Final Project

on

Hate Speech Detection using Machine Learning

CSIT 598_01

Instructor: Dr. Jing Peng

by

Yamini Pathuri, Koundinya Raghava Nerella

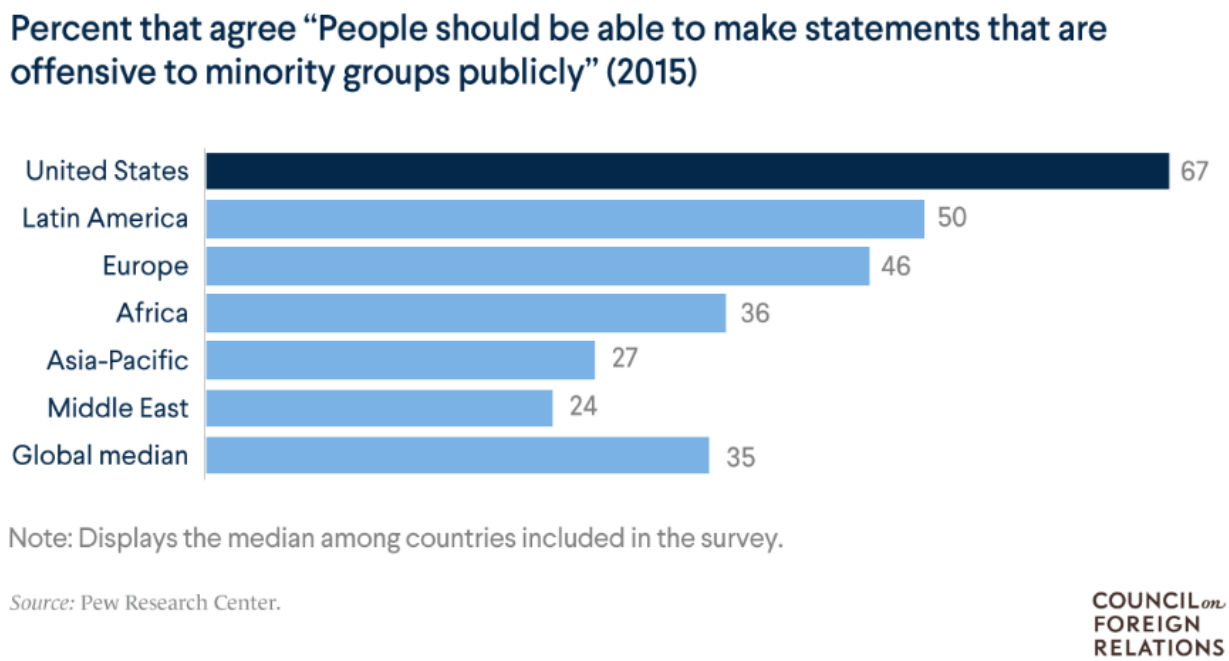
Contents:

1.Introduction:

One of the serious problems we see with social media platforms is hate speech. There have been incidents sparked from hate speech on social media on just about every continent. A large part of the world now communicates on social media, which is close to one third of the global population on Facebook alone. With more and more people moving online, experts say that the people who are inclined to racism, misogyny or homophobia have found niches that can strengthen their opinions and drive them to violence. We are motivated to use Machine Learning to build a hate-speech detection model which helps social media platforms to identify and eliminate such content from their platform. Hate speech is not legally defined because people's opinions cannot easily be considered hateful or offensive. Also, there is always a very thin line between hate speech and free speech which makes it difficult to put a stopper to hate speech while still protecting the people's right to free speech. In any case, the UN defines hate speech as any type of verbal, written or behavioral communication that may attack or use discrimination.

2.Background:

Sociologists and others have observed that the social media posts and other online discourses can inspire violent acts. The stats below show how inclined people are to adopt hate speech under the disguise of free speech [6].



Below are few examples [6] from around the globe where hate speech on social media has inspired acts of violence:

- In the United States, perpetrators of white supremacist attacks have circulated among racist communities online, and also embraced social media to publicize their acts. Prosecutors said the Charleston church shooter, who killed nine black clergy and worshippers in June 2015, engaged in a "self-learning process" online that led him to believe that the goal of white supremacy required violent action.

- In India, lynchings and other kinds of community violence, in many cases are sparked from rumors on WhatsApp groups, and are increasing since the Hindu-nationalist Bharatiya Janata Party (BJP) took power in 2014.
- Sri Lanka has also seen the vigilante inspired by rumors spread online, targeting Tamil Muslims.
- In Germany, a correlation has been found between anti-refugee Facebook posts by the extreme right-wing party Alternative for Germany and attacks against refugees.

3. Specifications & Design:

We have built a detection model with two different vectorizers and compared the results to find out the effective vectorizer that identifies the hate speech more efficiently.

- TF-IDF Vectorizer
- Countvectorizer

Also, we have used a decision tree classifier to identify each statement either as a hate-speech, offensive language or no hate or offense based on the training dataset.

3.1. TF IDF Vectorizer:

We have used the Tf-idf vectorizer to measure originality of a word by comparing it with the number of times a word appears in a tweet to the number of tweets the word appears in.

3.2.Countvectorizer:

The Count Vectorizer removes punctuation marks and converts all the letters to lowercase. It considers the occurrences of each word and converts each tweet to a vector of numbers. It forms a vocabulary of known words which is used to encode new words later.

3.3. Decision Tree Classifier:

We have used Decision Tree Classifier to generate the training model that can be used to predict the class of the selected variable which is vectorized using Tf-idf or Count vectorizer by learning the simple decision rule assumed from the training data.

4. About the Dataset:

The dataset of tweets that we are using for this project is downloaded from kaggle [7] which was originally collected from twitter. The dataset contains the following columns:

- index
- count
- hate_speech
- offensive_language
- neither
- class
- tweet

5. Implementation:

We have chosen Python language to build this model because of its wide range of library ecosystems and also it allows us to keep our rewriting program code lines to minimum. Below are the steps that we are followed in the implementation:

5.1. Importing necessary Libraries:

Firstly, we have started with importing the necessary libraries to build the model. Below is list of libraries that we've imported:

1. pr from nltk.util
2. Pandas
3. Numpy
4. CountVectorizer from skllearn.feature_extraction.text
5. TfidfVectorizer from skllearn.feature_extraction.text
6. Train_test_split from sklearn.model_selection
7. DecisionTreeClassifier from sklearn.tree
8. Re
9. snowballstemmer(english)
10. Stopwords from nltk.corpus
11. String

In [1]:
```
#Importing Libraries
from nltk.util import pr
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import re
import nltk
stemmer = nltk.SnowballStemmer("english")
from nltk.corpus import stopwords
import string
stopword=set(stopwords.words('english'))
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\kound\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[1]:
```
True
```

5.2. Data Loading and Data manipulation:

We have loaded the data which is in csv format using pandas and added a new column to the dataset as labels which contains the values as:

Hate Speech
Offensive Language
No hate and Offensive.

In [2]:
```
#Data Loading
data = pd.read_csv("twitter.csv")
print(data.head())
```

```
   Unnamed: 0  count  hate_speech  offensive_language  neither  class  \
0           0      3            0                   0        3      2
1           1      3            0                   3        0      1
2           2      3            0                   3        0      1
3           3      3            0                   2        1      1
4           4      6            0                   6        0      1

                                               tweet
0  !!! RT @mayasolovely: As a woman you shouldn't...
1  !!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2  !!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3  !!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4  !!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
```

In [3]:
```
#Adding new Column to the dataset:
data["labels"] = data["class"].map({0: "Hate Speech",
                                    1: "Offensive Language",
                                    2: "No Hate and Offensive"})
print(data.head())
```

```
   Unnamed: 0  count  hate_speech  offensive_language  neither  class  \
0           0      3            0                   0        3      2
1           1      3            0                   3        0      1
2           2      3            0                   3        0      1
3           3      3            0                   2        1      1
4           4      6            0                   6        0      1

                                               tweet                labels
0  !!! RT @mayasolovely: As a woman you shouldn't...  No Hate and Offensive
1  !!!!! RT @mleew17: boy dats cold...tyga dwn ba...      Offensive Language
2  !!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...      Offensive Language
3  !!!!!!!!! RT @C_G_Anderson: @viva_based she lo...      Offensive Language
4  !!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...      Offensive Language
```

Once each tweet is labeled as mentioned above, we have considered only two columns “tweet” and “labels” from the data for our further programming.

In [4]:
```
data = data[["tweet", "labels"]]
print(data.head())
```

```
                                               tweet                labels
0  !!! RT @mayasolovely: As a woman you shouldn't...  No Hate and Offensive
1  !!!!! RT @mleew17: boy dats cold...tyga dwn ba...     Offensive Language
2  !!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...     Offensive Language
3  !!!!!!!!! RT @C_G_Anderson: @viva_based she lo...     Offensive Language
4  !!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...     Offensive Language
```

5.3. Data Cleaning:

We have created a function to remove punctuation marks, unnecessary characters, unnecessary spaces and stemmer words and applied the cleaning function to the data.

In [5]:
```
#Data Cleaning
def clean(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text=" ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text=" ".join(text)
    return text
data["tweet"] = data["tweet"].apply(clean)
```

5.4. Building the model with Tf-idf Vectorizer and Decision Tree classifier:

As mentioned earlier, we have used the Tf-idf vectorizer to measure originality of a word by comparing it with the number of times a word appears in a tweet to the number of tweets the word appears in. Once the data is vectorized, we split the dataset into training and test sets and train a machine learning model for the task of hate speech detection with the help of a decision tree classifier.

In [6]:
```
#Applying TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [7]:
```
tfidf = TfidfVectorizer()
```

In [8]:
```
result = tfidf.fit_transform(data)
```

```
In [11]:
x = np.array(data["tweet"])
y = np.array(data["labels"])

Tfidf = TfidfVectorizer()
X = Tfidf.fit_transform(x) # Fit the Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

clf = DecisionTreeClassifier()
clf.fit(X_train,y_train)
```

```
Out[11]: DecisionTreeClassifier()
```

5.5. Building the model with Countvectorizer and Decision Tree classifier:

In the second attempt, we have used Countvectorizer instead of Tf-idf to vectorize the words in tweets and used Decision Tree Classifier to generate the training model that can be used to predict the class of the selected variable which is vectorized using Count vectorizer by learning the simple decision rule assumed from the training data.

```
In [9]:
x = np.array(data["tweet"])
y = np.array(data["labels"])

cv = CountVectorizer()
X = cv.fit_transform(x) # Fit the Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

clf = DecisionTreeClassifier()
clf.fit(X_train,y_train)
```

```
Out[9]: DecisionTreeClassifier()
```

6. Results and Evaluation:

Let us take a look at the results of each model and then we can come to a conclusion with regards to the vectorizer that most suits the hate speech detection model.

6.1. Results of Tf-idf model:

We have given an input with a hate speech to the model to test if it could identify and return the output as “Hate Speech”, however from the below example we see that the model fails to identify the hate speech.

```
In [23]:
sample = "Kill all the protestors"
data = Tfidf.transform([sample]).toarray()
print(clf.predict(data))
```

```
['No Hate and Offensive']
```

The above sample clearly gives out a call to kill all the protestors, however the model fails to identify it as a ‘hate speech’.

6.2. Accuracy of Tf-idf model:

Let us take a look at the accuracy of this model so that we could compare it with the accuracy of the other model and conclude.

```
In [21]: #Accuracy of decision tree classifier using Tf-idf Vectorizer
         score = clf.score(X_test, y_test)
         score

Out[21]: 0.8730896197579167
```

The accuracy of the Tf-idf model is 0.87308 which is around 87.30%.

6.3. Results of Countvectorizer model:

We have given 3 types of inputs to this model, one with a no hate or no offensive speech, the second with a hate speech and finally a sample with an offensive language to test if it could identify and return the outputs accordingly.

```
In [10]: #Sample Testing and Output
         sample = "Be kind to all the protestors"
         data = cv.transform([sample]).toarray()
         print(clf.predict(data))

['No Hate and Offensive']
```

```
In [11]: sample = "Kill all the protestors"
         data = cv.transform([sample]).toarray()
         print(clf.predict(data))

['Hate Speech']
```

```
In [12]: sample = "Fuck all the protestors"
         data = cv.transform([sample]).toarray()
         print(clf.predict(data))

         ['Offensive Language']
```

From the above screenshots we see that this model classifies the samples as desired and it is absolutely working for the given samples. Now let us dig deeper and check the efficiency of this model.

6.4. Accuracy of Count vectorizer model:

```
In [16]: #Accuracy of decison tree classifier using Countvectorizer
         score = clf.score(X_test, y_test)
         score

Out[16]: 0.87577943513877
```

The accuracy of the Count vectorizer model is 0.87577 which is around 87.58% and the accuracy of this model is slightly higher when compared to the earlier 87.30% of the Tf-idf model.

7. Future Work:

Though our model successfully identifies most of the hate speech based on the words used in a post, it still fails to identify underlying data for hate speech detection such as, accounting for speaker identity and dialect.

We are further motivated to build a dialect-identifiable module which could identify the underlying hate speech across the english speaking/ tweeting/ posting countries with different geographical, political and linguistic situations.

The Dialect-identifiable model could be achieved by establishing dialect parameters from the data that is used to train the model and using a classifier to identify and differentiate each dialect. Post which, we could define a different set of rules for each dialect to identify it as a hate speech.

8. Conclusion:

To conclude our project, we mainly concentrated on Tf-idf vectorizer and Countvectorizer with the help of Decision Tree Classifier and we analyzed that Countvectorizer is more effective when compared to the Tf-idf vectorizer here. However, there is a scope for testing this model with different combinations of vectorizers (Word2Vec, GloVe) and classifiers (RandomForestVectorizer, Logistic regression, Naive Bayes). Our research on this topic made us learn that more classifiers are being developed in Machine Learning for detection of Hate Speech on different social media platforms. As mentioned above we are looking forward to building a more efficient model which has a capability to detect hate speech at various levels and nuances of natural language.

9. References:

1. N. S. Mullah and W. M. N. W. Zainon, "Advances in Machine Learning Algorithms for Hate Speech Detection in Social Media: A Review," in *IEEE Access*, vol. 9, pp. 88364-88376, 2021, doi: 10.1109/ACCESS.2021.3089515.
2. Sindhu Abro, Sarang Shaikh, Zahid Hussain Khand, Zafar Ali, Sajid Khan and Ghulam Mujtaba, “Automatic Hate Speech Detection using Machine Learning: A Comparative Study” International Journal of Advanced Computer Science and Applications(IJACSA), 11(8), 2020. http://dx.doi.org/10.14569/IJACSA.2020.0110861
3. Kovács, G., Alonso, P. & Saini, R. Challenges of Hate Speech Detection in Social Media. *SN COMPUT. SCI.* 2, 95 (2021). https://doi.org/10.1007/s42979-021-00457-3
4. Decision Tree Algorithm by Nagesh Singh Chauhan on February 9, 2022
5. https://raw.githubusercontent.com/amankharwal/Website-data/master/twitter.csv
6. https://www.cfr.org/backgrounder/hate-speech-social-media-global-comparisons
7. https://raw.githubusercontent.com/amankharwal/Website-data/master/twitter.csv
8. https://patents.google.com/patent/US20170011739A1/en#:~:text=The%20computing%20device%20can%20generate%2C%20based%20on%20the,applying%20the%20dialect%20classifier%20to%20candidate%20content%20items.