Varun Chilukuri, Rahul Nair, Anirudh Poruri, Shreya Shete
CMSC 421
Final Project Report
9 May, 2024

# Using Sentiment Analysis to Detect Online Hate Speech

[Link to Github]

## I. Literature Review

The following section will introduce and highlight existing research related to Online Hate Speech Detection. Each source helped direct our project scope and our solution's implementation.

### Motivation and Key Definitions

The U.S. Government Accountability Office [1] (USGAO) defines hate speech as "prejudiced comments about race, national origin, ethnicity, gender, gender identification, religion, disability, or sexual orientation."

After extremist attacks in Charleston and El Paso, the USGAO also concluded that the Internet has served as a vehicle for "disseminating hateful materials- such as manifestos containing disparaging and racist rhetoric prior to the attacks."

Experiencing hate speech takes a toll on youth. Julia Kansok-Dusche et al. [2] found that youth who interact "online hate speech (cyberhate) commonly experience negative feelings" such as anger, sadness, or shame.

The burden falls on these social media platforms to monitor, flag, and remove perpetrators of online hate speech. Manual identification of hate speech is time-consuming, costly, and introduces variability/human bias. Our team was motivated to create a model that uses sentiment analysis to detect hate speech within text.

### Previous Research

In 2023, Yadav et al. [3] found that LSTM deep learning techniques with word embeddings can achieve over 92% performance and outperform machine learning detection methods. This inspired us to look into various deep learning methods: Specifically we looked into CNN, LSTM, and Bidirectional LSTM (Bi-LSTM) models.

Mahadevaswamy, U.B., and Swathi P. [4] evaluated these three methods for sentiment analysis. They included steps for pre-processing and tokenizing the data before passing it to the neural network. Their evaluation revealed that Bi-LSTM outperforms CNNs and simple LSTMs in terms of classification accuracy for sentiment analysis. Specifically, while CNNs achieved an accuracy of 86.28%, and simple LSTMs achieved 85.74% accuracy, the Bi-LSTM model achieves a notably higher accuracy of 91.4% when classifying (Positive, Negative).

Bi-LSTM (Bidirectional Long Short-Term Memory) models are a type of RNN (Recurrent Neural Network) that processes sequential data, like text, in forward and backwards directions simultaneously. Huan, Hai et al. [5] noted that Bi-LSTM models effectively capture broader contextual meaning. Our team believes that this can help distinguish nuances when words have multiple interpretations (depending on their context).

We were inspired to implement our own Bi-LSTM model with the same text preprocessing and tokenization conventions as [4]. We wanted to see how this model, which predicts on three classes (Hate Speech, Offensive Language, Neither), would perform on medium length comments (i.e. youtube comments, Tweets, etc.).

## II. Introduction to the Problem

Millions of people use social media platforms like Youtube, Reddit, and Instagram every day, a large population of which are impressionable youth. Certain things are not okay to say, and it is unacceptable that young children are exposed to such vulgarity.

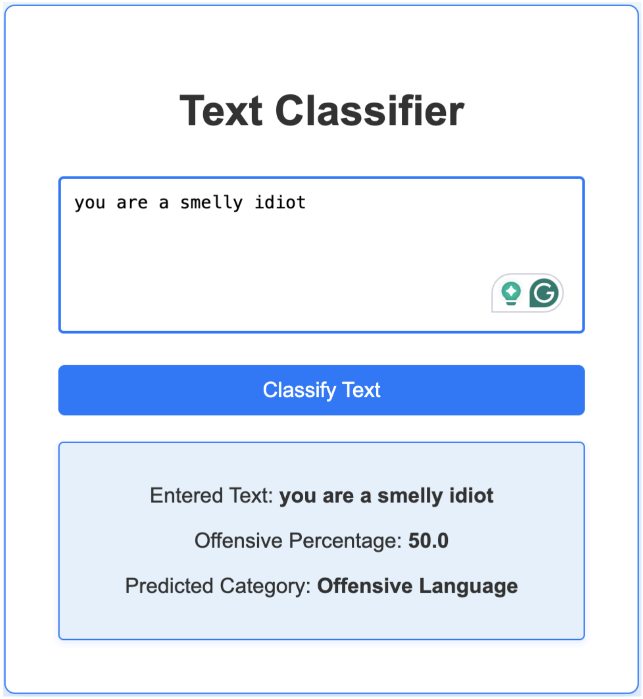
There are many real world applications for our model. For example, any social media platform can use it to identify hateful comments that should be removed, thereby improving the safety and user experience of their respective platform. Additionally, our model can help to flag users who repeatedly post hateful comments, ultimately banning such users, in efforts to further minimize toxicity online. Using our model, we can help make the internet a safer place for all users.

However, hateful comment detection remains a challenging problem. Hate speech is nuanced and heavily context-dependent. Words that are neutral or positive in one context can just as easily be used hatefully in another context. Furthermore, it is difficult to balance sensitivity and specificity—overly aggressive models may result in high false positives, impacting free speech and causing user dissatisfaction, whereas very conservative models can let hate speech go unfiltered.

## III. Initial goals vs. achieved results:

Our initial goals for this project were to (1) identify hateful/offensive comments so they can be removed on these platforms and (2) develop a user-friendly tool that utilizes our model's results to classify comments.

We were indeed able to accomplish this. Our custom model achieved an overall classification accuracy of 95.46% on the validation set. In addition, we built two tools that allowed users to use our trained model to identify hateful comments on social media—which went beyond our initial goal of just one tool. The first of these tools is a simple text classifier. Users can enter any text and our model will return an offensive percentage score and classify it accordingly:



```
you are a smelly idiot
```

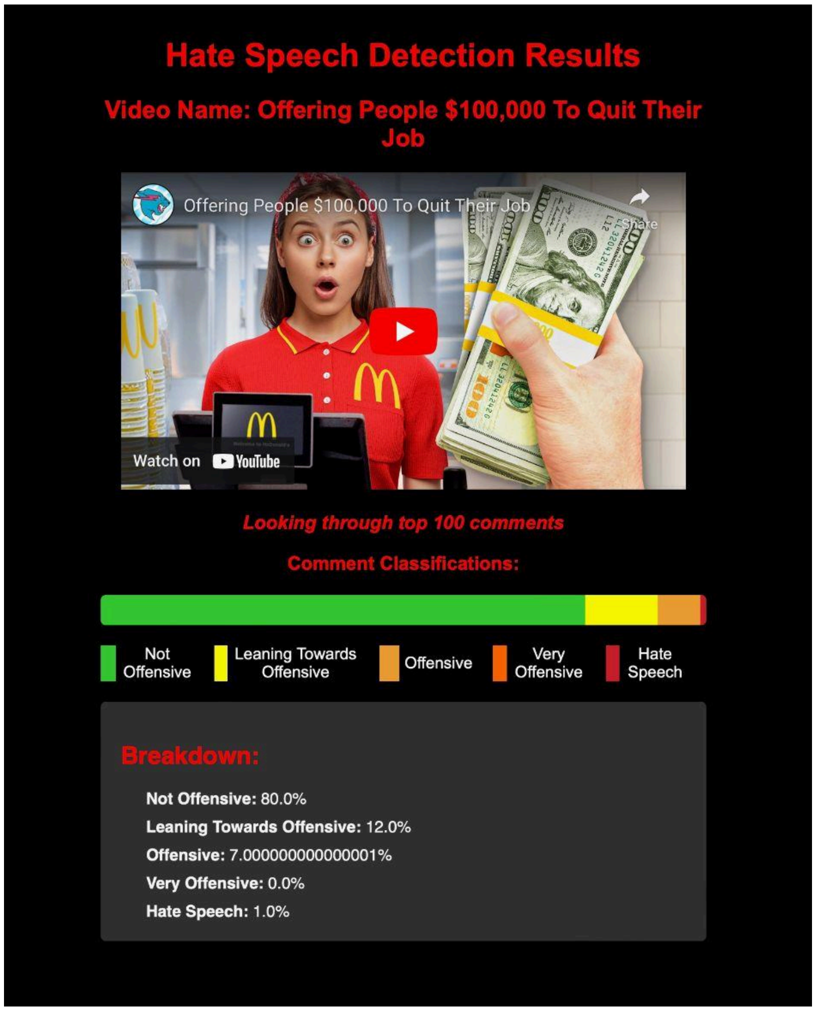
Entered Text: **you are a smelly idiot**

Offensive Percentage: **50.0**

Predicted Category: **Offensive Language**

**Application 1: Hate Speech Detector**

Our second application analyzes and classifies the top 100 comments of any given YouTube video to see if hate speech is present. We got the following results when we ran it on a video titled "Offering People $100,000 To Quit Their Job" by Mr. Beast:


Hate Speech Detection Results

Video Name: Offering People $100,000 To Quit Their Job

Looking through top 100 comments

Comment Classifications:

🟩 Not Offensive
🟨 Leaning Towards Offensive
🟧 Offensive
🟧 Very Offensive
🟥 Hate Speech

Breakdown:

**Not Offensive:** 80.0%
**Leaning Towards Offensive:** 12.0%
**Offensive:** 7.000000000000001%
**Very Offensive:** 0.0%
**Hate Speech:** 1.0%

**Application 2: Youtube Comments Analyzer**

As seen above, YouTube has not completely filtered out all hate speech. A large portion of Mr. Beast's channel audience includes children, meaning that they are being exposed to any hateful comments that pass through YouTube's filtering system.

## Unexpected findings or changes in direction

We did have a change in direction. Initially, we planned to implement a Convolutional Neural Network (CNN) as our model. However, while reviewing modern literature, we realized that a Bi-LSTM is better for our use case of context-based text classification. A Bidirectional LSTM Model is capable of learning long term dependencies in sequence data and processes data via an internal state similar to memory."

Our initial proposed dataset was ucberkeley-diab: Measuring Hate Speech. This dataset labelled text with a "Hate Speech Score," with a range of values being considered hate speech, but we had to switch datasets since this first one improperly labeled hate speech values as "neutral."

## IV. Solution to the problem (strengths, weaknesses, and limitations)

### What Exactly is Our Solution

Our solution is a bidirectional long short-term memory model that predicts on three classes ("Hate Speech", "Offensive Language", and "Neither"). The model outputs a probability distribution, which we then use to compute a score ranging from 1-100. The reason we did this was to capture nuances amongst various probability distributions. Doing such prevents us from outright classifying close calls (based on the majority probability) which is important if this model were to be onboarded on a solution that automatically flags hate speech. A score of 0-20 is not offensive, 20-40 is leaning towards offensive, 40-60 is offensive, 60-80 is very offensive, and 80-100 is hate speech.

### Strengths, Weaknesses, Limitations

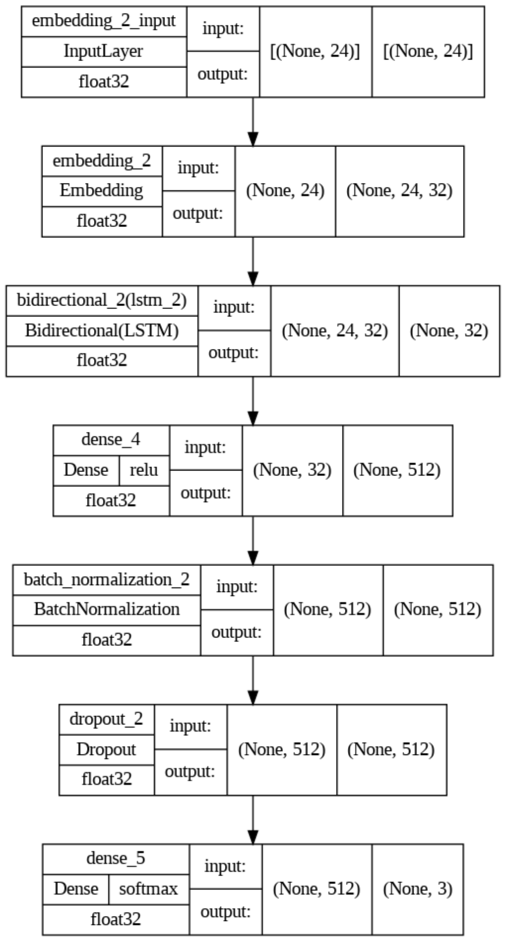
The strengths of our solution are that profanity and true hate speech is classified well along with friendly and supportive text. Another strength is that we also achieved a high accuracy and built two user-friendly tools that allow us to analyze individual comments or of an entire YouTube video. As a result, our project has really strong real-world applications.

However, a weakness is that mildly offensive text is classified more inconsistently. For example, our model wasn't confident that a phrase like like "you suck" was offensive, even though it obviously is. A few phrases were also classified way too extremely and adding negations to statements also resulted in incorrect classifications. An example is "you are not an idiot", which was classified as "offensive language" when it was neither offensive nor hateful. This is likely due to the model weighing the word "idiot" heavily when doing the classification.

Limitations of our project include that we were only able to train with one dataset, thus our model cannot provide meaningful insights beyond the scope of the data contained in one dataset. In addition, we also saw some overfitting meaning that the model might be too specific in its classification and has difficulty generalizing. Finally, there are slight inherent biases in the dataset, leading the model to amplify these biases in its predictions. This is why it's beneficial to have multiple datasets.

### How our solution works and how we developed it

The dataset we chose for this project is the Hate Speech and Offensive Language Dataset from Kaggle. This dataset contained a column with the Tweets, and a column containing a classification of the Tweet, either 0 for hate speech, 1 for offensive language, or 2 for neither hate speech or offensive language. Data preprocessing was done first. This involved balancing the dataset by upsampling hate speech and offensive language. Then, we removed punctuation and stop words. Afterwards, we split text into words and employed lemmatization to break down words into their root forms. Next, we tokenized the sequences, converting them into integers. Finally, we padded sequences to have the same length. A Bidirectional LSTM Model was developed as our solution. The input for the model was a 1D sequence of padded tokens and output was the probability for each class: Hate Speech, Offensive Language, Neither. Below is our model architecture:


embedding_2_input | InputLayer | float32 | input: [(None, 24)] | output: [(None, 24)]
embedding_2 | Embedding | float32 | input: (None, 24) | output: (None, 24, 32)
bidirectional_2(lstm_2) | Bidirectional(LSTM) | float32 | input: (None, 24, 32) | output: (None, 32)
dense_4 | Dense | relu | float32 | input: (None, 32) | output: (None, 512)
batch_normalization_2 | BatchNormalization | float32 | input: (None, 512) | output: (None, 512)
dropout_2 | Dropout | float32 | input: (None, 512) | output: (None, 512)
dense_5 | Dense | softmax | float32 | input: (None, 512) | output: (None, 3)

The input layer is an embedding layer for creating word vectors, [None, 24], where None represents any batch size and takes in a sequence of integers size 24. The second embedding layer transforms each integer input (representing words) into dense vectors of fixed size 32. The third layer, a Bidirectional LSTM Layer, processes the sequence in forward and backward directions simultaneously, outputting a 32-dimensional vector that summarizes the sequence from both directions. The fourth layer is a fully connected layer where each input is connected to the output by a learned weight, expanding the representation to 512 dimensions.

The ReLU activation function was applied to introduce non-linearity, allowing it to learn more complex patterns. A Batch Normalization Layer for normalization and to stabilize and accelerate the training process. Next, we added a dropout layer which implements L2 regularization and randomly selects 30% of neurons to ignore during training in order to prevent overfitting. Essentially, downstream neurons are temporally not activated on forward pass and weight updates are not applied to these neurons on backward pass. This results in the same 512-dimesnional vector but with 30% of values set to zero.

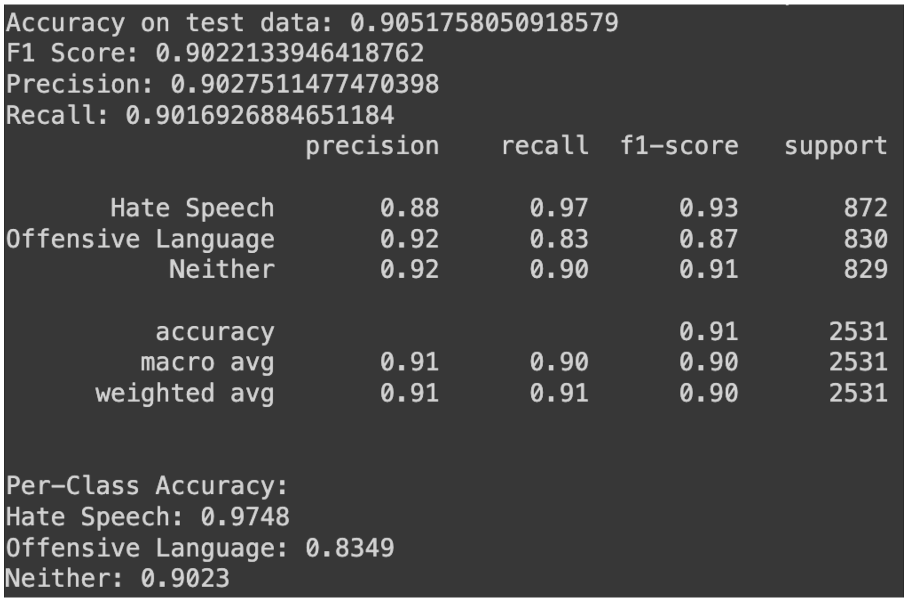
The output layer is a dense fully connected layer that reduces dimension from 512 to 3, which is the number of output classes. The softmax activation function was used since it is good for multi-class classfication, outputting a probability distribution over the 3 classes.

The model was then trained using the training set and performance was measured on the validation set. We used a callback function to monitor and adjust the learning rate if the loss wasn't improving after 2 epochs, as well as one to stop early if the validation accuracy wasn't improving after 5 epochs. Accuracy and Loss was plotted for both training and validation data over epochs. Overall accuracy, per-class accuracy, precision, recall, F1 score for Test data.

## Pretrained models or existing libraries

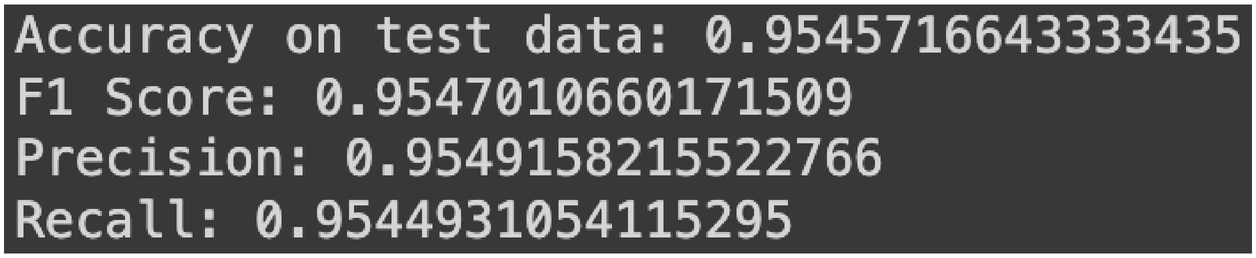
For this project, we utilized several existing libraries, including TensorFlow, Keras, Numpy, Pandas, NLTK, Scikit-learn, and Matplotlib, due to their capabilities in machine learning and natural language processing tasks. TensorFlow and Keras were chosen for their powerful deep learning frameworks, allowing for efficient, straightforward model building and training, particularly with building the RNN and using LSTM and Dense layers. Numpy and Pandas were essential for efficient data pre-processing and analysis. NLTK was used for text processing tasks such as tokenization and stopword removal, which helped feed data into our model. Scikit-learn provided essential tools for model evaluation and performance metrics, ensuring our models were accurately assessed. Finally, Matplotlib was used for visualizing data and training progress, making it easier to interpret model performance.

## V. Results and Evaluation



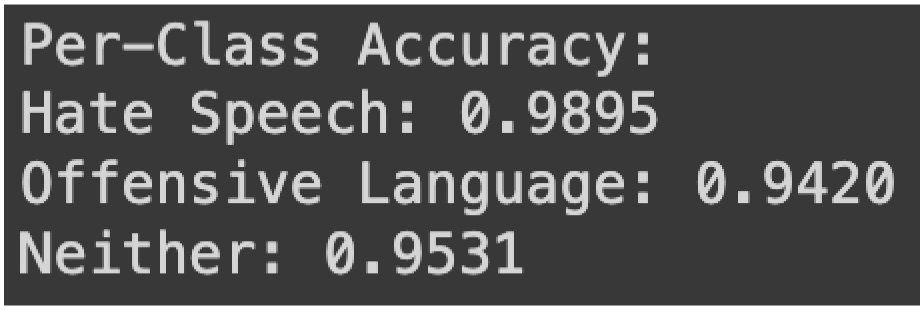
```
Accuracy on test data: 0.9051758050918579
F1 Score: 0.9022133946418762
Precision: 0.9027511477470398
Recall: 0.9016926884651184
                    precision    recall  f1-score   support

       Hate Speech       0.88      0.97      0.93       872
Offensive Language       0.92      0.83      0.87       830
           Neither       0.92      0.90      0.91       829

          accuracy                           0.91      2531
         macro avg       0.91      0.90      0.90      2531
      weighted avg       0.91      0.91      0.90      2531


Per-Class Accuracy:
Hate Speech: 0.9748
Offensive Language: 0.8349
Neither: 0.9023
```

*Figure 1. Overall & Per Class Metrics After Initial Downsampling of "offensive language".*



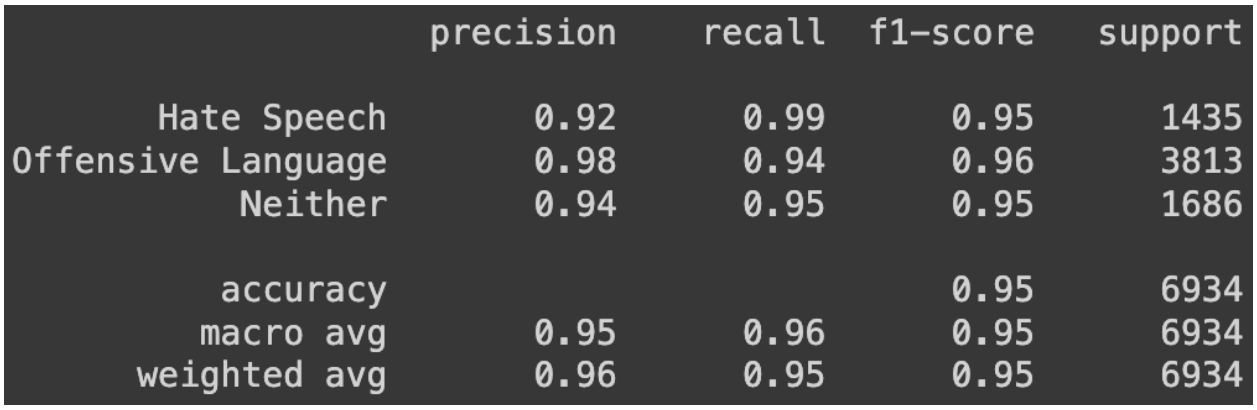
```
Accuracy on test data: 0.9545716643333435
F1 Score: 0.9547010660171509
Precision: 0.9549158215522766
Recall: 0.9544931054115295
```

*Figure 2. Overall Metrics. This displays the Accuracy, F1 Score, Precision, Recall, and support values for the whole validation set.*



```
Per-Class Accuracy:
Hate Speech: 0.9895
Offensive Language: 0.9420
Neither: 0.9531
```

*Figure 3. Per Class Accuracy Comparison***.** *This displays accuracies for classification for all three classes: Hate Speech, Offensive Language, Neither.*



```
                    precision    recall  f1-score   support

       Hate Speech       0.92      0.99      0.95      1435
Offensive Language       0.98      0.94      0.96      3813
           Neither       0.94      0.95      0.95      1686

          accuracy                           0.95      6934
         macro avg       0.95      0.96      0.95      6934
      weighted avg       0.96      0.95      0.95      6934
```

*Figure 4. Per Class Metrics Comparison.* This figure displays the F1 Score, Precision, Recall, and support values for all three classes: Hate Speech, Offensive Language, Neither in our final model

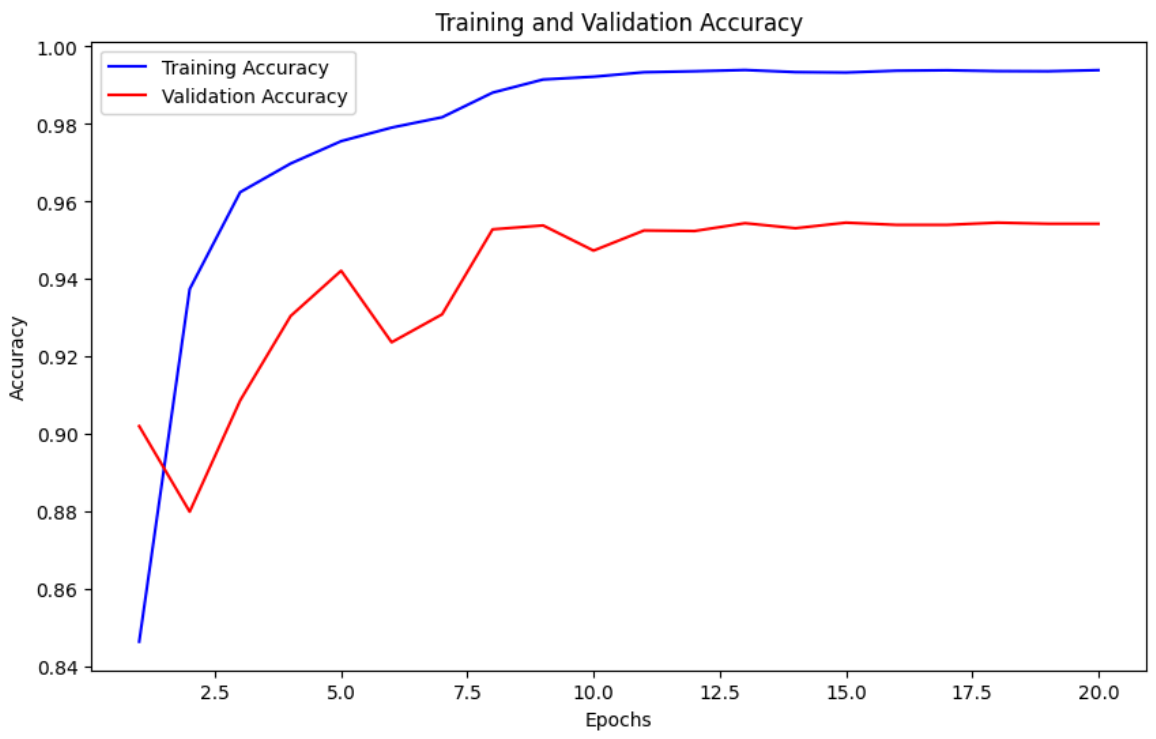


*Figure 5. Training and Validation Accuracy Over Epochs.* This graph illustrates the improvement in accuracy of our model on training and validation sets, as the number of epochs increases.

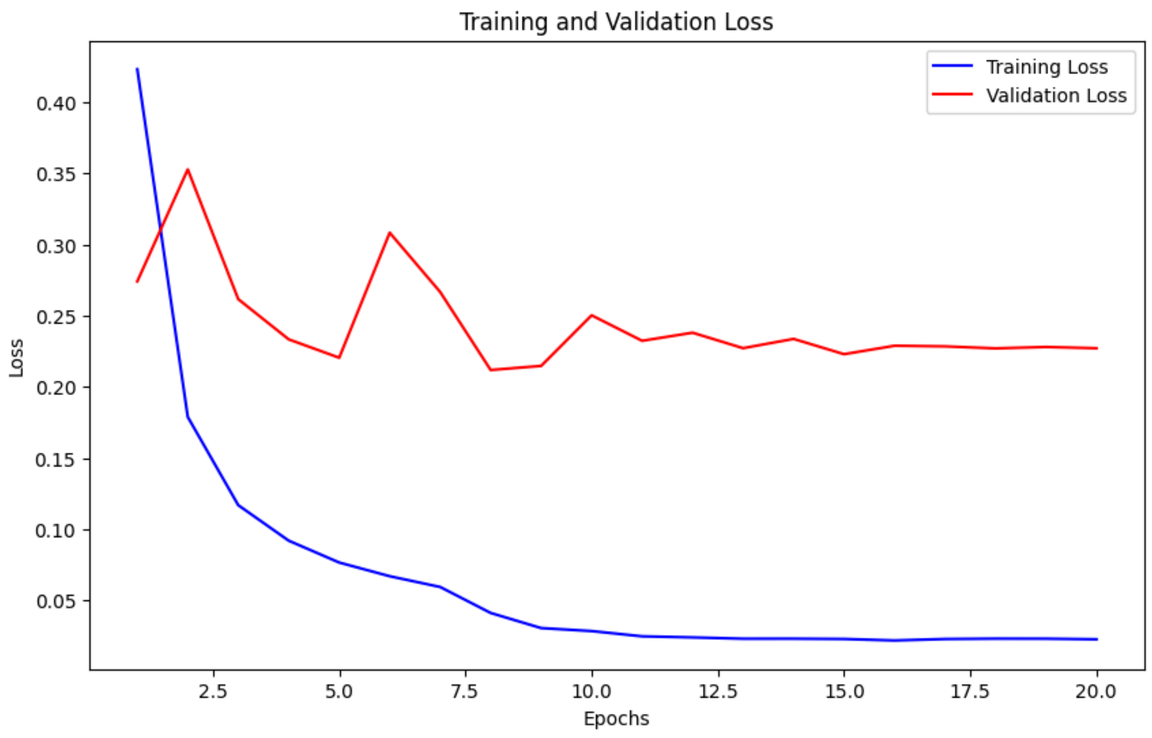


*Figure 6. Training and Validation Loss Over Epochs.* This graph illustrates the decrease in loss of our model for training and validation sets, as the number of epochs increases.

Figure 2 displays the overall metrics for accuracy, F1 Score, Precision, Recall. We found an overall accuracy of 95.46% on our validation dataset. This suggests that our pre-processing of data was good and that the LSTM Model we built was well tuned for its classification task. In addition, our model also had high classification metrics such as a F1 score of 95.47%, Precision of 95.49%, and Recall of 95.45%, indicating that our model performed well. Our high F1-score shows that there is a good balance between precision and recall, meaning our model gives good results for an imbalanced classification problem.

Figure 3 displays the accuracies per class with classification accuracies for Hate Speech, Offensive Language, and Neither being 98.95%, 94.20%, and 95.31% respectively.  Figure 4 portrays our final model's per class metrics for F1 Score, Precision, Recall, and Support. Support is the actual number of occurrences of each class in the dataset with the highest number of occurrences for "Offensive Language" , which was 3813. There were 1686 occurrences for "Neither", and 1435 occurrences of "Hate Speech".

Figure 5 displays the training and validation accuracy over epochs, illustrating the increase in both accuracies over 20 epochs. It shows that the model trains to nearly 100% accuracy on the training data and that the validation accuracy at the end is lower than the training accuracy, hinting at slight overfitting. There is also an initial decrease in validation accuracy over the first three epochs. Figure 6 displays the training and validation losses over 20 epochs. The training loss at the end is lower than the validation loss, which is also another sign of slight overfitting. There is also an initial increase in validation loss over the first three epochs.

## Unexpected outcomes or challenges faced during evaluation

We noticed that our dataset (see tweets.csv) was unbalanced. 77% of the records were labeled as "Offensive" language, whereas "Hate Speech" and "Neither" made up 6% and 17% of rows respectively. We thought that would lead to the model developing a bias towards offensive language when classifying text. To avoid this, we downsampled offensive language, and upsampled hate speech by 3 times. This left us with a roughly even (33% each class) split.

After training the model and evaluating it's per-class accuracy, we saw that downsampling "Offensive" data led to offensive language being classified less accurately (see Figure 1).

Our team theorized that hate speech is classified most accurately because there are a limited number of words or phrases that can be hate speech (profanity, slurs, etc.) Conversely, offensive language has a very wide range. Something as simple as "you suck" can be considered offensive, as well as something as extreme as profanity, depending on the context of use. In our final solution, we decided to not downsample the offensive language category, since we evidently needed more offensive language data to better classify it.

We experimented with different factors for the upsampling of "Hate Speech" and "Neither". For the latest trained model, we decided on upsampling "Hate Speech" by a factor of 5 times and "Neither" by a factor of 2. This led to much better accuracy per class and also overall, at 95 percent (see results).

One unexpected result in our project was the initial decrease in validation accuracy (Figure 5) and initial increase in validation loss (Figure 6) over the first three epochs. A possible explanation for this is the model complexity, causing the model to struggle in generalizing to the validation set initially but improving as training progresses. Another possible explanation could be that the learning rate scheduler or optimizer that we implemented may have caused a rapid exploration of parameters, leading to sharp initial decreases in performance on the validation set.

However, as training continues, the learning rate may stabilize, allowing the model to converge to a better solution. We also didn't expect to have overfitting indicated by the training accuracy at the end being 4% higher than the validation accuracy (Figure 5) and the training loss at the end (Figure 6) being lower than the validation loss by 0.2. Eliminating overfitting was a significant challenge in improving our model, though for the most part, we were successful. Strategies such as early stopping, learning rate scheduler, batch normalization, and L2 regularization via dropout were used.

An unexpected positive outcome in our project was the development of two applications, the Simple Text Classifier and Youtube Comment Section Analyzer. These tools implemented our model and provided qualitative, specific feedback on the performance of our model, which was useful.

A challenge we faced was having to switch datasets. We had to find a new one since our initial dataset, ucberkeley-dlab/measuring-hate-speech, improperly labeled several obvious hate speech records as “neutral”. This was a significant roadblock since we had already preprocessed the dataset and developed two models, another LSTM and Quantum Neural Model with Qiskit. Switching to our new dataset required significant preprocessing and manipulation of layer sizes and training parameters.

## VI. Member Contributions

All team members worked together to create and fine-tune the Bi-LSTM model proposed in this report, which was truly a collaborative effort. Anirudh found the final dataset, pre-processed/explored data, and created the “Text Classifier” tool. Rahul worked on data exploration for both datasets and created the “Youtube comment section Analyzer” tool that uses the model. Shreya worked on quantum model for the initial dataset, and data metrics for the final deliverable. Varun worked on preprocessing and the LSTM model for initial dataset.

## VII. Bibliography

1. Office, U. S. Government Accountability. “Online Extremism Is a Growing Problem, but What’s Being Done about It? | U.S. GAO.” Www.gao.gov, 13 Feb. 2024, www.gao.gov/blog/online-extremism-growing-problem-whats-being-done-about-it#:~:text=Online%20hate%20speech%20is%20widespread.

2. Kansok-Dusche, Julia et al. “A Systematic Review on Hate Speech among Children and Adolescents: Definitions, Prevalence, and Overlap with Related Phenomena.” Trauma, violence & abuse vol. 24,4 (2023): 2598-2615. doi:10.1177/15248380221108070

3.Yadav, Dharmveer, et al. “Comparative Analysis and Assesment on Different Hate Speech Detection Learning Techniques.” Journal of Algebraic Statistics, vol. 14, no. 1, Jan. 2023, pp. 29–48. EBSCOhost, search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=asn&AN=164596172&site=ehost-live.

4. Mahadevaswamy, U.B., and Swathi P. "Sentiment Analysis using Bidirectional LSTM Network." Procedia Computer Science, vol. 218, 2023, pp. 45-56, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2022.12.400.

5. Huan, Hai, et al. “A Text Classification Method Based on a Convolutional and Bidirectional Long Short-Term Memory Model.” Connection Science, vol. 34, no. 1, Dec. 2022, pp. 2108–24. EBSCOhost, https://doi-org.proxy-um.researchport.umd.edu/10.1080/09540091.2022.2098926.