Yashraj Parmar

Data Glacier

2 June 2025

Week 9: Deliverables

1) **Information:**
    a) Group Name: NLP Sentinels
    b) Name: Yashraj Parmar
    c) Email: yparmar2024@gmail.com
    d) Country: United States of America
    e) College: Stevens Institute of Technology
    f) Specialization: Natural Language Processing
    g) Internship Batch: LISUM44
2) **Problem Description:**
    a) Social media platforms rely on hate speech detection systems to monitor and filter content that may contribute to online harm, including cyberbullying. However, these systems are not always fully effective, and harmful content can still go undetected. The goal of this project is to develop a machine learning model using natural language processing (NLP) techniques that can accurately assess and classify the level of hate speech in a given piece of text.
3) **Data Cleansing:**
    a) After checking for null values in the dataset, I realized that there were no null or missing values in any of the columns. Therefore, nothing had to be done for the null/missing values.

b) Since the dataset had only two possible values: 0, or non-hate speech, and 1, hate speech, outlier detection was not applicable in this case.

c) The class imbalance of significantly more non-hate speech examples than hate speech examples is addressed through the use of the class_weight = "balanced" parameter which allows the model training phase to ensure fair learning from both classes.

d) Most of the text has inconsistency in casing, the whole tweet can be reduced to lowercase to avoid treating same words with different casing as separate tokens. This can be removed through Python's .lower() function which lowers the case of all contents of a string.

e) The text has user mentions such as "@user" which had no relevant information to the sentiment of the text, therefore this can be removed as well. This can be removed through Python's .replace() function which replaces all instances of "@user" with "".

f) The text contains hashtags, and while these contain sentimental value, the "#" symbol is irrelevant, while the word is useful therefore the word will stay. This can be removed using Python's .replace() function as well which will replace all instances of "#" with "".

g) The text contains numerous special characters which will only cause noise in the model's interpretation, therefore this can be removed as well using Python's Regex library. This can be removed using Python's re.sub() function which will replace everything except lowercase letters, numbers, and spaces.

h) Some of the text also contains trailing spaces occasionally which is standardized spacing and removes leading/trailing whitespace. This can be removed using Python's re.sub() function as well which will replace all white spaces with nothing.

4) **Transformation Done on Data:**

a) The first featurization technique I will be using is TF-IDF Vectorization from scikit-learn. This method transforms text into numerical vectors based on the frequency of words and their relative importance across the dataset (Term Frequency–Inverse Document Frequency). Common words across all documents receive lower scores, while more distinctive or informative words are given higher weights. The process involves tokenizing the text and converting it to lowercase, removing stop words, and generating a sparse matrix of TF-IDF scores–where each row corresponds to a tweet and each column represents a word feature. This matrix is then used to train a Linear Support Vector Machine (SVM) model, which serves as a strong traditional baseline for text classification. Linear SVMs are particularly effective for high-dimensional, sparse datasets like those produced by TF-IDF, and are known for their strong performance in binary classification tasks such as hate speech detection. The purpose of this approach is to benchmark traditional NLP performance against transformer-based models like BERT.

b) The second featurization technique I will be using is BERT Embeddings, a transformer-based representation derived from the bert-base-uncased model provided by the Hugging Face's Transformers library. This method leverages a pre-trained BERT model to convert each tweet into a dense 768-dimensional vector that captures rich semantic meaning and contextual relationships between words. The process involves cleaning the tweets, tokenizing them using BERT's tokenizer, padding and truncating the sequences to a fixed length, and extracting the [CLS] token embedding from the final hidden state to represent the overall meaning of each tweet. These embeddings are then used to train a deep learning classifier, such as a feedforward neural network. This transformer-based featurization is designed to capture nuanced language patterns and significantly outperform traditional methods like TF-IDF on complex tasks such as hate speech detection.