



# **Dhirubhai Ambani University**

Summer Research Internship

## **Hate Speech**

**Prof. Purbasha Das**

**Member 1 : Denil Antala**

**Student ID : 202201090**

**Member 2 : Aaryan Sinh Shaurya**

**Student ID : 202201075**

**Member 3 : Jami Sidhava**

**Student ID : 202201038**

# Hate Speech Detector

As per our outline

## 1) Setting Up Our Toolkit

We started by importing the essential libraries - our tools for every stage of the pipeline. These libraries formed the backbone of our entire project pipeline, from reading raw data to getting our final predictions.

- **Data Handling:**  
We used `pandas` and `numpy` for loading, processing to structured data efficiently.
- **Visualization:**  
`matplotlib.pyplot` and `seaborn`, helped us visualize label distributions, word frequencies.
- **Utilities & File Handling:**  
Libraries `os`, `re`, `urllib`, `zipfile`, and `pickle/joblib` supported tasks such as file operations, downloading data, and saving/loading models.
- **Models**  
`sklearn`, `tensorflow.keras`, and `transformers` were essential for actually building and running our AI models.
- **Pre-Processing**  
The `nltk` library was used for text preprocessing, to clean text, removing stop words and etc.

## 2) Data Preparation

- Dataset is taken from Kaggle : "[Hate Speech Detection Curated Dataset.](#)"
- First, we saw the dataset had text in many languages. Since we aren't building a tool for all languages, we only kept the English ones.

- Then, we found a big problem: some messages were wrongly labeled. For example, a really hateful comment might be marked as Not Hate “0(zero)” or vice-versa. To fix this, we used a pre-trained transformer called [toxic-bert](#). This tool helped us re-label the messages pretty accurately by understanding what they really meant. This made our data much more reliable to learn from and use it further.
- So the updated dataset is saved as [finalhateull.csv](#)

### 3) EDA and Pre-Processing

We peek into data, and try to get more idea about the spread of data for better understanding.

#### Pre-Processing

- Made everything lowercase: So "Hate" and "hate" were seen as the same word.
- Removing website links, emojis, HTML codes, special characters.
- Removing Punctuation, to focus on text.
- Removing stop words like "the," "is," "at" don't carry much meaning for hate detection.
- Tokenization : serves as the basis for text segmentation, Lemmatization aids in standardizing words and improving language understanding and analysis.

How?

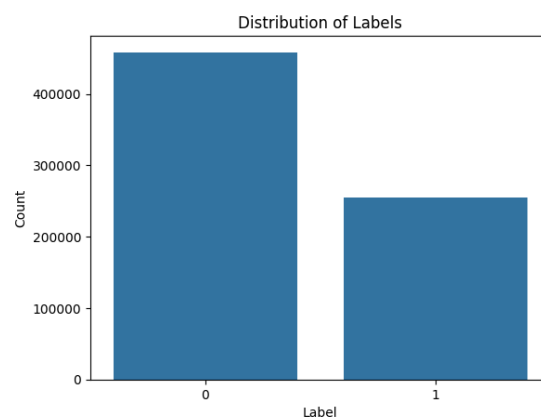
We wrote a function called [clean\\_text\(text\)](#). This code processed every single tweet/comment. It made sure everything was cleaned the same way, which will help our models learn better and faster in future.

#### Data Visualization

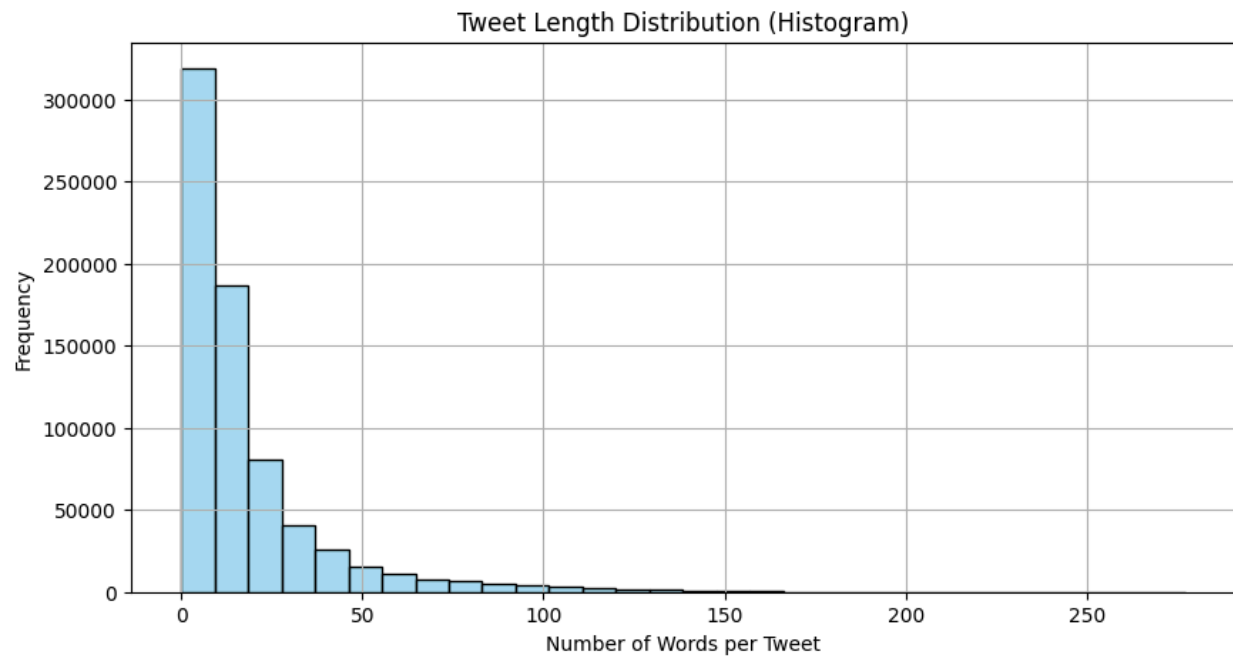
First of all labels, How many are hateful or non-hateful tweets

Non-Hateful -> 0  
Hateful -> 1

label	
0	458588
1	255522



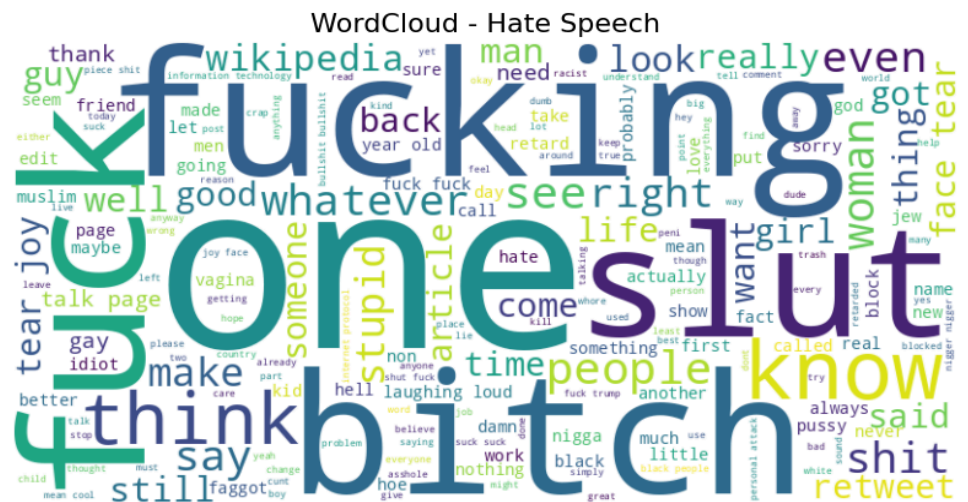
Distribution of word lengths: (to know what is the avg no. of words per tweet )



## Common words: (WordCloud)

Non-Hateful  
tweets (o)

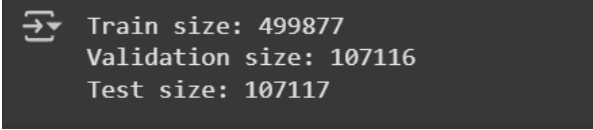
Hateful  
Tweets (1)



## 4) Dataset splitting for training and testing

We splitted our dataset into

- Training Data (70%): This is data from which our model "learns" from.
- Validation Data (15%): We used this to make sure our model wasn't just memorizing the training data (overfitting).
- Test Data (15%): This is the hidden data, which is not fed to the model while training and it is kept for testing the model later to see how it performs on unseen data.



```
➡ Train size: 499877  
Validation size: 107116  
Test size: 107117
```

## 5) ML models

Before starting to work on models, We first converted our `clean_text` into numerical format (as the models do not understand the plain words ) using a method called **TF-IDF** (term-frequency and inverse document frequency)

We used `TfidfVectorizer` to turn clean text into feature vectors. Limiting the size to the top 10,000 important tokens. Includes single words and pairs of words, improving context understanding by taking a window of text and processing.

We started with simple ML models.(Logistic Regression, Naive Bayes and Support Vector Machine (SVM))

- Logistic Regression: A good starting point, often surprisingly effective.
- Naive Bayes: Works well for text, as it assumes words are independent.
- Support Vector Machine (LinearSVC): Known for being strong with this type of numerical data.

## 6) Evaluation

We used common measures to measure our model performance

- Accuracy: How often the model was accurate/right overall

- Precision: When the model said "hate",but how often was it *actually* hate? Or When the model said " Not-hate",but how often was it *actually* Not-hate?
- Recall: How well the model found *all* the hateful messages.
- F1-Score: A mix (harmonic mean) of precision and recall, showing overall balance.

All these models showed decent performance.

## 7) Issues and Limitations

- Despite re-labeling, some misclassified samples might remain due to ambiguity in language or sarcasm.
- Classical ML models struggle with understanding deep contextual meaning or sarcasm, which may affect hate speech detection accuracy.

## 8) Deep learning

To capture meaning of words and richer linguistic patterns and context, we move to Deep learning. We used GloVe, which is word maps trained on billions of words.

### Pre-processing for DL

Tokenization: We used Keras's tool ([Tokenizer](#)) to convert text into sequences of integers, based on word frequency.

Word Embeddings :Low dimension (Compact) representation of something, here the size is 100 (100D)

## 9) Deep Learning Model with GloVe embeddings & GRU

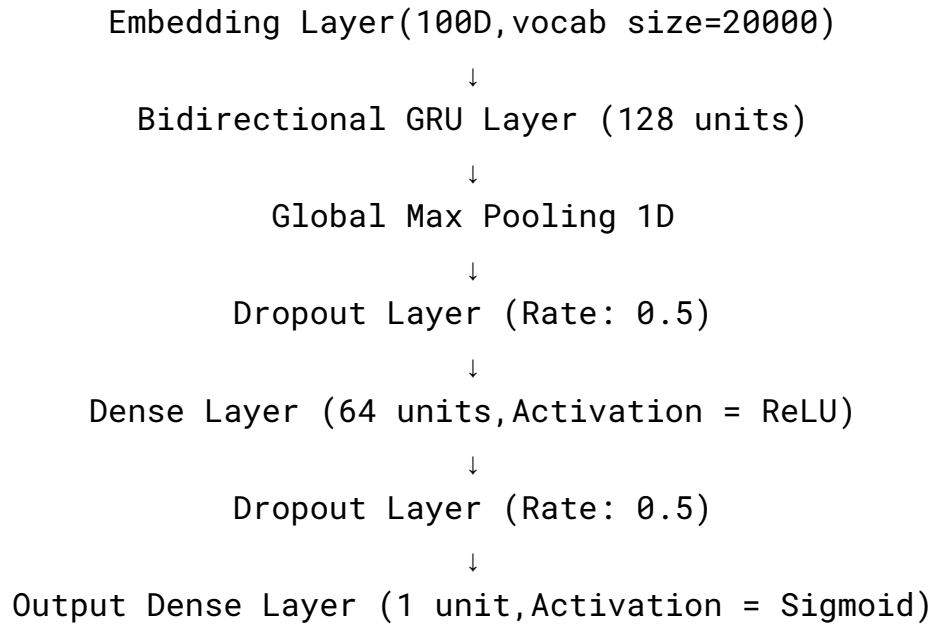
GloVe Word Map Layer: This is where the AI learned the meaning of words.

Bidirectional Gated Recurrent unit (GRU): A special "brain" that reads messages both forwards and backward, helping it understand context better.

Global Max Pooling: A step to pick out the most important features from the GRU.

Dense Layers: More "brain" layers that refine the prediction, with a final "yes/no" output for hate speech.

## DL Model Architecture



Output: Probability (0 to 1) → Binary Class (0: Non-Hate, 1: Hate)

## Training the GloVe-GRU model

We split our dataset into training (80%) and validation (20%) sets for training for 10 epochs and monitoring model performance with early stopping (Stops training if the model stops improving)

Loss Function: We used Binary Crossentropy, common for two-class problems.

Optimizer: We chose Adam, a popular and effective optimizer.

## Evaluation of the model

After training, we evaluated the model on the validation set using Accuracy, Precision & recall, F1-score

Final Validation Accuracy: 0.9580 (~95%)

## Result

Our GRU model was a huge step forward. It was much better at finding hate speech hidden in tricky, subtle language. This was our first major success!

## 10) Transformers

- **RoBERTa** : A state-of-the-art transformer model, fine-tuned for hate speech detection.
- **Toxigen-BERT** : Specifically trained on detecting toxicity and abusive language.

This is fine-tuning the already pre-trained model on our dataset. The architecture is below, how we are doing it with our DL model

Tokenizer(Converts text to input IDs)

↓

Pretrained Transformer Encoder  
(Model: roberta-base / Toxigen-bert)

↓

[CLS] Token Representation

↓

Dropout Layer (Rate= 0.3-0.5)

↓

Fully Connected Dense Layer

↓

Output(1 unit,Activation = Sigmoid)



## 11) Ensembling

We noticed the model sometimes got confused by subtle cases. But our deep learning model was better and to push performance further, we explored **ensemble modeling** - a technique that combines multiple models to make more robust predictions

Instead of relying on just one model, we built a hybrid system that aggregates predictions from **3** powerful models, We gave each model a specific "weight" based on how well it performed individually and how diverse its predictions were:

- **GRU-based Deep Learning Model : 20%**
- **RoBERTa : 45%**
- **Toxic-BERT : 35%**

$$\text{final\_score} = 0.2 * \text{gru\_prob} + 0.45 * \text{roberta\_prob} + 0.35 * \text{toxic\_prob}$$

Each model predicts a probability score for whether a tweet/comment/statement is hateful. We compute a weighted average of the three scores. If the final score is above **0.5**, the tweet/comment is classified as hateful.

## 12) Final Result

The final ensemble model combined GRU, RoBERTa, and Toxic-BERT to make more accurate and context-aware predictions.

It effectively captured subtle hate speech, including sarcasm, slang, and implicit toxicity, which previous individual models often missed.

By leveraging the unique strengths of each model, the ensemble improved overall precision, recall, and robustness.

This made it significantly more reliable for real-world hate speech detection compared to any single model.

Model	Text-Convert	Model Type	Accuracy
Logistic Regression	TF-IDF	Classical ML	94.08
Naive bayes	TF-IDF	Classical ML	86.83
SVM	TF-IDF	Classical ML	94.52
GloVe - GRU	GloVe-embedding	Deep-Neural Network	95.80
RoBERTa/Toxic-BERT		Transformer	
Ensemble		Hybrid	

## 13) Deployment: Making Our Model Available to Everyone

To allow users to test our model easily, we created a **Hugging Face** Space. It provides a clean-simple, web-based UI where users can simply enter text and the model will instantly classify it as Hateful or Not Hateful.

This deployment brings the model to life, enabling real-world experimentation without requiring users to install anything.

Link :

<https://aaryan24-hate-speech-detector.hf.space/?text=>