Detecting Toxicity in Online Discussions: A Comparative Study of Linear and Non-Linear Models

Real-Time Text Toxicity Detection

ECS 171

Team 6

Aleena Basil, Desiree Bishnoi, Pranaya Rao Ganta, Serina Moon, Roshni Sandhu, Divyansh Singh, Maanit Shah, Pritul Vachhani, Aiden Xie, Michelle Yeoh, Ehab Raza Zaidi

Abstract

The exponential growth of user-generated content across digital platforms has amplified the need for scalable, accurate systems that can identify toxic language. Traditional moderation pipelines struggle to keep up with the volume, speed, and evolving nature of online discourse, making machine learning essential for automating toxic comment detection.

Maintaining healthy online discussions requires balancing open expression with civil and safe communication. Human moderation alone cannot handle the scale of content on modern platforms. Recent advances in natural language processing and machine learning have enabled automatic detection and classification of harmful language patterns in text.

In this work, we examine binary toxic comment classification on the Kaggle Toxic Comment Classification dataset. We evaluate three modeling approaches with distinct architectural assumptions: logistic regression, Random Forest ensemble, and Convolutional Neural Network (CNN). We compare models across multiple performance dimensions using accuracy, precision, recall, F1-score, ROC-AUC, and threshold sensitivity analysis.

Our analysis reveals critical trade-offs between interpretability, computational efficiency, and detection capability across the three approaches, providing practical guidance for deploying toxicity classifiers in content moderation systems.

1 Introduction

Online platforms such as Reddit, YouTube, Twitter, and news forums facilitate global conversation at unprecedented scale. While these platforms create space for meaningful social interaction and community-building, they also introduce significant risks: harassment, hate speech, threats, and other forms of toxic language can degrade discourse quality and cause real-world harm. This toxicity strains individual users, entire communities, and erodes societal trust.

Traditional human moderation is essential for resolving edge cases and enforcing community norms. However, no team of moderators can feasibly review millions of comments per day. Even well-resourced technology companies struggle with the speed, volume, and subtlety of potentially harmful content. These limitations motivate the need for automated toxicity detection, enabling platforms to filter, flag, or down-rank harmful comments proactively.

Machine learning and natural language processing (NLP) methods offer promising solutions to this challenge. Over the past decade, toxicity detection has evolved from simple keyword matching to sophisticated classification models using n-grams, neural networks, and transformer-based architectures. Prior work has demonstrated strong potential for automating toxicity detection through approaches ranging from traditional feature-based classifiers to modern deep learning methods. However, several limitations remain. Many models lack interpretability, making moderation decisions difficult to justify. Others struggle with misspellings, slang, or subtle forms of toxicity such as sarcasm and coded language. Research has also shown that toxicity classifiers can inherit or amplify social biases, disproportionately misclassifying comments referencing certain identity groups.[1]

Despite these advances, a key question remains: what are the practical trade-offs between different modeling paradigms for toxicity detection? While deep learning approaches often achieve strong re-

[1] https://aclanthology.org/P19-1163/

sults on benchmark datasets, their computational requirements and lack of interpretability may limit deployment in production systems where decisions must be explained. Conversely, simpler models may offer sufficient performance with greater efficiency and transparency. Understanding these trade-offs is critical for selecting appropriate models for real-world content moderation.

This work provides a systematic comparison of three fundamentally different approaches to toxic comment classification. We implement logistic regression and Random Forest models using Term Frequency-Inverse Document Frequency (TF-IDF) vectorization to convert text into numerical features, alongside a CNN that learns dense word embeddings, which are vector representations that capture semantic relationships between words. By evaluating these models under identical conditions on the same dataset, we isolate the impact of model architecture on detection performance, computational efficiency, and interpretability. This comprehensive comparison reveals which modeling choices best balance the competing demands of accuracy, speed, and explainability in toxicity detection systems.

2 Methods

Our approach centers on comparing three fundamentally different model architectures for toxic comment classification: a linear classifier, an ensemble method, and a neural network. Because of this, we have slightly differing preprocessing steps for each of the models. The logistic regression and Random Forest models rely on engineered features derived from the text, while the CNN model learns dense word representations through an embedding layer. This diversity allows us to evaluate how different modeling assumptions affect performance on real-world toxic comment data characterized by class imbalance, linguistic variation, and noisy labeling.

2.1 Dataset

This study uses the Jigsaw Toxic Comment Classification Challenge dataset from Kaggle, consisting of approximately 160,000 Wikipedia talk page comments annotated by human raters for toxic behavior. Each comment is labeled across six toxicity categories as shown in Table 1. These labels represent a spectrum of harmful behaviors ranging from mild verbal aggression to explicit threats and identity-based harassment.

Table 1: Toxicity Label Categories

toxic	severe_toxic	obscene
threat	insult	identity_hate

The dataset originates from Wikipedia's talk page edits, providing realistic examples of online discourse, including both constructive discussion and various forms of abuse. The original training set contains 159,571 observations, though our analysis uses a subset after preprocessing and data cleaning operations.

A critical characteristic of this dataset is severe class imbalance. Toxic comments comprise just over 10% of the data, with the rarest categories, threat and identity_hate, appearing in less than 1% of comments. This imbalance reflects real-world comment distributions where most content is non-toxic, but it poses significant challenges for model training. Without corrective measures, classifiers tend to achieve high accuracy by simply predicting the majority class while failing to detect rare but critical toxicity types.

2.2 Data Preprocessing

Raw text from online platforms contains significant noise that can degrade model performance. Our preprocessing pipeline standardizes the text while preserving meaningful linguistic patterns relevant to toxicity detection. We apply consistent text cleaning across all three models, then employ model-specific feature extraction techniques.

2.2.1 Text Cleaning

We implement comprehensive text normalization for all models using a cleaned dataset file. All text is converted to lowercase to ensure consistent treatment of words regardless of capitalization. We replace URLs and web addresses with a placeholder token **url**. Similarly, hashtags and user mentions are replaced with the generic token **tag**. Special characters and punctuation are removed or replaced with spaces, and multiple consecutive spaces are collapsed to single spaces. Missing comment text values are filled with empty strings to avoid errors during feature extraction. This cleaning procedure reduces feature space dimensionality while maintaining the semantic content of comments.

2.2.2 Label Construction

Recall that the dataset has six different types of toxicity, which conflicted with our initial goal of creating a binary toxic/not toxic classifier. This presented a choice between two main options: (1) creating a single binary label by applying a logical OR operation across all six categories, or (2) treating all six labels as independent targets for a multi-label prediction. The single binary label is computationally simpler and provides a definitive 'flag' for human review, but the multi-label approach retains information about the type and severity of toxicity. We wanted to investigate this trade-off by adopting both strategies. The logistic regression model performed binary classification, using the logical OR operation to create a unified toxicity label. If a comment scored 1 in any of the six columns, it received a binary label of 1 (toxic). In contrast, the Random Forest and CNN models were designed for multi-label classification. The Random Forest used a OneVsRest strategy, essentially training six separate binary models, one for each category, while the CNN was designed with a multi-output final layer. This allowed these models to predict the specific types of toxicity present in a comment, supporting more nuanced moderation decisions.

2.3 Feature Pre-Processing

2.3.1 TF-IDF vectorizer

The cleaned text is transformed into numerical features for the Logistic Regression and Random Forest models to use. This is handled using the TF-IDF Vectorizer (Term Frequency-Inverse Document Frequency), which calculates the Term Frequency (TF), how often a word appears in a comment, and scales it by the Inverse Document Frequency (IDF), how rare the word is across the entire dataset. This gives more weight to rare, informative words while removing high-frequency, low-value words like 'is', 'the', and 'a', as they do not contribute meaningfully to the classification of toxicity.

The logistic regression model does a feature union of word-level and character-level vectors into a single feature set. TF-IDF vectorizer is used to calculate a score for both a word and character basis to omit stop words, typos, and slang. The random forest model instead uses the TF-IDF vectorizer to limit the vocabulary size to 50000 most frequently occurring words, reducing dimensionality and removing stop words as well. This vectorization step ensures all modeling is performed on a dense, quantitative feature set.

2.3.2 CNN Feature Pre-Processing

The Convolutional Neural Network (CNN) model utilizes word embeddings, representing a fundamental departure from the TF-IDF feature engineering used by the classical models. Word embeddings allow the model to capture the semantic and contextual similarity of words in a dense vector space, moving beyond simple word counts. This rich feature representation is crucial for identifying context-dependent toxicity and the nuances of online language.

In our implementation, the first step is to transform the cleaned text into a sequence format. A Keras Tokenizer converts each word into an integer index based on a fixed vocabulary built from the training data. In addition, a challenge in neural network design is that all input samples must have the same dimension. To solve this, these integer sequences are then either padded with zeros or truncated to a fixed length of 200 tokens, which establishes a uniform input dimension required by the neural network.

The core of the feature transformation happens in the initial Embedding layer of the CNN architecture. This layer maps each word index w_i in the sequence into a 100-dimensional dense vector $\mathbf{e}_i$. These vectors are not based on pre-trained knowledge; rather, they are randomly initialized and trained concurrently with the rest of the CNN. This end-to-end training process is highly effective, as it learns a task-specific vector space where toxic and non-toxic linguistic features are optimally distinguished. This sequence-based representation preserves the original word order, providing a critical advantage over TF-IDF's bag-of-words assumption.

2.4 Model Classifiers

Following feature representation, the subsequent stage involves training three distinct model architectures. These models—a linear classifier, an ensemble method, and a deep neural network—each embody fundamentally different strategies for learning from the data, providing a comprehensive assessment of which paradigm is best suited for the noise and imbalance inherent in online toxic comment classification.

2.4.1 Logistic Regression Classifier

The Logistic Regression classifier was chosen to establish a highly efficient and interpretable baseline for binary toxic comment detection. The model operates by learning a linear combination of input features which is then passed through the sigmoid function to model the probability of a comment belonging to the toxic class. This probability is given by:

$$P(\text{Toxic}|\mathbf{x}) = \frac{1}{1 + e^{-z}}$$

where z is the linear component $z = \mathbf{w}^T\mathbf{x} + b$, $\mathbf{x}$ is the composite TF-IDF feature vector, $\mathbf{w}$ is the learned weight vector, and b is the bias term.

The model was trained on the composite TF-IDF feature set (Section 2.3.1). Hyperparameter tuning was performed using 3-fold cross-validation via GridSearchCV on the training set, searching over the regularization strength C. The regularization is used to prevent overfitting by penalizing large coefficients in the weight vector $\mathbf{w}$. The cross-validation process systematically tested $C \in \{0.5, 1.0, 2.0\}$ and selected $C = 2.0$ as the value that yielded the highest average F1-score across the validation folds, indicating the optimal balance between bias and variance for the given dataset. The model was trained using the *liblinear* optimization solver, which is an implementation of a coordinate descent algorithm specialized for optimizing linear classifiers on large, sparse datasets, and was allowed a maximum of 700 iterations to converge.

To manage the severe class imbalance, a *balanced* class weighting strategy was applied. This strategy automatically computes weights that are inversely proportional to the class frequencies, ensuring that the minority (toxic) class contributes proportionally more to the loss function, preventing the model from ignoring rare but critical examples.

A critical step for deployment was the Threshold Tuning for Deployment. The model's output probability must be converted into a binary classification. In content moderation, a false negative (missing a toxic comment) is often more detrimental than a false positive, meaning the default 0.5 threshold is unsuitable. Therefore, Precision-Recall curve analysis was performed on the validation set, revealing 0.67 as the optimal operating point. By setting the decision boundary to this value, the classifier prioritizes prediction confidence, ensuring that a comment must have a significantly higher probability of toxicity to be flagged, which drastically reduces the manual review load caused by false positives. This optimal threshold value, 0.67, is considered a critical operational parameter.

Table 2: Logistic Regression Hyperparameters

Parameter	Value
Regularization Strength (C)	2.0
Class Weighting Strategy	balanced
Optimization Solver	liblinear
Maximum Iterations	700
Optimized Threshold	0.67

2.4.2 Random Forest Classifier

The Random Forest model was selected to provide a non-linear, ensemble-based contrast to the linear baseline, and was designed for multi-label classification across all six toxicity categories. The core algorithm is an ensemble of 100 individual decision trees, with the final prediction determined by aggregating the votes of these trees. This ensemble approach increases robustness and allows the model to capture non-linear interactions between tokens in the sparse TF-IDF feature space.

Since the base Random Forest algorithm can only handle a single label, the OneVsRest (OvR) strategy was employed for multi-label classification. OvR decomposes the six-label problem into six separate, independent binary classification tasks. This requires training six dedicated Random Forest models: one for each toxicity category. Consequently, the final model is composed of six distinct Random Forest classifiers, each with 100 trees, resulting in a total of 600 decision trees making the overall prediction.

To manage the inherent class imbalance within each of the six binary problems, a *balanced* class weighting strategy was applied to all underlying OvR classifiers. This ensures that the rare toxic comments in each category receive increased attention during training. Additionally, training time was a major concern. Running 3-fold cross-validation on 600 decision trees (6 models $\times$ 100 trees) across multiple hyperparameter settings would have been computationally infeasible within the project timeline. Our solution was a two-stage hyperparameter tuning process: Initial tuning was performed on a reduced ensemble size of 20 trees to efficiently search for optimal internal parameters. The final model was then trained using the full 100-tree ensemble with the optimized parameters. The parameters

selected included allowing the maximum tree depth to be Unlimited and setting the minimum samples required to split an internal node to 2.

Table 3: Random Forest Hyperparameters

Parameter	Value
Number of Trees	100
Class Weighting	balanced
Maximum Tree Depth	Unlimited
Min Samples per Split	2

2.4.3 CNN Classifier

The CNN model represents the deep learning approach, utilizing a multi-filter architecture to exploit the contextual and semantic information captured by the learned word embeddings for multi-label classification. The multi-label nature of the task requires the model to learn six independent probabilities, one for each toxicity class.

The architecture is built sequentially, starting with the learned word embeddings (Section 2.3.2), which convert the padded sequence of tokens into a matrix of dense vectors. The classification logic is based on a multi-channel (parallel) CNN, a robust design for text classification.

The network flow proceeds as follows: The input sequence passes through the Embedding layer and then into three parallel Conv1D layers. A Conv1D layer functions by applying a filter (a weight matrix $\mathbf{W}$) across a sequence of word embeddings, effectively performing a mathematical convolution to detect specific n-gram patterns. The output c_i for a filter at a specific position is calculated as:

$$c_i = f(\mathbf{W} \cdot \mathbf{e}_{i:i+k-1} + b)$$

Where f is the ReLU activation function, $\mathbf{W}$ is the convolutional filter weight matrix, and $\mathbf{e}_{i:i+k-1}$ is the k-gram window of word embedding vectors. The network uses 128 filters for each of the three parallel layers, applied with varying kernel sizes of 3, 4, and 5. This parallel design enables the model to simultaneously extract features corresponding to tri-gram, 4-gram, and 5-gram token patterns.

Following the convolutional layers, a Global Max-Pooling layer (GlobalMaxPooling1D) is applied to each parallel output. This layer extracts the maximum activation value across the entire sequence for each filter, creating a fixed-length feature vector that summarizes the most salient local patterns present. The three resulting vectors are concatenated into a single feature vector.

The concatenated features are then passed through the classification head of the network. This consists of a Dropout layer (rate 0.5) for regularization, followed by a dense layer (64 units, ReLU activation), and a second Dropout layer (rate 0.3). The model finishes with a final Dense layer that has six output units, corresponding to the six toxicity classes, and uses a *sigmoid* activation function to provide an independent probability for each label.

The model was trained for 5 epochs using a batch size of 64. The optimizer used is Adam, which is an adaptive learning rate optimization algorithm that computes individual learning rates for different parameters. The loss function used is *binary_crossentropy*, which is ideal for multi-label problems as it measures the loss independently for each class's prediction. The formula for binary cross-entropy (BCE) for a single sample is:

$$BCE = -\sum_{i=1}^{C} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Where $C = 6$ is the number of classes, y_i is the ground truth label (0 or 1), and $\hat{y}_i$ is the predicted probability for class i. Per-class weighting is applied during training, and custom per-class probability thresholds are used during inference to maximize the recall for critically rare categories such as 'threat' and 'severe_toxic'.

Table 4: CNN Hyperparameters

Parameter	Value
Embedding Vector Size	100
Max Comment Length	200
Convolutional Filters	128
Filter Sizes (N-gram)	3, 4, 5
Learning Rate	0.001
Batch Size	64
Training Epochs	5
Dropout Rates	0.5, 0.3

3 Results

We present the results of the comparative evaluation for the Logistic Regression, Random Forest, and CNN models. The analysis begins by defining the metrics used to assess performance, followed by

a detailed breakdown of the outcomes for each classifier.

3.1 Evaluation Metrics

Performance across all models is assessed using a standard suite of metrics critical for evaluating multi-class and imbalanced classification tasks.

- **Accuracy:** The ratio of correctly predicted instances to the total number of instances. For multi-label classification (Random Forest and CNN), this refers to subset accuracy, where all six labels for a comment must be correctly predicted.
- **Precision (Macro-Averaged):** Measures the proportion of positive identifications that were actually correct. The formula for a single class is:

$$Precision = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Positive (FP)}}$$

  The macro average is the unweighted mean of the precision score for each class.
- **Recall (Macro-Averaged):** Measures the proportion of actual positives that were correctly identified. The recall formula is:

$$Recall = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Negative (FN)}}$$

  The macro average is the unweighted mean of the recall score for each class.
- **F1-Score (Macro-Averaged):** The harmonic mean of precision and recall. It is a more robust measure than accuracy for imbalanced datasets, as it equally considers both false positives and false negatives. The formula for a single class is:

$$F1\text{-}Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

  The macro average is the unweighted mean of the F1-score for each class.
- **ROC-AUC (Macro-Averaged):** The Area Under the Receiver Operating Characteristic curve. This score measures the model's ability to distinguish between classes across all possible probability thresholds, providing a measure of how well the model ranks the positive instances above the negative instances.

3.2 Logistic Regression Model Results

The Logistic Regression model performed a binary classification task and demonstrated strong performance, particularly after adjusting the decision threshold.

The initial evaluation, performed at the default 0.5 probability threshold, resulted in an F1-score of 0.79 for the toxic class, with a recall of 0.86 and precision of 0.73. The model's overall Macro-Averaged ROC-AUC score was 0.979.

The primary goal of the Logistic Regression model was to achieve optimal performance for real-world moderation: maximizing recall while controlling the false positive rate. This led to a deliberate adjustment of the decision boundary. By raising the threshold from the default 0.5 to the optimized value of 0.67 (Table 2), the model's performance on the test set was significantly refined.

The optimized threshold of 0.67 yielded a final classification accuracy of 0.96. This tuning achieved a critical balance: the toxic class precision increased substantially from 0.73 to 0.81, while recall was maintained at 0.80, meeting the minimum target set for the project. The false positive count was reduced dramatically from 1,024 to 597 (a 42% reduction), making the model highly reliable for production use where incorrectly flagging a non-toxic comment is costly. The full classification report for the optimized model is summarized below.

Table 5: Logistic Regression Optimized Classification Report (Binary)

Metric	Non-Toxic	Toxic	Macro	Supp.
Precision	0.98	0.81	0.90	28668
Recall	0.98	0.80	0.89	3245
F1-Score	0.98	0.81	0.89	31913

Overall Accuracy: 0.96, ROC-AUC: 0.979

3.3 Random Forest Classifier Results

The Random Forest model was evaluated on the multi-label classification task. The model achieved an overall subset accuracy of 0.9099 and a Macro-Averaged ROC-AUC score of 0.9552, demonstrating solid overall performance in distinguishing toxic comments from non-toxic ones.

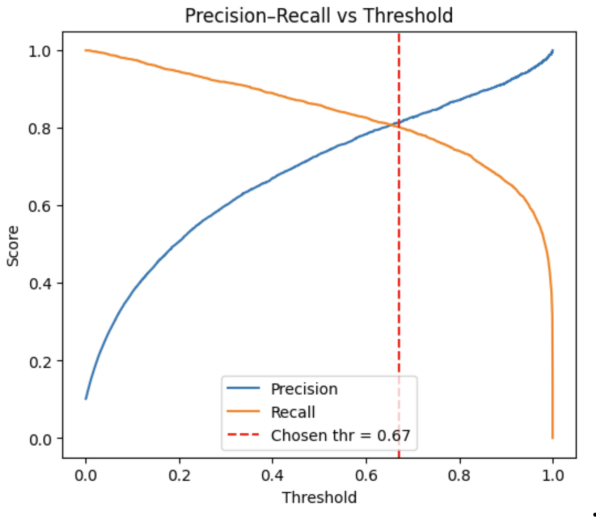

Figure 1: Logistic Regression Performance Metrics (Precision-Recall vs Threshold)

The detailed classification report (Table 5) reveals significant performance disparities across the six individual toxicity categories, primarily due to the severe class imbalance. For the two most frequent categories, 'toxic' and 'obscene', the model achieved strong precision (0.91 for both) and moderate F1-scores (0.69 and 0.73 respectively).

The model showed weakness in detecting the rarest categories:

Severe toxic: The recall score was 0.16 (F1 0.23).

Identity hate: The recall was 0.10 (F1 0.10).

Threat: The model achieved a recall of 0.07 (F1 0.13). With only 86 true instances in the test set, the confusion matrix confirms only 6 true positive predictions out of 86 actual threat comments, with 80 false negatives.

This pattern highlights that while the Random Forest classifier is highly accurate at identifying the majority non-toxic class, its reliance on sparse features and its struggle with imbalance led to low recall for the most infrequent forms of toxicity.

Overall Accuracy: 0.9099, Macro-Averaged ROC-AUC: 0.9552

3.4 CNN Classifier Results

The CNN model, leveraging learned word embeddings and custom per-class thresholds, also performed multi-label classification. It achieved an overall subset accuracy of 0.9033 and a Macro-Averaged ROC-AUC score of 0.9730. The high ROC-AUC suggests that the embeddings-based model is highly effective at ranking toxic instances.

Table 6: Random Forest Classification Report (Multi-Label)

Category	Prec.	Recall	F1	Supp.
toxic	0.91	0.56	0.69	3013
severe_toxic	0.45	0.16	0.23	314
obscene	0.91	0.61	0.73	1681
threat	0.67	0.07	0.13	86
insult	0.82	0.47	0.60	1605
identity_hate	0.10	0.10	0.10	285

The custom per-class threshold tuning was implemented specifically to boost recall for rare classes. The results show better performance across the least frequent categories:

Severe toxic: The recall score was 0.6951 (F1 0.4559).

Identity hate: The recall was 0.5637 (F1 0.4022).

Threat: The model achieved a recall of 0.3673 (F1 0.3618).

This demonstrates that the CNN's architecture, which captures semantic relationships and local n-gram patterns via the convolutional layers, proved more effective in detecting the rare, complex toxicity patterns. The full classification report is summarized below.

Table 7: CNN Classification Report (Multi-Label, Optimized Thresholds)

Category	Prec.	Recall	F1	Supp.
toxic	0.7453	0.7725	0.7587	3046
severe_toxic	0.3392	0.6951	0.4559	305
obscene	0.7783	0.8363	0.8063	1662
threat	0.3564	0.3673	0.3618	98
insult	0.7227	0.7176	0.7202	1576
identity_hate	0.3126	0.5637	0.4022	259

Overall Accuracy: 0.9033, Macro-Averaged ROC-AUC: 0.9730

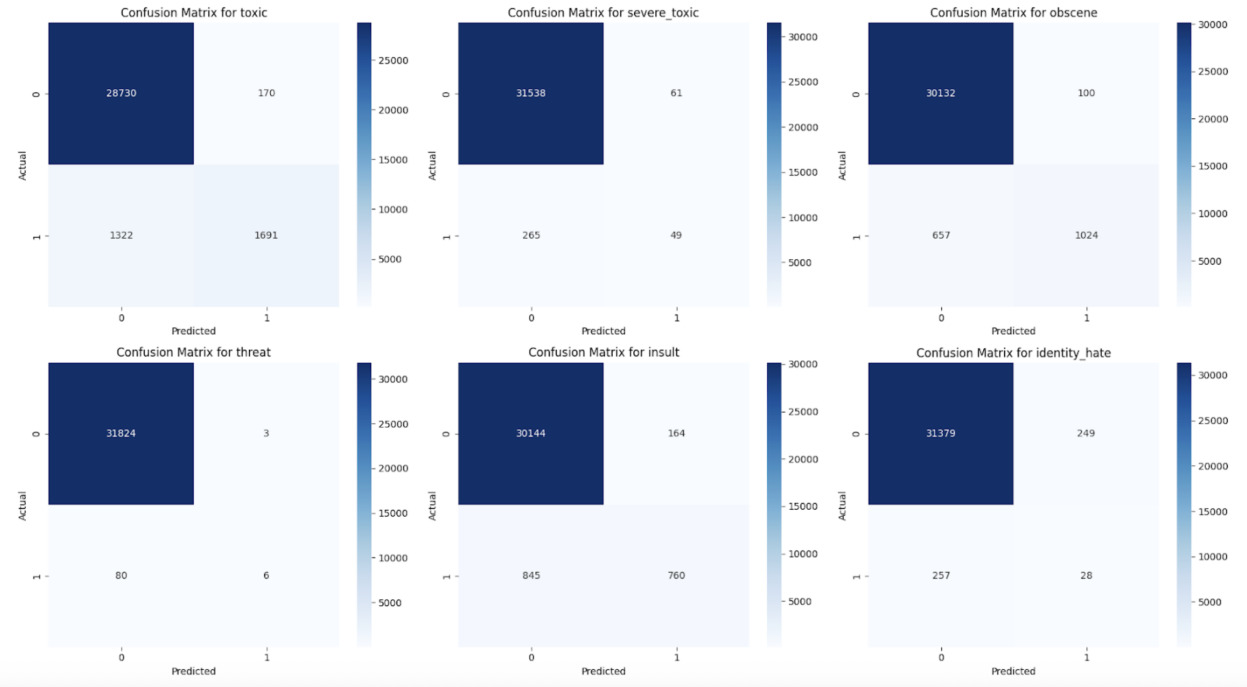

Figure 2: Random Forest Confusion Matrices Per Toxicity Category

4 Discussion

4.1 Feature Representation and Model Suitability

The choice of feature representation proved to be the single most defining factor in model performance.

The Logistic Regression and Random Forest models relied on the sparse Term Frequency-Inverse Document Frequency (TF-IDF) representation, which quantifies word importance based on frequency. For Logistic Regression, the combination of word-level and character-level n-grams provided a feature set that was sufficiently robust for the binary task. Its linear nature allowed it to efficiently establish a strong decision boundary, achieving a high ROC-AUC (0.979). The simplicity of its feature set was not a detriment for the consolidated binary goal.

In contrast, the Random Forest model struggled significantly with the sparse, high-dimensional nature of the TF-IDF feature space. Tree-based models are generally less adept than linear models at handling extreme sparsity. This architectural mismatch resulted in the model having great difficulty in detecting the rarest toxicity categories, exemplified by the very low recall of 0.07 for 'threat'. While the Random Forest achieved high precision for frequent categories like 'toxic' and 'obscene' (0.91), its inability to generalize to sparse, critical examples made its overall multi-label performance unreliable.

The CNN Model bypassed the limitations of TF-IDF by utilizing learned word embeddings. This transformation converts sparse counts into dense, contextual vectors, allowing the model to capture semantic and structural patterns. This ability to capture nuanced linguistic information is the primary reason the CNN showed significantly better recall on rare and complex categories, achieving 0.3673 for 'threat' and 0.5637 for 'identity_hate', confirming the value of learned representations for overcoming the sparsity challenge.

4.2 Handling Class Imbalance

The severe class imbalance was the main challenge across all three models, necessitating explicit mitigation strategies.

The balanced class weighting strategy applied to both the Logistic Regression and Random Forest models was successful in improving overall learning but was insufficient for the Random Forest on the smallest categories. For the multi-label models, the

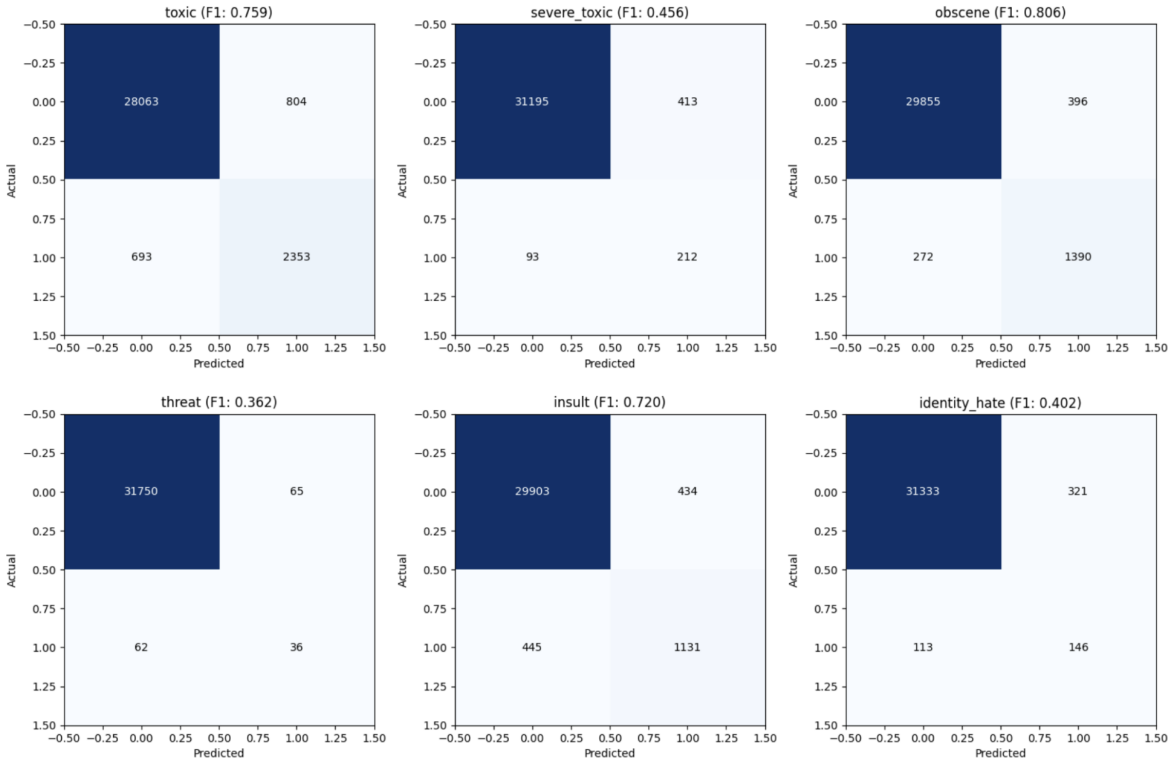

Figure 3: CNN Confusion Matrices Per Toxicity Category (Optimized Thresholds)

number of training examples for 'threat' was extremely low, and the OvR decomposition provided little additional signal for the Random Forest to learn a reliable pattern.

The CNN model's success in boosting recall for rare classes was due to its multi-pronged approach involving per-class loss adjustments during training and custom per-class threshold tuning during inference. By setting lower confidence thresholds for rare but critical classes like 'threat' (0.2) and 'severe_toxic' (0.3), the model prioritized the detection of these instances. This operational tuning is a recognition that the optimal performance threshold differs significantly based on a category's rarity and impact, a flexibility that significantly enhanced the model's detection capability.

4.3 Summary and Final Model Assessment

In summary, the comparison reveals clear trade-offs across the three architectures:

4.3.1 Computational Efficiency vs. Feature Richness

The Logistic Regression model is the most resource-efficient, making it the fastest for high-volume inference. It leveraged its simple architecture to excel at the binary task using a robust, engineered feature set. The CNN model is the most complex, but this complexity provided superior feature learning, resulting in the best performance on rare, high-impact categories.

4.3.2 Detection Reliability

The Logistic Regression model offers the most reliable overall performance for the binary task. Its 0.979 ROC-AUC and the highly refined threshold tuning that minimized false positives (a 42% reduction) established it as a robust, high-quality filter. The model's simplicity combined with disciplined optimization delivered a stable and effective classifier for the core screening need.

4.3.3 Random Forest Limitations

The Random Forest model proved to be the least suitable, as its architecture struggled with the sparse features, resulting in low recall for the most critical categories. While it achieved high precision for common categories, its failure to reliably detect threats makes it impractical for real-world moderation.

5 Limitations

Even though our study achieved strong results with all the three models, there are possible limitations to consider and take into consideration when the results are being interpreted.

5.1 Subjectivity and Label Nuance

Even for humans, it is difficult to consistently distinguish between truly harmful content, sarcasm, and aggressive but non-threatening discourse. The labels in the dataset reflect this challenge, as the annotations themselves are subjective judgements, meaning that a degree of human bias is built into the ground truth data used for training. Nuances like context, intent, and subtle coded language are often difficult to capture reliably, even by trained human annotators. Consequently, the models learn to classify comments based on this inherently noisy and complex ground truth, which limits their ability to make perfectly objective distinctions in real-world scenarios.

5.2 Severe Class Imbalance

The severe class imbalance of the underlying dataset is a significant technical limitation. The non-toxic class overwhelmingly outnumbers the toxic class, and the rarest categories, such as 'threat' and 'identity hate', have extremely low support in the training data. For example, 'threat' contained only 86 true instances in the test set. This extreme sparsity fundamentally limits the reliability of the models on these critical instances. Although explicit mitigation strategies like class weighting were applied, the models could not learn robust patterns from such limited positive examples, a constraint clearly demonstrated by the low recall of the Random Forest model on these rare categories.

5.3 Limited Generalizability Due to Data Source

Although the dataset is large, its comments are sourced exclusively from Wikipedia talk pages. This source represents a particular style of discourse that may not fully capture the continually evolving slang, coded language, or unique patterns of toxicity prevalent in other platforms like social media or smaller online communities. Therefore, the models' ability to generalize to forms of toxic speech and patterns not represented in the Wikipedia environment is constrained. This source limitation suggests that the models, despite their overall accuracy, may fail to detect new or community-specific types of abuse, and could potentially misclassify language patterns associated with certain user communities due to inherent data bias.

6 Future Work

6.1 Expanding to Multilingual Toxicity Detection

The current models, relying on English-specific techniques like TF-IDF stop-word removal and English word embeddings, are inherently limited to comments written in English. However, toxic behavior is a global issue that occurs across all languages on international platforms.

Future work should focus on expanding the system to handle multilingual toxicity detection. This requires moving away from language-specific feature extraction and integrating multilingual language models. By utilizing universal sentence encoders or large, pre-trained multilingual embeddings, the system can be trained to recognize cross-lingual patterns of toxic speech, significantly increasing the generalizability and practical applicability of the moderation tool to a global user base.

6.2 Developing a Moderation Triage and Severity Ranking System

While the current multi-label models predict the type of toxicity (e.g., 'insult', 'threat'), a critical need for production systems is to assess the severity or impact of the comment. This severity is currently a lost dimension in the final binary decision.

The next step is to leverage the predicted toxi-

city types and their probabilities to create a sophisticated Moderation Triage and Severity Ranking System. This system would not just flag content by type, but assign a final severity score or tier (e.g., Tier 1: Low-Severity, Tier 3: Frequent Harm). This involves calibrating the multi-label outputs to create a ranked prediction that enables more nuanced and actionable moderation decisions, such as automatically removing content ranked as 'Frequent Harm' versus simply prioritizing 'Low-Severity' content for human review.

7 Conclusion

The objective of this project was to provide a rigorous comparative analysis of the effectiveness of linear, ensemble, and deep learning approaches for the task of binary toxic comment classification. This was accomplished through the implementation and evaluation of a Logistic Regression model, a Random Forest classifier, and a Convolutional Neural Network (CNN) on the challenging, imbalanced Kaggle Jigsaw corpus. The comprehensive analysis successfully illuminated the strengths and weaknesses of each architectural paradigm, providing valuable insight into creating an effective, practical moderation system.

We confirmed that the **Logistic Regression** model offered the most practical and robust performance for the core binary classification task. Its efficiency and simplicity were paramount. By employing a balanced class weighting strategy and meticulously tuning the decision boundary to 0.67, the model achieved a strong 0.979 ROC-AUC and successfully prioritized reliability. This threshold adjustment substantially reduced false positives by 42%, while maintaining a strong 0.80 recall for the toxic class. This stability and high-quality output make it the most suitable model for a fast, initial screening system.

The non-linear models demonstrated varied performance, exposing the trade-offs of their architectures. The **Random Forest** model struggled significantly with the class imbalance and sparse feature representation, leading to unreliable performance on critical, rare toxicity categories like 'threat' (recall of 0.07). The **CNN model**, leveraging dense word embeddings, offered the most nuanced linguistic capture, resulting in significantly higher recall for rare categories (e.g., 'threat' recall of 0.3673). However, its complexity and resource demands limit its current applicability for high-volume, low-latency filtering.

Overall, the Logistic Regression classifier offers the most robust solution for binary toxicity detection. While the CNN captured more subtle patterns, the project demonstrates that with disciplined optimization and strategic feature engineering, simple linear models can deliver a highly stable and effective solution for the core demands of automated content screening.

References

[1] K. Khieu and N. Narwal, "Detecting and Classifying Toxic Comments," CS224N: Deep Learning for Natural Language Processing, Stanford Univ., Stanford, CA, Course Project Report, 2018.

[2] L. Breiman, "Random Forests," University of California, Berkeley, 2001.

[3] G. Louppe, "Understanding Random Forests," University of Liége, 2014.

[4] M. S. Jahan and M. Oussalah, "A systematic review of hate speech automatic detection using natural language processing," *Neurocomputing*, vol. 546, p. 126232, 2023. https://doi.org/10.1016/j.neucom.2023.126232

[5] A. Rashid, S. Mahmood, U. Inayat, M. F. Zia, "Urdu Toxicity Detection: A Multi-Stage and Multi-Label Classification Approach," *AI*, vol. 6, no. 8, p. 194, 2025. https://doi.org/10.3390/ai6080194

[6] V. Mishra and M. Tripathi, "Detecting Toxic Comments Using Convolutional Neural Network Approach," in *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, Al-Khobar, Saudi Arabia, 2022, pp. 252-255, doi: 10.1109/CICN56167.2022.10008301.

[7] S. Macavaney, H. R. Yao, E. Yang, K. Russell, N. Goharian, and O. Frieder, "Hate speech detection: Challenges and solutions," *PLoS One*, vol. 14, no. 8, pp. 1-16, 2019. https://doi.org/10.1371/journal.pone.0221152

[8] S. Awasthi, S. K. Shukla, D. Sharma, D. Gupta, and A. Tripathi, "Combating Cyber Abuse: A Toxic Comment Detection Model

Using Deep Learning," in *2025 3rd International Conference on Communication, Security, and Artificial Intelligence (ICCSAI)*, Greater Noida, India, 2025, pp. 1723-1729, doi: 10.1109/ICCSAI64074.2025.11064521.

[9] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, Antalya, Turkey, 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

[10] M. A. Khan, "Determination of toxic comments and unintended model bias minimization using Deep learning approach," arXiv preprint arXiv:2311.04789, Nov. 2023.

Author Contributions:

- **Aleena Basil:** Created the outline for the report. Worked on the data set (section 2.1) of the report about all three datasets. Worked on the presentation slides, transferring information from the report and editing it down. Added Demo/visuals to the slides.
- **Desiree Bishnoi:** Updated and expanded the README with full setup instructions, tested the frontend and backend locally, and contributed to the report by outlining key sections and drafting discussion content on dataset issues, preprocessing, and model performance.
- **Pranaya Rao Ganta:** Worked on developing the frontend and integrating it with the backend API so the model could be used directly through the interface. I also helped set up the virtual environment and contributed to the written report.
- **Serina Moon:** Worked on the initial data and feature preprocessing section and model comparison section of the report. Wrote the introduction for the model classifiers. Helped with initial setup of report documents and presentation.
- **Roshni Sandhu:** Worked on the graphing and visual aids of the logistic regression model and primarily wrote the Abstract, Introduction, and Methodology of the final report. Additionally, assisted in the final editing of all the sections to ensure cohesiveness.
- **Divyansh Singh:** Created and developed the Random Forest model for Toxicity Classification. Created diagrams of confusion matrices for the model and assisted with the report and several presentation slides.
- **Maanit Shah:** Worked on the entire overleaf file. Fixed the formatting of the entire report and graphs using Latex. Helped with the initial set up of the report documents and presentation. Presentented the project in class as well.
- **Pritul Vachhani:** Made the logistic reasoning model, fine tuned it and worked on finding the optimal threshold value. Summarized the logistic reasoning model on the report. Connected the Logistic Reasoning model to Frontend and made necessary changes to the frontend so it works with the model better.
- **Aiden Xie:** Implemented and trained CNN model for multi-label toxicity classification. Optimized model performance through threshold tuning and addressing class imbalance. Summarized CNN model on report.
- **Michelle Yeoh:** Started initial outline of abstract and introduction section of report. Wrote report sections for feature preprocessing, random forest model and parts of logistic regression model. Worked on visual aspects of presentation slides.
- **Ehab Raza Zaidi:** Worked on cleaning both testing and training data to get the data ready for model training and testing.