Project 1: Content Moderation and Toxicity Classification

Problem setup:

The contemporary digital landscape, characterised by extensive communication across online platforms, presents a critical challenge in the form of harmful and toxic content. As user-generated content proliferates, maintaining a secure and respectful online environment becomes imperative. The task of content moderation is pivotal in upholding the quality and safety of digital platforms, necessitating efficient and automated solutions. This project specifically addresses the detection and classification of potentially harmful content, with a focus on the Jigsaw Toxic Comment Classification Dataset. The objective is to develop a machine learning model capable of identifying various types of comment toxicity, including mild toxicity, severe toxicity, obscenity, threats, insults, and identity hate.

Methodology and Findings:

The chosen methodology involves using Logistic Regression, Sequence Models like RNNs (LSTMs) and Pretrained Encoder-Transformers BERT (Bidirectional Encoder Representations from Transformers). All experiments are implemented using Google Colab. The initial steps for each involve loading the dataset, consisting of both training and testing data, from Google Drive, followed by distinct preprocessing, and model-building techniques.

Prior to predictive modelling, exploratory data analysis (EDA) was carried out to gain insights into the dataset and prepare it for effective modelling. The observation of predominantly short comments influenced the choice of a simpler model architecture, such as Logistic Regression, to capture essential patterns without overcomplicating the model. Moreover, the class imbalances in toxicity labels leaning toward the “toxic” category, as well as the disproportionate number of clean/unlabeled comments guided the emphasis on weighted metrics in model evaluation including a focus on AUC-ROC scores to address potential bias.

With these insights in mind, the model-building process began, starting with a shallow Logistic Regression classifier. For this experiment, initial text preprocessing involved lowercasing, emoji replacement, stopword removal, punctuation removal, non-ASCII character removal, numeric character removal, and whitespace reduction. Subsequently, TF-IDF vectorization was employed to transform the preprocessed text into feature-rich TF-IDF vectors.

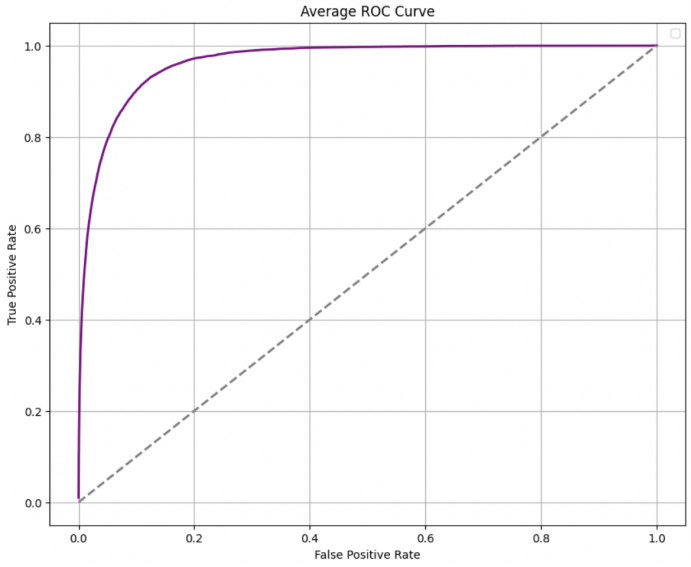
In experiment 1, Logistic Regression and Multinomial Naive Bayes models were considered, with their performance evaluated through cross-validation, splitting the training data to produce a 30% sized validation set, and employing a range of metrics such as accuracy, precision, recall, F1 score, and AUC-ROC. The results indicated that the Logistic Regression model outperformed the Multinomial Naive Bayes model in effectively classifying toxic comments, with a weighted AUC-ROC score of 0.9692, a substantial improvement from the Multinomial NB’s 0.8289. The emphasis on weighted metrics, considering the imbalanced nature of the dataset, provided a balanced evaluation of the models' performance.

After selection of Logistic Regression, the model’s hyperparameters were tuned using RandomizedSearchCV. Parameters, including solver algorithms ('lbfgs' and 'liblinear'), regularization strength (C values), and class weight settings, were optimized to maximize the area

under the ROC curve (AUC-ROC). The selected hyperparameters enhance the model's ability to classify toxic comments effectively, with a focus on minimizing false positives and false negatives.

The chosen Logistic Regression model, armed with optimized hyperparameters, was trained on the entire dataset and rigorously evaluated on the test set. Evaluation metrics included precision, recall, F1 score, accuracy, and AUC-ROC. The model exhibited robust recall for toxic, severe toxic, obscene, and insult classes, effectively identifying instances of these categories. Precision was relatively lower, indicating a trade-off between minimizing false negatives and false positives. The fact that the AUC-ROC was high, but precision was relatively low can be explained by the imbalanced dataset, and indicates that the model is biased towards predicting the majority class. Threat and identity hate classes also displayed moderate precision and recall, suggesting potential for improvement. The results are observed below.

	Precision	Recall	F1 Score	Accuracy	AUC ROC
Weighted Percentage	34.6%	92.03%	50.1%	76.3%	96.5%



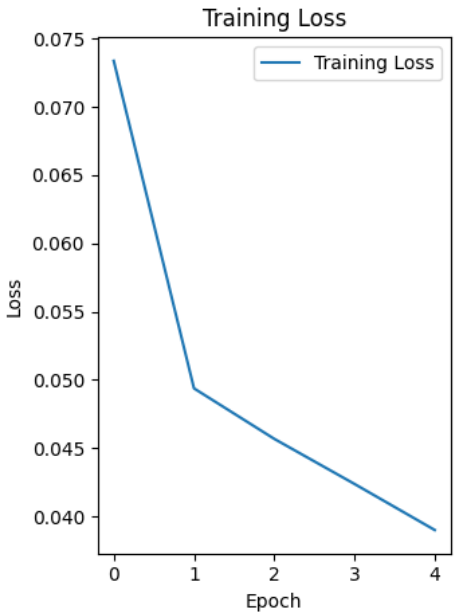
Analyzing misclassified texts for the "threat" label highlighted the need for a more representative training set, specifically including words like "vandal" and "traitor" for improved classification. Future improvements for enhancing the toxic comment classification model include exploring advanced techniques such as word embeddings, incorporating deeper feature engineering like sentiment scores and spelling correction, experimenting with advanced models like LightGBM and ensemble methods (e.g., stacking), integrating deep learning models such as LSTM for sequential language patterns, and employing advanced hyperparameter tuning methods like Bayesian Optimization.

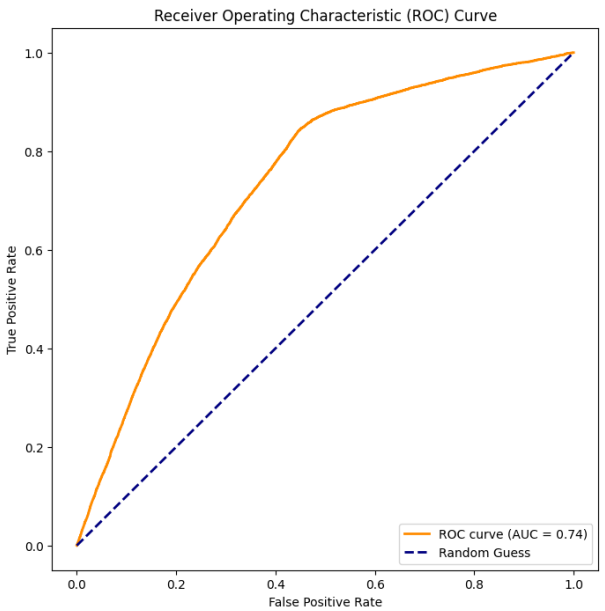
While Logistic Regression demonstrates strong performance in classifying toxic comments, particularly in recall across various toxic categories, precision is relatively lower. The AUC-ROC scores highlight the model's ability to discriminate between toxic and non-toxic comments effectively. The identified areas for improvement provide a roadmap for refining the model and advancing its capabilities, which are explored in subsequent experiments.

This was followed by a second experiment which leveraged Sequence Models like RNNs (LSTMs). After data loading, preprocessing steps for this experiment involved removing the 'ids' column and correcting mislabeled labels in the test dataset, specifically eliminating instances where labels were incorrectly marked as -1, avoiding any inconsistency within the dataset labels. The rows of text were converted to lowercase, special characters were removed. The text was also tokenized and rejoined removing extra white spaces and to standardise the formatting. Tokenization and vectorization of input tokens were executed using TensorFlow's TextVectorization layer. This layer was configured to tokenize comments, constrain vocabulary size of about 10,000 words, and pad sequences to a specified length of about 100. Tokenization facilitated breaking down the text into smaller words or tokens, which were then vectorized to convert them into a numerical format compatible with the model. The train-test-split function from the scikit-learn library was used to extract the validation dataset from the train data with a ratio of 80:20 (train:val).

The deep learning model architecture was implemented using TensorFlow's Sequential API. The architecture included an embedding layer, a bidirectional LSTM layer, and fully connected layers. LSTM (Long Short-Term Memory) was chosen due to its ability to address the vanishing gradient problem commonly encountered in traditional RNNs. The bidirectional layer allows it to make use of two separate LSTM layers processing the input sequence in both forward and backward directions. This bidirectional nature allows the network to capture information from past and future time steps simultaneously. LSTMs excel in learning and retaining information over extended sequences, making them suitable for tasks requiring an understanding of context across time. The Sequential model from Keras was first initialised. An embedding layer was then used to transform integers representing words into dense vectors of fixed size. This was followed by the LSTM layer and two dense layers with Softmax as the activation function. Softmax was used to ensure a robust approach to multi-label classification.

The model then underwent training on the vectorized train dataset for 5 epochs, during which the loss of the train gradually decreased from about 0.07 to 0.04. A graph depicting the loss versus the number of epochs was generated using the matplotlib library. Following training, the model was evaluated on a test set using metrics such as precision, recall, and accuracy, leveraging TensorFlow metrics. The model was designed to produce binary classifications for toxicity across six categories: mild toxicity, severe toxicity, obscenity, threats, insults, and identity hate.





The accuracy of the model obtained on the validation set was 86.3% while the test data had an accuracy of about 87.6%. The results can be summarised in the table below:

	Precision	Recall	F1 Score	Accuracy	AUC ROC
Weighted Percentage	93.8%	86.3%	89.7%	87.6%	73.7%

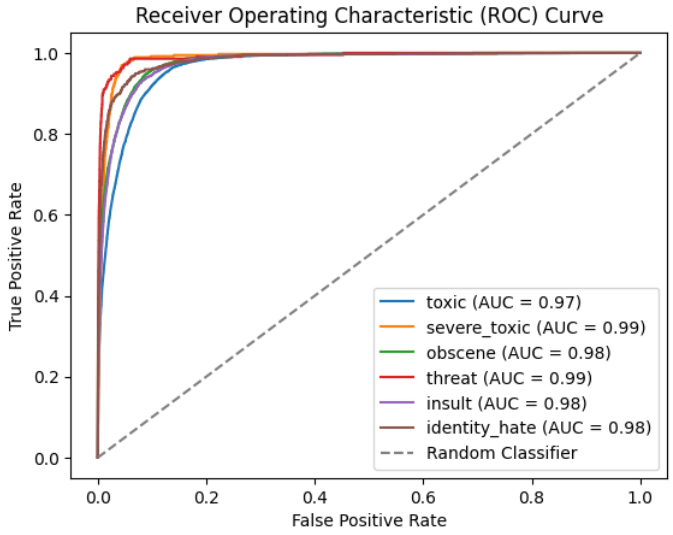
To further inform our analysis, a third experiment was conducted. This utilised Pretrained Encoder-Transformers like BERT: further incrementing candidate model complexity. Our first step in training a BERT classifier was to prepare the textual data for model input, whereby the data is preprocessed using the built-in BERT tokenizer. No manual preprocessing was done since it was seen to degrade the model's performance. This tokenization process converted the raw text into a format compatible with the BERT model, ensuring consistent and effective representation. All rows in the test set with label as -1 were removed from the data since the labels are supposed to be binary 1's and 0's.

Next, the chosen model, BERT for Sequence Classification, a pre-trained transformer model known for its ability to capture contextual information in both forward and backward directions was trained. The model was configured to handle the multi-label classification nature of the toxicity classification task, where a comment can exhibit multiple toxic attributes simultaneously. Training was conducted using an AdamW optimizer with a learning rate of 1e-5. The model underwent five training epochs, during which it learned to fine-tune its parameters based on the toxicity labels in the training data.

Once the model was trained, it was evaluated on a separate validation and test dataset to gauge its performance in classifying toxic comments. During evaluation, the model processed batches of validation and test data, producing probability scores for each toxicity category. To derive binary predictions, a threshold of 0.5 was applied to the probability scores. The model's predictions were then compared against the ground truth labels from the test dataset. Evaluation metrics such as precision, recall, AUC-ROC and F1 score were computed for each toxicity category, providing a detailed understanding of the model's strengths and weaknesses in differentiating between toxic and non-toxic comments. The total accuracy score of the model was also calculated so that comparisons could be made. The model gave an accuracy of 92.16% on the validation set and 85.84% on the test set showing strong classification performance. In the same vein, the AUC-ROC was also extremely high for this model indicating the model's capability of accurately differentiating between the classes.

The observed disparity in accuracy between the BERT-based model and other models could be attributed to various factors, including differences in data preprocessing, hyperparameter tuning, data imbalance, and threshold selection. Firstly, the distinct preprocessing pipelines for BERT and the other models might lead to variations in how the models capture and interpret textual information. Second, the hyperparameter configurations, such as learning rates and batch sizes, can significantly impact the model's ability to generalise patterns in the data. Additionally, an imbalanced distribution of toxic and non-toxic comments might skew the training process, affecting the model's performance. Ensuring a balanced dataset or using appropriate techniques to handle class imbalances is crucial. Lastly, the choice of the threshold for converting probability scores to

binary predictions can influence the trade-off between precision and recall, ultimately impacting the overall accuracy of the toxicity classification. Fine-tuning these aspects and conducting thorough analyses are vital steps in optimising the BERT model for improved performance.



	Precision	Recall	F1-Score	Accuracy	AUC-ROC
Weighted Percentage	54.26%	82.85%	65.28%	85.84%	98.85%

Evaluation and Best model:

Taken together, we conducted a comprehensive evaluation of our models using various metrics, including precision, recall, F1-score, accuracy, and AUC-ROC. Notably, RNNs excelled in precision, while Logistic Regression performed best in recall. RNNs also demonstrated superior performance in terms of F1-score and accuracy. However, traditional metrics, when studied in isolation, can be unreliable in imbalanced datasets due to biases towards the majority class, leading to inflated values and a potentially misleading assessment of model performance [1]. On the other hand, assessing AUC-ROC alongside metrics like precision and recall, known for their effectiveness in handling class imbalances, revealed a discernible trend: as model complexity increased, there was a corresponding enhancement in the model's ability to distinguish between different labels. Given AUC-ROC's widespread support and reliability in machine learning frameworks [2], its application underscores BERT as the optimal model for the project, highlighting its superior discriminatory ability across diverse label distinctions.

References

[1] Zhang, Jiajia, et al. “A Novel Evaluation Metric for Deep Learning-Based Side Channel Analysis and Its Extended Application to Imbalanced Data.” IACR Transactions on Cryptographic Hardware and Embedded Systems, tches.iacr.org/index.php/TCHES/article/view/8583. Accessed 10 Dec. 2023.

[2] Richardson, Eve, et al. “The ROC-AUC Accurately Assesses Imbalanced Datasets.” SSRN, 6 Dec. 2023, papers.ssrn.com/sol3/papers.cfm?abstract_id=4655233.