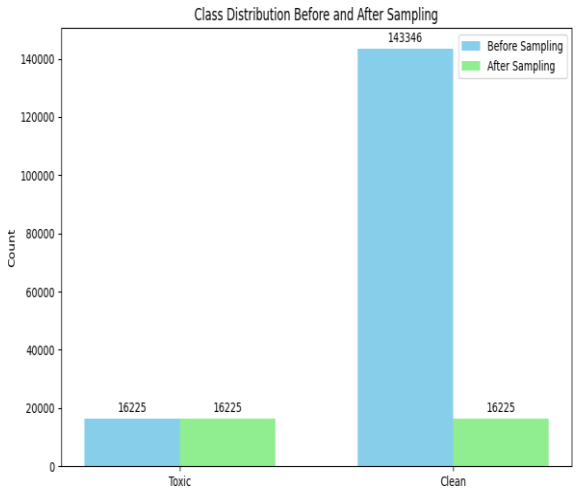
Project 1: Content Moderation and Toxicity Classification

Overview

We have used the Jigsaw Toxic Comment Classification Dataset to provide insight into six possible types of comment toxicity: **mild toxicity, severe toxicity, obscenity, threats, insults, and identity hate.** We have used **three** routes in our approach:

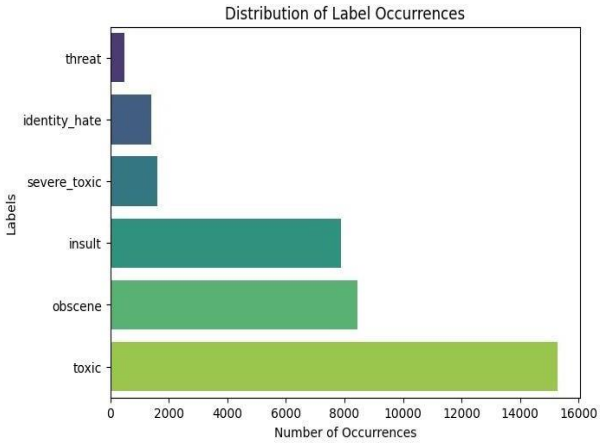
1. **Naive Bayes and Logistic Regression**
2. **Sequence Model RNN**
3. **Pretrained Encoder-Transformer BERT**

Clean Vs. Toxic Comments Imbalance in our Training Dataset



Considering the significant class imbalance between “**clean**” and “**toxic**” comments in the original dataset, a strategic approach was implemented to address this issue. By sampling a subset of **16225** clean comments and merging them with the existing toxic comments, a balanced dataset was created, comprising an **equal** proportion of both clean and toxic comments. This approach aimed to mitigate the imbalance and foster better learning and generalisation capabilities within the model by ensuring a more representative distribution of classes.

Class Imbalances in our training dataset



The "toxic" class has the highest number of samples, followed by **"insult"** and **"obscene"**. Conversely, the classes **"severe toxic," "threat,"** and **"identity hate"** exhibit significantly fewer samples. While any model may perform well in detecting content labelled as **"toxic," "insult,"** and **"obscene"** due to the abundance of training examples, it may face challenges in accurately predicting classes with fewer instances, such as **"severe toxic," "threat,"** and **"identity hate."** In each of our models, we have tried to mitigate the imbalance by using different techniques for each model.

Naive Bayes and Logistic Regression

Methodology:

1. Data Preprocessing: The dataset, consisting of 159,571 training data entries and 153,164 test data entries, underwent preprocessing steps like lowercasing, punctuation removal, lemmatization, and handling of non-ASCII characters. Such preprocessing is essential to reduce noise and variability in the text data, making it more amenable to analysis.

2. Feature Extraction: TF-IDF (Term Frequency-Inverse Document Frequency) was chosen for feature extraction. This method was preferred for its efficiency and interpretability, highlighting important words for classification.

3. Class Imbalance Handling: Given the class imbalance in the dataset, random sampling was employed to balance the clean and unclean comments. This step is crucial to avoid model bias towards the majority class and improve generalisation, as well as handling weights for the class imbalances by both the respective models.

Results:

The results were quantified using metrics like Recall, F1-Score, and Accuracy. Naive Bayes showed an F1-score of 0.857 for 'toxic' comments and as low as 0.034 for 'severe_toxic' comments. In comparison, Logistic Regression exhibited higher F1-scores across all categories, such as 0.864 for 'toxic' and 0.312 for 'severe_toxic'. This indicates Logistic Regression's better balance between precision and recall for each label.

Analysis:

Multinomial Naive Bayes

Label	Recall	F1	Accuracy
toxic	0.859748	0.856585	0.864314
severe_toxic	0.017555	0.034155	0.951248
obscene	0.646468	0.722960	0.871002
threat	0.000000	0.000000	0.985239
insult	0.552367	0.635833	0.846410
identity_hate	0.027758	0.053653	0.957627

Logistic Regiession

Label	Recall	F1	Accuracy
toxic	0.829737	0.863775	0.876672
severe_toxic	0.216928	0.311967	0.953097
obscene	0.668008	0.766553	0.894083
threat	0.136075	0.225389	0.986317
insult	0.565189	0.662339	0.860123
identity_hate	0.221352	0.336462	0.962219

Logistic Regression generally achieved higher Recall, F1-score, and Accuracy across multiple toxicity categories compared to Naive Bayes. Naive Bayes, due to its simplicity, struggled in classes with limited data, showing lower Recall, F1-score, and Accuracy. In contrast, Logistic Regression, with its ability to provide probabilities for outcomes, was more effective, especially in scenarios requiring interpretability of model predictions.

Sequence Model - Recurrent Neural Network(RNN)

Methodology

1. Data Preprocessing: The dataset, loaded into a pandas dataframe, contains labelled toxicity categories. The preprocessing includes tokenization, transforming text into sequences of integers using Keras's Tokenizer class. This step is essential to convert variable-length textual data into a uniform format for the RNN. The maximum vocabulary size is set to 20,000 and sequence length to 200, balancing information capture and computational efficiency.

2. RNN Model Architecture: The RNN model consists of an Embedding Layer to convert text data into dense vectors, a SimpleRNN Layer with 32 units to capture sequential patterns, and a final Dense Layer with 6 units (equal to the number of toxicity classes) using a sigmoid activation function. The choice of 32 units in the SimpleRNN layer balances model complexity and efficiency. The addition of more units or layers was observed to increase overfitting and reduce validation and testing accuracy.

3. Training and Evaluation: The dataset is split into training and validation sets with a test size of 20%. Class weights are computed and applied during training to address the class imbalance issue. The model is trained for 2 epochs, as more epochs led to overfitting. For evaluation, the F1-score is chosen due to the class imbalance, as accuracy might not be suitable in this scenario.

Results and Analysis

The model achieved an overall test accuracy of 89.61%. Here is the detailed report:

Class	Precision	Recall	F1-Score	Support
Toxic	0.67	0.58	0.62	6090
Severe Toxic	0.22	0.05	0.09	367
Obscene	0.61	0.6	0.6	3691
Threat	0.05	0.03	0.03	211
Insult	0.56	0.5	0.53	3427
Identity Hate	0.37	0.02	0.04	712
Micro Avg	0.61	0.51	0.56	14498
Macro Avg	0.41	0.3	0.32	14498
Weighted Avg	0.59	0.51	0.54	14498
Samples Avg	0.05	0.04	0.04	14498

The results demonstrate that while the RNN can effectively identify some forms of toxicity, it struggles significantly with classes that have fewer instances, such as severe toxic, threat, and identity hate. In conclusion, the RNN model shows promise in classifying toxic comments but requires further refinements, especially in handling class imbalance and capturing context more effectively for minority classes.

Pre-Trained Encoder Transformer - BERT

Methodology

1. Data Preprocessing: We employed a dataset comprising various labels representing toxicity categories. Critical steps in preprocessing included:
    - Label Summarization: To understand label distribution, aiding in balancing the dataset.

- Data Separation and Sampling: Categorised comments into 'toxic' and 'clean', followed by random sampling of clean comments to balance the dataset and prevent model bias.
- Shuffling and Splitting: Ensured a randomised data distribution and created a validation set for model evaluation.

2. Tokenization and Encoding
   BERT requires tokenized and encoded input. We used the `BertTokenizer` for this, ensuring consistent text representation. Special tokens ([CLS], [SEP]) and attention masks were utilised for delineating sentences and highlighting relevant tokens.

3. Model Training
   The BERT model, pre-trained on a large corpus, was fine-tuned using our dataset. We employed the AdamW optimizer for its effective handling of weight decay, crucial in preventing overfitting. The learning rate was set to 2e-5. Training involved iterating over epochs, calculating losses, and updating model parameters. A separate validation phase assessed model generalisation.

Results and Analysis

The model showcased encouraging performance in categorising toxic comments, exhibiting an accuracy of 74.83%. While achieving a high recall rate of 88.29% indicates its proficiency in identifying toxic comments, the precision score of 39.87% implies a likelihood of false positives. Despite this, the model maintains a balanced trade-off between precision and recall, evidenced by its robust F1 score of 54.93%. These metrics demonstrate the model's strong capability to recognize toxic comments while acknowledging the need to minimise false positives for enhanced precision.

Which model is better?

Considering the results and analysis, BERT Transformer emerges as the most promising model among the three. Its high recall rate is particularly valuable in toxic comment classification, where it is crucial to classify as many toxic comments as possible. Although its precision is lower than Logistic Regression, the trade-off is often acceptable in content moderation contexts where missing toxic content can be more damaging than falsely flagging non-toxic content. Additionally, BERT's advanced capabilities in understanding language nuances give it an edge, especially for more complex or subtle forms of toxicity.Moreover, since BERT is pre-trained on a large corpus, hence it gives more favourable results when fine tuned on our data.

Future Directions

- **Model ensembling**: Combining the strengths of BERT with the precision of Logistic Regression in a hybrid model could offer us a more balanced performance in identifying toxic comments and reducing false positives.
- **Extended Dataset and Augmentation**: We can include a more diverse and extensive dataset, possibly augmented with synthetic data, to improve the training of our models, especially for underrepresented categories.

- **Managing Class Imbalance**: Employing more sophisticated techniques for handling class imbalance, such as **SMOTE (Synthetic Minority Over-sampling Technique)** or **adaptive resampling methods**, can provide us with even better results.