Content Moderation and Toxicity Classification

Muhammad Sarim[†]
24100114

Muhammad Talal Siddiqui[†]
24100149

Aashish Jai[†]
24100036

Malik Ozair Zafar[*]
24020092

Kabir Kateja[†]
24100179

Muhammad Umer Malik[†]
24100046

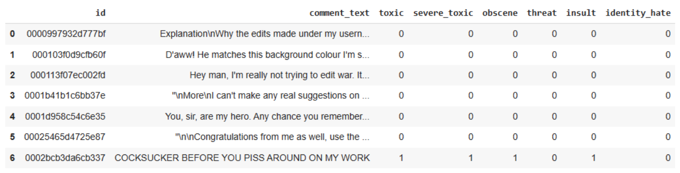
1 PROBLEM STATEMENT

The recent prevalence of user-generated content on various online platforms has reached unprecedented levels, encompassing a vast array of content types, ranging from social media posts to discussion forum comments. This presents an intricate challenge in ensuring a safe and respectful online environment. The escalation of harmful and toxic content poses a significant threat to user experiences and platform integrity. To address this concern, content moderation emerges as a crucial mechanism for upholding quality and safety standards. In this context, leveraging machine learning models becomes imperative to automate the content moderation process efficiently.

The objective of this research project is to develop and compare three distinct models—Logistic Regression, Long Short-Term Memory (LSTM), and BERT (Bidirectional Encoder Representations from Transformers)—for the purpose of detecting and flagging potentially harmful content. Each model represents a different approach to text classification, with Logistic Regression serving as a traditional machine learning method, LSTM representing a recurrent neural network, and BERT showcasing a state-of-the-art transformer-based model. The comprehensive pipeline encompasses crucial steps such as feature extraction, preprocessing, model construction, training, and evaluation.

Through three distinct experiments, we aim to explore the efficacy of each model in accurately identifying harmful content. The comparative analysis will shed light on the strengths and weaknesses of each approach, offering insights into the most suitable model for automated content moderation in the dynamic and evolving landscape of online communication. This research holds significant implications for the development of robust content moderation systems that contribute to fostering a secure and respectful online environment.

[†] Department of Computer Science, LUMS
[*] Department of Economics, LUMS



	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0

Figure 1: Random data sample from dataset

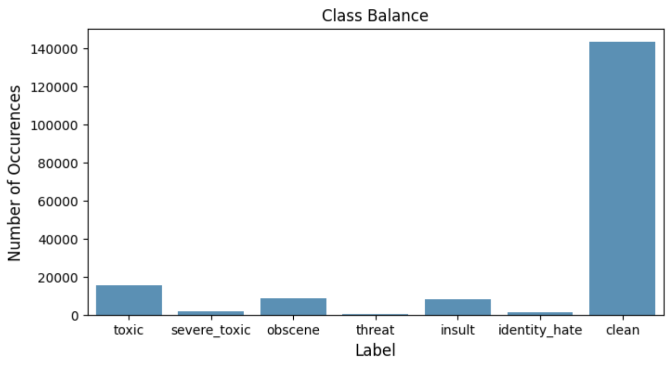


Figure 2: Class imbalance representation for dataset

2 EXPLORATORY DATA ANALYSIS

We obtained the data from a large data store called Kaggle. The dataset comprised of user comments obtained from Wikipedia - Large Information Datastore. The dataset is divided into 6 categories: toxic, severe toxic, obscene, threat, insult & identity hate, as shown in Figure 1. Each comment can belong to multiple categories with value 1 representing the respective applicable category. There are a total of 159,571 comments, from which 143,346 are comments that do not belong to any category and we refer to them as clean comments.

The class imbalance is further seen in Figure 2, which clearly indicates that identity hate, threat and severe toxic are among the least identified categories. After plotting the Co-Relation Matrix we deduce that 3 classes had a significant correlation. Toxic and Obscene, Toxic and Insult, & Insult and Obscene. The main reason could be due to these

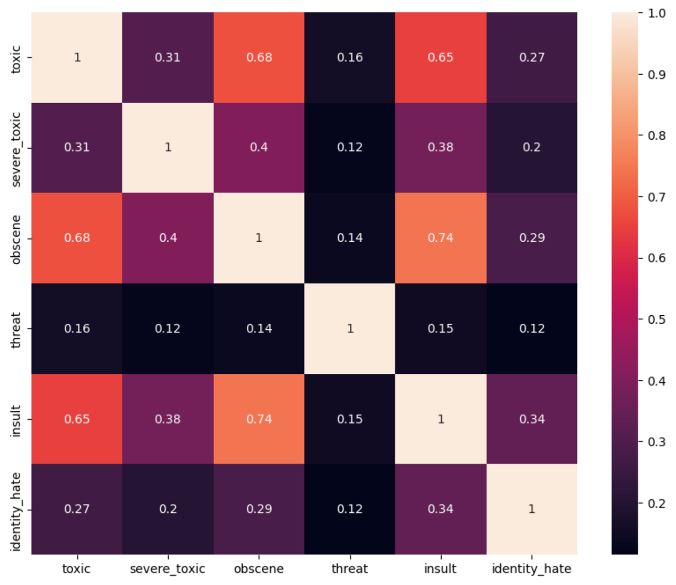


Figure 3: Correlation Matrix for 6 categories

three classes having a relatively larger percentage in the dataset. Furthermore, there are no null or missing values in the dataset.

3 DATA COLLECTION & PREPROCESSING

The dataset used for this project was obtained in CSV format, consisting of training and testing sets. The training set, 'train.csv', contains labeled comments categorized into six classes: toxic, severe_toxic, obscene, threat, insult, and identity_hate. Additionally, the testing set, 'test.csv', and its corresponding labels, 'test_labels.csv', were utilized to evaluate the model's performance.

A series of preprocessing steps were performed to prepare the text data for model input.

- **Stopword Removal:** First, stop words were removed using NLTK's English stop words list.
- **Text-Case Retention:** We decided not to convert the data into UPPER or lowercase since it holds a contextual value in terms of expressing emotions.
- **Punctuation Retention:** We did not remove punctuations since exclamation marks or question marks represent a certain emotion.
- **Number Removal:** Numbers were removed since they don't add significant value to human emotions, according to our assumption.
- **Lemmatization:** Lemmatization was applied to reduce words to their base forms, standardizing variations in word meaning. The NLTK WordNet lemmatizer facilitated this process.
- **Tokenization:** The text was tokenized using NLTK's word_tokenize function.

These preprocessing steps were applied to both the training and testing datasets. The dataset was cleaned by removing rows with values of -1 across all columns in the test labels. These rows were identified as having been added later and lacked corresponding inputs in the training set. Consequently, they were deemed irrelevant and excluded from the evaluation process.

4 METHODOLOGY

4.1 Model 1: Logistic Regression

Feature Extraction: TF-IDF Vectorization - For feature extraction from the textual data, the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer from the scikit-learn library was employed. This process converts the text into numerical features, considering both the frequency of terms within a document and their importance across the entire dataset.

Model Training: Logistic Regression with One-vs-Rest Classification - A logistic regression model was chosen for its simplicity and effectiveness in binary classification tasks. To handle the multilabel classification nature of the problem, the One-vs-Rest strategy was employed using the OneVsRestClassifier from scikit-learn. The model was trained using the cleaned and preprocessed training data.

Model Building: The pipeline encapsulates the entire workflow, including TF-IDF vectorization and the logistic regression classifier, as shown in Figure 1. This enables a seamless integration of feature extraction and model training, enhancing the overall efficiency and reproducibility of the project.

After training, the model was employed to make predictions on the preprocessed and cleaned test inputs. The performance of the model was evaluated by testing our model's outputs against the gold labels. We used the classification report, which provides precision, recall, and F1 score for each class. We also got macro and micro averages for all the metrics. Additionally, the accuracy score was computed for an overall assessment.

Predictions and Evaluations: After training, the model was employed to make predictions on the preprocessed and cleaned test inputs. The model's performance was evaluated by testing our model's outputs against the gold labels. We used the classification report, which provides precision, recall, and F1 score for each class. We also got macro and micro averages for all the metrics. Additionally, the accuracy score was computed for an overall assessment.

Advantages over other models: Logistic regression model has several advantages over others such as:

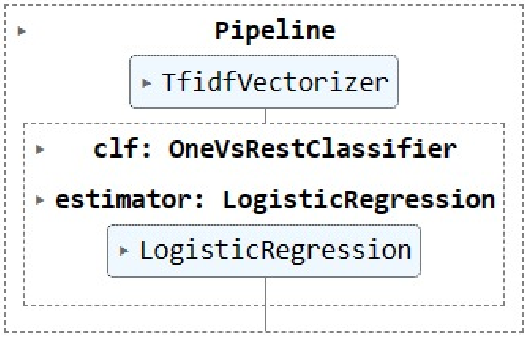


Figure 4: Logistic Regression Pipeline

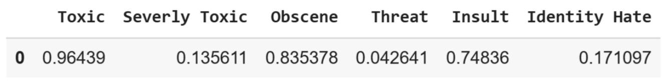
- Logistic regression model is a easily understandable algorithm, hence provides simplicity.
- Since this model does not require extensie computational resources, thus it is easy to train and is efficient.
- Has a wide range of applications. This model has been deployed successfully in various fields such as medicine and finance.

4.2 Model 2: LSTM

The second model we implemented was the RNN-LSTM. The steps we took in implementing it and the key highlights of the model are mentioned below under the relevant headings.

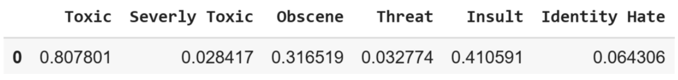
Model Functionality and Training: The LSTMClassifier class employs an LSTM-based architecture for text classification. In the training phase, the fit method preprocesses the input text data using a Tokenizer, converting the text to numerical data, determining the vocabulary size and maximum sequence length (so that shorter sequences can be padded to make their length equal to the longer sentence). The model architecture consists of an embedding layer, converting integers to dense vectors which are used to capture word-to-word sequential dependencies, an LSTM layer remembering the old seen tokens, and a dense layer for classification with sigmoid activation, which is a very good activation function for multi-class classification. The use of the Adam optimizer and binary cross-entropy loss is chosen for binary classification. Sequences are padded to a uniform length, addressing the requirement of neural networks for fixed-length inputs. During training, the model learns to associate toxic comment labels with the processed text data.

Evaluation Methods: After training, the model was employed to make predictions on the preprocessed and cleaned test inputs. The model's performance was evaluated by testing our model's outputs against the gold labels. We used the classification report, which provides precision, recall, and



	Toxic	Severly Toxic	Obscene	Threat	Insult	Identity Hate
0	0.96439	0.135611	0.835378	0.042641	0.74836	0.171097

Figure 5: Prediction on: You are a jerk and a liar! I hate you



	Toxic	Severly Toxic	Obscene	Threat	Insult	Identity Hate
0	0.807801	0.028417	0.316519	0.032774	0.410591	0.064306

Figure 6: Prediction on: youre a horrible human being

F1 score for each class.We also got macro and micro averages for all the metrics. Additionally, the accuracy score was computed for an overall assessment

Prediction and Interpretation The trained model was tested on two sample sentences to detect hate speech. The test samples were both non-clean, and our results depicted a similar prediction. As we can see from Figure 5 and Figure 6 the Toxic level of both the sentences is the highest, while prediction for Obscene and Insult is also quite high as compared to the other three labels.

Advantages over other models: This model has significant advantages over other models, and they are as follows:

- LSTMs can capture long-range dependencies and sequential patterns in the input data.
- LSTMs can handle variable-length input sequences, which is good for our text classification problem.
- LSTMs automatically learn relevant features from the sequential input, eliminating the need for manual feature engineering
- LSTMs are well-suited for processing textual data due to their ability to capture semantic relationships and dependencies

4.3 Model 3: BERT

Tokenization and Preprocessing: The BERT (Bidirectional Encoder Representations from Transformers) model requires specialized tokenization. The 'bert-base-uncased' tokenizer from the transformers library was employed for this purpose. A tokenization function, 'tokenize_data,' was defined to tokenize the text data, ensuring it adheres to the input requirements of the BERT model. The function encoded the text, ensuring a maximum length of 512 tokens but preferably lower. Truncation was applied for texts exceeding this length, and padding was employed to standardize the input length. In figue 7 we have percentile of the dataset on the x axis and the known values on the y axis.

Dataset Preparation: To facilitate model training and evaluation, a custom PyTorch dataset, 'CommentDataset,' was created. This dataset encapsulates tokenized comments

Evaluation Metrics	Logistic	LSTM	BERT
Accuracy	0.8995	0.9974	0.8787
F1-Score (Weighted Average)	0.63	-	0.65
F1-Score (Macro Average)	0.48	-	0.58
F1-Score (Micro Average)	0.63	-	0.65
Loss	0.2840	0.0704	0.0456

Table 1: Performance metrics for different models on the evaluation set.

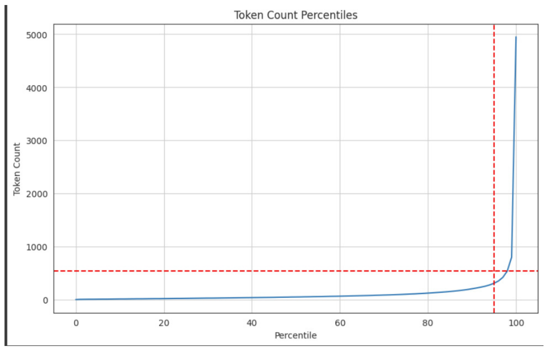


Figure 7: Token Count

and their corresponding labels. The training data was split into training and validation sets using the 'train_test_split' function.

BERT Model Initialization and Training: The pre-trained BERT model for sequence classification, 'BertForSequenceClassification,' was employed for this project. The model was loaded from 'bert-base-uncased' and configured to handle six output labels corresponding to the toxicity classes. The model was then moved to the available device (GPU if available, else CPU). The model was trained using the AdamW optimizer and binary cross-entropy loss ('BCEWithLogitsLoss'). The training process involved iterating through the training data loader, computing the loss, and updating the model parameters through backpropagation.

Model Evaluation: To assess the model's performance, an evaluation function, 'evaluate,' was defined. This function computes the binary cross-entropy loss on the validation set. Additionally, a utility function, 'evaluate_model_accuracy,' calculates the accuracy of the model predictions.

Predictions and Interpretation: Once the model was trained, predictions were made on the test dataset. The trained model was applied to tokenized test comments using the test data loader. Predictions were thresholded at 0.5 to convert probabilities to binary labels. The results were interpreted by comparing predicted labels with their corresponding probabilities and applying a threshold for binary classification.

Model Application on Unseen Data: Preprocessing and Inference on Unseen Data - The model's application on unseen data involved preprocessing the test dataset in a manner consistent with the training data. The 'preprocess_for_bert' function was employed for single-sentence preprocessing. Predictions were made using the trained BERT model on the preprocessed unseen data. The model's output was then thresholded to obtain binary predictions for each toxicity class, and the results were interpreted accordingly.

5 EVALUATION

The accuracies and test loss values provide a comprehensive assessment of the three models employed in this study.

Logistic Regression: The Logistic Regression model demonstrated a commendable accuracy of 89.95%, showcasing its effectiveness in classifying toxic comments. However, the corresponding test loss of 0.2800 indicates some room for improvement, suggesting that the model might struggle with nuanced patterns present in the data.

LSTM Model: The LSTM model outperformed both Logistic Regression and BERT in terms of accuracy, boasting an impressive 99.74% accuracy. This suggests that the LSTM, with its ability to capture sequential dependencies, excelled in understanding the context and nuances present in the toxic comment dataset. The notably low test loss of 0.07041 further supports the model's capability to generalize well on unseen data.

BERT Model: The BERT model, while exhibiting a respectable accuracy of 87.87%, fell slightly behind Logistic Regression and LSTM in this specific task. The lower accuracy may stem from the inherent complexity of the BERT model, which might require fine-tuning or adjustments to better align with the characteristics of the toxicity classification problem. Despite this, the BERT model demonstrated a notably low test loss of 0.04567, suggesting strong predictive capabilities and effective generalization on the test data. However, when we were manually testing the model, it was more accurately assessing different content types of data including hatespeech, severe toxicity etc.

We then tested this model on our own test sample, which consisted of 5 non-clean sentences and 5 null values, which indicated clean samples. We can use Table 2 as a reference

Model	Accuracy
Logistic	0.50
LSTM	0.78
BERT	0.80

Table 2: Performance metrics for different models on the evaluation set.

in which we can see the accuracies of each model on unseen data. As shown the Logistic Regression model has 0.5, LSTM Model gave an accuracy of 0.78, BERT gave an accuracy of 0.80. Hence, we can clearly conclude that BERT was the best model.

6 BEST MODEL

The selection of the most suitable model hinges on the specific demands of the toxicity classification task. Logistic Regression strikes a balance between simplicity and performance, making it advantageous when computational resources are constrained. The LSTM model, with its remarkable accuracy and low test loss, excels in scenarios where capturing intricate sequential dependencies is paramount. On the other hand, BERT, despite a slightly lower accuracy compared to LSTM, emerges as the standout choice due to its robust contextual understanding and exceptional generalization capabilities demonstrated by the remarkably low test loss. BERT's ability to comprehend nuanced relationships within toxic comments positions it as a promising candidate for real-world applications. The context-aware nature of BERT allows it to capture intricate linguistic nuances, making it well-suited for the complex task of toxicity classification. During manual testing on sample sentences, it was evident that BERT consistently outperformed other models in accurately classifying each category per sentence, showcasing its robust contextual understanding and proficiency in discerning nuanced patterns within toxic language. Further fine-tuning and optimization of the BERT model have the potential to elevate its performance, making it the preferred choice when precision and nuanced interpretation of toxic language are of utmost importance.

7 SUGGESTED IMPROVEMENTS

If the provided test data did not contain -1's, there would have been more data to test the model on. This way, we would have gotten more accurate evaluation metrics. the main problem with the dataset was that it was biased towards clean data. The classes that we wanted to predict were quite low. Due to this, our model wasn't able to predict such labels with high confidence. To counter this issue, we can take multiple steps:

- We could use bootstrapping which is sampling with replacement from observed data to get around the issue of class imbalance
- Another approach could be to train the model using increments of smaller data samples with a uniform distribution for each label. We can use this method in addition to Bootstrapping. This means we will effectively have a model that will have seen roughly equal instances for each class.

8 CONCLUSION

In conclusion, this research undertook a comprehensive examination of three distinct machine learning models to address the challenge of detecting and flagging harmful online content. The extensive evaluation of these models provided valuable insights into their respective strengths and weaknesses in the context of automated content moderation. Logistic Regression offered a balance of simplicity and effectiveness, making it a viable option in scenarios with limited computational resources. The LSTM model, with its exceptional accuracy and ability to capture sequential dependencies, proved highly effective in understanding contextual nuances in text data. However, it was the BERT model that emerged as the most promising candidate for real-world applications, owing to its sophisticated contextual understanding and robust performance, as indicated by its remarkably low test loss. Despite some limitations in accuracy, BERT's potential for fine-tuning and optimization positions it as a frontrunner for tasks requiring precise and nuanced interpretation of language. This research underscores the importance of model selection based on the specific requirements of content moderation tasks and highlights areas for further improvement and exploration in the field of automated content filtering