Overview:

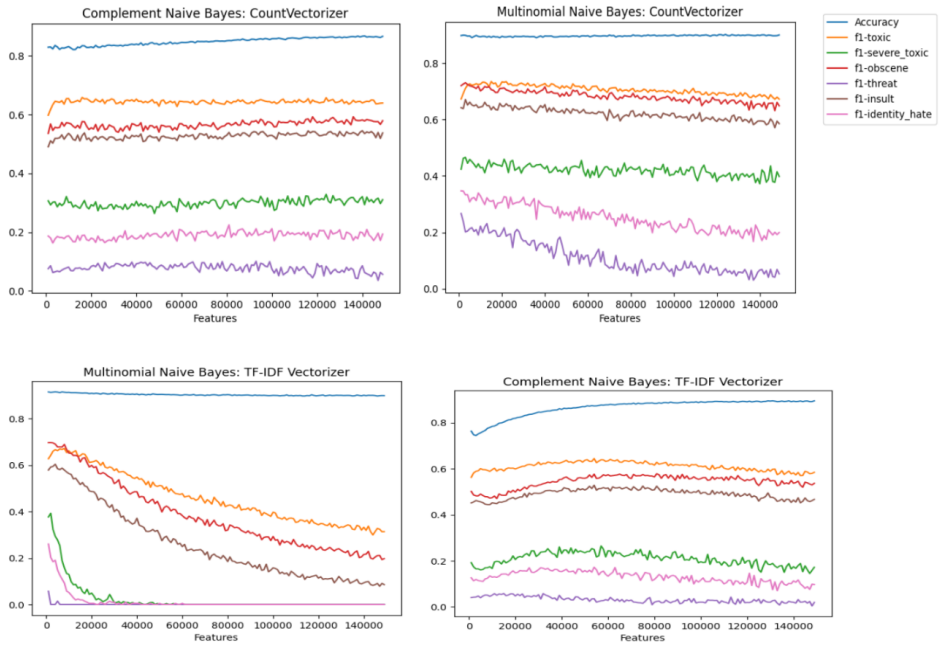
This project investigates the performance of three approaches: Naive Bayes, a simple yet powerful text classification technique, Recurrent Neural Networks (RNNs) & Long Short-Term Memory (LSTMs), which capture the sequential nature of language, and fine-tuning BERT, a pre-trained language model with remarkable natural language understanding capabilities. Each model is evaluated on the "Jigsaw Toxic Comment Classification Challenge" dataset, analyzing their strengths and weaknesses in identifying different types of toxic content, including threats, obscenity, and hate speech.

Initial Preprocessing:

The comments for both the train and test dataset were processed by lower casing, removing special characters, and stop words (words not useful for classification), followed by tokenization,and lemmatization to ensure that similar meaning words are considered as the same instance. An aim of our experiments was to investigate effects of data imbalances and so undersampling of the non-toxic data to create relatively balanced datasets (train and test). This was necessary also since the toxic: non-toxic ratio was too low (great imbalance). Due to these imbalances, F1-score and AUC-ROC were the primary metrics for testing model performance rather than accuracy.

Model 1: Naive Bayes & Sparse Representations:

Naive Bayes is a simple probabilistic classifier that employs the naive assumption of independence of features and uses conditional probabilities of features given a class to predict the probability of an instance belonging to a class. To motivate the use of Naive Bayes we first used a simple architecture with One-Hot Vectors and Bernoulli Naive Bayes. The results were a high accuracy (85.29%) albeit poor F1 scores. We then employed other types of sparse vectorizers like TF-IDF and Count as those provide more information. The NB to be employed was adjusted accordingly since our train features don't remain binary (they may be but not necessarily) any longer. Correspondingly, we use TF-IDF Vectors and Count Vectors sparse representations. . The corresponding Naive Bayes used were Multinomial NB assumes that the features (word counts or term frequencies) follow a multinomial distribution, and Complement NB assumes that features are conditionally independent given the class.It is designed to address the issue of imbalanced class distribution. The effect of feature size (maximum number of features) was also investigated simultaneously by running experiments testing different permutations of NB, Vectors and maximum feature sizes as shown in the graphs below:



We expected the ComplementNB to perform better since it is resilient to data imbalances, however unexpectedly MultinomialNB performed better on both accuracy (by ~2%) and f1-scores (by ~6%). A quick observation of the graphs also tells us that the median values of feature vectors are suitable for good F1-Scores except for TF-IDF with Multinomial which decays rapidly, and therefore, for balanced datasets we employed this median range as well.

We then tested the effect of balanced sets, by running the different permutations as before over balanced training sets but over a smaller feature range (40,000 to 80,000). The results indicated stability over this feature range, but more importantly very high validation F1 scores were obtained, the likes of which weren't obtained in any other model. This suggested balanced datasets are indeed more valuable, or at least equally as valuable as large datasets for good training.

The optimal Naive Bayes model obtained (MultinomialNB with CounteVectors and 70,000 features) was then tested on both balanced and unbalanced testing data (while using balanced training data). We calculated the accuracy, AUC and F1 scores for all the different categories.After calculating all this, we took a weighted average for all these metrics. The results for balanced training and testing data are shown below:

```
Average AUC: 80.761 %
Average F1-Score: 71.392 %
Average Accuracy: 91.306 %
Weighted Average AUC: 80.519 %
Weighted Average F1-Score: 70.138 %
Weighted Average Accuracy: 90.401 %
```

Balanced subsets of training data trained on balanced subsets of test data gave the best results. This hinted towards the importance of balanced data. A slight digression on our end was to try and employ dense representations (which completely violates the NB assumptions) and investigate the results (which were poor as expected). This reinforced that it is not best to use Naive Bayes with such representations and instead other models like RNN and LSTMs are to be used for such vector representations.

Model 2: RNNs & LSTMs - Dense Representations (GloVe and Word2Vec):

We then employ pre-trained dense representations (embeddings) using Word2Vec and GloVe vectorization to find out how these results would hold up against our previous results where we used the Naïve Bayes model and sparse representations. After generating balanced datasets, like we did previously, we then used pre-trained embeddings in our RNN and LSTM models. The primary motivation behind using these sequential models was their ability to capture long semantic and contextual dependencies which is key for sentiment analysis/ toxicity classification.

Using RNN:

In our initial experiments, we utilized basic RNN architectures. The model's first layer was an embedding layer, constructed from pre-trained embeddings and tailored to the vocabulary of each instance. This was succeeded by a SimpleRNN layer comprising 256 units and employing the ReLU activation function. During training, we tracked Loss, F1-Score, and Accuracy. A learning rate of 0.01 was maintained, yet we employed the ReduceLROnPlateau method to adjust the learning rate dynamically, mitigating plateaus in loss for smoother learning. Larger learning rates led to escalating losses, while smaller rates slowed training. We also implemented EarlyStopping, designed to halt training if the loss stagnated across consecutive iterations. However, our model was trained for only 5 epochs (other models for 2 or 3 epochs) due to observed overfitting indicated by rising validation loss, and declining val_f1 and val_accuracy, rendering EarlyStopping unnecessary in this scenario.

Discussion on Results for RNN:

We tested our RNN model with Word2Vec vectorization first and found a much greater accuracy than our previous Naïve Bayes model, however the F1-Score was significantly less. This was further investigated when measuring the test F1-scores and AUC. The results for the test data were also similar and are shown below:

```
Test Accuracy: 0.9633
toxic :
F1 score: 14.138 %
severe_toxic :
F1 score: 0.000 %
obscene :
F1 score: 21.017 %
threat :
F1 score: 0.000 %
insult :
F1 score: 15.250 %
identity_hate :
F1 score: 0.000 %
Test AUC: 0.8697
```

The accuracy and AUC increase significantly from our NB classifier, yet the F1-scores are terrible. We repeat this by using GLoVe in our RNN model and found a major improvement in our results. The accuracy increased to 99% while the F1 score, which was almost 0 as can be seen in the figure pasted above, now increased to around 40% to 45% which shows substantial growth over our previous experiment where we were using Word2Vec. The model still struggled on severe_toxic classification giving an F1 of 0 for this class alone. The improved results from GloVe may be due to the higher embedding dimension leading to higher number of parameters to train and the implicit difference is in the quality of embeddings (becomes apparent later on). This motivates the importance of pretrained embeddings and the right ones being used for different use cases. We also investigated the effects of balanced and imbalanced datasets with the same RNN model which was inconclusive since certain F1-scores (for certain classes) were higher for either of the two datasets. We therefore moved to LSTMs for further investigation.

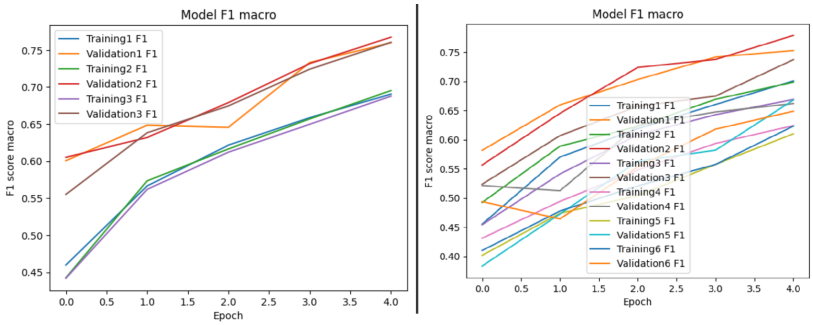
Using LSTM:

We now moved over to the LSTM model to test our dataset and find any conclusive results, if any. Our LSTM architecture was structured with an embedding layer, a bidirectional LSTM layer, and a final output layer generating probabilities for six classes through softmax. Training settings mirrored those of the RNN, but we included a dropout factor in the LSTM layer to prevent overfitting by randomly deactivating a fraction of neurons during training.

To explore optimal model depth, we conducted experiments with six variations, each incorporating an additional dense layer with the same units and ReLU activation function. Our aim was to assess the impact of increased network depth on the loss and F1 scores. Deep architectures, known for excelling with intricate data, prompted this investigation. A further set of similar experiments were carried out for investigating the effect of wide layers (once the depth (number of layers)) was decided empirically.

Discussion on Results for LSTM:

We only used GLoVe embedding in our LSTM model and then plotted our results for each variation of the model: the 1st model started off with 3 layers while the 6th final model had 8 layers. The results are shown below:



Effect of Layer Width (No. of Units) Effect of Number of Layers (Depth)

As can be seen, two projection layers of 128 neurons lead to the highest the validation Macro F1-score as well as the steepest increase in Macro-F1 score with epochs/ fastest descent of loss with epochs. This implies that this is the optimum layer depth (with layer size invariant under depth differences). It is also interesting to note that just increasing the depth doesn't keep on improving performance (quantified as validation F1- Scores), and for a certain use case, extensively deep networks may hurt performance rather than leading to performance gains.

We used this model for further experiments as it had the highest validation and training Macro F1 score and repeated the experiment by testing on balanced as well as the unbalanced dataset. What was interesting to note was this model worked better on unbalanced, raw training and test datasets than on smaller (~4x smaller) balanced train and test datasets. This might have been because complex models like bi-LSTMs, capable of capturing sequential dependencies, could have benefited from the unbalanced dataset by learning more intricate patterns and representations present in the majority class.

Model 3: Pre-trained BERT

Finally, we test our dataset on BERT (Bidirectional Encoder Representations from Transformers) which is designed to understand the context of words in search queries more accurately by considering the surrounding words (both before and after) in a sentence or text passage. It uses a transformer architecture, a neural network architecture known for its effectiveness in sequence-to-sequence tasks, which has revolutionized various NLP tasks.

BERT-specific parameters such as learning rate, batch size, and epochs were utilized for fine-tuning. The maximum sequence length played a critical role due to BERT's token limit, directly impacting the model's performance.

Discussion on Results:

The results exhibited an impressive overall accuracy of 92.78%, highlighting BERT's effectiveness in toxicity classification. However, the F1 scores varied significantly across different classes, with 'obscene' and 'insult' achieving notable scores of 0.70, while challenges were observed in obtaining high F1 scores for classes like 'severe_toxic' (0.42) and 'threat' (0.59).
One surprising finding was the variability in F1 scores across classes, showcasing BERT's capability to excel in certain toxic categories while encountering difficulties in others. The final choice of BERT was based on its remarkable ability to capture contextual dependencies and nuances in language. Despite varying F1 scores, the model's overall accuracy and performance across multiple classes made it the preferred choice. Comparison with other models revealed BERT's superiority.
It outperformed Naive Bayes in overall accuracy (92.78% vs. ~90%) and demonstrated superior F1 scores, particularly in capturing nuanced toxic language. Additionally, BERT showcased improved F1 scores over LSTMs for several classes, emphasizing its effectiveness in understanding contextual nuances.
The improvement in performance from our previous models to the BERT model can be attributed, in part, to the fact that transformers are a more powerful model architecture than RNNs. Unlike RNNs, transformers are not limited by the vanishing gradient problem, which can make it difficult for RNNs to learn dependencies over long sequences.

Future Work:

Future work possibilities involve fine-tuning BERT hyperparameters to enhance performance on challenging classes. Experimentation with different pre-trained transformer models or ensemble methods is also on the horizon. Addressing class imbalances and exploring interpretability tools for insights into decision-making and potential bias represent additional areas for exploration.