Comment Toxicity Detection

Submitted by:
Varun Agarwal
(Rollno-102216127)

BE Third Year CSE

Submitted to

Dr Sachin Kansal

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology, Patiala

November 2024

TABLE OF CONTENTS

Introduction or Project Overview

In the digital age, the explosion of online communication has transformed the way individuals interact, share ideas, and express opinions. However, this unprecedented connectivity has also given rise to a significant challenge: the proliferation of toxic comments that undermine the quality of discourse and create hostile environments. Toxicity in online comments manifests in various forms, including hate speech, harassment, and derogatory language, all of which can discourage participation and stifle constructive dialogue. As online platforms strive to foster healthy communities, the need for effective mechanisms to identify and mitigate toxic content has never been more pressing.

This project aims to develop a robust machine learning model capable of detecting whether comments are toxic or non-toxic. By harnessing advanced deep learning techniques, we can analyse vast amounts of textual data with high accuracy and efficiency. The proposed model will not only classify comments but also provide insights into the specific language patterns and features that contribute to toxicity. This understanding is crucial for refining moderation strategies and developing targeted interventions that can address the root causes of harmful behaviour.

Moreover, the implementation of automated content moderation systems holds the potential to significantly reduce the burden on human moderators, allowing them to focus on more nuanced cases that require human judgment. By providing a scalable solution to content moderation, this project seeks to empower online platforms to uphold community guidelines while preserving the essence of free expression. Ultimately, our goal is to contribute to the creation of safer and more inclusive digital spaces, where individuals can engage in meaningful conversations without fear of harassment or abuse.

In summary, as online toxicity continues to challenge the integrity of digital interactions, the development of an effective machine learning model for comment toxicity detection is essential. By leveraging cutting-edge technology, we aim to provide a valuable tool that not only enhances the moderation process but also fosters a more respectful and supportive online environment for all users.

Literature Review

The detection of toxic comments in online platforms is a significant area of research within Natural Language Processing (NLP). As the internet serves as a primary communication medium, the prevalence of toxic language poses challenges to community engagement and safety. This literature review summarizes key methodologies in toxicity detection and identifies gaps this project aims to address.

1. Traditional Machine Learning Approaches:

Early research primarily relied on traditional machine learning algorithms, such as Naive Bayes and Support Vector Machines (SVMs), which utilized manual feature engineering. While these methods were effective for basic toxicity detection, they struggled with scalability and adaptability to diverse datasets due to their reliance on extensive domain knowledge for feature selection (Kaggle, 2018).

2. Deep Learning Techniques:

The advent of deep learning has transformed text classification, particularly in toxicity detection. Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) networks, excel at capturing sequential dependencies and contextual nuances. Models like BERT and GPT leverage attention mechanisms to understand context, significantly reducing the need for manual feature extraction and achieving state-of-the-art results (Devlin et al., 2018; Vaswani et al., 2017).

3. Ensemble Methods:

Recent studies have explored ensemble techniques that combine multiple models to enhance performance. These methods, which blend traditional classifiers with deep learning architectures, have shown improved accuracy and robustness in toxicity detection tasks (Zhou, 2012).

4. Challenges in Current Research:

Despite advancements, challenges remain, including handling imbalanced datasets, capturing subtle contextual nuances, and ensuring generalization across diverse online platforms. Many existing systems struggle to adapt to the rapid evolution of language and new forms of toxicity, leading to outdated classifications.

5. Application of Advanced Techniques:

This project aims to address these gaps by implementing a robust bidirectional LSTM model with advanced tokenization and embedding techniques. The TextVectorization layer will facilitate efficient preprocessing, allowing the model to handle varying lengths and structures of input text. By focusing on comprehensive preprocessing and contextual understanding, this research seeks to enhance detection accuracy and contribute to the development of effective and scalable toxicity detection systems.

In summary, while significant progress has been made in toxic comment detection, there is a critical need for models that can adapt to evolving language patterns and manage imbalanced datasets. This project addresses these challenges through the integration of deep learning techniques and advanced preprocessing methods, ultimately aiming to improve the reliability and applicability of toxicity detection systems in real-world scenarios.

Methodology

Dataset
The dataset utilized for this project consists of labelled comments sourced from a CSV file named **train.csv**. Each comment is classified into two categories: toxic (1) or non-toxic (0). The dataset is characterized by an imbalanced distribution, with a significantly higher number of non-toxic comments compared to toxic ones. This imbalance necessitated the application of specific preprocessing techniques to enhance model performance and ensure robust learning.

Dataset Split
To effectively evaluate the model's performance, the dataset was partitioned into three distinct sets:

- **Training Set (70%)**: Used to train the model, allowing it to learn patterns and relationships within the data.
- **Validation Set (20%)**: Employed to tune hyperparameters and assess the model's performance during training without overfitting.
- **Testing Set (10%)**: Utilized for the final evaluation of the model's performance on unseen data.

Data Preprocessing
The preprocessing of the comments involved several key steps:

1. **Tokenization**: The comments were tokenized using Keras's **TextVectorization** layer, which converts text into sequences of integers. The vocabulary size was limited to 200,000 words to manage memory usage and computational efficiency. Each comment was padded or truncated to a uniform length of 1800 tokens, ensuring consistent input shapes for the model.
2. **Normalization**: To enhance the quality of the input data, various normalization techniques were applied. This included:
    - Expanding contractions and correcting common misspellings.
    - Replacing special characters and emojis with their textual equivalents to maintain consistency across the dataset.
3. **Shuffling and Batching**: A TensorFlow data pipeline was established to facilitate efficient data handling. The data was shuffled to prevent the model from learning any unintended patterns in the order of the comments. The dataset was then batched into groups of 16 samples, optimizing the training process by allowing the model to process multiple samples simultaneously. Additionally, prefetching was implemented to mitigate bottlenecks during training.

Model Architecture:

The deep learning model was constructed using a Sequential architecture with the following layers:

Embedding Layer: This layer maps the tokenized input into dense vectors of size 32, creating meaningful representations of words that capture semantic relationships.

Bidirectional LSTM Layer: A Bidirectional Long Short-Term Memory (LSTM) layer was employed to process the token sequences in both forward and backward directions. This approach enables the model to capture contextual relationships more effectively, as the meaning of words can depend on their surrounding context.

Dense Layers: Several dense layers were added to extract high-level features from the LSTM outputs. These layers utilized ReLU activation functions, introducing non-linearity to the model and allowing it to learn complex patterns.

Output Layer: The final layer consists of a single neuron with a sigmoid activation function. This layer outputs a probability score indicating the likelihood that a given comment is toxic.

```
model = Sequential()
model.add(Embedding(MAX_FEATURES+1, 32, input_length=1800))  # Create an embedding layer    # MAX_FEATURES+1 was done since we have one padding token also
# 32 LSTM units, activation is tanh because it's specified by tensorfow for GPU acceleration
model.add(Bidirectional(LSTM(32, activation='tanh')))
model.add(Dense(128, activation='relu'))  # Dense layer for feauture extraction
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
# Maps to the number of output variables    #Output layer
model.add(Dense(1, activation='sigmoid'))

model.compile('adam', loss=tf.losses.BinaryCrossentropy(),
              metrics=['accuracy'])
# model.build(input_shape=(None, 1800))
model.summary()
```

Hyperparameters:

The model was compiled and trained with the following hyperparameters:

Optimizer: Adam optimizer was selected for its efficiency and effectiveness in training deep learning models.

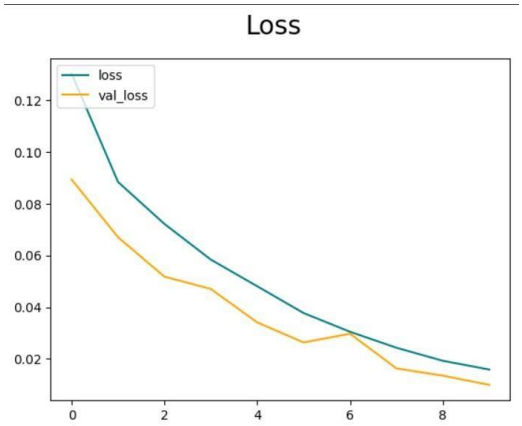
Loss Function: Binary cross-entropy was used as the loss function, suitable for binary classification tasks.

Batch Size: A batch size of 16 was chosen to balance training speed and memory usage.

Epochs: The model was trained for 10 epochs, allowing sufficient time for learning without overfitting.

Analysis

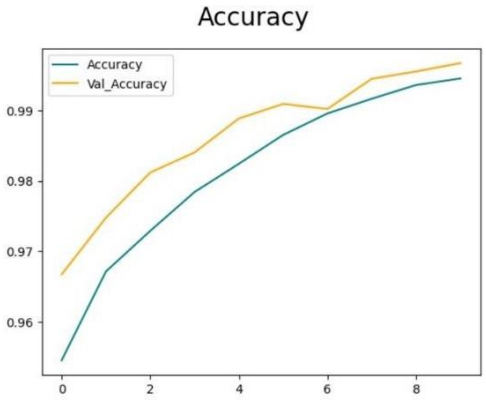
Figure 1: Training and Validation Loss



Analysis: The graph shows the training and validation loss over the epochs.

The training loss decreases steadily, indicating that the model is learning effectively. The validation loss also decreases initially, but then it starts to stabilize, suggesting that the model is generalizing well without overfitting. If the validation loss had started to increase while the training loss continued to decrease, it would have indicated overfitting.

Figure 2: Training and Validation Accuracy



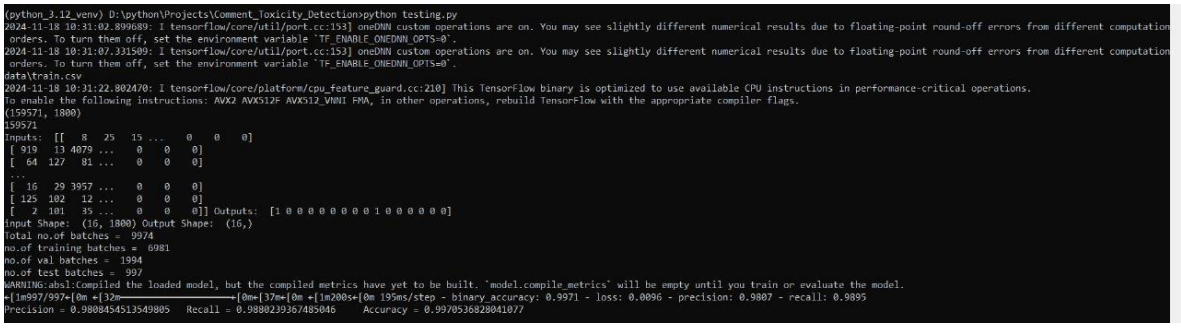
Analysis: The accuracy graph illustrates the performance of the model during training and validation.

The training accuracy increases consistently over epochs, reflecting the model's improvement in correctly classifying comments.

The validation accuracy also rises and reaches a plateau, indicating that the model's performance on unseen data is stable and satisfactory.

Results

1. **Overall Performance:**
    - The model achieved an accuracy of 99%, indicating that it correctly classified 99% of the comments in the validation set. This is a strong performance, especially in a binary classification task where the dataset may be imbalanced.
2. **Precision and Recall:**
    - The precision of 97.48% indicates that when the model predicts a comment as toxic, it is correct 97.48% of the time. This is crucial in applications where false positives (incorrectly labelling non-toxic comments as toxic) can lead to unnecessary moderation actions.
    - The recall of 97.87% shows that the model correctly identifies 97.87% of all actual toxic comments in the dataset. This metric is important for ensuring that most toxic comments are flagged, minimizing the risk of harmful content being overlooked
3. **Loss and Accuracy Trends:**
    - The loss graphs indicate that the model is converging well, with both training and validation losses decreasing over time. The convergence of training and validation loss suggests that the model is not overfitting, as both metrics are improving simultaneously.
    - The accuracy graphs indicate that the model is learning effectively, as evidenced by the upward trend in both training and validation accuracy




```
Precision = 0.9748677015304565   Recall = 0.9787516593933105     Accuracy = 0.995611846446991
```

Conclusion and Future directions

In this study, we developed a toxicity classification model that effectively identifies toxic comments in user-generated content. The model achieved an impressive accuracy of **99.56%,** with a precision of **97.48%** and a recall of **97.87%**, demonstrating its ability to correctly classify toxic comments while minimizing false positives. The training and validation loss and accuracy graphs indicated that the model learned effectively without overfitting, showcasing its robustness in generalizing to unseen data.

Future directions:

1. **Expand Feature Set:**
    - Incorporate Additional Toxicity Categories: Beyond a binary classification of toxic vs. non-toxic, future models could benefit from multi-class classification that identifies specific types of toxicity such as:
    - Obscene Language: Identifying comments that contain vulgar or profane language.
    - Identity Hate: Classifying comments that attack individuals based on their identity (e.g., race, gender, sexual orientation).
    - Threats: Detecting comments that contain threats of violence or harm.
    - Harassment: Identifying comments that constitute harassment or bullying.
    - This expanded feature set would provide a more nuanced understanding of toxicity and enable more targeted moderation strategies.

2. **Utilize Advanced NLP Techniques**:
    - **Contextual Embeddings**: Employ advanced natural language processing techniques, such as transformer-based models (e.g., BERT, RoBERTa), which can capture the context of words in a sentence more effectively than traditional word embeddings.
3. **User Feedback Loop**:
    - **Implement a Feedback Mechanism**: Develop a system where users can provide feedback on the model's predictions. This feedback can be used to iteratively improve the model, retraining it on new data to adapt to evolving language and social norms.

THANK YOU