distilling BERT's knowledge into a more compact, faster model without compromising accuracy. These innovations signify significant progress in advancing toxic text classification methods, making them more adaptable and accessible for practical applications.

2.4 Pre-trained Language Models

Pre-trained models have exerted a transformative influence on the landscape of natural language processing (NLP) by consistently achieving state-of-the-art performance across a diverse array of tasks. Moreover, pre-trained models facilitate efficient transfer learning, reducing the need for large labeled datasets and computational resources for task-specific model training (Han et al., 2021). Pre-trained model was applied to detect toxic texts as well in some studies (Baldini et al., 2022). They investigate techniques for fine-tuning pre-trained models and harnessing varied performance across different pre-trained models (Zhao et al., 2021). Fine-tuning a pre-trained language model is more efficient and yields superior performance compared to constructing and training a deep neural network from scratch.

3 Approach

In our approach, we use a dataset from the Toxic Comment Challenge (cjadams, 2017) on Kaggle to

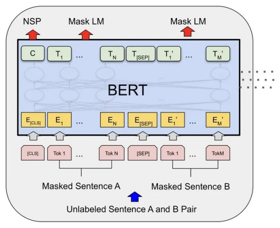

Figure 1: This is the language model that we are using to perform fine-runing. BERT utilizes a bidirectional Transformer architecture to capture contextual information from both left and right contexts in text sequences. Input to BERT consists of tokenized text sequences, while its output will go through a classification head and return the probability vector

fine-tune a BERT model, enabling it to determine whether a sentence is a toxic comment and to which category of toxic comment it belongs. We used accuracy and AUC-ROC as indicators to judge the model's capability.

3.1 Base Model

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained deep learning model that is widely used in various tasks in the field of natural language processing (NLP)(Devlin et al., 2018), such as text classification, question answering, named entity recognition, etc. The innovation of BERT is its ability to understand the meaning of words bidirectionally based on context. This method can more accurately capture the complexity and nuances of language than previous one-way models or shallow bidirectional models. The model structure of BERT is based on the 12 layers of Transformer encoder architecture. This architecture consists of multiple layers of Transformer encoders, each layer containing two main substructures: the Multi-Head Self-Attention mechanism and the Feed-Forward Neural Network. The BERT model uses a large amount of text data in the pre-training stage, which allows it to learn a wide range of linguistic rules and knowledge. This phase consists of two main types of tasks: Masked LM and Next Sentence Prediction. Through these two pre-training tasks, BERT is able to learn the relationships between words and the logical relationships between sentences. Pre-trained BERT can be adapted to a wide range of NLP tasks by fine-tuning on task-specific datasets without major changes to the model architecture. This flexibility is one of the reasons why we chose Bert as the base model.

3.2 Tokenizer

BERT's tokenizer uses a tokenization method called WordPiece. This approach allows the model to efficiently process unknown or rare words while maintaining efficient encoding of common words. The basic idea of the WordPiece word segmentation method is to break down words into smaller meaningful units (called tokens), which can be complete words, parts of words, or even individual characters. We choose "bert_uncased" as our model and tokenizer. The "bert_uncased" model and tokenizer are particularly suitable for case-insensitive tasks, where all English letters are converted to lowercase when preprocessing text data, ignoring the case distinction between letters. For example, in our toxic comment classification task, preserving case information may not result in additional performance gains, but rather interfere with the efficiency of training. Therefore, we chose to use the "bert_uncased" version to simplify the preprocessing steps and potentially speed up the training.

3.3 Pre-processing Data

We've pre-processed our finetune data. The idea is to think that there are some noise cases specified, and then delete them in the form of regular expressions. We think the noise could be, the square brackets and what's inside them, the input in the form of a URL (starting with https or www), the HTML tag "<>", all punctuation, the line break "\n", and any word or string that contains a number. This operation is extremely risky, because improper preprocessing logic can lead to severe noise in our training data, but at the same time, it is necessary to preprocess, and the original training data is mixed with too much text noise as shown above, which will greatly affect our training efficiency.

3.4 Configurations

We chose sigmoid as the last fully-connection layer (classification head). Because our task is multi-label classification, this means that a prediction is usually a binary probability of multiple independent labels. Therefore, sigmoid is the most suitable choice for our task.

In our task, the model needs to make predictions for each category independently, judging whether each category appears or not, rather than choosing one from multiple categories. Binary cross-entropy loss is particularly useful in this case, as it can independently assess the difference between the forecast and the actual situation for each category.

We chose AdamW as its optimization algorithm. It provides an effective way for AdamW to implement adaptive learning rate adjustment and a more reasonable weight decay mechanism at the same time. AdamW is a variant of the Adam optimization algorithm, which modifies the weight decay strategy in Adam to solve some problems of the original Adam algorithm when applying weight decay.

4 Experiment

4.1 DataSet

We leverage the toxic text comment dataset from Kaggle (cjadams, 2017) to perform training and testing. The dataset contains comment texts and their related labels. The categories are displayed in table 1, where each text can have multiple categories. For example, if a text is toxic and severe toxic, the value for its label vector will be $t(\vec{x}) = (1, 1, 0, 0, 0, 0)$. Therefore, the dataset can be regarded as six independent binary classification tasks.

Category
toxic
severe toxic
obscene
threat
insult
identity hate

Table 1: Categories

The text comments' raw data contain significant noise, including various signs that disrupt the meaning of sentences. Table 2 displays the most common noises. To address this issue, we propose a data cleaning method to preprocess the data before feeding it into our model. The input to our model is batched text comments and our model will return batched toxic classification probability vector.

We split the train dataset into train and validation. The raw test dataset contains a lot of garbage data which has no corresponding labels (as shown in table 3). We write an algorithm to filtered out these comments and generated the new test dataset. The detail data split strategy is proposed in table 4.

Noise	Example
newline character	\n\n It says it
contents in middle brackets	[Dear god this site] is
punctuation	'.', ',', ';'
(http),://, 'and' www.	(http),://, 'and' www.
words sticking with numbers	app1e is sweet, and...

Table 2: This table shows the main noise types of our dataset. While some of noises may play a meaningful role in the text, most of them are not helpful for our task. Therefore, we decide to remove these noises to enhance the training of our model.

== From RfC...	-1	-1	-1	-1	-1	-1
"== Sources == ...	-1	-1	-1	-1	-1	-1

Table 3: The example comments is grabbed from test dataset. When texts don't have labels, values of its label cells will be filled with -1. These rows of data will be eliminated by our algorithm and only those with meaningful labels will remain.

train	validation	test
127657	31914	63978

Table 4: We decided to split training dataset into training and valid sub-dataset with percentage being 8:2.

4.2 Implementation

Our project implementation comprises two components: one consisting of code chunks authored by ourselves, and another comprising parts sourced from online repositories. Initially, we developed data pre-processing algorithms designed to filter out garbage test data and remove noise. This step is fundamental to our project, as it ensures that the dataset is well-formatted and properly prepared for subsequent stages. Following that, we implemented the dataloader component, utilizing the tokenizer from the pre-trained BERT model obtained from Hugging Face ('Bert-base-uncased'). We opted for the 'Bert-base-uncased' model since the task of toxic comments classification is not case-sensitive. We did not construct the model architecture ourselves; instead, we solely wrote the code capable of invoking it. The model is an open source Bert model lying under library of hugging

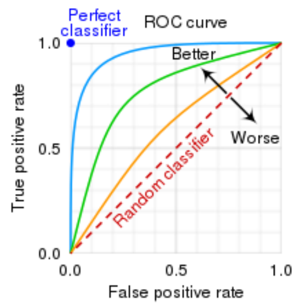

Figure 2: True positive rate measures the proportion of actual positive instances that are correctly identified by a classification model. False positive rate measures the proportion of negative instances that are incorrectly classified as positive by a classification model. A larger Area Under the ROC Curve (AUC score) indicates better model performance.

face. Then training methods and evaluation metrics are implemented by ourselves, where we used AUC-ROC and classification accuracy for evaluation. The detail of evaluation will be unfolded in later part.

4.3 Evaluation Metrics

Our model's performance evaluation is segmented into six categories, each requiring an independent assessment. To this end, we opt to utilize the Area Under the Curve of the Receiver Operating Characteristic (AUC-ROC) metric(Fawcett, 2006). AUC-ROC considers the entire range of possible classification thresholds, allowing it to capture the trade-off between true positive rate (TPR) and false positive rate (1-TPR) for different threshold values. This makes it particularly suitable for imbalanced datasets or scenarios where the cost of false positives and false negatives differs (see figure 1). Due to the nature of our data labels, which consist of a six-dimensional vector where each dimension represents a separate label without mutual influence, we initially attempted to evaluate each label using AUC-ROC individually. However, this approach yielded suboptimal results due to significant differences in the distribution of positive instances across labels. For instance, the number of positive instances for the 'toxic' label far exceeded those for the 'identity hate' label. Consequently, evaluating the labels separately led to inaccurate assessments of the model's performance. To address this issue, we combined the six individual binary classification problems into a single, larger binary classification problem. By doing so, we leveraged the insensitivity of AUC-ROC to data imbalances, enabling us to accurately evaluate the model's overall performance across all labels. This integrated approach ultimately allowed us to achieve our goal of correctly assessing the model's effectiveness. Building upon this framework, we also incorporated an accuracy score, computed as the ratio of correctly classified labels to the total number of labels.

$$accracy = \frac{correct}{total}$$

4.4 Model Comparison

We conducted an ablation study to analyze the impact of our fine-tuning strategies. Our baseline model is the pre-trained Bert model, which remains unaltered without any fine-tuning procedures. There are three fine-tuned pre-trained models that we have built. Model A underwent training by fine-tuning all layers of the pre-trained Bert model, including the embedding layer, transformer layers, and output classification layer. Concurrently, we developed a second model Model B, fine-tuned with the embedding layer and the first three transformer layers frozen. To gauge the influence of training data size, we downsized the training dataset for our third model Model C based on the fine-tuning approach of the first model. Specifically, we used only the first 1,000 rows of data from the original training dataset. Subsequently, we will evaluate the performance of each model using the mentioned evaluation metrics.

4.5 Results

Our experimental results encompass the training cost, training loss, and model performance metrics for each model. Given that all models were trained on a 2080ti graphics card, the training cost is approximated as the training time. As the loss and training time of the model trained on a smaller dataset may significantly differ from that of models trained on the complete dataset, we only compared the training loss and training time of two models trained on the complete training data. Additionally, we evaluated model performance using the mentioned evaluation metrics.

4.5.1 Training Time

Excluding the baseline model and the model trained with a small dataset, we compare the remaining two models. In summary, the model fine-tuned with frozen layers demonstrates a training time approximately 20 to 30 percents faster than the model fine-tuned with all layers. Specifically, the frozen layer model achieves a training speed of about 12 to 13 iterations per second, while the all-layer model operates at around 9 to 10 iterations per second. Detailed training time data can be found in Table 5.

model	A	B
epoch=4	48min	36min

Table 5: This is the training time of two models with epoch=5

4.5.2 Training Loss

After four epochs of training, the validation loss of the model fine-tuned with frozen layers is slightly higher than that of the model fine-tuned with all layers. Specifically, the final validation loss for the frozen layer model is 0.0641, while for the all-layer model it is 0.0463. Refer to Figure 3 for a plot-form comparison. Additionally, it's worth noting that during the fine-tuning process, models with slightly higher learning rates may exhibit significant fluctuations in loss, as depicted in Figure 4.

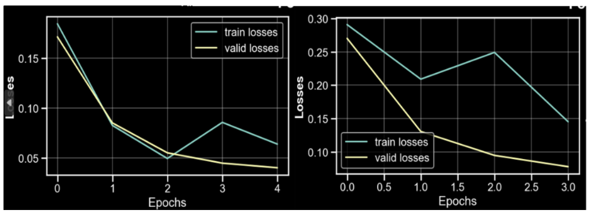

Figure 3: The plot on the left represents the fully fine-tuned model, while the plot on the right depicts the model with frozen fine-tuning. Since the model on the left has undergone a total of 5 epochs, we specifically examine its results at the 4th epoch when comparing losses. Observing the figure, after the fourth epoch of training, the validation loss of the fully fine-tuned model is slightly lower than that of the model with frozen layers.

4.5.3 AUC-ROC and Accuracy

We evaluated the models using AUC-ROC and accuracy metrics. The test dataset from the Kaggle competition was utilized to compute the AUC score and draw the ROC curve (cjadams, 2017). Given the significant differences between the test and training datasets, the AUC score on the test dataset typically remains relatively low, with the highest reaching only 0.65. Consequently, we additionally present the AUC score and ROC curve of the model on the validation dataset.

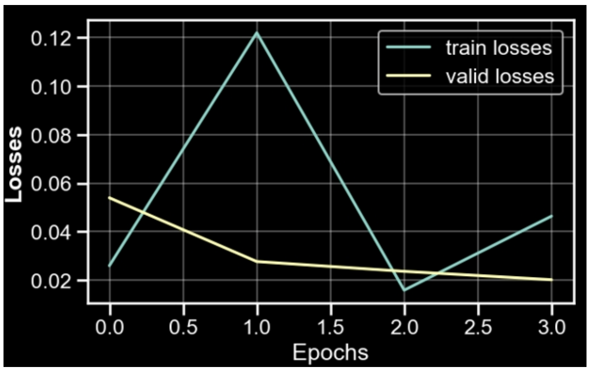

Figure 4: This is the figure of training loss and valid loss for model trained on full training dataset with learning rate being 1e-6

The comparative experiments results presented in Table 6 indicate that models fine-tuned on the complete training dataset, namely model A and model B, perform the best. Compared to the baseline model, both model A and model B exhibit significant improvements in performance on both the validation and test datasets. Conversely, the model fine-tuned on the small dataset performs the worst, with accuracy and AUC scores even lower than those of the baseline model. We put our plot of ROC curves w.r.t test and valid dataset of four models at appendix A for more specifically visualization of our model evaluation.

model	baseline	A	B	C
test auc	0.4796	0.6584	0.6498	0.4033
test acc	0.61	0.93	0.90	0.30
valid auc	0.5450	0.9786	0.9574	0.4764
valid acc	0.38	0.98	0.95	0.50

Table 6: This table compares four models from their test AUC score, test accuracy score, valid AUC score, and valid accuracy score, respectively.

5 Analysis

Drawing from our experiments and their corresponding results, we can conduct the following analyses to derive more reasonable conclusions.

5.1 Learning Rate

In our model fine-tuning process, we ultimately opted for a learning rate of 2e-7. Reflecting on the variations in loss resulting from different learning rates, as discussed in section 4.5.2, we observed a high sensitivity to learning rate adjustments for models fine-tuned on the complete training dataset. As depicted in Figure 4, training with a learning rate of 1e-6 led to considerable fluctuations in training loss. However, reducing the learning rate to 2e-7 or 1e-7 resulted in smaller fluctuations. We attribute this phenomenon to the substantial volume of training data, which prompts the model to optimize parameters more frequently in each iteration. Consequently, even minor adjustments to the learning rate exert a greater impact on the model's parameter optimization process.

5.2 Training Dataset

Initially, we anticipated that fine-tuning the model on a small dataset would enhance its performance compared to the baseline model. However, upon analyzing the evaluation metrics detailed in section 4.5.3, we discovered that fine-tuning on a small dataset (specifically, by utilizing only the first 1000 rows of the training dataset) actually resulted in poorer classification of toxic comments by the model. We attribute this outcome to either the dataset being too small or having numerous missing training data, which consequently limits the model's ability to classify words not present in the training data but existing in the vocabulary. Additionally, the model exhibits an excessively high recognition rate for words with toxicity weights in comments from the training dataset. Thus, even if a word's toxicity level is low, its frequent occurrence in toxic comments within the first 1000 lines may lead the model to incorrectly classify it as highly toxic, ultimately undermining the model's overall performance.

5.3 Freezing-layer Fine-tuning

Model A involves fine-tuning all Bert layers, while model B fine-tunes Bert after freezing certain layers. As observed from the results in Table 6, model A demonstrates slightly better performance compared to model B. We attribute this to model A's comprehensive fine-tuning of all Bert layers, rendering the model more adaptable to detecting toxic words and thus enhancing its ability to classify toxic comments. Conversely, model B fine-tunes Bert with fewer parameters by freezing certain layers, resulting in approximately a 20% improvement in training speed.

6 Conclusion

In conclusion, our primary objective in this project was to train a language model with a high sensitivity to toxic words. Based on the experimental and analytical findings, we have determined the following optimal fine-tuning strategy: fine-tune the model on the entire training dataset, set the learning rate to 2e-7, and simultaneously fine-tune all layers of the Bert model. This approach ensures that the final trained model achieves optimal expressiveness and the highest ability to accurately classify toxic comments.

Regarding areas for potential optimization in this project's approach, we believe that the data preprocessing process and the handling of special symbols should be more refined. While certain symbols may indeed constitute noise and lack relevance for analyzing sentence meaning, others can provide valuable context in detecting toxicity. For instance, variations in punctuation like "Hello!" and "Hello?" convey different meanings. Similarly, in toxic text classification, the usage of punctuation may impact the weight of toxicity in a comment. Thus, we suggest that enhancing the refinement and meaningfulness of filtering algorithm during data pre-processing could lead to improvements on model performance.

Due to time constraints, certain operations in our project were not undertaken, which could potentially influence the experimental results. We believe that comparing our fine-tuned models with a broader range of models, such as DistilBert(Sanh et al., 2019) or GPT(Yenduri et al., 2023), could enhance our understanding of their performance. By juxtaposing our fine-tuned models with additional open-source language models, we may uncover new phenomena, derive more insightful conclusions, and ultimately refine our fine-tuning strategy.

References

Ioana Baldini, Dennis Wei, Karthikeyan Natesan Ramamurthy, Mikhail Yurochkin, and Moninder Singh. 2022. Your fairness may vary: Pretrained language model fairness in toxic text classification.

Julia Elliott Lucas Dixon Mark McDonald nithum Will Cukierski cjadams, Jeffrey Sorensen. 2017. Toxic comment classification challenge.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Tom Fawcett. 2006. Introduction to roc analysis. *Pattern Recognition Letters*, 27:861–874.

Spiros V. Georgakopoulos, Sotiris K. Tasoulis, Aristidis G. Vrahatis, and Vassilis P. Plagianakos. 2018. Convolutional neural networks for toxic comment classification.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. 2021. Pre-trained models: Past, present and future.

Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving google's perspective api built for detecting toxic comments.

Md Azim Khan. 2023. Determination of toxic comments and unintended model bias minimization using deep learning approach.

Harte D. K. Tripp G. Lodge, J. 1998. Children's self-talk under conditions of mild anxiety. *Journal of anxiety disorders*.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey.

K. Poojitha, A. Sai Charish, M. Arun Kuamr Reddy, and S. Ayyasamy. 2023. Classification of social media toxic comments using machine learning models.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers.

Yau-Shian Wang and Yingshan Chang. 2022. Toxicity detection with generative prompt-based inference.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale.

Qinyuan Ye, Maxamed Axmed, Reid Pryzant, and Fereshte Khani. 2024. Prompt engineering a prompt engineer.

Gokul Yenduri, Ramalingam M, Chemmalar Selvi G, Supriya Y, Gautam Srivastava, Praveen Kumar Reddy Maddikunta, Deepti Raj G, Rutvij H Jhaveri, Prabadevi B, Weizheng Wang, Athanasios V. Vasilakos, and Thippa Reddy Gadekallu. 2023. Generative pre-trained transformer: A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions.

Zhixue Zhao, Ziqi Zhang, and Frank Hopfgartner. 2021. A comparative study of using pre-trained language models for toxic comment classification. In *Companion Proceedings of the Web Conference 2021*, WWW '21, page 500–507, New York, NY, USA. Association for Computing Machinery.

Appendices

A Plot of ROC Curves for Models

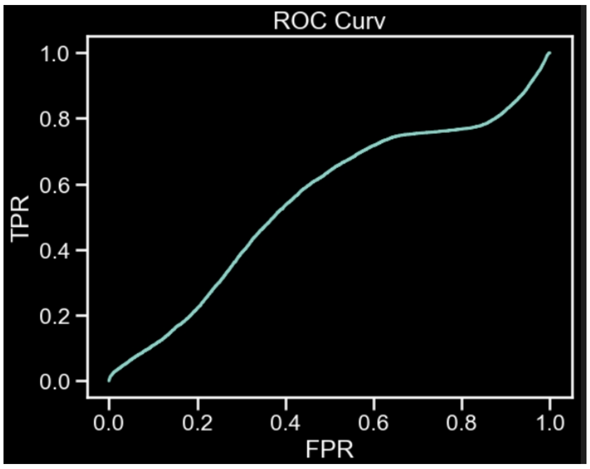

Figure 5: This is the ROC curve of baseline model on valid dataset

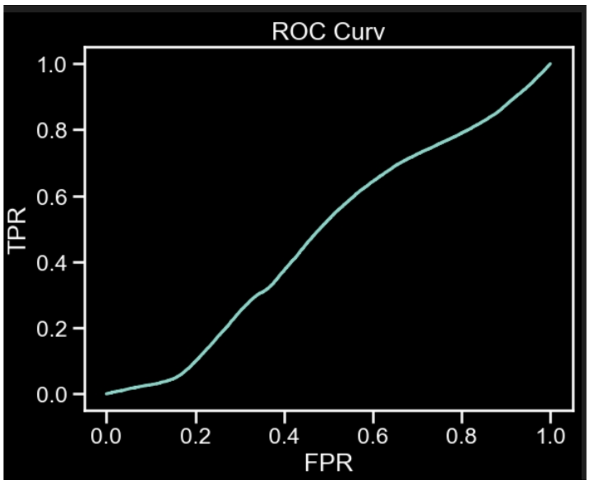

Figure 6: This is the ROC curve of baseline model on test dataset

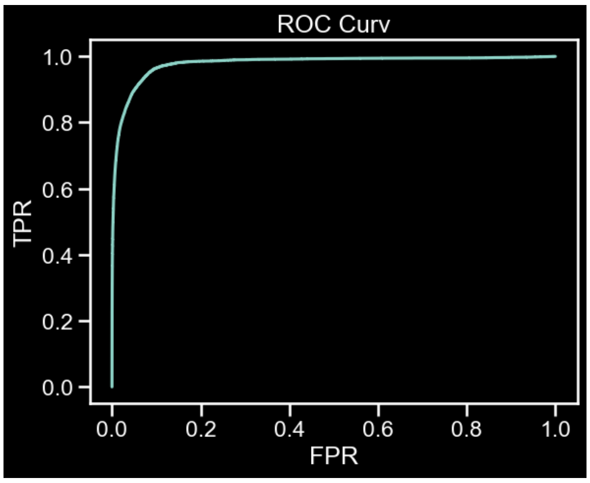

Figure 7: This is the ROC curve of model A on valid dataset

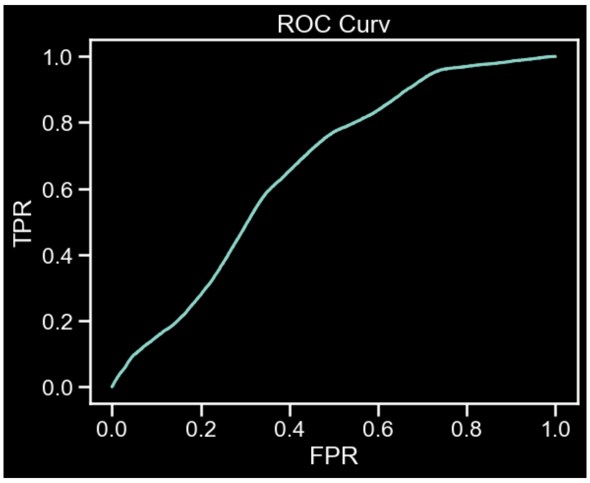

Figure 8: This is the ROC curve of model A on test dataset

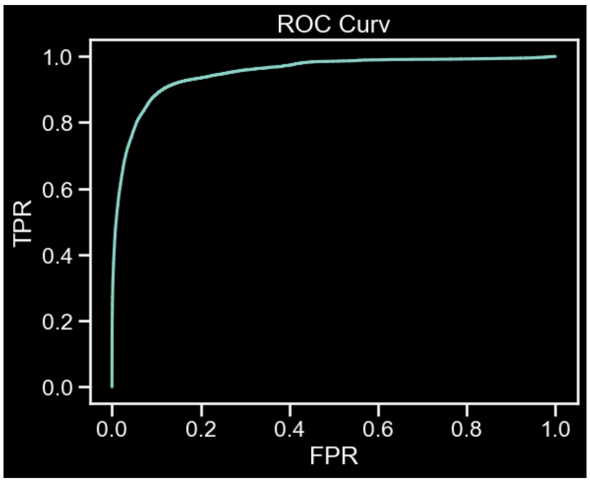

Figure 9: This is the ROC curve of model B on valid dataset

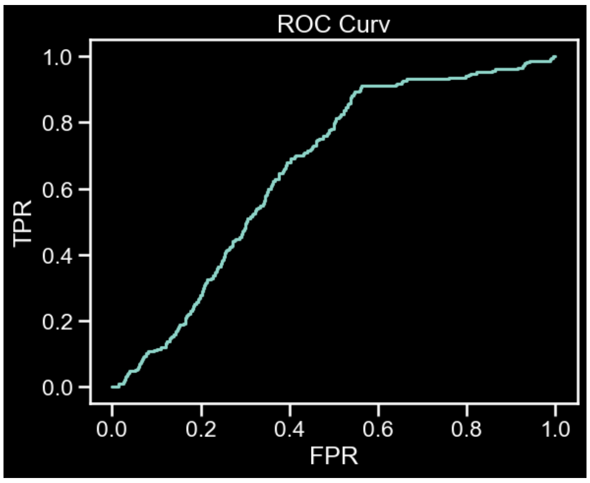

Figure 10: This is the ROC curve of model B on test dataset

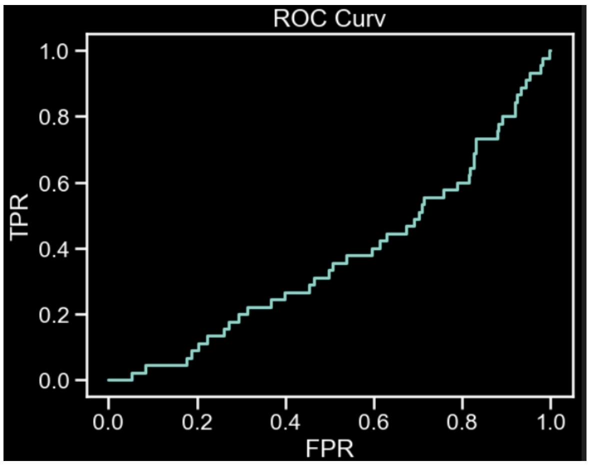

Figure 11: This is the ROC curve of model C on valid dataset

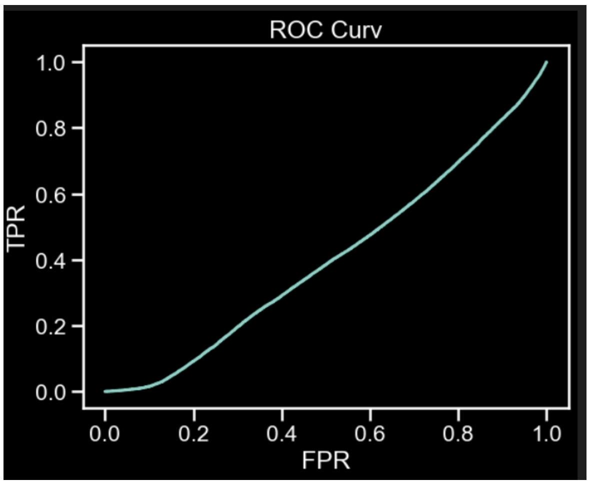

Figure 12: This is the ROC curve of model C on test dataset