Guilt Detection in Text

Ruojia Tao
Cornell tech master student
rt456@cornell.edu

Anqi Dong
Cornell tech master student
ad829@cornell.edu

Yaxuan Huang
Cornell tech master student
yh798@cornell.edu

Hongjiao Zhang
Cornell tech master student
hz452@cornell.edu

Abstract

Researchers have made significant progress in sentiment analysis, utilizing deep learning and natural language processing techniques to determine positive or negative sentiments in text. However, detecting complex emotions remains a challenge. In this study, we aim to develop a deep learning model, particularly focusing on natural language processing, to classify emotions, specifically guilt. Our work is based on a published paper, "Guilt Detection in Text: A Step Towards Understanding Complex Emotions". By improving the model's understanding and classification of guilt, we aim to facilitate proactive monitoring of user-generated content on social media platforms and enable timely support for individuals in need of assistance.

Keywords
Deep learning, Natural language processing, Sentiment analysis, Emotion recognition

1. Introduction

Deep learning and natural language processing models have made significant advancements in enabling artificial intelligence to comprehend human emotions. While the current focus has primarily been on recognizing positive and negative emotions, understanding complex emotions such as guilt remains a challenge.

Accurately classifying complex emotions opens up opportunities for wider applications of artificial intelligence. Guilt, in particular, arises when individuals feel they have acted inappropriately or committed wrongdoing. This emotion is closely associated with mental illness, making its classification and analysis crucial. Considering the prevalence of emotional expression in online social media, effectively classifying a broader range of emotions in social media posts could potentially aid in identifying individuals exhibiting suicidal tendencies and providing them with timely support.

Therefore, this research aims to develop a deep learning-based model that can accurately classify various emotions, including guilt, from online social media posts, thereby contributing to the early detection of individuals at risk and enhancing mental health interventions.

2. Related Work

To establish a foundation for our study, we draw upon the paper "Guilt Detection in Text: A Step Towards Understanding Complex Emotions" [1]. This work offers valuable insights into the literature review, available datasets, baseline models, and future directions for developing emotion classification models. This paper served as a valuable resource to navigate the research landscape and it is our main paper.

Another important reference was the review article titled "Text-based Emotion Detection: Advances, Challenges, and Opportunities" [2]. This comprehensive review summarized the main approaches used in text-based emotion detection systems and highlighted recent state-of-the-art proposals. It also provided a useful comparison of emotion-labeled datasets, aiding us in selecting an appropriate dataset for our study.

Additionally, the paper "ReDDIT: Regret Detection and Domain Identification from Text" [3] presents a deep learning model specifically focused on detecting regret, which shares similarities with our goal of emotion detection. We can leverage similar approaches from this study to train our deep learning model effectively.

To compare different models, we analyzed "Comparative Analyses of BERT, RoBERTa, DistilBERT, and XLNet for Text-Based Emotion Recognition" [4]. This paper evaluated the performance of various pre-trained transformer models, such as BERT, RoBERTa, DistilBERT, and XLNet, in recognizing emotions from the text. The findings provided insights into the effectiveness of these models for emotion recognition tasks.

We also explored papers specifically focused on the biLSTM model, including works by Balouchzahi et al. [3], [5], and Schuster, M. & Paliwal, K [6]. These studies provided valuable insights into the application and optimization of the biLSTM model in different contexts. Furthermore, the efficacy of combining the CNN network with an attention mechanism for emotional classification was demonstrated by He et al. [7], supporting our approach to integrating the attention mechanism into our model. Additional insights into the fusion of biLSTM and CNN models were gained from the works of Zhang et al. [8], Polignano et al. [9], and Rajabi et al. [10].

Considering more complex emotion detection problems, we found this paper Fine-grained Sentiment Classification using BERT [11]. There have been a lot of related papers using BERT models, but this paper focused on more significant or higher-level deep learning NLP for sentiment classification, especially in fine-grained ones. And their experiments also made a conclusion that the sentiment classifier using BERT models resulted in outperformance over other popular models even with simple architecture. This paper showed guidance to our project in model selection for a promising accuracy score. While ChatGPT is prevailing around the globe, we wanted to see if ChatGPT can help us with guilt detection. Thus, we found a paper named Is ChatGPT a Good Sentiment Analyzer? A Preliminary Study [12]. They discussed the difference in performance using various evaluation metrics with ChatGPT and BERT and SOTA models. Their result suggested that ChatGPT gave magnificent zero-shot sentiment analysis, and few-shot was even more impressive than fine-tuned BERT. Thus, for our project with BERT, we might not expect a very good accuracy score.

Furthermore, certain papers exploring the physiological aspects of guilt are relevant to our project and can contribute valuable insights.

3. Methodology

Our main method involves comparing the traditional machine learning model, deep learning model, and GPT API Prompt Model.

3.1. Datasets

Vent [13]: a large annotated dataset of texts (texts are from 33M posts), emotions, and social connections from the Vent social media platform. In Vent, each post is associated with emotion, self-annotated with emotion by the post's author at the posting time. It comprised 3 datasets: emotion_categories.csv, emotions.csv, and vents.csv. vents.csv is where the text data is located, while emotion_categories and emotions are emotion_id with specific emotion labels.

Data Privacy Notice: The data used in this research project, referred to as "vent", is classified as private and proprietary. We therefore regretfully cannot provide it in an academic manner to make it publicly available. However, we have followed standard research practices to ensure the validity and integrity of our findings, and we provide detailed information on the methodology, variables, and results from the dataset within this report.

3.2. Data Cleaning (EDA)

3.2.1 Dataset Combination

In this study, we utilized the Vent dataset package, which comprised three files: emotion_categories.csv, emotions.csv, and vents.csv. The vents.csv file contained the textual data, which was labeled by user_id and emotion_id. Our initial step involved joining the emotion_categories dataset with emotions to establish the specific emotion associated with each id. We solely extracted categories that were classified as "Feelings," as indicated in the paper. Subsequently, we selected the customer id and specific emotions from the joined dataset and merged them with the vent dataset, enabling us to obtain the corresponding text for each entry. This process yielded a combined dataset encompassing the emotion_id, corresponding emotion, user_id, creation time, reaction, and text.

3.2.2 Noise Cleaning

Due to the source of the text data being Twitter, it contained various forms of noise, including emojis, non-ASCII characters, URLs, and null values. Consequently, our next step involved cleaning the text column to address these issues. We commenced by removing a broad range of Unicode emojis,

encompassing emoticons, symbols, pictographs, and flags, utilizing a comprehensive regular expression (regex) pattern. Subsequently, we eliminated non-ASCII characters, URLs, single standalone characters, and overlong sequences of identical characters. Lastly, we filtered out null values and cells that contained only empty spaces, completing the data-cleaning process.

3.2.3 Data Imbalance

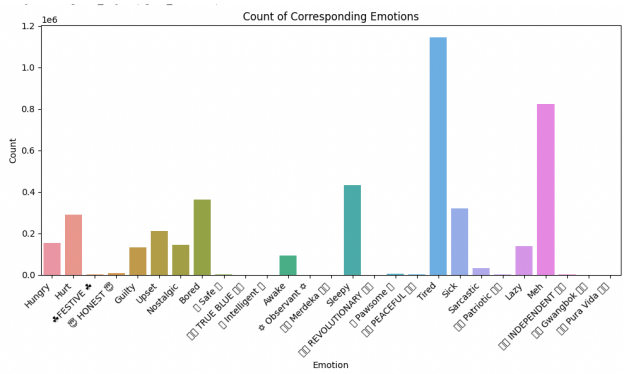

Fig 1. The distribution of emotions

We analyzed the distribution of emotions as shown in Fig1. and found out that “Guilty” only takes a small percentage of the whole population. Therefore, we have a huge imbalance problem to solve.

Addressing the challenge of severe data imbalance, where the minority class constituted merely 3.08% of the overall training dataset, we employed the undersampling technique to downsize the majority class, thus enhancing the model's performance.

3.2.4 Train-test split

We split the undersampled dataset into training and testing subsets by randomly assigning them under the 80:20 train-test split.

3.3 Feature engineering

Inspired by the research paper "Guilt Detection in Text: A Step Towards Understanding Complex Emotions"[1], we decided to use Term Frequency-Inverse Document Frequency (TF-IDF) as feature engineering for the traditional machine learning method. TF-IDF is a numerical representation technique commonly used in natural language processing and information retrieval. It assigns a weight to each word in a document that reflects its importance or relevance in the context of a larger collection of documents (corpus).

For the deep learning method, we used Bert tokenization and padding procedure to prepare the text data in a numerical format that can be fed into machine learning models, ensuring consistent sequence lengths and preserving the information of the original text.

3.4 Experiments plan

We start with the baseline model. For the baseline model in the Guilt Detection in Text project, our objective was to select a simple yet highly accurate model. Through a thorough analysis of traditional machine learning models, as outlined in the comparison table of the research paper "Guilt Detection in Text: A Step Towards Understanding Complex Emotions" [1], we observed that the model employing TF-IDF for feature engineering and Logistic Regression for classification exhibited the highest accuracy. Consequently, we chose to adopt this model configuration.

For the deep learning models, inspired by the experiments in [1], [4], and [6], we selected Bi-directional Long Short Term Memory (BiLSTM) for the initial. Bi-LSTM [7], is a form of recurrent neural network that uses LSTM cells (i.e. cell state representations) to adaptively change the size of its step length between inputs. This mechanism makes it possible for the network to learn long and short-term dependencies, as well as dynamics that span multiple time steps and hidden states. This NN is made of a Bidirectional LSTM layer and Dropout layers.

Also, the current research finds that the combination of the CNN network and the attention mechanism can obtain very good target-specific emotional classification results [8]. It can well solve the shortcomings of LSTM which cannot accurately indicate the importance of each word in the sentence. The proposed fusion model combines this advantage and the CNN model with a multiple-attention mechanism that is proposed to obtain the emotional polarity of keywords, which is an important dimension of the emotional classification of fusion models. Inspired by the [9], [10], and [11], we decided to combine the biLSTM and CNN models together.

Another model we want to use is BERT (Bidirectional Encoder Representations from Transformers), which is a powerful pre-trained language model that has revolutionized natural

language processing tasks. BERT provides contextual word embeddings, which means it captures the meaning of words based on their surrounding context in a sentence, which is what we need for understanding the sentiment of the context. It is also trained with a large amount of text data.

The last model we take is the ChatGPT API prompt model. ChatGPT, or similar language models, are primarily designed for natural language understanding and generation tasks, including text classification. ChatGPT has a strong language understanding capability. It can capture contextual information, identify relationships between words, and comprehend the meaning of the text, which is beneficial for text classification tasks that require understanding the content and context of the text.

In the current natural language processing, the Chat GPT model is popular for checking text sentiment. As a chatbot, it is famous for its outperformance in its understanding of human languages. As an industry application of natural language processing, it is also really likely to be used for sentiment detection. So in the last model, we want to use an industry model and want to see their performance of this specific task: How well the chatbot performs in understanding the guilty emotion.

In conclusion, we will compare six models including the Logistic Regression model, BiLSTM Model, CNN Model, BiLSTM+CNN Model, DistilBert Model, and GPT API Prompt Model. The full details of the experiments are detailed in the following subsection.

4. Implementation, experiments & Observations

As previously mentioned, we compared six models, and here's the final output as shown in Table 1, the details are in the following subsection. The BiLSTM model with L2 regularization achieved the highest accuracy in the test dataset at 77.00%.

Table 1. Model Accuracy Comparison

model	Training Accuracy in the last epoch	Validation Test Accuracy in the last epoch
Logistic Regression	77.00%.	75.00%
BiLSTM	80.40%	77.00%
CNN	72.37%	70.13%
BiLSTM+ CNN	81.16%	76.36%
DistilBert Model	95.20%	71.89%
GPT API Prompt Model	/	62.6%

4.1 Logistic Regression model(baseline):

To facilitate model training, we transformed the raw text data into numerical features using TF-IDF vectorization. Subsequently, we fitted the model and performed a grid search to identify the optimal hyperparameters. Our investigation revealed that utilizing a penalty value of 'l2' and employing the 'saga' solver yielded the best results.

The resulting model demonstrated an overall accuracy of 75%, with a precision of 96.24%, recall of 75.87%, and an f1-score of 83.82%.

4.2 BiLSTM Model:

The model consists of several key components. Initially, an Embedding layer is introduced to learn and represent input words as dense vectors, with an embedding dimension of 100. Subsequently, a Bidirectional LSTM layer is added to process the input sequences in both forward and backward directions, enhancing the model's ability to capture

long-range dependencies. This LSTM layer comprises 64 units and is configured to return the complete sequence of hidden states. Another Bidirectional LSTM layer follows, mirroring the previous one, but without the need to return the entire sequence. Lastly, a Dense layer with a single unit and a sigmoid activation function is incorporated to generate a probability score between 0 and 1 for binary classification tasks. The structure graph is in Fig.2:

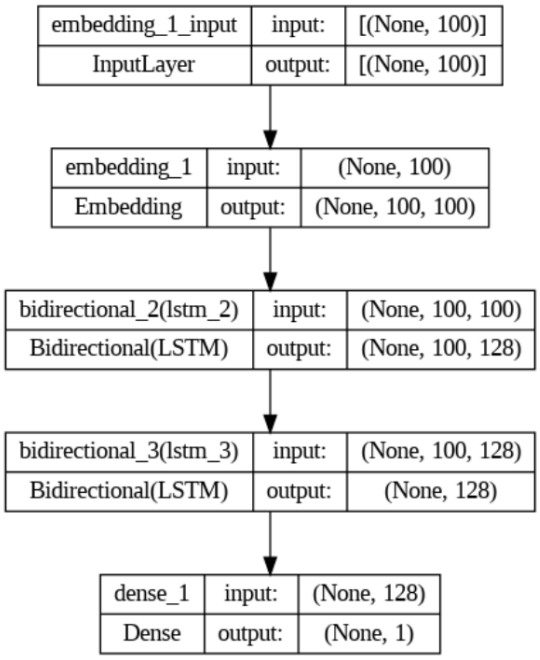

Fig2. BiLSTM model structure

The initial result is as follows: In the last epoch, the accuracy of the training dataset is 92.35%, while the accuracy in the test dataset is 72.80%, indicating that there is a likelihood of overfitting. So we tried different methods of controlling overfitting, including adding a dropout layer with 0.5 as the dropout rate, using l2 regularization with 0.005, and early stopping. The result is shown in Table 2:

Table 2. Comparison of methods preventing overfitting

method	Training Accuracy in the last epoch	Validation Test Accuracy in the last epoch	Average Training Loss in the last epoch	Average Validation Loss in the last epoch
original	0.9235	0.7281	0.1720	1.0137
dropout layer	0.9137	0.7005	0.1983	0.9874
L2 regularization	0.8040	0.7700	0.4380	0.4950
Early stopping	0.7649	0.7795	0.4800	0.4688

Upon analyzing the table, it shows that adding a dropout layer shows some regularization effect but also decreases the validation accuracy. L2 regularization and early stopping effectively control overfitting and improve generalization.

Considering the training and validation accuracy, as well as the training and validation loss, L2 regularization and early stopping appear to be effective methods for controlling overfitting and achieving better generalization in this scenario.

In conclusion, our final result is achieving 77.00% accuracy in the test dataset.

4.3 CNN Model:

After running through the TF-IDF vectorizer while limiting the top 100 features, we converted the processed data into PyTorch tensors and loaded them into Dataloaders. Before the model, we defined a grid of hyperparameters including kernel_dim, kernel_sizes, dropout, lr, and performed over all combinations of those hyperparameters. For each combination, a CNN model is instantiated and trained. The output of the final layer is passed through a log softmax function to provide the log probabilities for each class.

For each training process, we looped over a predefined number of epochs so that we can present all training examples in a mini-batch fashion. For each batch, the parameters are updated to minimize the log-likelihood loss.

After training, we evaluated the performance through a test set when the average of loss and accuracy is calculated. The process of training and testing is repeated for all combinations of

hyperparameters while hyperparameters yield the highest accuracy on the test set.

Upon completion, the parameters of the best model along with its corresponding test accuracy will be printed out, which can help us gain insights into the optimal set of hyperparameters for CNN.

Initially, we ran through a 1-dimensional convolutional layer with different kernel sizes followed by a max pooling layer. After concatenating the outputs, 2 fully connected layers are applied. But the training and testing accuracy was always fluctuating around 0.5. We figured out that it might not be complex enough to learn patterns from the data. Therefore, we added batch normalization layers, more fully connected layers, relu activation function, and drop-out layers in the original network. The final testing accuracy raised from 63% to 71.20% in consequence.

4.4 BiLSTM+CNN Model:

Our model begins with an Embedding layer, which transforms input tokens into dense vectors for effective representation. The BiLSTM layer captures sequential dependencies by processing the input sequence in both forward and backward directions. To further enhance feature extraction, a CNN layer with a ReLU activation function is applied, followed by a Global Max Pooling layer to extract the most salient features.

In order to introduce an attention mechanism, our model utilizes an attention size of 64. Attention weights are computed using Dense layers, applying hyperbolic tangent activation to encode non-linearity, and softmax activation to ensure proper weight distribution. By multiplying the attention weights with the BiLSTM output, the attention mechanism highlights the relevant parts of the input sequence. The resulting attention output is then fed into fully connected layers with Rectified Linear Unit (ReLU) activation functions to enable further learning and abstraction. Finally, a sigmoid-activated Dense layer serves as the output layer for binary classification tasks.

The model, created with the attention layer, is compiled using the Adam optimizer and binary cross-entropy loss function. By leveraging the combination of BiLSTM, CNN, and attention mechanism, this model aims to capture the both local and global context in text data for improved classification performance. The structure graph is in Fig 3.

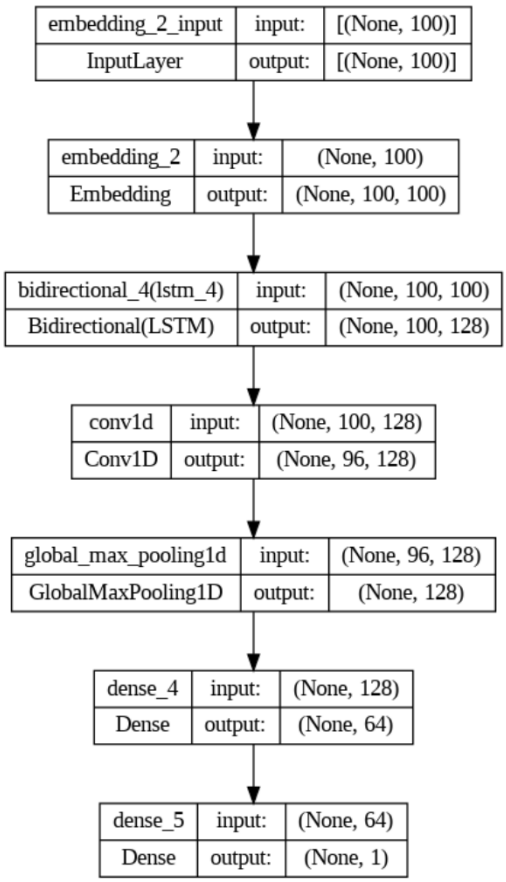

Fig3. BiLSTM+CNN model structure

The initial result is as follows: In the last epoch, the accuracy of the training dataset is 0.9337, while the accuracy in the test dataset is 0.6646, indicating that there is a likelihood of overfitting.

According to the previous experiment, using the l2 regularization performs well, so we add the l2 regularization for the rate of 0.05 for the BiLSTM Layer and Fully Connected Layers. Eventually, the overfitting phenomenon is eased, achieving 76.36% accuracy in the test dataset.

4.5 DistilBert Model:

4.5.1 Model Selection:

We use a pre-trained Bert model for the text classification task. We choose the Distil Bert model since this is a relevant small-size model. We have also tried other different Bert models, such as the

Bert base model[19]. However, those models are too large to be trained with the current hardware resources. The model we tried and their size is in Table 3:

Table 3. Comparison of models

Architecture Name	Shortcut name from hugging face	Number of Parameters
BERT	bert-base-uncased	110M
GPT	openai-gpt	110M
BERT	bert-small	60M
DistilBERT	distillery-base-uncased	66M

We tried upon models and after consideration of the advantages of each model, we finally decided to use distillery-base-uncased.

The reason we choose DistilBert[15] is that it is a distilled version of the BERT model. It has a smaller size and fewer parameters compared to the original BERT model while still retaining much of its performance. This makes DistilBERT more memory-efficient and faster to train and deploy.

The bert-base-uncased model and openai-gpt model are too large for us to use. Our hardware is not supportable for using these models since they have 110M parameters, and the system will crash with running out of all the RAM. So we turn our gears to other models that have relatively smaller sizes. So we tried distilbert-base-uncased and bert-small. However, the bert-small is a user upload model, so we finally decided to use distilBert model finally. The DistilBERT model should be a good choice after considering the model efficiency, performance, speed, and resources we have.

The distilBert model consists of a BERT encoder, which encompasses embeddings, an encoder, and a pooler. The embedding component handles the input token embeddings, position embeddings, and token type embeddings. The encoder component includes multiple transformer layers, each comprising attention mechanisms, intermediate and output layers, and layer normalization. The pooler aggregates the contextualized representations from the encoder and produces a fixed-length representation for the entire sequence. Following the BERT encoder, a dropout layer is applied to mitigate overfitting, and a linear classifier layer maps the pooled output to the number of output classes, which is 2 in this case, indicating a binary classification task. This model leverages the power of BERT to process input sequences, learn contextual representations, and perform sequence classification effectively. The overall structure of the distilBert model is as Fig4 shows:

```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 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)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=2, bias=True)
)
```

Fig4. BiLSTM+CNN model structure

4.5.2 Model Performance:

Distilbert model is a relatively complex model that includes 60M parameters inside of it. It is a huge model and it is really likely to have a problem of overfitting.

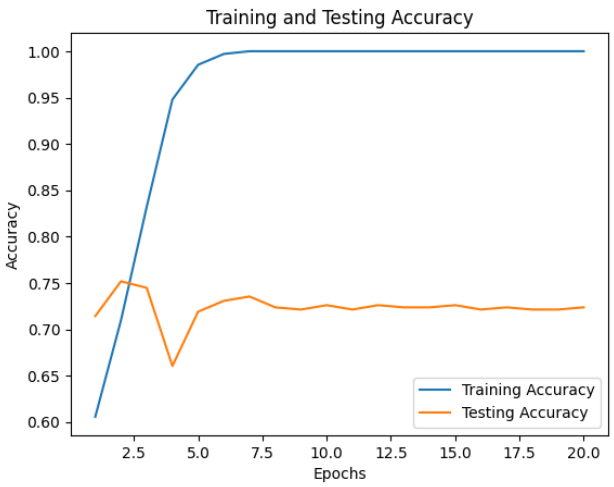

Fig5. Training and Testing Accuracy

Fig5. is the training result just with the learning rate parameter. It is really clear that the first several epochs of the model are improving both the training and testing accuracy however, after several epochs, the training accuracy is increasing to 1 while the testing accuracy is not improving a lot even declaring. This is because the DistilBert model is a relevantly complex model and is likely to overfit, so we need some feature tuning to improve its performance.

As the model already has a dropout layer, we decided to use l2 regularization to prevent overfitting. By setting the weight to 0.5, the training accuracy is 95.20% and the test accuracy is 71.89%. There is a slight improvement in using the l2 regularization method. However, the overfitting has not been thoroughly addressed. But as the training for Bert Model is time-consuming, we are running out of time to try new methods.

4.6 GPT API Prompt Model:

We first sample a small amount of data, since the API has a limited number of requests by time. We cannot use a large dataset. And with each post, we add a prefix with the question: “This post shows guilty emotion or not, please use yes no to answer” . Then, the API will return a sentence as the output. Usually, it is returning “yes” or “no” as the answer, and sometimes, it will refuse to answer the question, and send responses like: 'As an AI language model, I do not have the capability to judge emotions, hence I cannot determine whether the statement shows guilty emotion or not.' I will convert this as a positive which is a non-guilty emotion.

Table 4. Model Performance

Evaluation Methods	Performance
Accuracy	0.6261682242990654
Precision	0.5647058823529412
Recall	0.9411764705882353
F1-score	0.7058823529411765

Based on the evaluation scores shown in Table 4, we can know that the ChatGPT API has good performance on the True Positive, while it is not good at predicting True negative which is predicting the posts with guilty emotion. Looks like the overall performance for the ChatGPT model is not performing well on detecting the complex emotion, but it can easily identify the posts that do have that emotion.

4.7 Challenges:

In the whole process, as beginners in deep learning, we faced lots of challenges in the whole process:

4.7.1 Vent Dataset

Our first challenge arose from the considerable size of the Vent dataset we acquired. This dataset is big, containing a vast number of posts and associated emotional IDs. The original size of the dataset was (4359045, 7), making it computationally demanding and posing memory constraints. Unzipped, the Vent file amounts to 8.4 GB. Given that we conducted our program on Google Colab, we faced difficulties in storing and processing such a large dataset.

Another challenge was the inherent class imbalance within the dataset. Upon initial

exploration, we observed a significant disparity in the number of non-guilty posts compared to the number of guilty posts. Specifically, the dataset comprised 4,224,611 non-guilty posts and only 134,434 guilty posts. This class imbalance introduces bias and leads to overfitting when training the models, potentially resulting in inadequate performance on guilty posts.

We could see the impact of data imbalance specifically through the classification report in our baseline model in Fig 6:

	precision	recall	f1-score	support
0	0.09	0.72	0.15	26725
1	0.99	0.76	0.86	845084
accuracy			0.75	871809
macro avg	0.54	0.74	0.50	871809
weighted avg	0.96	0.75	0.84	871809

Fig 6. Classification report for baseline model

The data imbalance problem in this classification report is evident from the large difference in the number of instances between the two classes. This imbalance can lead to biased performance and a focus on the majority class. Addressing this imbalance is crucial to achieving a more balanced and accurate classification model.

To address these challenges, we employed the undersampling method. It involves reducing the number of samples from the majority class (in this case, non-guilty posts) to match the number of samples in the minority class (guilty posts). By randomly selecting a subset of samples from the majority class, undersampling helps create a more balanced dataset.

4.7.2 The model tuning and Complexity

The models we use have relatively complex structures. Like the Bert model we adopted, even with the relatively small distillery-base-uncased model, it still has 66M parameters.

The complexity of these models arises from their ability to capture intricate patterns and dependencies within the data, allowing for more sophisticated and nuanced analysis. However, this complexity also presents challenges during the tuning process. Fine-tuning such models necessitates extensive experimentation with hyperparameters, including learning rate, batch size, and regularization techniques, to achieve optimal performance.

Moreover, the increased number of parameters introduces computational complexities, requiring significant computational resources and time for training. Larger models with a greater number of parameters often demand powerful hardware infrastructure, including high-capacity GPUs or specialized processors, to effectively handle the computational load. Often we will wait for hours to test if our method works, which is time-consuming.

4.7.3 Hardware Not Supportable

As students without access to industry resources, we encountered challenges related to hardware limitations throughout our project. Specifically, our lack of hardware infrastructure capable of supporting extensive data processing and model training posed significant obstacles. To overcome this, we relied on Google Colab Pro for our model training. However, the storage capacity provided by Google Drive was insufficient for our needs. With a maximum storage limit of 15 GB, the dataset we worked with alone occupied 8.4 GB, with the zipped file itself amounting to 4 GB. This shortage necessitated additional payments to accommodate the dataset.

Moreover, the computational power of Google Colab was inadequate for our project requirements. While initially intending to leverage the Vent dataset's comprehensive labels and cleaned text, we encountered system crashes when attempting to tokenize the dataset using the BERT model. To mitigate this, we resorted to partial tokenization to fit within the system's available RAM. Unfortunately, even with this approach, we could only load two-fifths of the dataset before encountering system crashes. Consequently, we had to downsample the dataset to match the computational capacity required for our models.

In summary, our hardware limitations compelled us to rely on Google Colab Pro and undertake downsampling measures to overcome storage and computational constraints. Despite these challenges, we strived to maximize the utilization of available resources to ensure the successful execution of our training tasks.

4.7.4 The difference between industry and research

The distinction between industry and research becomes evident when considering renowned

models like BERT and GPT. These models have gained recognition for their exceptional performance. However, it is important to acknowledge the underlying factors contributing to their success: the extensive size of the models and the vast quantities of high-quality training data. Additionally, the hardware infrastructure required to support these models is often costly.

In industry settings, organizations typically possess the resources to develop and deploy large-scale models that yield impressive results. They have access to substantial computational power, enabling them to train and fine-tune models on massive datasets. This advantage is accompanied by the availability of substantial financial resources to invest in the necessary hardware infrastructure.

In contrast, research endeavors often face limitations due to limited access to such resources. Academic researchers, students like us, and those without industry affiliations encounter challenges when attempting to replicate the performance achieved by industry models. Constraints in computational power, access to extensive datasets, and financial resources often hinder the development and deployment of comparable models.

5. Ethical Considerations

Research involving the detection of guilt in text necessitates careful consideration of various ethical aspects. These considerations can be broadly categorized into two directions: dataset and model application.

5.1 Ethical Considerations for Vent Dataset

The first aspect pertains to the privacy and confidentiality of the dataset. As the dataset is sourced from the Vent social media platform, it is crucial to ensure that the data collection for this project is conducted with participants' consent and their privacy is adequately safeguarded. Therefore, it is necessary to take measures to anonymize the collected data, preventing the identification of individual participants. Moreover, the dataset should not contain any sensitive information or details that could be used to identify the users who authored the posts, thereby respecting their privacy.

Furthermore, as users of the dataset, we must also consider ethical implications. Since the dataset is private and requires the application for access, it is imperative to securely store the dataset to prevent unauthorized access. Moreover, we commit to refrain from including any results that could disclose the dataset's representative characteristics.

Additionally, it is important to emphasize the significance of analysis over outcome. Although our model may provide predictions on users' tendencies, it is important to acknowledge that immediate assistance may not be feasible, considering the limitations of the model's accuracy. Consequently, we should approach the data primarily as research data rather than practical data, prioritizing comprehensive analysis and meaningful insights.

By addressing these ethical considerations, we can ensure the protection of participants' privacy, securely handle the dataset, and maintain a research-oriented perspective when utilizing the data.

5.2 Ethical Considerations for Model Application

In addition to the data-related ethical considerations, the application of models also demands careful evaluation of their ethical implications. This entails assessing whether the intended applications align with ethical principles and safeguard the interests of individuals and society at large.

5.2.1 Ethical Considerations for GPT API Model

One of our implementations is using the GPT API and requesting the answer from the ChatGPT [22]. In the API call we made, we have to send a small sample of post text from the vent dataset into the ChatGPT. There is a probability that ChatGpt will use the input to improve its performance. However, by checking its announcement of OpenAi, we found that from March 1st, 2023, they will keep the data for 30 days, but not use it for training their model anymore. So this would not cause unauthorized data leakage.

While this addresses the ethical considerations related to the training phase, it is essential to extend ethical deliberations to the future application of this model. Users and implementers should consider the ethical use of the generated outputs and ensure alignment with ethical principles. It is crucial to assess the potential impact on individuals, society, and any potential biases or discriminatory outcomes that may arise from the model's application.

5.2.2 Ethical Considerations for Model Usage

The model usage could be applying our model for detection the of potential suicide or depression emotion on social media. However, using social media posts also needs to consider the user's privacy. When using the models from our training results, people should get permission to use their post data.

6. Conclusions and Future Work

Our project aimed to develop and compare various models for text classification, specifically focusing on detecting guilty emotions in online posts. Through extensive experimentation and analysis, we obtained valuable insights into the performance and challenges associated with each model.

Among the models we evaluated, the BiLSTM model with L2 regularization achieved the highest accuracy of 77.00%. The Logistic Regression model served as a baseline with an accuracy of 75%, while the CNN model demonstrated competitive performance with an accuracy of 71.20%. The BiLSTM+CNN model, incorporating an attention mechanism, achieved an accuracy of 76.36% after addressing overfitting through L2 regularization. The DistilBert model, limited by hardware constraints, achieved an accuracy of 71.89% after applying L2 regularization. The GPT API Prompt model showed strength in identifying guilty emotions but struggled with detecting non-guilty emotions.

Throughout the project, we faced challenges such as the large size of the Vent dataset, class imbalance, model complexity, and hardware limitations. We addressed class imbalance through undersampling, controlled overfitting through hyperparameter tuning and regularization techniques, and optimized available resources within the constraints of our hardware limitations.

Our findings underscore the importance of preprocessing, model architecture, and regularization techniques in achieving optimal performance. However, we recognize that there is still room for improvement. In the future, we plan to further enhance our models. For example, with the DistilBert model, we aim to tackle the issue of overfitting and explore the use of smaller Bert models to improve performance. Additionally, we have ideas for utilizing preprocessed tokens from Bert models in conjunction with other structures such as multi-layer perceptrons.

Due to time and resource constraints, we were unable to fully explore all the possibilities. Moving forward, we intend to continue refining our models by leveraging more powerful cloud services like AWS, which can provide greater computational power and storage capacity. This will enable us to experiment with larger models and datasets, ultimately improving the overall performance and accuracy of our text classification models.

References

[1] Meque, Abdul Gafar Manuel, et al. "Guilt Detection in Text: A Step Towards Understanding Complex Emotions." arXiv preprint arXiv:2303.03510 (2023).

[2] Acheampong, Francisca Adoma, Chen Wenyu, and Henry Nunoo‐Mensah. "Text‐based emotion detection: Advances, challenges, and opportunities." Engineering Reports 2.7 (2020): e12189.

[3] Balouchzahi, Fazlourrahman et al. “ReDDIT: Regret Detection and Domain Identification from Text.” ArXiv abs/2212.07549 (2022): n. pag.

[4] Adoma, Acheampong Francisca, Nunoo-Mensah Henry, and Wenyu Chen. "Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition." 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP). IEEE, 2020.

[5] Balouchzahi, F., Sidorov, G. & Gelbukh, A. Polyhope: Two-level hope speech detection from tweets, DOI: 10.48550/ARXIV.2210.14136 (2022).

[6] Schuster, M. & Paliwal, K. Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing 45, 2673–2681, DOI: 10.1109/78.650093 (1997).

[7] He, R.; Lee, W.S.; Ng, H.T.; Dahlmeier, D. An Unsupervised Neural Attention Model for Aspect Extraction. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, BC, Canada, 30 July–4 August 2017; pp. 388–397.

[8] Zhang J, Liu F, Xu W, et al. Feature fusion text classification model combining CNN and BiGRU with multi-attention mechanism[J]. Future Internet, 2019, 11(11): 237.

[9] Polignano M, Basile P, de Gemmis M, et al. A comparison of word-embeddings in emotion detection from text using bilstm, cnn and self-attention[C]//Adjunct Publication of the 27th

Conference on User Modeling, Adaptation and Personalization. 2019: 63-68.
[10] Rajabi Z, Shehu A, Uzuner O. A multi-channel bilstm-cnn model for multilabel emotion classification of informal text[C]//2020 IEEE 14th International Conference on Semantic Computing (ICSC). IEEE, 2020: 303-306.
[11] Haile, Sileshi Bogale et al. “Emotion Classification for Amharic Social Media Text Comments Using Deep Learning.” SSRN Electronic Journal (2022): n. pag.
[12] Wang, Z. et al. (2023) Is Chatgpt a good sentiment analyzer? A preliminary study, arXiv.org. Available at: https://arxiv.org/abs/2304.04339 (Accessed: 17 May 2023).
[13] Lykousas, N., Patsakis, C., Kaltenbrunner, A. & Gómez, V. Sharing Emotions at Scale: The Vent Dataset. Proc. Int. AAAI Conf. on Web Soc. Media 13, 611–619, DOI: 10.1609/icwsm.v13i01.3361 (2019).
[14] Mubeen, S. Mohammad Malik et al. “Linguistic Based Emotion Detection from Live Social Media Data Classification Using Metaheuristic Deep Learning Techniques.” International Journal of Communication Networks and Information Security (IJCNIS) (2022): n. pag.
[15] Abbaschian, Babak Joze et al. “Deep Learning Techniques for Speech Emotion Recognition, from Databases to Models.” Sensors (Basel, Switzerland) 21 (2021): n. pag.
[16] Wang, Z. et al. (2023) Is Chatgpt a good sentiment analyzer? A preliminary study, arXiv.org. Available at: https://arxiv.org/abs/2304.04339 (Accessed: 17 May 2023).
[17] Balahur, A., Hermida, J. M. & Montoyo, A. Detecting implicit expressions of sentiment in text based on commonsense knowledge. In Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2.011), 53–60 (Association for Computational Linguistics, Portland, Oregon, 2011).
[18] Bert¶ (no date b) BERT - transformers 3.3.0 documentation. Available at: https://huggingface.co/transformers/v3.3.1/model_doc/bert.html (Accessed: 17 May 2023).
[19] Distilbert¶ (no date) DistilBERT - transformers 3.3.0 documentation. Available at: https://huggingface.co/transformers/v3.3.1/model_doc/distilbert.html (Accessed: 17 May 2023).
[20] OpenAI GPT¶ (no date) OpenAI GPT - transformers 3.3.0 documentation. Available at: https://huggingface.co/transformers/v3.3.1/model_doc/gpt.html (Accessed: 17 May 2023).
[21] M. Munikar, S. Shakya and A. Shrestha, "Fine-grained Sentiment Classification using BERT," 2019 Artificial Intelligence for Transforming Business and Society (AITB), Kathmandu, Nepal, 2019, pp. 1-5, doi: 10.1109/AITB48515.2019.8947435.
[22] OpenAI API. Available at: https://platform.openai.com/docs/guides/chat (Accessed: 17 May 2023).