Multi-Label Toxic Comment Classification for Improved Online Conversations

Rohit Anilkumar, 3, ranilkum@syr.edu

Deepika Nandan, 42, dnandan@syr.edu

Introduction

Effective online communication fosters diverse viewpoints and constructive discussions. However, the pervasive issue of abusive and toxic comments hampers this potential, leading to the stifling of voices and a reluctance to engage in meaningful conversations. The project endeavors to enhance online dialogues by addressing negative behaviors, particularly toxic comments. The efforts include building machine learning models to identify toxicity. Yet, existing models have limitations, often making errors and lacking granularity in distinguishing various types of toxicity.

This project tackles the challenge of multi-label toxic comment classification, aiming to create a sophisticated model capable of detecting different forms of toxicity, such as toxic, severe toxic, threats, obscenity, insults, and identity-based hate. Leveraging a dataset from Wikipedia's talk page edits.

Problem Statement

The task at hand involves developing a multi-label classification model capable of discerning and labeling various categories of toxicity present in online comments. Multi-label text classification is the task of assigning multiple labels or categories to a single piece of text, accommodating scenarios where the text belongs to more than one category simultaneously.

While current models may flag general toxic content, they often lack the nuance required to differentiate between different types of toxic behavior. The objective here is to construct an advanced model that not only detects toxicity but also identifies and

categorizes specific types of toxic content within comments, thereby enabling more targeted moderation on online platforms.

Methodology

Exploratory Data Analysis (EDA)

The project commences with an exploratory analysis of the dataset. This phase includes visualizations such as bar plots showcasing the distribution of toxic comments across different categories, insights into the number of tags per comment, and correlation heat maps to identify relationships between various toxicity types.

The **EDAVisualizer** class encapsulates a collection of methods designed to facilitate Exploratory Data Analysis (EDA) on a given dataset. In the realm of data science and machine learning, EDA is an essential preliminary step for understanding the underlying patterns, distributions, and relationships within the data before diving into modeling.

This versatile class offers a suite of functions tailored to visualize various aspects of the dataset:

1. **Visualizing Feature Counts:** This method allows for a quick visual representation of feature counts within the dataset, aiding in understanding the distribution and occurrence of individual features.

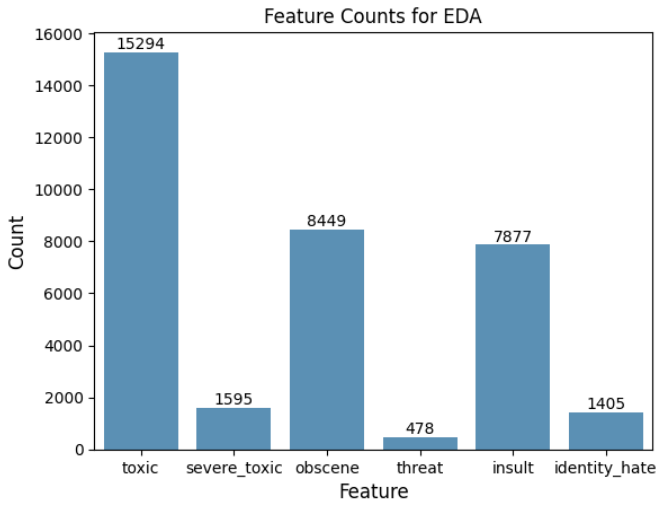

Fig 1.a: Feature counts

Functionality Description:

1. Calculates the count of each feature by summing occurrences across the dataset.
2. Utilizes seaborn's barplot to create a bar chart displaying feature counts.
3. Adds text labels on each bar for clear count representation.

Data Description: This visualization highlights the occurrence or frequency of individual features within the dataset. Each bar represents a feature, and its height signifies the count of occurrences of that feature.

Interpretation and Inferences: The below graph provides insights into the prevalence of features. It depicts the count of data entries per Feature. We can also get information on Patterns of feature occurrence and disparities in feature counts become apparent, aiding in identifying influential or less significant features.

Conclusion and Importance: This visualization method serves as a crucial step in EDA, facilitating the identification of dominant features like “Toxic” comments here and their distributions, guiding subsequent data processing and modeling decisions. Understanding feature frequencies aids in selecting relevant features for modeling and analysis.

2. **Visualizing Tags per Comment:** Understanding the count of tags per comment is crucial in certain contexts. This function enables a clear visualization of the frequency of tags in the dataset, contributing to insightful exploratory analysis.

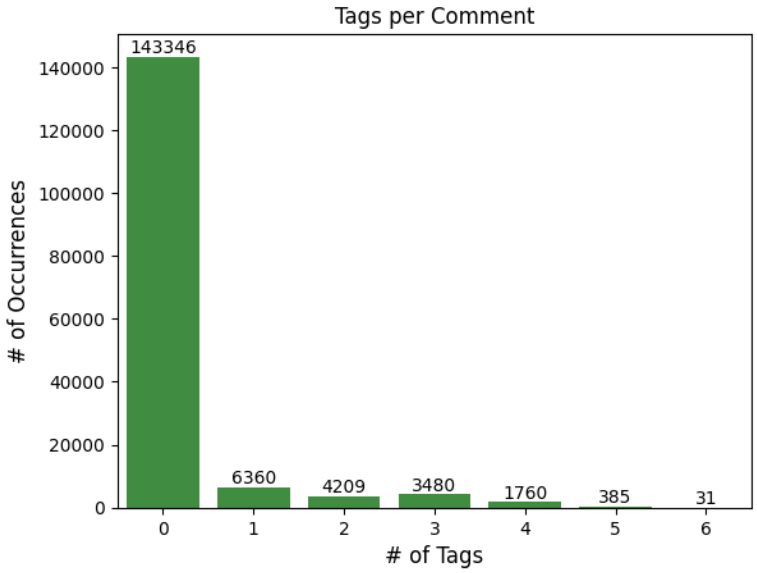

Fig 1.b: Tags per comment

Functionality Description:

1. Computes the count of tags per comment by summing row-wise.
2. Creates a bar plot using seaborn's **barplot** function to display tag counts.
3. Labels bars with counts for clarity.

Data Description: This visualization highlights the distribution of tag counts per comment in the dataset. Each bar represents the count of tags per comment.

Interpretation and Inferences: Insights into the frequency of tags in comments can be derived. The visualization helps identify common tag occurrences and the distribution of tag counts across comments.

Conclusion and Importance: This method aids in understanding the prevalence and distribution of tags within comments. It assists in identifying frequently occurring tag combinations and their potential impact on comment classification or analysis. As in this case we can infer that around 143,346 comments have no tags and are categorized as clean comments.

3. **Correlation Heatmap Visualization:** Analyzing correlations between different features is vital for feature selection or understanding the interdependence between variables. This method generates a heatmap illustrating the correlations among features within the dataset.

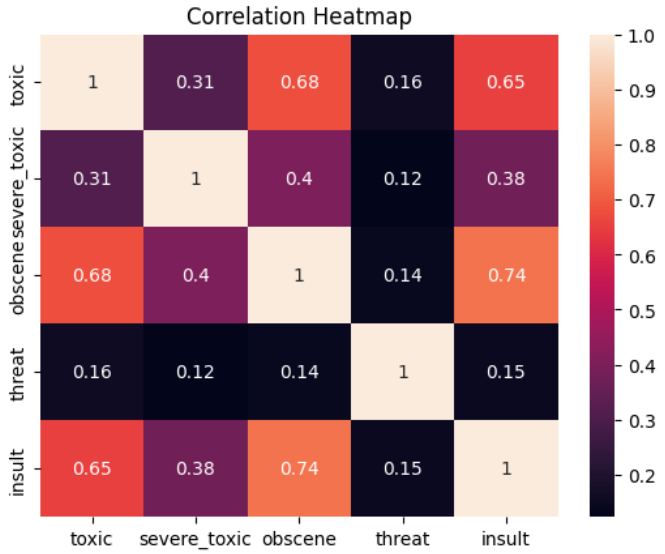

Fig 1.c: Correlation heatmap

Purpose: Visualizes the correlation heatmap for features in the DataFrame.

Functionality Description:

1. Creates a copy of the DataFrame to avoid modifying the original data.
2. Utilizes seaborn's **heatmap** to generate the correlation heatmap.

Data Description: The heatmap visualizes the correlation coefficients between different features, represented by a color scale.

Interpretation and Inferences: This heatmap reveals the degree and direction of linear relationships between features. High correlation values indicate strong relationships, aiding in feature selection and multicollinearity assessment.

Conclusion and Importance: Understanding feature correlations assists in identifying potentially redundant or highly influential features. This visualization aids in feature selection and model building by considering correlated features' impact on predictions.

4. **Checking Overlaps:** The function replaces categorical data values, aiding in visualizing shared occurrences or differences across columns. It enables a clear examination of overlaps within the dataset's categories, enhancing the understanding of intersecting elements and unique features.

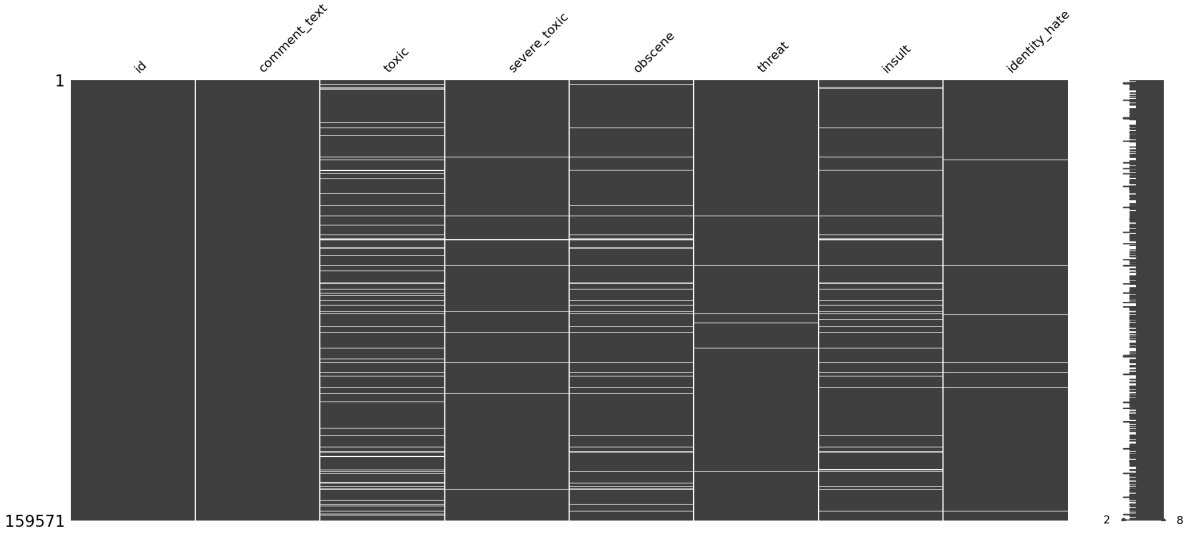

Fig 1.d: Overlap check using msno

Functionality Description:

1. Replaces '1's with NaN in the DataFrame.
2. Utilizes missingno's **matrix** function to visualize overlaps among columns.

Data Description: The visualization shows overlaps among different columns by indicating missing or overlapping data patterns.

Interpretation and Inferences: The below matrix representation of our data distribution indicates the if there are any empty values pertaining to that column. The grey indicates the values present and white indicates missing values. As we can infer, this dataset is relatively having low number of missing values. Patterns in missing or overlapping data become evident from such a graph, indicating shared occurrences or distinct data regions among columns.

Conclusion and Importance: This visualization helps identify missing values or discrepancies among columns, guiding data cleaning and identifying relationships between different categorical or binary features.

5. **Creating Venn Diagrams:** This method generates visual representations of overlaps among designated columns. These Venn diagrams offer insights into the relationships and commonalities between categorical data, highlighting intersections and distinct categorical occurrences within the dataset.

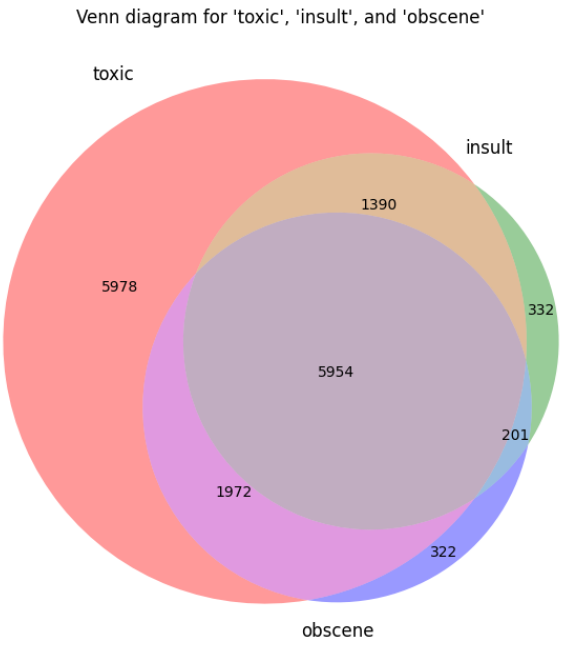

Fig 1.e: Venn diagram for checking overlap

Functionality Description:

1. Computes counts for various combinations of 'toxic', 'insult', and 'obscene' columns.
2. Utilizes matplotlib's **venn3** function to create the Venn diagram.

Data Description: The Venn diagram visually represents intersections and unique occurrences among the specified columns.

Interpretation and Inferences: Insights into shared occurrences and unique instances among 'toxic', 'insult', and 'obscene' columns become apparent through the diagram's overlapping sections. Here we can observe the overlap between how many comments can be categorized as toxic + insult, insult + obscene, toxic + obscene or all three.

Conclusion and Importance: This visualization method offers a clear representation of overlaps among specific categories, aiding in understanding relationships and shared occurrences among different categorical columns.

6. **Generating Word Clouds:** Using specific columns as input, this function creates word clouds, displaying frequently used words within the specified dataset subset, assisting in understanding textual patterns.

Fig 1.f: Word cloud for identity hate

Functionality Description:

1. Filters the DataFrame based on the specified column.
2. Generates a word cloud using the WordCloud library.
3. Displays the word cloud using matplotlib.

Data Description: The word cloud displays frequently occurring words from the specified column, highlighting their importance or prevalence.

Interpretation and Inferences: The word cloud visually represents frequently used words, providing insights into the most common words within the specified column.

Conclusion and Importance: This visualization helps identify prevalent terms or words within a specific column, aiding in textual data analysis and understanding prominent words within a corpus.

Data Preprocessing

The preprocessing phase involves preparing the textual data for model training. This includes steps such as text cleaning, normalization, lemmatization, and tokenization, ensuring that the comments are appropriately formatted and devoid of noise for effective model training. GloVe word embeddings were employed for enhancing the model's understanding of text semantics.

Multi-Label Neural Network Architecture

The crux of the project lies in the design of a multi-label neural network architecture. This architecture consists of parallel channels, each responsible for processing comments to identify a specific type of toxicity. Leveraging Convolutional Neural Networks (CNNs) and Gated Recurrent Units (GRUs), the model extracts meaningful features from comments. These features are then merged and fed into fully connected layers for multi-label toxicity classification.

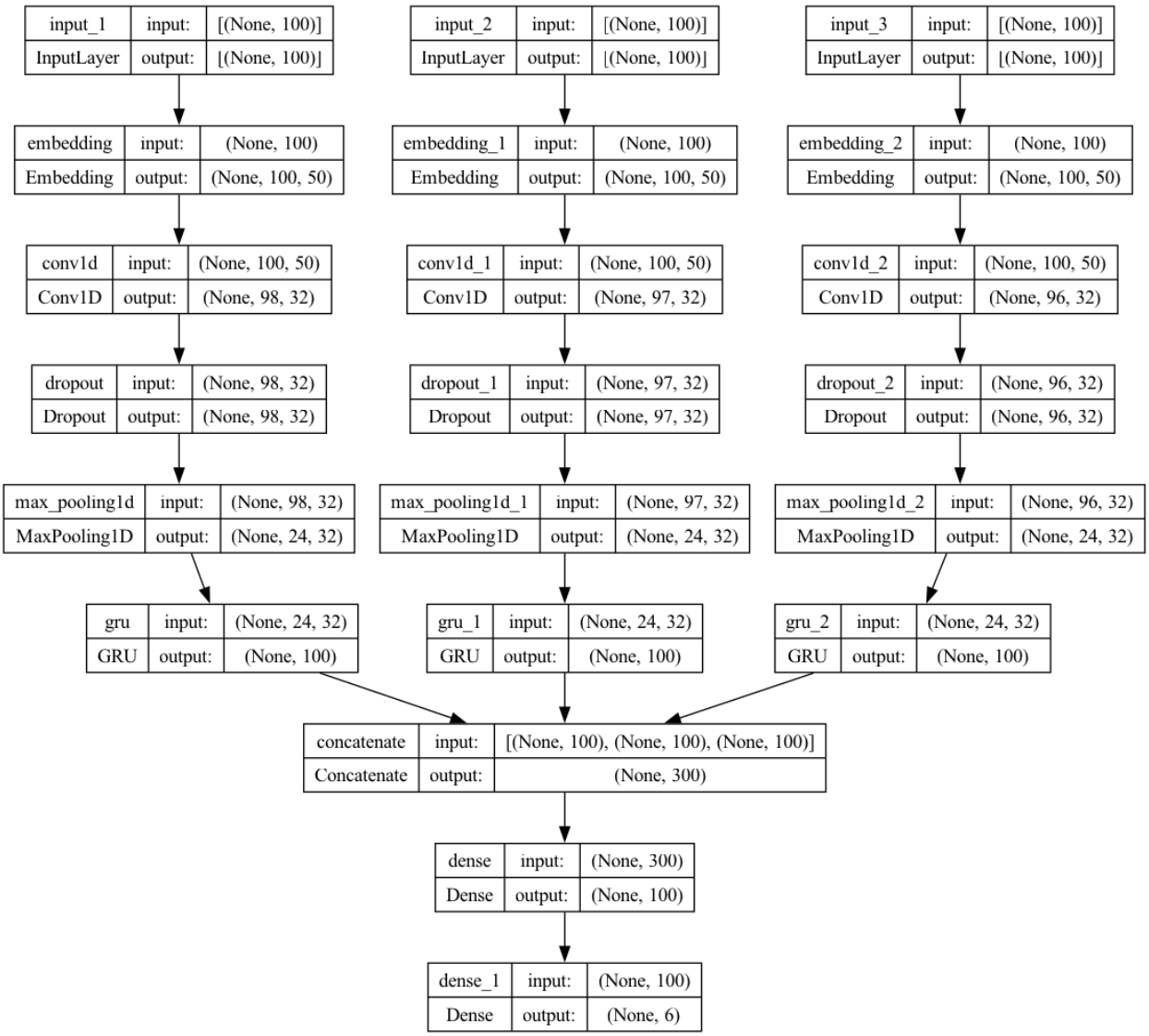

Fig 2 . Architecture of the proposed multi headed CNN + GRU neural network generated using plot_model method.

A Multi-Channel Convolutional Neural Network (CNN) coupled with Gated Recurrent Units (GRUs) presents a promising architecture for improved multi-label toxicity classification in online comments due to several key advantages:

1. **Capturing Local and Sequential Patterns:** CNNs excel at learning local patterns within data through filters of varying sizes. By employing multiple channels, each with different filter sizes, the model can capture diverse patterns in the text, enabling it to discern nuanced features associated with different types of toxicity.

2. **Hierarchical Feature Extraction:** The combination of CNNs and GRUs offers a hierarchical approach to feature extraction. CNNs initially identify low-level features like word sequences and combinations, while GRUs, with their recurrent nature, capture contextual information and longer-term dependencies within the text, allowing the model to comprehend the sequential nature of language, which is crucial in understanding the context of toxic comments.
3. **Parameter Sharing and Dimensionality Reduction:** CNNs employ parameter sharing, which reduces the model's complexity by reusing weights across different parts of the input, enhancing efficiency and reducing overfitting. Additionally, pooling layers in CNNs help in dimensionality reduction, focusing on the most relevant features extracted by the filters, aiding in better representation of the data.
4. **Flexibility in Learning Representations:** GRUs possess the ability to adaptively learn and update representations of the text as it processes sequential data. This adaptability allows the model to understand complex relationships between words and phrases, making it adept at capturing semantic nuances associated with different types of toxicity.
5. **Parallel Processing:** The multi-channel architecture processes the input data in parallel through different channels. This enables the model to simultaneously learn from diverse perspectives and representations of the text, enhancing its ability to extract meaningful features from comments representing various types of toxicity.
6. **Ensemble Learning Benefits:** The fusion of CNNs and GRUs operates as an ensemble, combining the strengths of both architectures. This amalgamation not only provides a more comprehensive understanding of the data but also mitigates the weaknesses inherent in individual architectures, resulting in a more robust and accurate model for multi-label toxicity classification.

Model Training and Evaluation

The model undergoes training using a portion of the dataset while being validated on another segment to gauge its performance. Key evaluation metrics, including accuracy, loss, and ROC-AUC scores, are tracked to assess the model's effectiveness in multi-label toxicity classification. The model has given a training accuracy of 97%, validation accuracy of 97% and ROC-AUC score of 0.98. The model, initially set for 25 epochs, stopped at the 18th due to early stopping triggered by minimal improvement in validation loss, ensuring optimal generalization and preventing overfitting. Since we

have set *restore_best_weights to True,* weights from the epoch with the best value of the monitored quantity will be used.

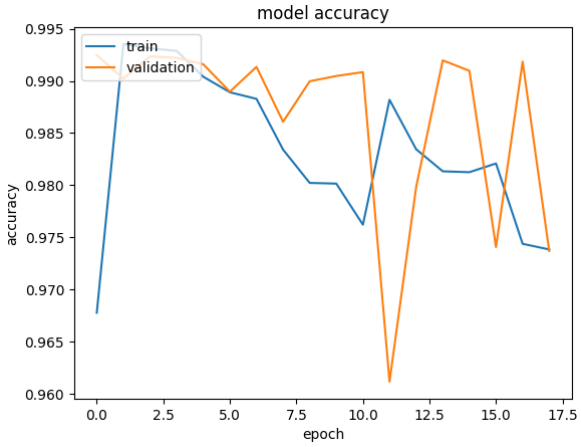

Fig 3.a Model Accuracy

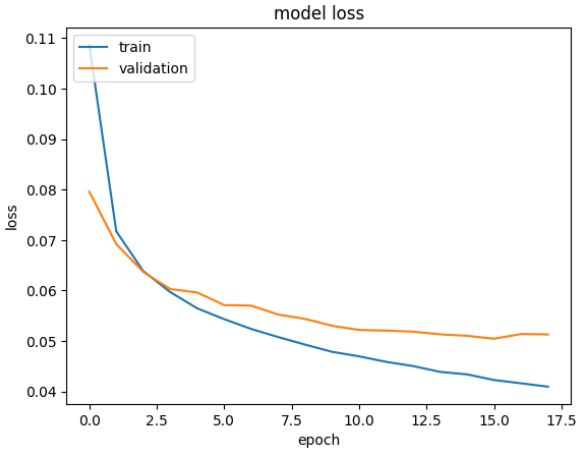

Fig 3.b Model loss

Results and Discussion

Post-training, the model's performance metrics such as accuracy, loss, and ROC-AUC scores are analyzed. Visualizations of training and validation metrics offer insights into the model's learning process. Additionally, predictions are made on the test dataset to evaluate the model's proficiency in accurately classifying multiple types of toxicity within comments.

The model was able to accurately identify non problematic comments by giving the specific comments a toxicity value of less than 0.5, which is arbitrarily set as the threshold.

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is, IMO.	0.001885	0.000034	0.000972	0.000023	0.000334	0.000029
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lapland â€ / "	0.005989	0.000349	0.003211	0.000301	0.002411	0.000497
3	00017563c3f7919a	:If you have a look back at the source, the information I updated was the correct form. I can only guess the source hadn't updated. I shall update the information once again but thank you for your message.	0.000337	0.000003	0.000143	0.000014	0.000048	0.000004
4	00017695ad8997eb	I don't anonymously edit articles at all.	0.003661	0.000091	0.001512	0.000066	0.000784	0.000175
5	0001ea8717f6de06	Thank you for understanding. I think very highly of you and would not revert without discussion.	0.001275	0.000011	0.000307	0.000034	0.000247	0.000048

```
HTML(test_df.where((test_df['toxic']<0.5) & (test_df['severe_tox
                   (test_df['obscene']<0.5) & (test_df['threat']<0.5
```

```
                (test_df['insult']<0.5) & (test_df['identity_hate
               ).dropna(subset=['comment_text'], how='any').head(
```

The model was able to accurately identify comments having derogatory terms and assign toxicity labels accordingly.

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll ever be whats up with you and hating you sad mofuckas...i should bitch slap ur pethedic white faces and get you to kiss my ass you guys sicken me. Ja rule is about pride in da music man. dont diss that shit on him. and nothin is wrong bein like tupac he was a brother too...fuckin white boys get things right next time.,	0.986524	0.301072	0.818010	0.069429	0.860365	0.530812
7	000247e83dcc1211	:Dear god this site is horrible.	0.500445	0.009313	0.104390	0.015007	0.135108	0.025006
48	0013fed3aeae76b7	DJ Robinson is gay as hell! he sucks his dick so much!!!!!	0.987785	0.237324	0.798933	0.026147	0.797629	0.503441
50	001421530a1aa622	I have been perfectly civil in what quite clearly idiotic trolling.	0.621290	0.001688	0.106845	0.001358	0.141253	0.010237
56	0016b94c8b20ffa6	I WILL BURN YOU TO HELL IF YOU REVOKE MY TALK PAGE ACCESS!!!!!!!!!!!!	0.866619	0.036408	0.282741	0.181700	0.271728	0.040154

```
HTML(test_df.where((test_df['toxic']>=0.5) | (test_df['severe_t
               (test_df['obscene']>=0.5) | (test_df['threat']>=0
               (test_df['insult']>=0.5) | (test_df['identity_hate
              ).dropna(subset=['comment_text'], how='any').head(
```

Conclusion and Future Work

The developed multi-label neural network demonstrates notable improvements in accurately identifying and categorizing various types of toxicity present in online comments. This advancement holds promise in enabling platforms to implement more nuanced and targeted moderation strategies, fostering healthier and more inclusive online discussions.

Future iterations of this project could delve deeper into refining the model architecture using hyper parameter tuning, exploring advanced preprocessing techniques, use contextual embeddings such as BERT, GPT etc and incorporating user feedback mechanisms to enhance the model's accuracy and usability in real-world online platforms.

References

1. Keras documentation [https://keras.io/]

2. Machine Learning Mastery blog [https://machinelearningmastery.com/]
3. Dataset: [https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data]

This project report encapsulates the development and implementation of a multi-label neural network for toxicity classification, aiming to facilitate respectful and productive online conversations by identifying and categorizing diverse forms of toxic behavior within comments.

Work

Exploratory Data Analysis → Deepika Nandan

Neural Network and analysis → Rohit Anilkumar