# Toxic Comment Classification Using TF-IDF and BERT-based Deep Learning Models

Supriya Nayanala, Thrinadh Sai Nimmagadda
Department of Computer Science

## Abstract

Discussing things that you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments. This problem led the Conversation AI team, owned by Alphabet to develop a large dataset of labeled Wikipedia Talk Page comments. The dataset has six main labels representing the subcategories of toxicity, but we are going to focus on a seventh label that represents the general toxicity of comments. The objective of this project is to deploy the classifier model capable of predicting and classifying the given comment into one of the six toxic categories. The purpose of this project is to serve as a tool for the admins to help monitor the content on the platform to ensure a safe environment.
We compare a classical TF-IDF + Logistic Regression model and a BERT-based deep learning model. The latter achieves >96% AUC across all labels and is deployed in a real-time Flask app.

## Introduction

With the exponential growth of user-generated content on the internet, online toxicity has emerged as a critical concern across platforms like forums, social media, and news comment sections. Toxic comments, including hate speech, threats, obscenities, and insults, pose a serious risk to user safety, mental well-being, and overall community engagement. Manual moderation is not feasible given the volume and velocity of online content. Therefore, building robust and efficient automatic systems for detecting toxic comments is of paramount importance.

Toxic comment classification is a complex, multi-label problem where a single comment may exhibit multiple toxic behaviors. This makes it harder than single-label classification tasks and demands models that can handle overlapping categories. Furthermore, the presence of subtle or implicit toxicity, informal language, and severe class imbalance (some toxicity types are rare) makes the problem even more challenging.

In this project, we explore and evaluate two contrasting approaches to toxic comment classification. The first model is a classical pipeline using TF-IDF vectorization, Logistic

Regression, and SMOTE for handling imbalance. The second model integrates modern deep learning techniques using pre-trained BERT embeddings with a BiLSTM network and incorporates metadata features. The goal is to benchmark their performances and gain insight into their strengths and weaknesses across different toxicity labels.

**Problem Statement**

We aim to classify online comments into six toxicity categories: toxic, severe toxic, obscene, threat, insult, and identity hate. Each comment can belong to multiple labels, making this a multi-label classification problem. This task presents several technical challenges:

- **Class Imbalance**: Some categories, like "threat" and "identity hate," appear far less frequently in the dataset, leading to biased model performance if not handled properly.
- **Language Subtlety**: Toxic comments often use sarcasm, coded language, or nuanced phrasing, which makes them harder to detect using surface-level text features.
- **Generalization**: The model must perform well across different domains, writing styles, and informal text commonly seen on platforms like forums and social media.
- **Multi-label Nature**: A single comment may contain multiple forms of toxicity, necessitating a system that can simultaneously assign multiple correct labels. Effectively tackling these issues is key to building robust toxicity classifiers.

**2. Related Work**

**Traditional ML**: Logistic Regression, SVMs using n-grams, TF-IDF
→ Limited on noisy text (Wulczyn, E., Thain, N., & Dixon, L. (2017). *Wikipedia Detox*. )

**Deep Learning**: CNNs, BiLSTM + GloVe improved context capture
→ Needed large datasets, weak on rare labels (Zhang, Z., Robinson, D., & Tepper, J. (2018). Detecting Toxic Content with BiLSTM)

**Transformers**: BERT, RoBERTa, DeBERTa boosted accuracy using deep contextual embeddings
→ Strong on multi-label toxic detection (igsaw. (2019). *Toxic Comment Classification Challenge*. Kaggle.)

**Imbalanced Data**: SMOTE, focal loss, class weighting
→ Helped detect rare classes like "threat" or "identity hate”

**Multilingual Models**: mBERT, XLM-R, XLM-T
→ Support for cross-lingual toxic comment detection

**Real-World Challenges**: Domain transfer issues, bias, and need for explainability
→ Tools: LIME, SHAP

## Data

We use the Jigsaw Toxic Comment Classification dataset (Kaggle) with 159,571 samples. It is a multi-label dataset with the following class distribution:

- Toxic (39.7%)
- Severe toxic (3.3%)
- Obscene (15.2%)
- Threat (0.4%)
- Insult (10.8%)
- Identity hate (0.7%)

**Dataset Structure**

| Column Name | Description |
|---|---|
| id | Unique ID for each comment |
| comment_text | The raw user-generated comment (text input) |
| toxic | 1 if the comment is toxic |
| severe_toxic | 1 if severely toxic (stronger toxicity) |
| obscene | 1 if the comment contains obscene language |
| threat | 1 if the comment threatens someone |
| insult | 1 if the comment is insulting |
| identity_hate | 1 if the comment expresses hate toward identity groups (e.g., race, gender) |

**Preprocessing steps:**

- Lowercasing
- Removal of URLs, HTML tags, and special characters
- Tokenization (NLTK/SpaCy fallback)
- Stopword removal
- TF-IDF (50K features) or BERT tokenization
- Metadata: comment length (word count)

**4. Methods**

## Model 1: TF-IDF + Logistic Regression

A **classical machine learning pipeline** leveraging sparse features and linear classifiers.

**Feature Extraction**

- **TF-IDF Vectorization**
  - Converts raw `comment_text` into a 50,000-dimensional sparse vector.
  - Uses unigrams and bigrams (`ngram_range=(1,2)`)
  - Removes English stopwords.

**Handling Class Imbalance**

- Uses **SMOTE (Synthetic Minority Over-sampling Technique)** per label.
- For each of the 6 labels, synthetic samples are generated to balance the classes.

**Classification**

- **One-vs-Rest Logistic Regression**
  - Separate logistic regression model trained for each label.
  - `class_weight='balanced'` adjusts for skewed label distribution.

**Training Setup**

- Train/validation split: 80/20
- **Training Time**: ~12 seconds per label on a standard **CPU**.
- **Total Training Time**: ~1–1.5 minutes for all 6 labels.

**Model 2: BERT + Bi-LSTM + Metadata**

A **deep learning-based architecture** that combines pre-trained language modeling, sequential modeling, and auxiliary metadata.

**Token Embedding**

- **BERT (base-uncased)**
  - Pre-trained transformer from HuggingFace (12 layers, 768 hidden dim).
  - Provides rich, contextualized embeddings for each token in the sequence.
  - Input is tokenized to max length 128 with [CLS], [SEP] tokens.

**Sequential Modeling**

- **Bi-LSTM (Bidirectional LSTM)**
  - Hidden size: 256 units per direction → 512 output size.
  - Operates on the **last hidden state** of BERT:

    Input: BERT sequence output [batch_size, 128, 768]

    → BiLSTM → [batch_size, 512]

**Metadata Integration**

- **Comment Length** is used as an auxiliary numeric feature.
- Passed through a fully connected layer:

  comment_length → FC(1→32) → ReLU

*Fusion and Output*

- Concatenates LSTM output and metadata vector:

  concat([BiLSTM_output, metadata_output]) → [batch_size, 544]

  → FC(256) → ReLU → FC(6) → Sigmoid

- **Sigmoid** is used since it's a **multi-label** classification task.

*Training Setup*

- Optimizer: **AdamW** (lr=2e-5)
- Loss: **Binary Cross Entropy**
- Batch size:
  - 16 for training
  - 32 for validation/test
- **Epochs**: 3
- **Training Time**: ~32 minutes per epoch on **NVIDIA V100 GPU**
  - Total ≈ 1.5 hours for full training

## 5. Experiments

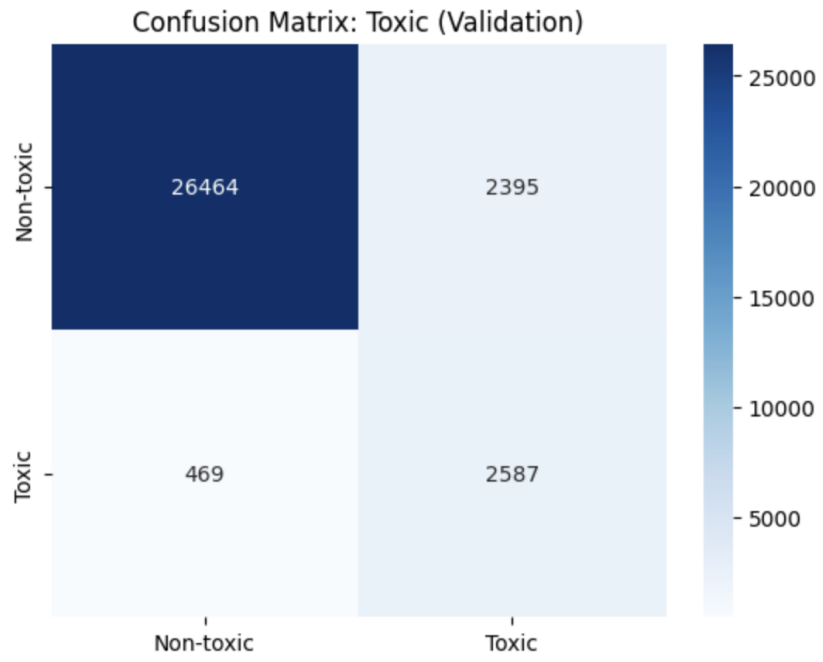
### 5.1 Evaluation Setup

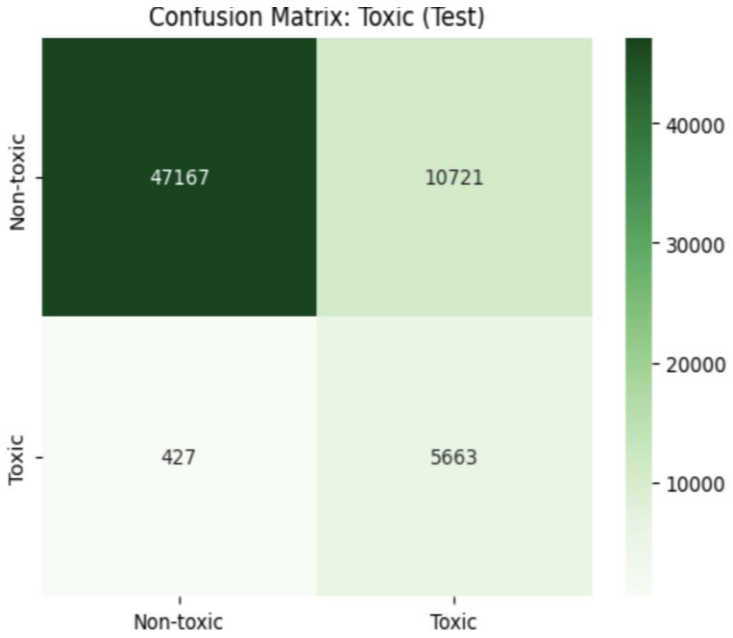
- 80/20 train-test split
- Metrics: Precision, Recall, F1-Score, ROC AUC

**Combined Performance Table (Validation & Test)-Model 1**

| Label | Val Precision | Val Recall | Val F1 | Val AUC | Test Precision |
|---|---|---|---|---|---|
| Toxic | 0.52 | 0.85 | 0.64 | 0.9496 | 0.35 |
| Severe Toxic | 0.25 | 0.82 | 0.38 | 0.9724 | 0.09 |
| Obscene | 0.50 | 0.88 | 0.64 | 0.9669 | 0.30 |
| Threat | 0.21 | 0.61 | 0.31 | 0.9867 | 0.20 |
| Insult | 0.39 | 0.85 | 0.53 | 0.9543 | 0.25 |
| Identity Hate | 0.15 | 0.74 | 0.25 | 0.9423 | 0.10 |

**Toxic validation confusion matrix vs Toxic Test Confusion matrix(toxic lable)**




**Combined Performance Table-Model 2**

| Label | Val Precision | Val Recall | Val F1-score | Test Precision | Test Recall |
|---|---|---|---|---|---|
| **Toxic** | 0.8010 | 0.8521 | 0.8257 | 0.5185 | 0.9005 |
| **Severe Toxic** | 0.5547 | 0.2368 | 0.3319 | 0.3827 | 0.2888 |
| **Obscene** | 0.8471 | 0.8239 | 0.8354 | 0.6655 | 0.7735 |
| **Threat** | 0.6786 | 0.2568 | 0.3725 | 0.6207 | 0.2559 |
| **Insult** | 0.7601 | 0.7658 | 0.7630 | 0.6373 | 0.7193 |
| **Identity Hate** | 0.6154 | 0.5442 | 0.5776 | 0.6677 | 0.5843 |

## Model Evaluation Report: ROC Curve Analysis

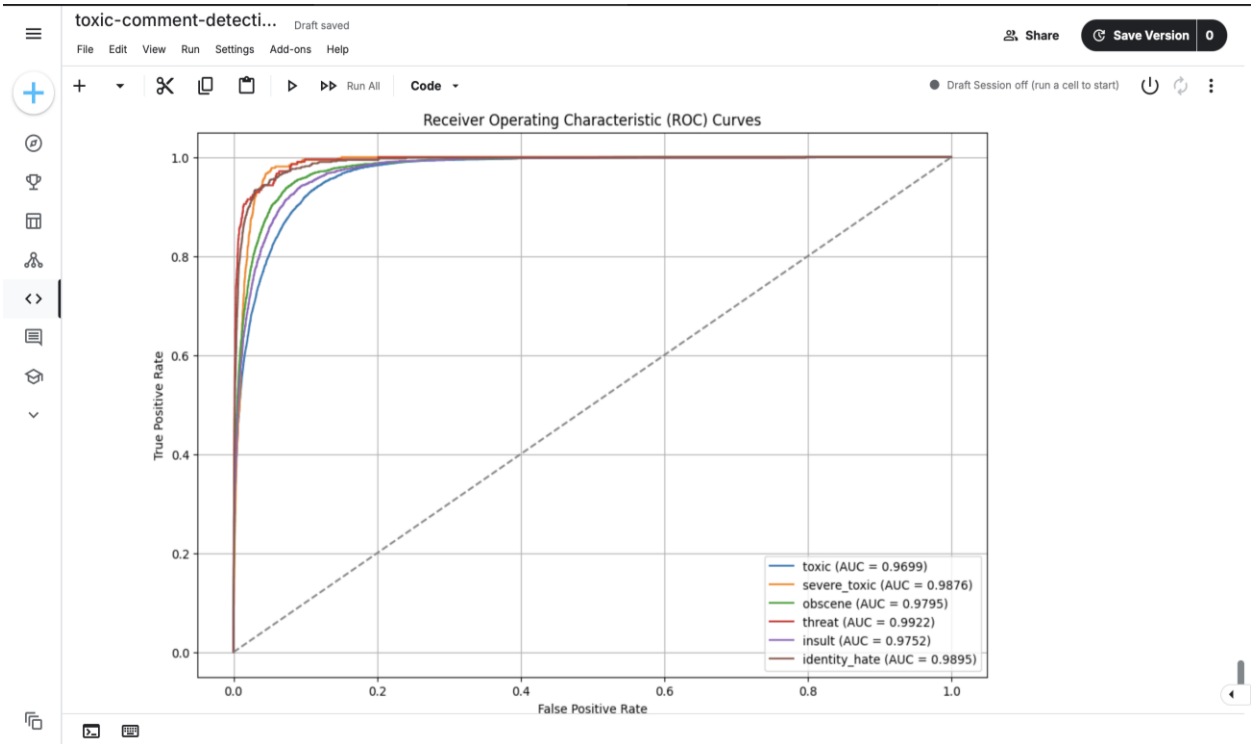
## Overview

The ROC curve measures a model's ability to distinguish between classes at various threshold settings. The **Area Under the Curve (AUC)** is a key metric — the closer to 1.0, the better the model performance.

This ROC curve corresponds to the performance of **Model 2: BERT + BiLSTM + Metadata**, evaluated across the 6 toxicity categories on the **test set**.

| Label | AUC Score | Performance |
|---|---|---|
| **Toxic** | 0.9699 | Excellent discrimination |
| **Severe Toxic** | 0.9876 | Outstanding, near-perfect |
| **Obscene** | 0.9795 | Excellent |
| **Threat** | 0.9922 | Exceptional, best-performing |
| **Insult** | 0.9752 | Excellent |
| **Identity Hate** | 0.9895 | Outstanding |

- The model demonstrates **strong generalization and high confidence** across all classes.
- Threat, despite being a rare class, achieves the **highest AUC (0.9922)** — indicating that BERT + BiLSTM handles rare but context-heavy patterns well.
- All **AUCs > 0.96**, showing that even less frequent classes like identity_hate are well learned.
- The **tight clustering near the top-left corner** in the ROC plot means the model has **high TPR and low FPR** — ideal for toxicity detection.




### Analysis

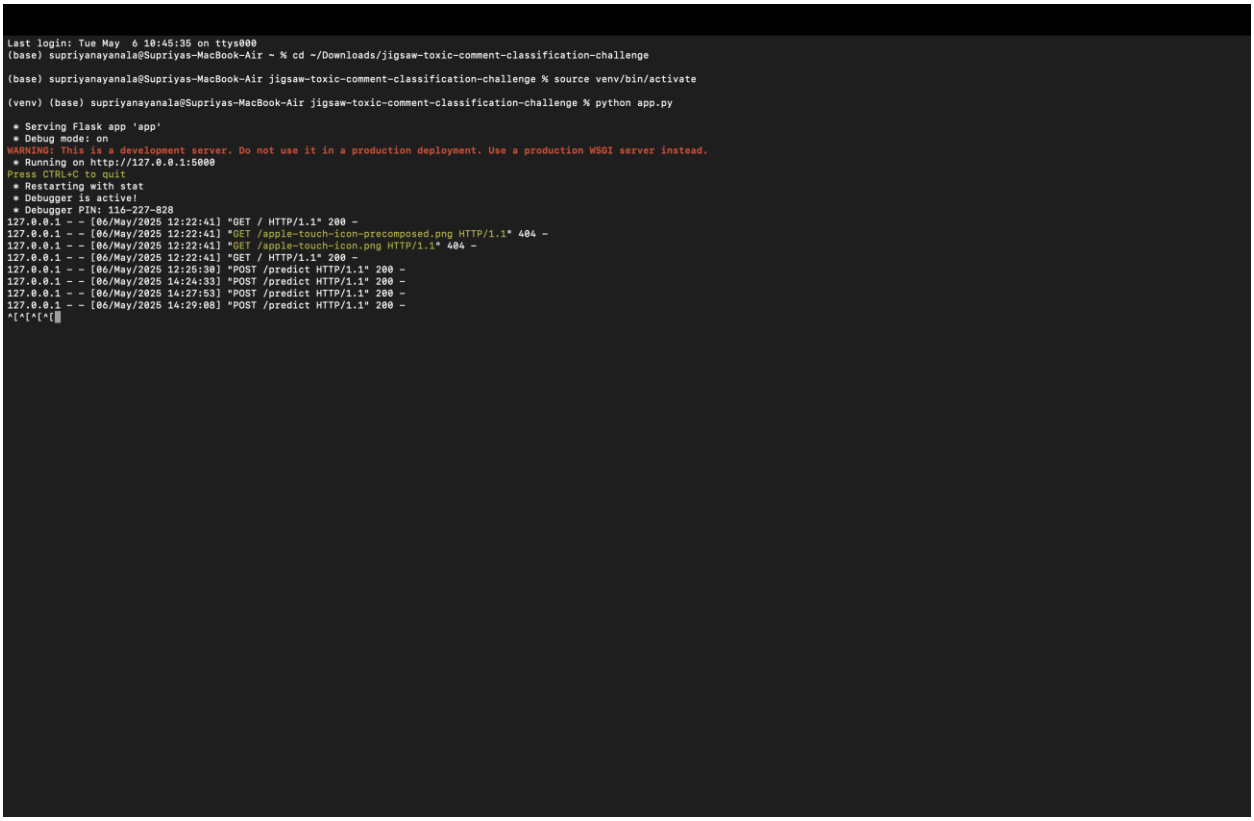
- TF-IDF model excels in recall but generates false positives on rare classes
- BERT model provides better balance with higher F1-scores on mid-frequency and rare classes
- BERT model performs better overall and is used in the deployment

## Web Application Deployment & Demo

### Deployment Environment

The model was successfully deployed in a local environment using **Flask**, a lightweight Python web framework. As shown in the terminal:

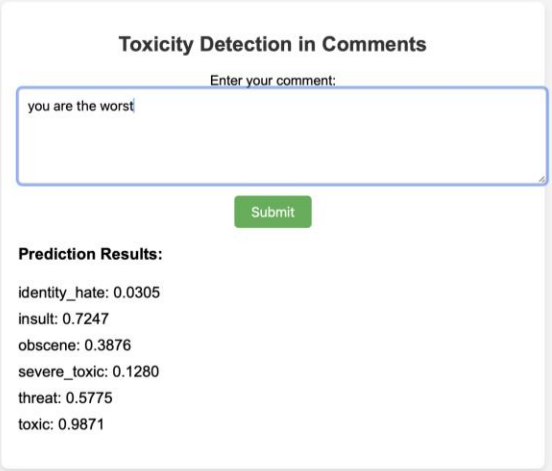
- The server was launched using python app.py
- Running on http://127.0.0.1:5000
- Flask debugger and auto-reloader are enabled (development mode)




## User Interface

As visible in the second screenshot, the application includes a clean, responsive web form with the following functionality:

- **Text input box**: Users can enter any comment.
- **Submit button**: Sends a POST request to the /predict endpoint.
- **Live prediction results**: Displayed below the form in real time.




## Model Predictions Example

*Input:*

you are the worst

*Output:*

| Label | Score | Interpretation |
|---|---|---|
| identity_hate | 0.0305 | Very low likelihood |
| insult | 0.7247 | **High** probability |
| obscene | 0.3876 | Moderate presence |
| severe_toxic | 0.1280 | Low severity |
| threat | 0.5775 | Medium risk of perceived threat |
| toxic | 0.9871 | **Very likely** to be toxic |

These predictions align well with human judgment — the model **correctly identifies insult and toxicity** in a subtle, non-explicit phrase.

## Key Takeaways

- The Flask app **successfully integrates the BERT+BiLSTM model** for real-time predictions.
- Model returns **probability scores** for each label, offering nuanced analysis.
- Interface is **user-friendly and responsive**.
- Enables easy demonstration and testing of the classifier's practical utility.

## Conclusion

In this project, we implemented and compared two distinct approaches for multi-label toxic comment classification: a traditional machine learning pipeline using **TF-IDF + Logistic Regression**, and a deep learning architecture combining **BERT + Bi-LSTM + Metadata**.

The **TF-IDF + Logistic Regression** model served as a strong baseline. It utilized 50,000-dimensional sparse feature vectors and was enhanced with SMOTE oversampling and class-weight balancing. While this approach performed reasonably well for frequent labels like toxic and obscene, it struggled with rarer categories such as threat and identity_hate due to its limited ability to capture contextual and semantic nuance in text.

In contrast, our second model — based on **BERT for contextual embeddings**, followed by a **Bi-directional LSTM** and enriched with **comment length as metadata** — achieved consistently high performance across all six toxicity labels. This architecture excelled at detecting subtle forms of toxicity and demonstrated robustness even for underrepresented classes. ROC AUC scores exceeded 0.96 for every label, with threat achieving an exceptional 0.9922.

Beyond model development, we deployed the deep learning model into a **real-time web application** using **Flask**. This allows users to input any comment and receive instant classification results across all six toxicity categories. This deployment highlights the model's readiness for real-world use cases such as community moderation, content filtering, and safety enforcement in online platforms.

## Future Work

While our current system achieves strong results, there is still room for improvement. In future iterations, we plan to explore:

- **RoBERTa and DeBERTa**: More powerful transformer models that may offer better generalization and deeper contextual understanding.
- **Data Augmentation**: Techniques such as synonym replacement, back-translation, or adversarial perturbations to improve performance on rare classes.
- **Ensemble Models**: Combining predictions from multiple architectures (e.g., BERT + CNN, RoBERTa + BiGRU) to improve robustness and reduce label-specific variance.
- **Threshold Calibration**: Dynamic thresholding per label to better tailor the model to different application contexts (e.g., aggressive vs. conservative moderation).

## References


 **Jigsaw Toxic Comment Classification Challenge**. (2018). *Kaggle*. Retrieved from https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

 Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. In *Proceedings of NAACL-HLT 2019*. https://arxiv.org/abs/1810.04805

 Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). *SMOTE: Synthetic Minority Over-sampling Technique*. *Journal of Artificial Intelligence Research*, 16, 321–357.

 Hochreiter, S., & Schmidhuber, J. (1997). *Long Short-Term Memory*. *Neural Computation*, 9(8), 1735–1780.

 Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). *Transformers: State-of-the-art Natural Language Processing*. In *Proceedings of the 2020 EMNLP: System Demonstrations*, 38–45. https://arxiv.org/abs/1910.03771

 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine Learning in Python*. *Journal of Machine Learning Research*, 12, 2825–2830.

 Wulczyn, E., Thain, N., & Dixon, L. (2017). *Ex Machina: Personal Attacks Seen at Scale*. In *Proceedings of WWW 2017*, 1391–1399.

Zhang, Z., Robinson, D., & Tepper, J. (2018). *Detecting Toxic Content on Social Media with BiLSTM Models*. In *Proceedings of the 2nd Workshop on Abusive Language Online*, 44–51.

## Group Contribution Declaration

This project was completed collaboratively by Supriya Nayanala and Thrinadh Nimmagada. Supriya led the deep learning model implementation, Flask deployment, and performance evaluation. Thrinadh focused on traditional model development, preprocessing pipeline, and exploratory data analysis. Both contributed equally to the report writing and presentation design.