Project Report: Hate Speech Detection Using Transformers (Deep Learning)

Name: Rakshit Grover
University: University of Southern California
Specialization: NLP
Batch Code: LISUM35
Email: rakshitg@usc.edu
Country: Poland
Date: AUGUST 18, 2024
Submitted to: Data Glacier

Table of Contents:

Project Plan

Week 1 (August 18, 2024):

- Deliverables: Problem Statement, Data Collection, Data Report

Week 2 (August 25, 2024):

- Deliverables: Data Preprocessing

Week 3 (September 1, 2024):

- Deliverables: Feature Extraction

Week 4 (September 8, 2024):

- Deliverables: Model Building

Week 5 (September 15, 2024):

- Deliverables: Model Evaluation

Week 6 (September 22, 2024):

- Deliverables: Flask Development, Heroku Deployment

Week 7 (September 30, 2024):

- Deliverables: Final Submission (Report, Code, Presentation)

2. Problem Statement

Hate speech refers to any form of verbal, written, or behavioral communication that attacks or uses derogatory or discriminatory language against individuals or groups based on factors such as religion, ethnicity, nationality, race, gender, or other identity markers. This project aims to build a machine learning model that can detect hate speech in text using Python.

The task of hate speech detection is often treated as a sentiment classification problem. For this project, we will focus on classifying hate speech using Twitter tweets as our data source. By training the model on data typically used for sentiment classification, we can create an effective tool to identify tweets containing hate speech.

3. Business Understanding

Hate speech detection is essential for online platforms to maintain a safe and inclusive environment.

1. Market Need: With the rise of social media, hate speech has become more prevalent, requiring effective detection systems to foster positive engagement.

2. Brand Impact: Companies that fail to address hate speech risk damaging their reputation and losing users. Proactive measures enhance corporate responsibility and build community trust.

3. Regulatory Compliance: As regulations against hate speech tighten, effective detection helps organizations avoid legal issues and fines.

4. Operational Efficiency: Automating detection reduces the workload on human moderators, enabling faster responses to incidents.

5. Data Insights: A machine learning model provides valuable insights into user behavior, guiding policy adjustments and interventions.

This project addresses a critical societal issue while positioning businesses for success in a digital landscape.

4.Data Collection

The dataset for this project, sourced from Kaggle, consists of Twitter data used for research in hate speech detection. The dataset contains 31,962 observations and three features. The text is categorized into three classes: hate speech, offensive language, and neither. It is important to note that the dataset includes content that may be considered offensive, including racist, sexist, or homophobic language.

5.1 Text Cleaning:

- **5.1.1 Convert to Lowercase:**
  All text was converted to lowercase to ensure uniformity. For example, words like "Racism" and "racism" should be treated as the same in a vector space model, preventing the creation of unnecessary dimensions.

- **5.1.2 Remove Punctuation:**
  Punctuation marks do not contribute meaningful information to the model, so they were removed using regular expressions.
- **5.1.3 Remove URLs:**
  Since URLs are not relevant to the hate speech detection model, they were removed from the text to ensure only the core content is processed.
- **5.1.4 Remove @tags:**
  User mentions, signified by @tags, were removed because they are irrelevant to the model’s task of detecting hate speech.
- **5.1.5 Remove Special Characters:**
  Special characters such as [!”#$%&’()*+,-./:;<=>?@[]^_{|}~] were eliminated as they hold no significance for the model. This was accomplished using Python’s isalnum()` function.

5.2 Preprocessing Operations

In this section, we apply various preprocessing operations to prepare the text data for modeling.

5.2.1 Tokenization

Tokenization is the process of breaking down raw text into smaller, meaningful units called tokens. Since our data is composed of paragraphs, we use the `nltk.word_tokenize` library to convert the text into individual words (tokens). This step is essential for Natural Language Processing (NLP), as it helps the model interpret the context of the text by analyzing the sequence of words.

5.2.2 Removing Stopwords

Stopwords are common words like 'a,' 'is,' 'the,' and 'are,' which do not contribute significant meaning in NLP tasks like hate speech detection. To remove these irrelevant words, we first tokenize the text as mentioned above, and then exclude any tokens that match the stopwords provided by the `nltk.corpus.stopwords` collection. This ensures that the model focuses on the more meaningful words in the text.

5.2.3 Lemmatization

Lemmatization involves grouping together different inflected forms of a word, treating them as a single item. Unlike stemming, lemmatization takes the context of the word into account, ensuring that similar words are linked to their base form. For example, words like "intelligently" and "intelligence" are converted to their root form, "intelligent." This process enhances the model’s ability to understand the true meaning of the text.

5.2.4 WordCloud

A WordCloud is a visual representation of text data where the most important words appear larger or in different colors. This tool helps to visually distinguish between keywords in datasets.

Below, we provide WordClouds for both hate speech and free speech, highlighting the most frequent terms in each category.

5.3 Feature Extraction

5.3.1 TF-IDF Model

After preprocessing the text, we apply the Term Frequency-Inverse Document Frequency (TF-IDF) model to extract relevant features. The model calculates the importance of each word in the dataset by assigning a score based on its frequency across documents. From this, we select the top 2,000 most frequent words for both hate speech and free speech categories. The resulting word count vectors represent the frequency of these words in the entire dataset, and these vectors serve as input for training our model.

5.4 Splitting the Data into Train and Test Sets

For model development, we divide the dataset into two parts: 80% for training and 20% for testing. This data split allows the model to learn patterns from the training data, while the testing set is used to evaluate its performance. Proper data splitting is crucial for creating accurate and reliable machine learning models, as it ensures that the model generalizes well to unseen data.