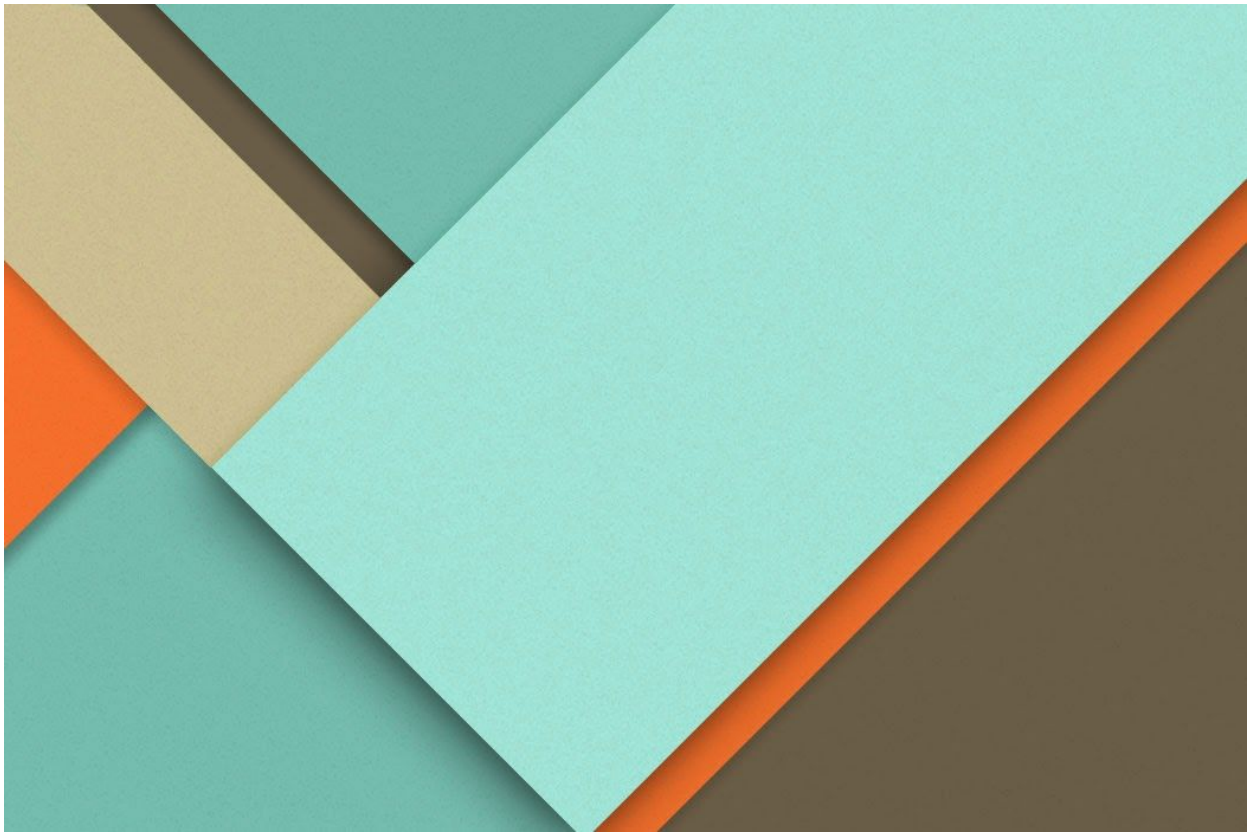

Hate Speech Recognition using Natural Language Processing

Adwait Toro
Chavi Singal
Rishab Ballekere
Sneha Lakshmikanthaiah
Sushrut Shendre
Tanvir Brar
Vedant Kshirsagar

Overview

We are exploring this topic as we think it is important to categorize hate speech and understand the sentiment behind it. There is a very fine line on whether to classify a tweet as hate speech and we are looking to define this threshold.

Goals

To build a model which classifies tweets as hate speech/racist one or not. We are also considering an alternative model which assigns these tweets a score between 0 and 1 which signifies the hatred/racism measure.

Data

We will potentially use the following datasets to build our model(s)

https://www.kaggle.com/vkrahul/twitter-hate-speech#train_E6oV3lV.csv

https://data.world/thomasrdavidson/hate-speech-and-offensive-language/workspace/file?filename=labeled_data.csv

Once our model is finalized, we intend to scrape live Twitter data and apply our model in real-time.

Approach

The data we have in our dataset is in the form of a binary variable table. Each observation in the table is a tweet followed by its label, where 1 corresponds to the tweet being a hate-speech/racist tweet and 0 corresponds to it being a non-hate speech tweet.

In this project, we intend to use these sample tweets and their labels to design a model which can understand the sentiment related to the tweet. We aim to do so, by converting the standalone tweets into features that would give us more insight as to whether it was intended as a hate speech comment. We will employ NLP techniques to decipher whether the tweet will be classified as a 1 or a 0.

Like numerical data, text data also can be untidy, and therefore, our first step would be to clean the dataset.

- We will remove all empty rows (missing values) if any. Then, from each tweet, we will remove irrelevant characters.

- Case doesn't usually hinder with the intent and meaning of the written text. Thus, in order to remove discrepancies that may arise due to small and upper case words, we will convert all our text into small case.
- Tokenization - Sentences will have punctuation marks, like commas, full stops, exclamation marks etc. We will remove these such that we have tokenized words.
- Stemming and Lemmatization - Words like 'carry', 'carried' and 'carrying' convey the same meaning. We need to assign all such words to the same form/root. In order to do this, while 'Stemming' chops off the ends of words, lemmatization usually returns the morphological base of the word.

After data cleaning and preprocessing, our primary approach is to build a Bag of Words encoding model. We would also be exploring other encoding models which might help us achieve our objective.

In Bag of Words, we encode the *n* unique words in our dataset into *n* columns, where each column corresponds to a word. This will be a binary column. For example, if we have a row, which is "Hi, how are you", the columns corresponding to 'hi', 'how', 'are' and 'you' will be 1, and every other column will be 0.

Once we have converted the entire textual dataset into a numerical one, and have the labels associated with it, we will employ classification algorithms to design the model. We will start with a simple logistic regression model and further explore other classification models to understand which model fits into our requirements and works best with our dataset. These will include decision trees (random forests, gradient boosting), Support Vector Machines, Neural Networks, Naive Bayes etc.