Università degli studi di Milano-Bicocca

Advanced Machine Learning

Final Project

Toxic Comments Challenge

Authors:
Gargiulo Elio - 869184 - e.gargiulo4@campus.unimib.it
Gervasi Alessandro - 866140 - a.gervasi4@campus.unimib.it
Piacente Cristian - 866020 - c.piacente@campus.unimib.it

February 21, 2025

Contents

Abstract

Online discussions are frequently disrupted by toxic behavior, discouraging users from engaging in meaningful conversations. The **Toxic Comments Challenge**, organized by **Jigsaw and Google**, aims to improve the detection of harmful content, including threats, obscenity, insults, and identity-based hate speech. In this project, we analyze the dataset provided in the competition to explore word distributions and identify patterns commonly associated with toxic comments. Several **Recurrent Neural Networks (RNNs)**, including **GRU, LSTM, Bidirectional LSTM, and Bidirectional GRU**, are employed for multi-label binary classification of comments based on toxicity type. The effectiveness of these models in distinguishing toxic content is evaluated, providing insights that could guide future improvements in automated content moderation systems.

1 Introduction

Online discussions play a crucial role in modern communication, but they are often disrupted by toxic behaviors that discourage participation and harm digital communities. The **Toxic Comments Classification Challenge**, organized by **Jigsaw** and **Google**, aims to develop models capable of detecting various forms of toxicity, including threats, obscenity, insults, and identity-based hate speech [1]. The task is a **multi-label binary classification problem**, where a comment may belong to multiple toxicity categories.

Traditional keyword-based approaches struggle to capture the complexity of toxic language, making **deep learning** methods a more effective alternative. In this project, **Recurrent Neural Networks (RNNs)**, including **GRU, LSTM, and their Bidirectional variants**, were employed to classify toxic comments. Given the strong imbalance in the dataset, where non-toxic comments vastly outnumber toxic ones, getting very strong performances from the models will be particularly difficult and challenging.
The effectiveness of the proposed models is evaluated using **macro F1-score** and **Mean ROC-AUC**, ensuring a comprehensive assessment across all toxicity categories.

2 Dataset

The dataset used in this project comes from the **Toxic Comments Classification Challenge** and consists of user comments from Wikipedia talk pages. Each comment is labeled with one or more toxicity categories, making this a **multi-label classification problem**. In particular the labels are the following:

- **toxic**, **severe_toxic**, **obscene**, **threat**, **insult**, **identity_hate**

2.1 Exploratory Data Analysis

To better understand the dataset, an exploratory data analysis (EDA) is necessary. This process focuses on analyzing the dataset to ensure it is clean and consistent, while also examining the general distribution of the data and relationships between variables. Specifically:

- **Data Cleaning**: Checking for missing values and duplicate rows.

- **Comment Distribution**: Analyzing the proportion of toxic vs non-toxic comments.
- **Univariate Analysis**: Visualizing individual labels (e.g., word clouds, class distribution and sentiment analysis).
- **Covariate Analysis**: Investigating relationships between labels (e.g., correlation matrix, class distribution against clean comments, and crosstabs).

The dataset resulted already cleaned, without missing values and duplicate rows, which allowed a smooth analysis process.

The following sections will provide more insights about some of the most important operations done in the analysis.

2.2 Dataset Distribution

The dataset is highly imbalanced, with most comments being non-toxic. Figure 1 shows the dataset split, given by the Kaggle page, while Figure 2 highlights the class imbalance.

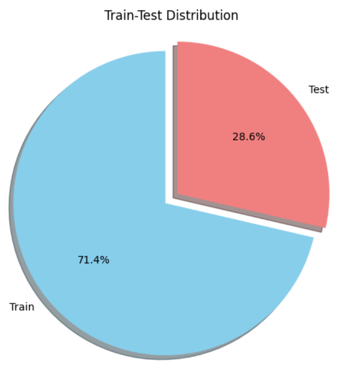


Figure 1: Training vs. Test set distribution.

Additionally, the distribution of toxic labels is uneven, with some categories appearing far less frequently than others. Figure 2 illustrates the class imbalance, while Figure 3 shows the number of toxicity labels per comment.

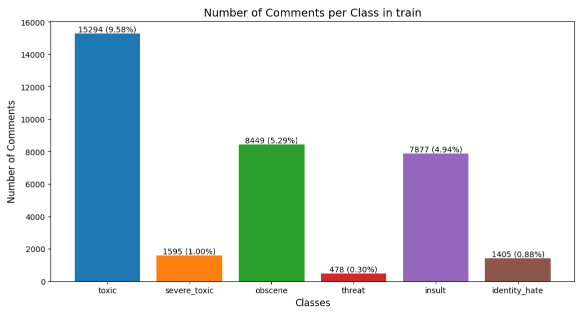


Figure 2: Class distribution in the training set. Threat and Identity Hate are very rare

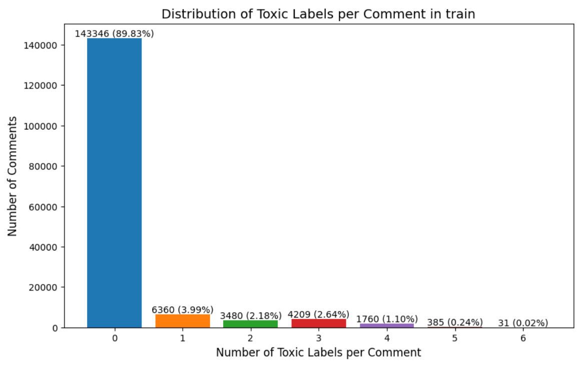


Figure 3: Number of toxicity labels per comment. 89% of comments are non-toxic

2.3 Comments and Tokens

Focusing specifically on words (tokens), a **word cloud** was generated to visualize the most frequently used terms in the comments. As shown in Figure 4, many common words are related to Wikipedia discussions, reflecting the nature of the dataset.

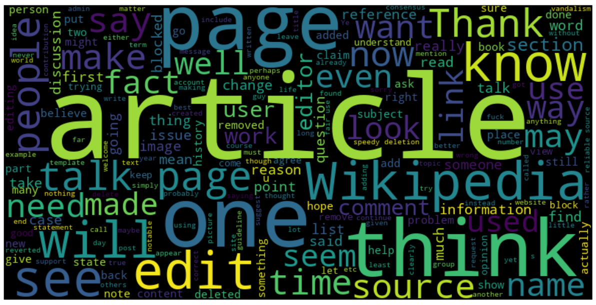


Figure 4: Word cloud of the most frequent words.

Furthermore, most comments are relatively short, but some exceed 1000 tokens. The distribution of comment lengths is summarized in Table 1.

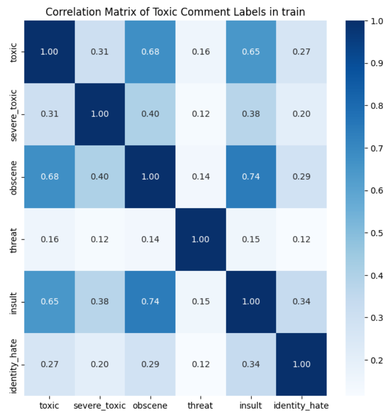
Table 1: Distribution of Comment Lengths (Token Count)

Token Range	**Number of Comments**
0 - 50	125,328
51 - 100	20,839
101 - 200	6,694
201 - 300	2,558
301 - 400	1,281
401 - 500	690
500+	617

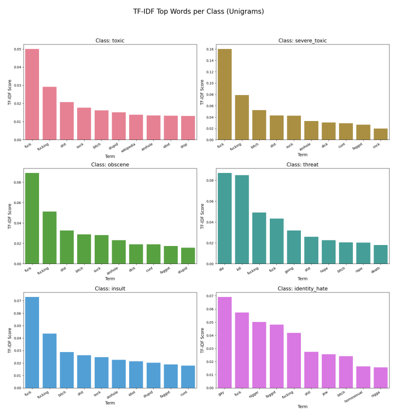
2.4 Label Correlation and Key Terms

Certain toxicity labels often appear together, indicating strong correlations. Figure 5a shows the relationships between toxicity categories, with notable overlaps between *toxic* and *insult*, as well as *obscene* and *insult*.

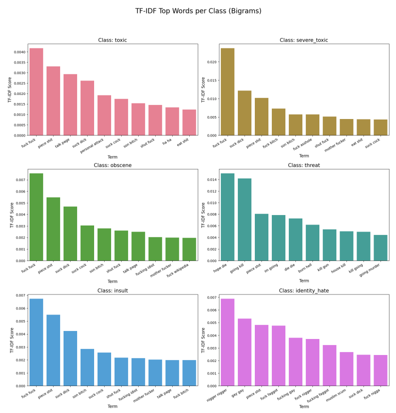
To further analyze toxicity patterns and better understand the distinguishing characteristics of each classes, a TF-IDF analysis was performed to extract the most relevant words for each category. The results, shown in Figure 5, highlight the key unigrams and bigrams associated with different toxic labels.



(a) Correlation matrix of toxicity labels.



(b) Top words per class using TF-IDF - Unigrams



(c) Top words per class using TF-IDF - Bigrams

Figure 5: Label Correlation and Key Terms

2.5 Sentiment Scores Analysis

Sentiment analysis is a technique in natural language processing (NLP) used to determine the emotional tone of a text. In the context of toxic comment classification, sentiment scores help in understanding the polarity of comments and their potential correlation with toxicity. By quantifying sentiment, it is possible to analyze whether toxic comments tend to be more negative, while non-toxic ones exhibit a more neutral or positive sentiment.

The sentiment scores are computed using a lexicon-based approach, assigning numerical values to different sentiment components:

- **Negative**: Represented the degree of negative sentiment, ranging from 0 to 1.
- **Neutral**: Represented the degree of neutrality, ranging from 0 to 1.
- **Positive**: Represented the degree of positive sentiment, ranging from 0 to 1.
- **Compound**: A normalized overall sentiment score that ranged from -1 to 1, where -1 indicated strong negativity and 1 indicated strong positivity.

Figure 6 shows that most comments exhibit neutral sentiment, as highlighted by the predominance of yellow (neutral) and the compound sentiment distribution (light blue).

While both negative (red) and positive (green) sentiments are present, they are less frequent.

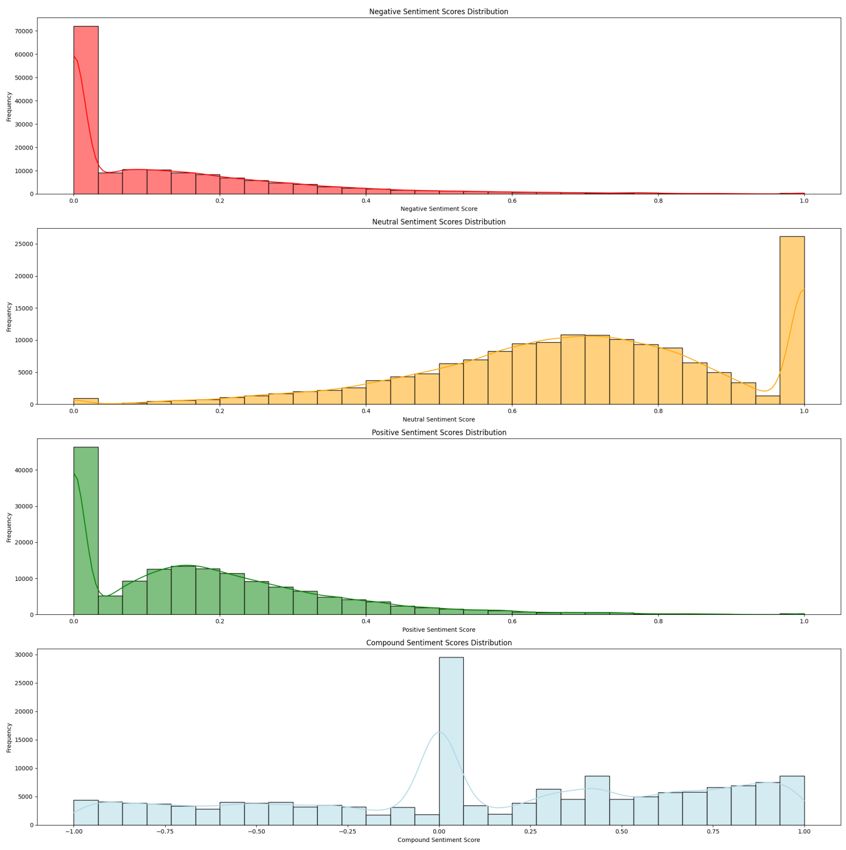


Figure 6: Sentiment scores: Red for Negative, Yellow for Neutral, Green for Positive, and Light Blue for Compound sentiment.

3 The Methodological Approach

This section outlines the methodology used to develop the toxicity classification model, covering data preprocessing, sequence transformation, model selection, training strategies, and challenges encountered.

3.1 Initial Text Processing

The dataset consists of raw text comments that need to be cleaned and preprocessed to extract meaningful and relevant words. Since the models operate on numerical inputs, the text needs to be first cleaned and tokenized, and later used for sequences creation. The following preprocessing steps were applied, based on Natural Language Processing (NLP) techniques:

- **Normalization**: Standardized text by converting all words to lowercase, while also removing links, username tags, newline tags, and emoticons present in the comments.
- **Tokenization**: Comments were split into individual word tokens.
- **Stopword Removal**: Eliminating common words (e.g., “the”, “is”) that provide little semantic value.

- **Lemmatization**: Reducing words to their base forms, ensuring they're valid vocabulary words, for better text consistency.

3.2 Feature Engineering

Once the raw comments have been processed, the tokens obtained need to be converted into numerical sequences (the actual features) before being used as input for deep learning models. In particular, the steps are the following:

- **Vocabulary Building**: Only the **10,000** most frequent words were retained, with an additional token for out-of-vocabulary (`<OOV>`) words. This is to lighten the complexity of the input for the models.
- **Sequence Encoding**: Each token was mapped to a unique integer ID.
- **Padding and Truncation**: All sequences were set to a fixed length of **200** tokens using post-padding (zeros added at the end if shorter).

These transformations led to the creation of features used by the models through an **embedding layer**, which reduces the input dimension to 50.

3.3 Train-Validation Split

The dataset provided by Kaggle was already split into training and test sets. Therefore the splitting has been done following traditional splitting percentages and using only the training part of the set, keeping **80% of training data**, and getting **20% data for validation**.

3.4 Model Architectures

Given the sequential nature of textual data, various **Recurrent Neural Network (RNN)** architectures were implemented. At the same time, a benchmark using a simple Fully-Connected Neural Network has been done. Specifically:

- **Fully-Connected Neural Network (Benchmark)**: A baseline model with an embedding layer, a `ReLU`-activated dense layer, and a final `sigmoid` output layer. Acts as a starting point for the model performances.
- **LSTM (Long Short-Term Memory)**: Captures long-term dependencies in text using memory cells. Suitable for sequences of text.
- **GRU (Gated Recurrent Unit)**: A more computationally efficient alternative to LSTM, with comparable performance.
- **Bidirectional LSTM and GRU**: Process sequences in both forward and backward directions, improving contextual understanding.

3.5 Baseline vs. Regularized Models

The main approach that the development followed was to create two versions of each RNN model (particularly the best ones):

- **Baseline Models**: Standard architectures without additional regularization. Built using simple thumb rules to have a starting point. The hidden layers use **64 units**, as they are very close to the **embedding output** dimension of 50.
- **Regularized Models**: Enhanced with dropout layers (0.3) to prevent overfitting, **ReduceLROnPlateau** to adjust the learning rate dynamically, and **early stopping** to halt training when validation performance plateaued. The amount of layers hasn't been altered, since after several attempts, it only increased the amount of overfitting while not improving the performance.

Regularization was applied to mitigate overfitting, particularly given the dataset's class imbalance, while trying to improve the baseline performances.

3.6 Training Strategy

The models were trained using the following configuration:

- **Batch Size: 128**, balancing computational efficiency and stable gradient updates.
- **Epochs: 5**, since more epochs did not bring any improvements. The models generally converged in the first epochs.
- **Loss Function:** Binary cross-entropy, suitable for **independent** multi-label classification if used with **sigmoid**.
- **Optimizer:** Adam optimizer with default hyperparameters.
- **Regularization:** Dropout layers (0.3) and learning rate adaptation (ReduceLROnPlateau).
- **Early Stopping:** Training stops if validation loss doesn't improve for three epochs (patience), restoring the best weights to prevent overfitting and unnecessary computations.

3.7 Ensemble Learning

In addition to individual models, an **ensemble** approach was tested by averaging the predictions of the best-performing regularized models (**Bidirectional LSTM and Bidirectional GRU**). This method aimed to improve robustness and generalization by leveraging complementary strengths of both architectures.

3.8 Evaluation Metrics

The following metrics were used to assess model performance:

- **Macro F1-score:** Evaluates classification performance across all toxicity categories equally, mitigating class imbalance effects. Moreover, metrics like accuracy were not reliable while evaluating the model. Accuracy stayed around 99% for every model, making it not a significant metric to analyze.
- **Mean ROC-AUC:** Measures the model's ability to distinguish between toxic and non-toxic comments. It was used as the main evaluation method in the Kaggle Challenge.

The best-performing architecture was selected based on these metrics and further analyzed in the next section.

4 Results and Evaluation

This section presents the performance of the models in terms of classification effectiveness and computational efficiency.

4.1 Training and Validation Performance

Table 2 summarizes the results on the training and validation sets. The benchmark model achieved the highest training F1-score but exhibited poor generalization, indicating overfitting. Among RNN architectures, GRU outperformed LSTM in both standard and bidirectional configurations. Regularization techniques, including dropout and learning rate scheduling, helped mitigate overfitting but slightly reduced validation scores.

Table 2: Performance on Train and Validation Sets

Model	**Train Macro F1**	**Train Loss**	**Val Macro F1**	**Val Loss**
NN Benchmark	0.7133	0.0240	0.4907	0.0777
LSTM Baseline	0.3901	0.1078	0.3810	0.0563
GRU Baseline	0.5745	0.0379	0.5103	0.0525
Bi-LSTM Baseline	0.4930	0.0398	0.4395	0.0521
Bi-GRU Baseline	0.5502	0.0377	0.4909	0.0512
Bi-LSTM Regularized (Best Epoch)	0.4556	0.0472	0.4375	0.0511
Bi-GRU Regularized (Best Epoch)	0.4327	0.0477	0.4167	0.0496

4.2 Test Performance

Table 3 reports the models' performance on the test set, including Kaggle competition scores. The public score is computed on a subset of the test set and is visible during the competition. The private score is computed on a separate hidden portion of the test set and is used for final rankings.
The **Bidirectional GRU** achieved the highest **Macro F1-score (0.4470)** and **Mean ROC-AUC (0.9678)**, outperforming LSTM-based models. The ensemble model, combining Bidirectional LSTM and GRU, did not significantly improve performance over individual models, but overall reduced the Test Loss.

Table 3: Performance on Test Data

Model	Test Macro F1	Test Loss	ROC AUC	Kaggle Private Score	Kaggle Public Score
NN Benchmark	0.4696	0.0963	0.9573	0.9570	0.9589
LSTM Baseline	0.3446	0.0731	0.9595	0.9595	0.9588
GRU Baseline	0.4585	0.0887	0.9663	0.9660	0.9676
Bi-LSTM Baseline	0.3817	0.0843	0.9637	0.9638	0.9630
Bi-GRU Baseline	0.4470	0.0811	0.9678	0.9671	0.9703
Bi-LSTM Regularized	0.3667	0.0792	0.9615	0.9614	0.9606
Bi-GRU Regularized	0.3705	0.0749	0.9634	0.9635	0.9632
Ensemble	0.3734	0.0754	0.9643	0.9644	0.9631

4.3 Computational Efficiency

The computational cost of each model was assessed based on the average training time per epoch (Table 4). GRU models were generally faster than LSTM counterparts. The **Bidirectional GRU** offered the best trade-off between accuracy and efficiency, while regularization slightly increased training time.

Table 4: Time per Epoch

Model	Average Time per Epoch
NN Benchmark	15 seconds
LSTM Baseline	40 seconds
GRU Baseline	40 seconds
Bi-LSTM Baseline	80 seconds
Bi-GRU Baseline	75 seconds
Bi-LSTM Regularized	85 seconds
Bi-GRU Regularized	80 seconds

4.4 Testing on Custom Comments

To further evaluate a model's performance, a test was conducted using custom comments to assess the effectiveness of the classification. Specifically, the ensemble model was applied to predict the classes of manually written comments, including both clean and toxic examples.

The results, also partially shown in table 5, closely align with the values observed in the Macro F1 scores and class distributions, highlighting that the classes "threat" and "identity_hate" are more challenging for the model to classify accurately, seen in table 5b. Additionally, the model performs well in identifying clean comments shown in table 5a, which can be attributed to the class imbalance. However, the model is also effective in identifying toxic comments, as it is able to recognize words highlighted in the TF-IDF analysis, leading it to correctly classify such comments as toxic.

Table 5: Class Probabilities for "Clean" and "Toxic" Comments

(a) "Welcome to Wikipedia, I will help you find more knowledge :-)"

Class	**Probability**
toxic	0.002267
severe_toxic	0.000022
obscene	0.000344
threat	0.000290
insult	0.000976
identity_hate	0.000314

(b) "I will kill you, I found your house from your IP address! You are going to die!"

Class	**Probability**
toxic	0.808958
severe_toxic	0.016382
obscene	0.280066
threat	0.028481
insult	0.410670
identity_hate	0.060257

4.5 Summary

The **Bidirectional GRU** emerged as the best-performing model, achieving the highest test Macro F1-score (0.4470) and ROC-AUC (0.9678). Regularization strategies improved generalization but slightly reduced raw performance. The ensemble model did not provide a significant boost over individual models, besides the loss reduction.

5 Discussion

The results highlight the challenges of detecting toxic comments, particularly due to class imbalance and overlapping toxicity categories. The **Bidirectional GRU** demonstrated the best performance in terms of **Macro F1-score** and **Mean ROC-AUC**, indicating its superior ability to model sequential dependencies in textual data. However, despite its strong performance, absolute F1-scores remain relatively low, reflecting the difficulty of accurately predicting all toxicity labels.

Regularization techniques, such as dropout and adaptive learning rate scheduling, helped reduce overfitting but slightly lowered validation performance. This suggests that while regularization improves generalization, further hyperparameter tuning or alternative architectures may be required to maximize effectiveness. The ensemble model, combining Bidirectional LSTM and GRU, did not significantly outperform individual models, indicating that both architectures capture similar patterns in the dataset.

One key limitation of this approach is the reliance on a simple token-based representation without contextual embeddings. More advanced techniques, such as transformer-based models (e.g., BERT), could further enhance performance by capturing richer linguistic features. Additionally, handling data imbalance remains a major challenge. While techniques such as oversampling were not applied in this project, their potential impact on robustness should be explored in future work.
The results provide valuable insights into the effectiveness of different RNNs architectures for toxic comment classification. However, improvements in both model architecture and data augmentation strategies are needed to achieve more reliable and robust systems.

6 Conclusions

This study evaluated multiple deep learning architectures for toxicity classification, comparing baseline, regularized, and ensemble approaches. The **Bidirectional GRU** emerged as the best-performing model, achieving the highest **Macro F1-score** and **Mean ROC-AUC**. Regularization strategies improved generalization but did not significantly enhance validation performance, while the ensemble model provided only marginal improvements over individual architectures.
Table 6 summarizes the final model rankings based on the Kaggle private leaderboard score. The Bidirectional GRU achieved the highest ranking, reinforcing its effectiveness in multi-label toxicity classification.

Table 6: Model Ranking based on Kaggle Private Score

Rank	**Model**	**Private Score**
1st	GRU Bidirectional	0.96713
2nd	GRU Baseline	0.96602
3rd	Ensemble Model	0.96442
4th	GRU Bidirectional Regularized	0.96352
5th	LSTM Bidirectional	0.96376
6th	LSTM Bidirectional Regularized	0.96141
7th	LSTM Baseline	0.95952
8th	NN Benchmark	0.95695

6.1 Future Works

The results demonstrate that **RNNs are effective for toxicity detection**, but further improvements are needed. In summary:

- Techniques such as oversampling could improve F1-scores for underrepresented labels like *Threat* and *Identity Hate.*
- CNNs could be investigated as an alternative to RNNs, leveraging their efficiency in capturing local patterns while reducing training time.
- Combining CNNs with RNNs or Transformer-based models (such as BERT) may enhance feature extraction and sequence understanding.
- Using embeddings from models like FastText or Word2Vec could improve text representation, benefiting classification.
- Further tuning dropout rates, learning rates, and model depth could enhance generalization while maintaining efficiency, though at the cost of increased computational power.
- Model pruning techniques could be explored to reduce the size of existing models.

Overall, GRU-based models with regularization provide the best balance between performance and generalization, while ensembling helps stabilize predictions.

References

[1] Jigsaw and Google, “Toxic comment classification challenge,” 2018, accessed: 2025-02-20. [Online]. Available: https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge