DETECTION OF HATE SPEECH AND OFFENSIVE LANGUAGE IN TWITTER USING SENTIMENTAL ANALYSIS

Team members

- Thirunayan Manikantan R (113216104150)
- Aravind Kumar D (113216104016)
- Harish Krishna K (113216104045)

Guide:

Ms.R.Amirthavalli
(Assistant professor/
Department of CSE)

Domain

Machine learning

- **Random forest algorithm**

  Random forest is a supervised learning algorithm which is used for both classification as well as regression.

- **Logistic regression algorithm**

  Logistic regression algorithm uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity.

Abstract

- We show that it is a challenging task, as our analysis of the language in the typical datasets shows that hate speech lacks unique, discriminative features and therefore is found in the 'long tail' in a dataset that is difficult to discover.
- We then propose Deep Neural Network structures serving as feature extractors that are particularly effective for capturing the semantics of hate speech.
- Our methods are evaluated on the largest collection of hate speech datasets based on Twitter, and are shown to be able to outperform state of the art by up to 6 percentage points in macro-average, or 9 percentage points in the more challenging case of identifying hateful content.

Existing System

- Existing methods primarily cast the problem as a supervised document classification task.
- One relies on manual feature engineering that are then consumed by algorithms such as SVM, Naive Bayes; the other represents the more recent deep learning paradigm that employs neural networks to automatically learn multi-layers of abstract features from raw data (deep learning methods).

Disadvantage

- Existing studies on hate speech detection have primarily reported their results using micro-average Precision, Recall.
- The problem with this is that in an unbalanced dataset where instances of one class (to be called the ‘dominant class’) significantly out-number others (to be called ‘minority classes’), micro-averaging can mask the real performance on minority classes.

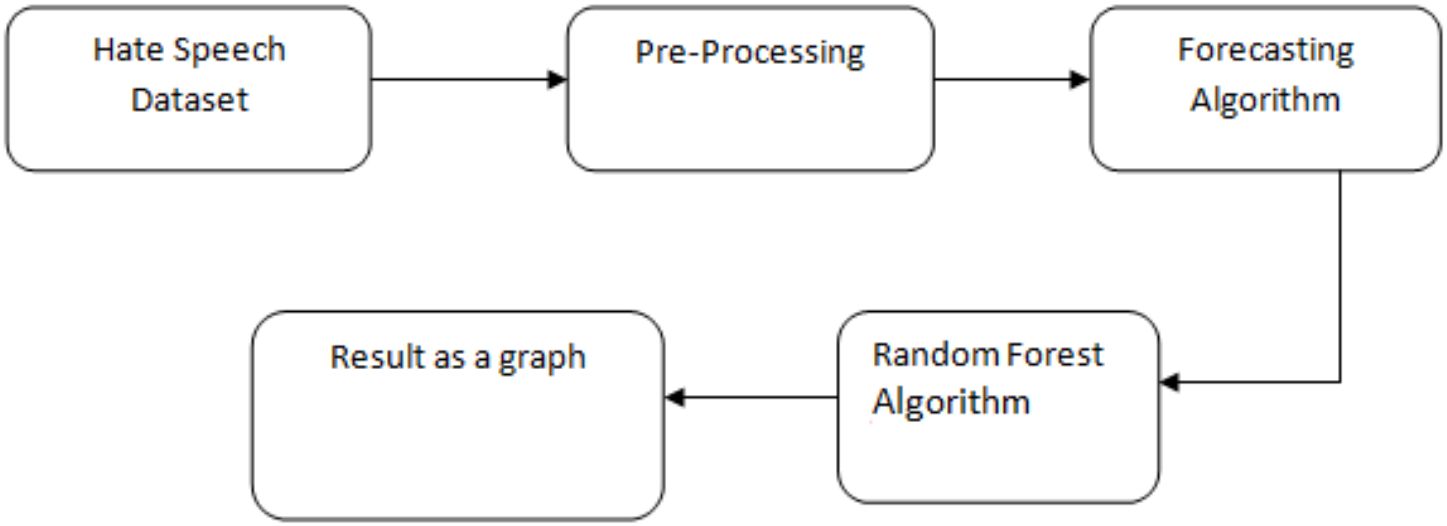
Proposed System

- All datasets are significantly biased towards non-hate, as hate Tweets account between only 5.8% (DT) and 31.6% (WZ). When we inspect specific types of hate, some can be even scarcer, such as ‘racism’ and as mentioned before, the extreme case of ‘both’.
- This has two implications. First, an evaluation measure such as the micro F1 that looks at a system’s performance on the entire dataset regardless of class difference can be biased to the system’s ability of detecting ‘non-hate’. In other words, a hypothetical system that achieves almost perfect F1 in identifying ‘racism’ Tweets can still be overshadowed by its poor F1 in identifying ‘non-hate’, and vice versa. Second, compared to non-hate, the training data for hate Tweets are very scarce.

Advantage

- This problem may not be easily mitigated by conventional methods of over- or under-sampling.
- Because the real challenge is the lack of unique, discriminative linguistic characteristics in hate Tweets compared to non-hate.
- As a proxy to quantify and compare the linguistic characteristics of hate and non-hate Tweets, we propose to study the ‘uniqueness’ of the vocabulary for each class.

System architecture



Software and Hardware Requirements

Hardware

- OS – Windows 7, 8 and 10 (32 and 64 bit)
- RAM – 4GB

Software

- Python
- Anaconda

Modules List

- Exploratory Data Analysis
- Data Cleaning
- Pre-processing & Transformation
- Data Partition
- Evaluation

Exploratory Data Analysis

During this step we performed some descriptive analysis and determined the target variable. We also explored how many classes were in the target and a selection of other possibly problematic (high cardinality) variables.

Data Cleaning

We dropped those high cardinality variables during this step as a precursor to the pre-processing step.

Pre-processing & Transformation

We removed the target variable from the entire data set and transformed the categorical variable into a model matrix with one-hot encoding. Other statistical software such as R, automates this step when generating models. I imputed the missing values in the data to 0. I scaled the continuous variables using min-max normalization which transforms values onto a scale from 0 to 1 to prevent variables on different scales heavily impacting the coefficients.

Data Partition

We partitioned the pre-processed data into a training and test data set. Modelling : We built a k-NN classifier model, using 10 neighbour classes and the Euclidean distance.

Evaluation

We scored the classifier on unseen test data and calculated the R squared values for both the training and test data.

Explanation

- The exponential growth of social media such as Twitter and community forums has revolutionised communication and content publishing, but is also increasingly exploited for the propagation of hate speech and the organisation of hate-based activities .
- The anonymity and mobility afforded by such media has made the breeding and spread of hate speech – eventually leading to hate.
- The term ‘hate speech’ was formally defined as ‘any communication that disparages a person or a group on the basis of some characteristics (to be referred to as types of hate or hate classes) such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics’.

Execution Snapshot

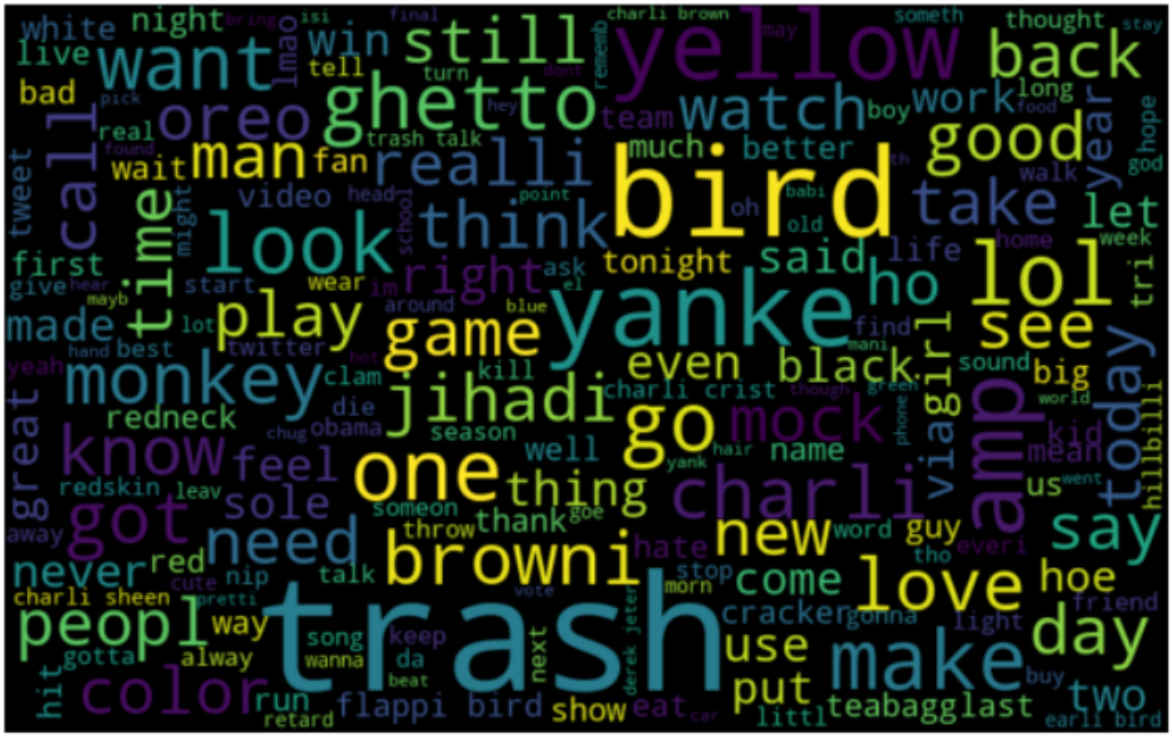
Visualizations

In [8]:
```
# visualizing which of the word is most commonly used in the twitter dataset

import matplotlib.pyplot as plt
from wordcloud import WordCloud

all_words = ' '.join([text for text in dataset['processed_tweets'] ])
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110).generate(all_words)

plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

In [9]:
```
# visualizing which of the word is most commonly used for hatred speech
hatred_words = ' '.join([text for text in dataset['processed_tweets'][dataset['class'] == 1]])
wordcloud = WordCloud(width=800, height=500,
random_state=21, max_font_size=110).generate(hatred_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

In [10]:

```
# visualizing which of the word is most commonly used for offensive speech
offensive_words = ' '.join([text for text in dataset['processed_tweets'][dataset['class'] == 2]])
wordcloud = WordCloud(width=800, height=500,
random_state=21, max_font_size=110).generate(offensive_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```



In [15]:

```
# Sentiment Analysis
sentiment_analyzer = VS()
def count_tags(tweet_c):

    space_pattern = '\s+'
    giant_url_regex = ('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|'
        '[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')
    mention_regex = '@[\w\-]+'
    hashtag_regex = '#[\w\-]+'
    parsed_text = re.sub(space_pattern, ' ', tweet_c)
    parsed_text = re.sub(giant_url_regex, 'URLHERE', parsed_text)
    parsed_text = re.sub(mention_regex, 'MENTIONHERE', parsed_text)
    parsed_text = re.sub(hashtag_regex, 'HASHTAGHERE', parsed_text)
    return(parsed_text.count('URLHERE'),parsed_text.count('MENTIONHERE'),parsed_text.count('HASHTAGHERE'))

def sentiment_analysis(tweet):
    sentiment = sentiment_analyzer.polarity_scores(tweet)
    twitter_objs = count_tags(tweet)
    features = [sentiment['neg'], sentiment['pos'], sentiment['neu'], sentiment['compound'],twitter_objs[0], twitter_objs[1],
                twitter_objs[2]]
    #features = pandas.DataFrame(features)
    return features

def sentiment_analysis_array(tweets):
    features=[]
    for t in tweets:
        features.append(sentiment_analysis(t))
    return np.array(features)

final_features = sentiment_analysis_array(tweet)
#final_features

new_features = panda.DataFrame({'Neg':final_features[:,0],'Pos':final_features[:,1],'Neu':final_features[:,2],'Compound':final_fea
                            'url_tag':final_features[:,4],'mention_tag':final_features[:,5],'hash_tag':final_features[:,6]})
```

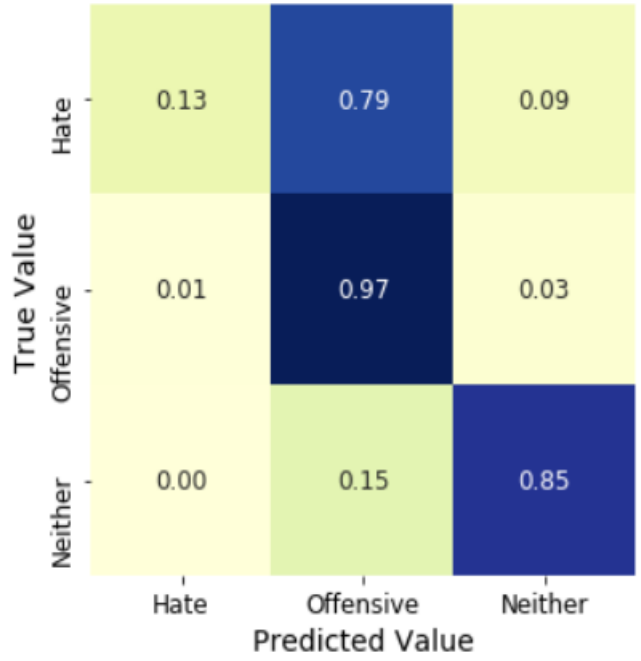
Building Models using Logistic Regression

In [13]:
```
# Using Bigram Features
X = panda.DataFrame(bigram)
y = dataset['class'].astype(int)
X_train_bow, X_test_bow, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.1)


model = LogisticRegression(class_weight='balanced',penalty="l1", C=0.01).fit(X_train_bow,y_train)
y_preds = model.predict(X_test_bow)
report = classification_report( y_test, y_preds )
print(report)

print("Accuracy Score:" , accuracy_score(y_test,y_preds))
```

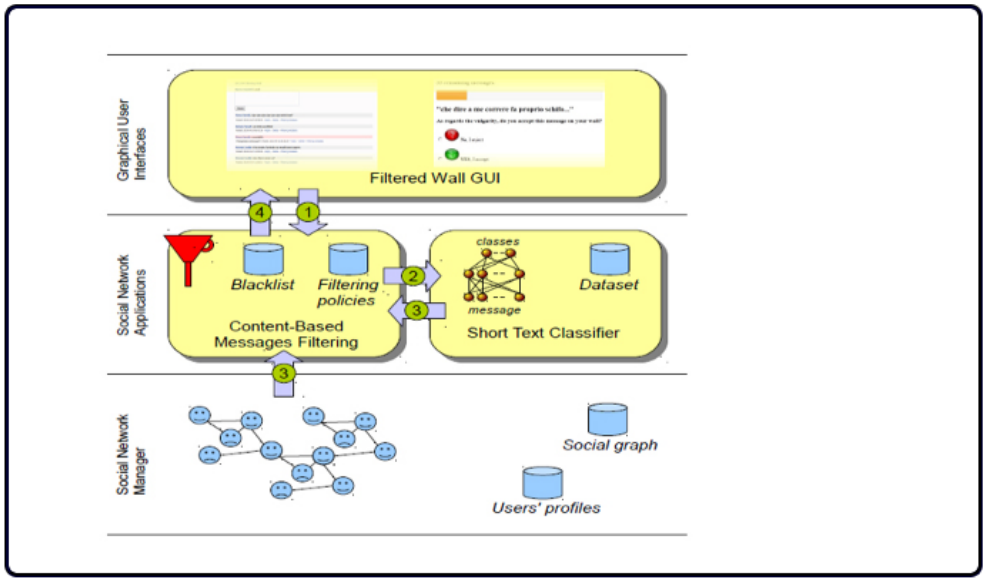
```
             precision    recall  f1-score   support

          0       0.43      0.41      0.42       164
          1       0.97      0.84      0.90      1905
          2       0.59      0.97      0.74       410

avg / total       0.87      0.83      0.84      2479

Accuracy Score: 0.8342073416700282
```

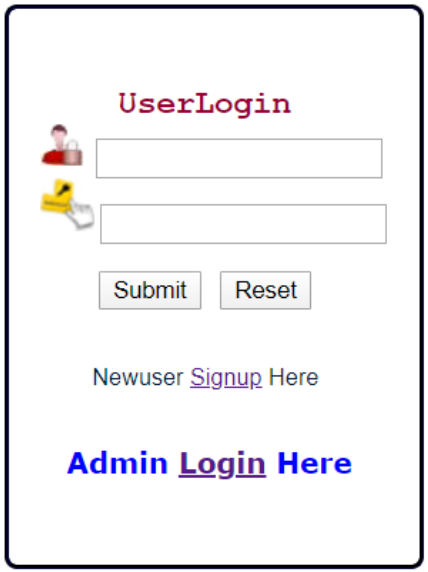
In [21]:

```
#Confusion Matrix for TFIDF with additional features
confusion_matrix = confusion_matrix(y_test,y_preds)
matrix_proportions = np.zeros((3,3))
for i in range(0,3):
    matrix_proportions[i,:] = confusion_matrix[i,:]/float(confusion_matrix[i,:].sum())
names=['Hate','Offensive','Neither']
confusion_df = panda.DataFrame(matrix_proportions, index=names,columns=names)
plt.figure(figsize=(5,5))
seaborn.heatmap(confusion_df,annot=True,annot_kws={"size": 12},cmap='YlGnBu',cbar=False, square=True,fmt='.2f')
plt.ylabel(r'True Value',fontsize=14)
plt.xlabel(r'Predicted Value',fontsize=14)
plt.tick_params(labelsize=12)
```



A-System-to-Filter-Unwanted-Messages-from-OSN-User-Walls





A-System-to-Filter-Unwanted-Messages-from-OSN-User-Walls

Welcome : thiru

Logout

hi

Friends Comments

aravind hai Reply Delete

aravind how are you? Reply Delete

Friends On Chat

aravind Comment

If You Are Going C

Update Status

Submit Reset

Change Profile Image

Choose File No file chosen

Submit Reset

People You May To Know

Ram Add Friend
Raju Add Friend
Selva Add Friend
Krish Add Friend
Arun Add Friend
Sam Add Friend
james Add Friend
aravind Add Friend

A-System-to-Filter-Unwanted-Messages-from-OSN-User-Walls

References and Publication

SNO	TITLE	YEAR	AUTHOR	CONCEPT	TECHNIQUE	FUTURE SCOPE	DRAWBACKS
1	Application of Machine Learning Techniques for Hate Speech Detection in Mobile Applications	2018	Bujar Raufi, Ildi Xhaferri	The proliferation of data through various platforms and applications is in constant increase. The versatility of data and its omnipresence makes it very hard to detect the trustworthiness and intention of the source. This is very evident in dynamic environments such as mobile applications. As a result, designing mobile applications that will monitor, control and block any type of malintents is important.	automatic hate speech detection, machine learning, artificial neural networks (ANNs)	The results indicated high level of classifier accuracy as well as applicability of the machine learning algorithms in mobile environments.	Not possible in hate speech and offensive language detection and prevention.
2	Analysis Text of Hate Speech Detection Using Recurrent Neural Network	2018	Arum Sucia Saksesi, Muhammad Nasrun, Casi Setianingsih	Twitter is very important for the success and destruction of one's image due to the many sentences of opinion that can compete the users. Examples of phrases that mean evil refer to hate speech to others. Evil perspectives can be categorized in hate speech, which hates speech is regulated in Article 28 of the ITE Law. Not a few people who intentionally and unintentionally oppose social media that contain hate speech. Unfortunately, social media does not have the ability to aggregate information about an existing conversation into a conclusion.	Hate Speech, Analysis text, Deep Learning, Recurrent Neural Network, RNN, LSTM	in this paper can make how to classify element of hate speech in the text by a computer, which later speech of hate can be recognized.	Needs an android device

SNO	TITLE	YEAR	AUTHOR	CONCEPT	TECHNIQUE	FUTURE SCOPE	DRAWBACKS
3	Text Analysis For Hate Speech Detection Using Back propagation Neural Network	2018	Nabiila Adani Setyadi, Muhammad Nasrun, Casi Setianingsih	They especially twitter is very important for the success and destruction of one's image due to the many sentences of opinion that can compete the users. To get hate speech information from the existing opinion data, in this final project will be done data processing with sentiment analysis using an Artificial Neural Network method optimized with back propagation algorithm. It can make how to classify element of hate speech in text by computer, which later speech of hate can be recognized	Hate Speech, Text Mining, Sentiment Analysis, Back propagation Neural Network	Hate speech is intimidates people from certain social groups oriented towards	Needs a high end processing platform
4	Detecting Hate Speech within the Terrorist Argument: A Greek Case	2018	Ioanna K. Lekea, Panagiotis Karampelas	Hate speech can be used by a terrorist group as a means of judging possible targets' guilt and deciding on their punishment, as well as a means of making people to accept acts of terror or even as propaganda for possibly attracting new members. To decide on how the automatic classification will be performed, we experimented with different text analyzing techniques such as critical discourse and content analysis and based on the preliminary results of these techniques a classification algorithm is proposed that can classify the communiqués in three categories depending on the presence of hate speech.	Greek terrorism, hate speech, tactics, targets, ideology, ethics, critical discourse analysis, content analysis	This paper presents a methodology for automatically detecting the presence of hate speech within the terrorist argument.	It is very limited because it is unable to handle a number of large existing data,

SNO	TITLE	YEAR	AUTHOR	CONCEPT	TECHNIQUE	FUTURE SCOPE	DRAWBACKS
5	Automatic Detection of Hate Speech on Facebook Using Sentiment and Emotion Analysis	2019	Axel Rodríguez, Carlos Argueta, Yi-Ling Chen	Hate speech has been an issue since the start of the Internet, but the advent of social media has brought it to unimaginable heights. To address such an important issue, in this paper, we explore a novel framework to effectively detect highly discussed topics that generate hate speech on Facebook. With the use of graph, sentiment, and emotion analysis techniques, we cluster and analyze posts on prominent Facebook pages. the definition of hate speech is conduct that uses direct assault with words on people who have particular traits, and this kind of assault usually has a tendency of violence or carries a tone of debasement.	Hate speech, Facebook, sentiment analysis, clustering	framework is able to identify the pages that promote hate speech in the comment sections.	Uses Matlab for processing