Content Moderation and Toxicity Classification

Muhammad Ayain Fida Rana	Muhammad Muneeb Ahmad	Muhammad Hamza Nisar	Murtaza Ur Rehman	Muhammad Affan Pasha	Haris Saeed
25100045	25100067	25100068	25100117	25020449	25100288

1. Introduction

In an era where communication occurs on a large scale and free speech is guaranteed, a safe space where social values are respected is imperative. Following this issue, we were inspired to create an ML model that can detect toxicity in any of the following forms: hate speech, threats, racial remarks, and disrespect. We have addressed the particulars of toxic content detection with three distinct models. We feed in a comment in the model, and it classifies the text as belonging to one or more of these: "toxic," "severely toxic," "obscene," "threat," "insult," and "identity hate." The classification of different types of toxicity allows for a more nuanced understanding of harmful language and behaviour, enabling targeted interventions and fostering a safer online environment for all users.

```
input_text = "Fat man go eat with your nigga brother"
predicted_labels = predict_input(input_text, model, tokenizer)
print("Predicted labels:", predicted_labels)

Predicted labels: ['toxic', 'insult', 'identity_hate']
```

Figure 1: Text Classification

2. Dataset

2.1 Data Description

To train our models, we use the Jigsaw Toxic Comment Classification Dataset from Kaggle. [1] There are approximately 160,000 entries in the dataset. Each row includes a comment and a target score of either 0 or 1, which corresponds to the aforementioned labels. The distribution of labels in dataset is shown in Figure 2, which is quite imbalanced for training. The dataset is split into an 80:10:10 split for training, validation, and testing.

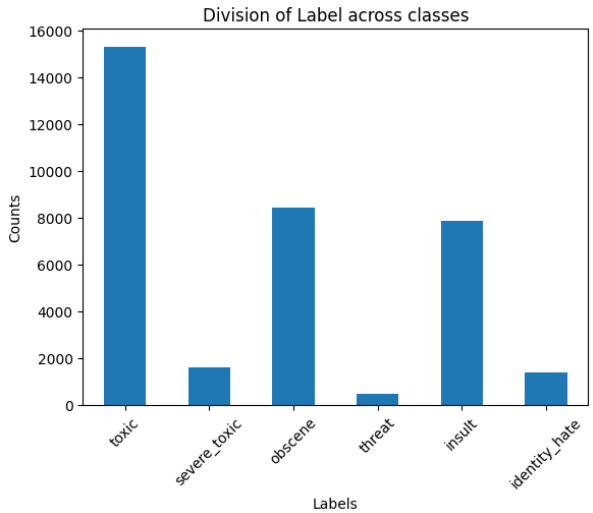

Figure 2

2.2 Data Imbalance

Figure 2 illustrates the imbalances in our dataset. It is evident that some classes, such as 'toxic', are significantly overrepresented, which can lead models to develop a bias, overfitting the majority class while underperforming the minority classes due to a lack of sufficient data to learn from. So, these kinds of imbalances can mess up the performance metrics. This means that the high accuracy might not really mean how well the model predicts across all classes; it may just mean how well it recognises the dominant class.

2.3. Data Pre-process

We preprocessed the dataset as follows in accordance with our observations of random comments extracted from the dataset:

- **Lowercasing:** Converts all characters in the comment to lowercase to maintain consistency.
- **Removing non-ASCII characters:** removes non-ASCII characters to simplify text and focus on English language words.
- **Expanding Contractions:** Replaces contractions like "can't" with "can not" for clarity and to maintain consistency.
- **Removing URLs:** Removes URLs starting with 'https', 'www'
- **Removing Punctuations:** Remove punctuation marks from the comments, as they are mostly irrelevant for the classification problem.
- **Removing Numbers:** Removes numbers as they aren't relevant in this classification problem.
- **Removing Stop Words:** Filters out stop words like "is," "the," etc. These words occur frequently and offer little value for analysis.

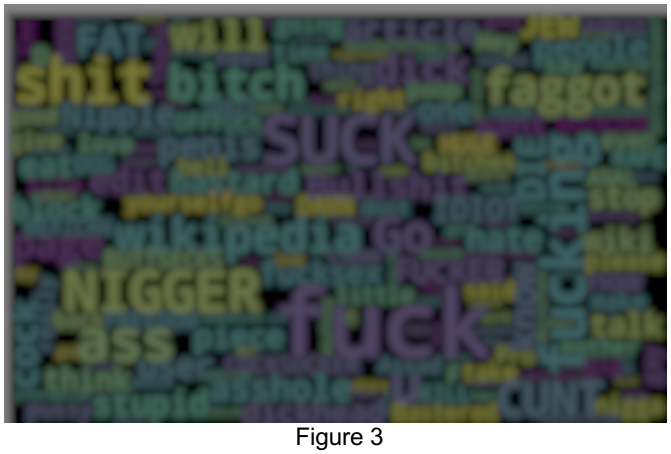

Figure 3

Some of the common words in the dataset are shown in Figure 3.

3. Methods

3.1 Hybrid Naïve Bayes and SGD Classifier

We combined Naïve Bayes and SGD classifiers to serve as our baseline model. The rationale behind this choice stems from the capabilities of SGD in optimising large-scale learning problems and the complementary nature of Naïve Bayes when it comes to managing high-dimensional data.

It augments the initial count vectors with probabilistic outputs from the Nave Bayes classifier in order to provide features for the SGD classifier. By doing so, we are able to enhance our depiction of features and take advantage of the predictive capability of Naive Bayes, whereas SGD optimises the decision boundary.

Not only did the resulting model outperform the best individual classifiers, but it was also more resistant to overfitting, which was crucial given the unbalanced nature of our dataset. Future research into more complex architectures can be built upon this model.

The model can be summarized as below:

Let X_{raw} be a raw input comment. The pre-processing converts it into X_{clean}:

$$X_{clean} = preprocess(X_{raw})$$

Then Count Vectorizer is applied on it to get X_{cv}, which is term-document matrix:

$$X_{cv} = CountVectorizer(X_{cv})$$

The Naïve Bayes classifier is then applied to these count vectors to compute the conditional probabilities of each class given the count vector:

$$P(y|X_{cv}) = NBClassifier(X_{cv})$$

The log-probabilities are then computed and then combined with count vectors X_{cv} to form new extended features:

$$X_{ext} = [\log(P(y|X_{cv}); X_{cv}]$$

The extended feature vector is then used to train the SGD Classifier:

$$z = SGDClassifier(X_{ext})$$

The SGD Classifier learns a weight vector θ and bias b; the sign functions assign the class labels based on sign of decision function:

$$z = sign(\theta^T X_{ext} + b)$$

The parameters θ and b are obtained by solving the optimization problem:

$$\min_{\theta,b} \frac{1}{2}\|\theta\|^2 + k\sum_{i=1}^{n} \mathcal{L}(\theta; X_{ext}, y_i)$$

k is the regularization parameter, $\mathcal{L}$ is the hinge loss used for SVMs within the SGD framework, and n is number of training instances.

This mathematical description provides a brief overview of the model's process flow. It outlines the key steps and equations involved in the model's calculations. The description aims to give a clear understanding of how the model progresses from input data to output results.

3.2 LSTM Model

LSTMs are special types of RNNs designed to be capable of learning long-term dependencies. With their feedback connections, LSTMs are able to model sequence data more effectively than traditional feed-forward neural networks.

These characteristics of LSTM make it suitable for text classification tasks in which context and word order are important for comprehending the overall sentiment or intent of the text. This is due to the fact that the order of words in a phrase can drastically alter its meaning and toxicity level. Its internal structure is shown in Figure 4.

There are often instances where the dependencies present in sequential data, such as sentences, can be observed in both directions. As a result, we used BiLSTMs, which consist of bidirectional layers that simultaneously combine features acquired in both directions to exploit forward and backward dependencies.

Our model can be summarized as below and is presented in Figure 5 as well:

Let X_{raw} be a raw input comment. The pre-processing converts it into X_{clean}:

$$X_{clean} = preprocess(X_{raw})$$

The pre-processed text is then vectorized, X_{vec}:

$$X_{vec} = Vectorize(X_{clean})$$

The vectorized tokens are passed through an embedding layer to get dense vector representations, X_{emb}:

$$X_{emb} = EmbeddingLayer(X_{vec})$$

Then the LSTM layer processes the sequence of embeddings in both directions and captures temporal dependencies.

$$h_t, C_t = LSTM(X_{emb,t}, h_{t-1}, C_{t-1})$$

These embeddings are then passed through the last classification layer of the model with parameters θ, b to obtain logits Z:

$$Z = \theta^T h_t + b$$

The label probabilities are then computed using SoftMax:

$$P = softmax(Z)$$

Briefly, the aforementioned mathematical representation illustrates the progression from unprocessed text to the projected probabilities, underscoring the crucial aspect of LSTM models' sequential processing that captures the context and interdependencies in text data.

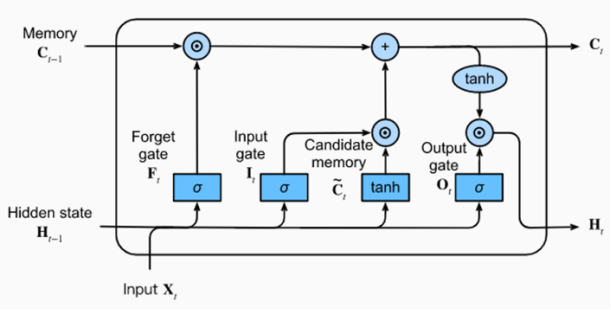

Figure 4

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_11 (Embedding)    (None, 100, 32)           3200032

 bidirectional_11 (Bidirect  (None, 64)                16640
 ional)

 dense_22 (Dense)            (None, 128)               8320

 dense_23 (Dense)            (None, 6)                 774

=================================================================
Total params: 3225766 (12.31 MB)
Trainable params: 3225766 (12.31 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 5

3.3 BERT Model

Finally, we used Bidirectional Encoder Representations from Transformers (BERT) as part of our solution pipeline. It is known for its state-of-the-art performance in natural language processing. Unlike previous models, which processed words in a sentence sequentially (from left to right or right to left), BERT reads the entire sequence of words at once, allowing the model to understand the context of a word based on its surroundings. This is because it is built upon the Transformer mechanism, which relies on self-attention mechanisms rather than sequence-aligned recurrence.

For our classification task, we used the BERT-base model, which is cased and has 12 layers with 768 hidden, 12-heads, and 110M total parameters. We fine-tuned it on the training data set derived from the original dataset. Because of this fine-tuning, the model was able to not only make use of its previously acquired context awareness, but it was also able to apply its understanding to the task of recognising different types of toxicity in written text.

Additionally, it is essential to keep in mind that the complexity of BERT necessitates a significant amount of computational resources and that the process of fine-tuning the model can be time-consuming. For this reason, we were only able to fine tune for 3 epochs.

The model can be summarized as below and is presented in Figure 6 as well:

$$X_{clean} = preprocess(X_{raw})$$

The pre-processed data is then tokenized, and these tokens are converted into token ids X_{tid} using BERT's vocabulary. Moreover, input id, X_{iid}, and attention masks, $X_{attn.mask}$ are also generated:

$$Tokens = BertTokenizer(X_{clean})$$
$$X_{tid} = BertVocabulary(Tokens)$$

The BERT model performs a froward pass generating an embedding C.

$$C = model(X_{tid}, X_{iid}, , X_{attn.mask})$$

These embeddings are then passed through the last classification layer of the model with parameters θ, b to obtain logits Z:

$$Z = \theta^T C + b$$

The label probabilities are then computed using SoftMax:

$$P = softmax(Z)$$

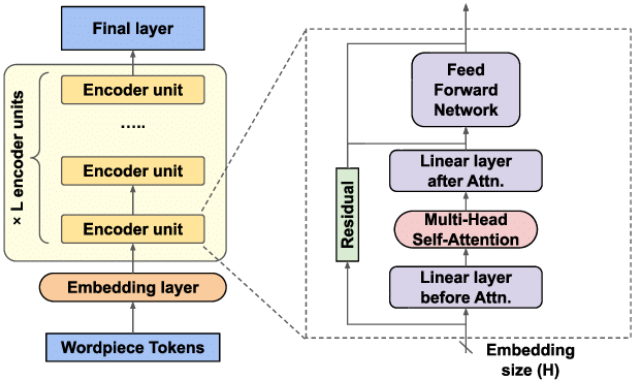

Figure 6

4. Evaluation Metrics:

Accuracy measures the proportion of correct predictions, both toxic and non-toxic, over all labels and samples. However, our dataset is imbalanced, so using this can be misleading. Imbalanced datasets often lead to inflated accuracy scores, as the model may perform well on the majority class but struggle with minority classes. To account for this, it is important to consider alternative evaluation metrics such as precision, recall, and F1 score, which provide a more comprehensive understanding of the model's performance across different classes.

$$accurracy = \frac{true}{true + false}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

$$F1\ score = \frac{2(precision)(recall)}{precison + recall}$$

The harmonic mean of precision and recall constitutes the F1 score. Furthermore, macro-average and micro-average metrics were also used. The macroaverage weights each class equally, even underrepresented classes, and then gets the average. Microaverage, on the other hand, considers all labels of all samples equally.

Figure 7 shows an example of a classification report generated:

	precision	recall	f1-score	support
toxic	0.79	0.84	0.81	1520
severe_toxic	0.65	0.14	0.22	162
obscene	0.83	0.84	0.83	856
threat	0.52	0.35	0.42	37
insult	0.81	0.74	0.77	808
identity_hate	0.64	0.47	0.54	138
micro avg	0.80	0.76	0.78	3521
macro avg	0.71	0.56	0.60	3521
weighted avg	0.79	0.76	0.77	3521
samples avg	0.07	0.07	0.07	3521

Figure 7

5. Results

5.1 Hybrid Naïve Bayes and SGD Classifier

The evaluation of our baseline method is shown in Figure 8:

Accuracy: 0.9091963026789911
Hamming Loss: 0.020538931536894876

	precision	recall	f1-score	support
toxic	0.77	0.73	0.75	3056
severe_toxic	0.44	0.12	0.18	321
obscene	0.86	0.70	0.77	1715
threat	0.20	0.20	0.20	74
insult	0.76	0.58	0.65	1614
identity_hate	0.48	0.15	0.23	294
micro avg	0.77	0.63	0.69	7074
macro avg	0.58	0.41	0.46	7074
weighted avg	0.76	0.63	0.68	7074
samples avg	0.07	0.06	0.06	7074

Figure 8

The model has an overall accuracy of 91.2%, suggesting that the model performs well for the majority of predictions. However, examining the recall values, especially for less represented classes such as 'threat' and 'identity_hate', reveals that the model struggles to identify these rarer cases. The low recall and F1-scores for these classes highlight the challenges of working with imbalanced datasets.

The micro average F1-score of 0.70 outperforms the macro average of 0.48, reflecting the impact of class imbalance on the model's ability to generalize across less frequent labels. Hamming loss at 0.0200 indicates a relatively low rate of incorrect label predictions to the total number of labels, which is encouraging for a multi-label classification task.

5.2 LSTM Model

Due to timing constraints, we only trained for 3 epochs. Batch size was set to 64 due to limitation of GPU memory and learning rate was set to 1×10^{-3}. The results are shown in Figure 8, and training loss across epochs is shown in Figure 9.

The model has an overall accuracy of 91.6%, suggesting that it performs well for the majority of predictions.

Accuracy: 0.9162802356184986
Hamming Loss: 0.018674019300664244

	precision	recall	f1-score	support
toxic	0.81	0.75	0.78	1520
severe_toxic	0.45	0.37	0.41	162
obscene	0.83	0.76	0.79	856
threat	0.00	0.00	0.00	37
insult	0.77	0.63	0.69	808
identity_hate	0.25	0.02	0.04	138
micro avg	0.79	0.67	0.73	3521
macro avg	0.52	0.42	0.45	3521
weighted avg	0.76	0.67	0.71	3521
samples avg	0.07	0.06	0.06	3521

Figure 1

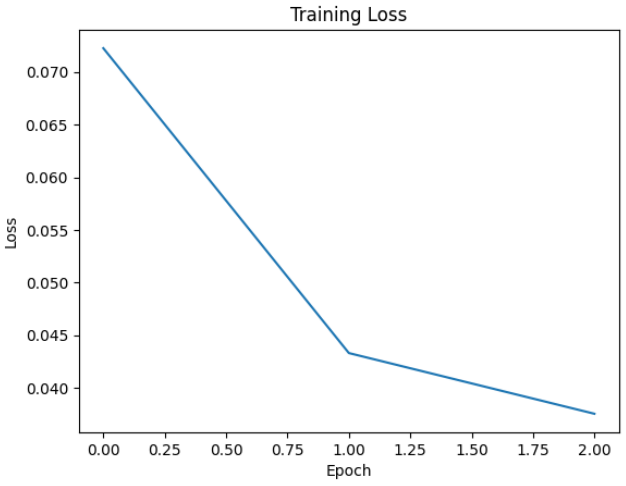

Figure 2

Examining the recall values, particularly for less-represented classes like 'threat' and 'identity_hate,' demonstrates that the model struggles to recognise these rarer cases. It couldn't even categorise a single instance of "threat" accurately.

To summarize, the model exhibits strong recall across all instances in the dataset, but its performance is not as strong across all classes, as the macro-average F1-score of 0.45 indicates. However, the micro-average F1-score of 0.73 outperforms the macro-average, highlighting the impact of class imbalance on the model's ability to generalise across less frequent labels. Additionally, hamming loss of 0.018 indicates a relatively low rate of incorrect label predictions for the total number of labels, which is encouraging for a multi-label classification task.

5.3 BERT Model

Due to timing constraints, we only trained for 3 epochs. Batch size was set to 64 due to limitation of GPU memory and learning rate was set to 2×10^{-5}. The results are shown in Figure 7, and training loss across epochs is shown in Figure 10.

The model has a overall accuracy of 92.4%, suggesting that the model performs well for the majority of predictions. However, examining the recall values, especially for less represented classes reveals that the model struggles to identify these rarer cases. While accuracy is a useful indicator, it does not always provide the full picture, especially in an imbalanced dataset.

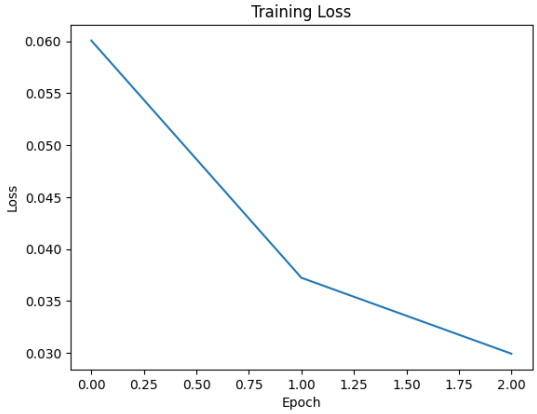

Figure 3

For toxic comments, f1-score is 0.81, which vows for high precision and recall. Despite, precision of 0.65, severe_toxic label have a very low recall. This suggests that while the model is relatively confident when it identifies a comment as severely toxic, it misses a significant number of such instances. Similarly, for other labels such as threat and identity_hate precision is moderate but recall is low.

F1-score	Model 1	Model 2	Model 3
toxic	0.75	0.78	0.81
severe_toxic	0.18	0.41	0.22
obscene	0.77	0.79	0.83
threat	0.20	0.00	0.42
insult	0.65	0.63	0.77
identity_hate	0.23	0.02	0.54

Table 1

micro	Model 1	Model 2	Model 3
Precision	0.77	0.79	0.80
Recall	0.63	0.67	0.76
F1 Score	0.69	0.73	0.78

Table 2

macro	Model 1	Model 2	Model 3
Precision	0.58	0.52	0.71
Recall	0.41	0.42	0.56
F1 Score	0.46	0.45	0.60

Table 3

6. Analysis

As shown in Tables 1, 2, and 3, Model 3 outperforms Model 1 and Model 2 across all metrics in a consistent manner, demonstrating a superior ability to generalise across the various categories of toxicity. According to the macro F1 scores, which take into account the trade-off between precision and recall for each class, Model 1 and Model 2 have trouble addressing class imbalances. According to the micro-F1 scores, which aggregate the contributions of all classes to compute the average metric, all models perform reasonably well across the entire dataset. However, Model 3's superior performance demonstrates its ability to effectively handle both majority and minority classes.

7. Conclusion

The Naïve Bayes + SGD model offers a solid baseline; however, the model's underlying feature independence assumptions limit its performance, which can be problematic for the complex language patterns that are inherent in toxic comment classification.

The LSTM model captures sequential data and long-term dependencies more effectively than Model 1. It shows a marked improvement in identifying 'severe toxic' comments, yet it falls short on classes with fewer examples like 'threat' and 'identity hate'. The dataset's imbalance hinders its recall capability, leaving room for improvement in recognizing minority classes.

The BERT model (Model 3) outperforms both predecessors with its contextualized embeddings and attention mechanisms. As shown by Model 3's amazing ability to deal with uneven data, transformer-based models are great for difficult multi-label classification tasks where context and class representation are very important.

For future work, we aim to explore the effect of data augmentation strategies and oversampling techniques on addressing the **problem of class imbalance**. Additionally, we plan to conduct experiments to fine-tune hyperparameters, which we were unable to achieve due to resource constraints. Furthermore, our objective is to investigate ensemble methods that integrate the advantages of various architectures in order to enhance performance for underrepresented classes even further.

8. Acknowledgements

We would like to thank the entire CS 535 teaching staff, particularly Zohaib Khan, for giving us project direction.

References

[1] https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/overview