Machine Learning Engineer Nanodegree

Capstone Report

Raghunadh Chilukamari
March 09th, 2019

1. Definition

1.1. Project Overview

Sentiment analysis is basically concerned with analysis of emotions and opinions from text. We can refer sentiment analysis as opinion mining. Sentiment analysis finds and justifies the sentiment of the person with respect to a given source of content.

Social media contain huge amount of the sentiment data in the form of tweets, blogs, and updates on the status, posts, etc. Sentiment analysis of this largely generated data is very useful to express the opinion of the mass. Hate speech is an unfortunately common occurrence on the internet. Often social media sites like Facebook and Twitter face the problem of identifying and censoring problematic posts while weighing the right to freedom of speech. The importance of detecting and moderating hate speech is evident from the strong connection between hate speech and actual hate crimes. Early identification of users promoting hate speech could enable outreach programs that attempt to prevent an escalation from speech to action. In spite of these reasons, NLP research on hate speech has been very limited, primarily due to lack of general definition of hate speech, an analysis of its demographic influences and an investigation of the most effective resources.

Twitter sentiment analysis is tricky as compared to broad sentiment analysis because of the slang words and misspellings and repeated characters. We know that the maximum length of each tweet in Twitter is 140 characters. So, it is very important to identify correct sentiment of each word.

In this project we will build a model for sentiment analysis of tweets. With the help of feature vectors and classifiers such as Random Forest, Logistic regression, Naïve Bayes etc., we will be classifying these tweets as positive, negative given sentiment of each tweet.

We will use the dataset provided in this competition for this problem. Datasets are described below

1. **train.csv** - For training the models, we provide a labelled dataset of tweets. The dataset is in the form of a csv file with each line storing a tweet id, its label and the tweet
2. **test.csv** - The test data file contains only tweet ids and the tweet text with each tweet in a new line.

1.2. Problem statement

The objective here is to detect hate speech in tweets. we say a tweet contains hate speech if it has a racist or sexist sentiment associated with it. So, the task is to classify racist or sexist tweets from other tweets. Given a training sample of tweets and labels, where label '1' denotes the tweet is racist/sexist and label '0' denotes the tweet is not racist/sexist, the goal is to predict label on the test dataset.

In this project, we will use supervised machine learning models to determine whether a tweet is a racist tweet or not. Our goal is to train these models to identify tweets that has hate speech. This is going to be a challenging task as the data can contain lot of noise and given a tweet sometimes it will be challenging for even a human to determine whether a tweet exhibits racisms or hate. we will use the words in the tweets as features to these algorithms. If a tweet consists of words or the context of tweets exhibits racism or hate speech, we will classify that tweet as a racist tweet. For example, words that can help us identify would be white, black, racist, hate etc. To generate best features, we will eliminate noise in the data by preprocessing it.

1.3. Metrics

The metric used for evaluating the performance of models would be F1-Score.

The metric can be understood as -

True Positives (TP) - These are the correctly predicted positive values which means that the value of actual class is yes, and the value of predicted class is also yes.

True Negatives (TN) - These are the correctly predicted negative values which means that the value of actual class is no, and value of predicted class is also no.

False Positives (FP) – When actual class is no, and predicted class is yes.
False Negatives (FN) – When actual class is yes but predicted class in no.

Precision = TP/TP+FP

Recall = TP/TP+FN

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Class distribution is uneven in the dataset that we are going to use in the project. We will see that only 2,242 (~7%) tweets are labeled as racist or sexist, and 29,720 (~93%) tweets labeled as non-racist/sexist. Large number of positive labels can give an edge to accuracy and precision. It is important to find a balance between precision and recall and F1 score does exactly that. F1 score is basically a harmonic mean of precision and recall and it is always closer to smaller number thus we can be sure that we won't create a high precision or a recall solution. F1 score also takes a parameter beta to choose whether we want a high precision model or a recall model. If beta is close to 0 that means we are looking for a high precision model and if close to 1 it is going to be high recall model.

2. Analysis

2.1. Data Exploration

We will use a dataset consisting of a total of 31962 tweets in training dataset a nd 17917 tweets in testing dataset. This data is made available in the form of a csv file. The training dataset has 3 columns tweet id, tweet and the label of th e tweet indicating whether it represents a racist tweet or not, 1 being racist tw eet and 0 being otherwise.

Sample records are as below –

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

We can see that the twitter handles are masked for security purposes and are given as @user. Also, the tweet ids are sequence of numbers.

The distribution of the dataset is as follows:

Train set has 31,962 tweets and
Test set has 17,197 tweets.

Label-distribution in the train dataset:

```
train_df["label"].value_counts()
0    29720
1     2242
```

We can see that in the train dataset, we have 2,242 (~7%) tweets labeled as racist or sexist, and 29,720 (~93%) tweets labeled as non-racist/sexist. It is an imbalanced classification challenge.

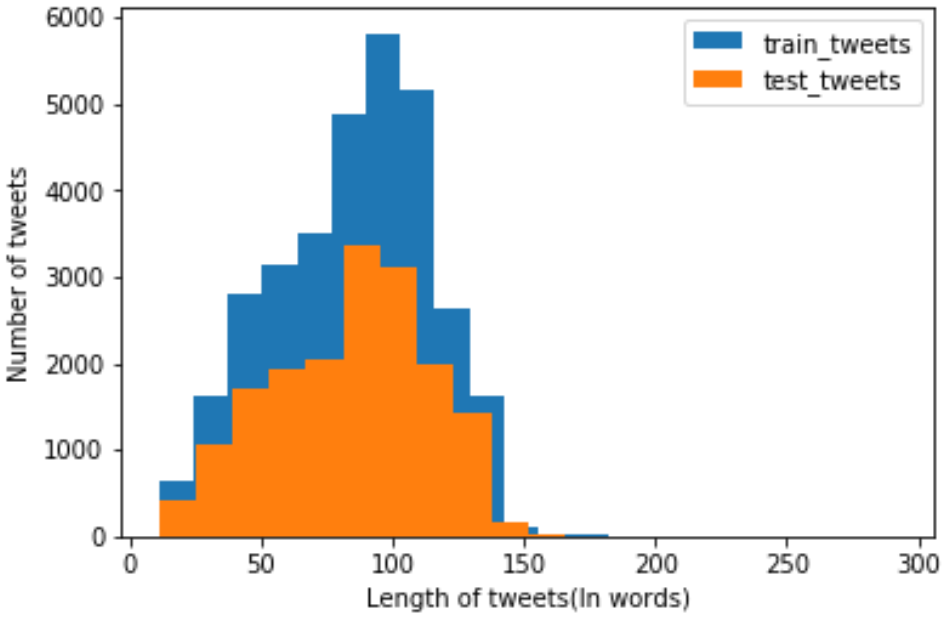
We will check if the dataset misses any information.

```
train_df['label'].isnull().sum()
train_df['tweet].isnull().sum()
test_df['label'].isnull().sum()
```

```
In training data :
number of missing labels : 0
number of missing tweets : 0
In testing data :
number of missing labels : 0
```

we can see that dataset contains all required information.

The distribution of length of the tweets, in terms of words, can be seen as below



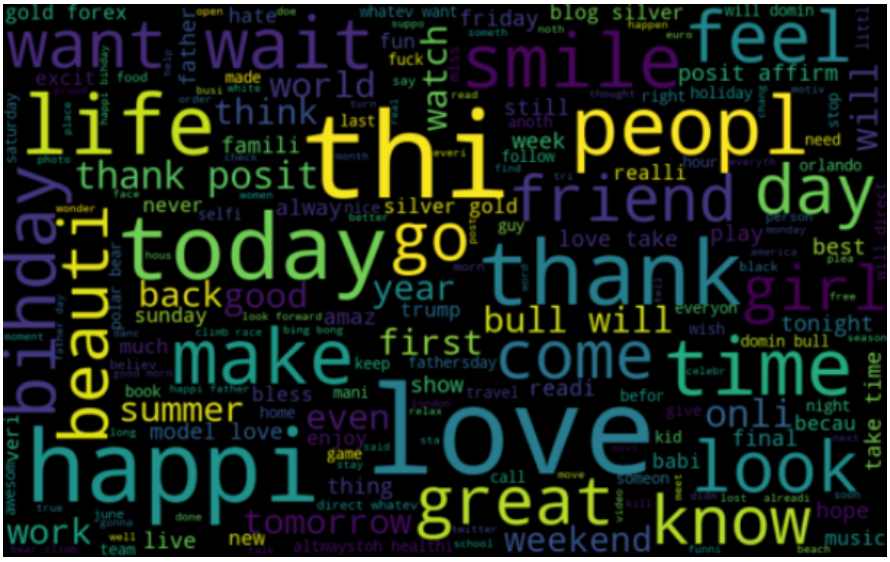
2.2. Exploratory Visualization

To understand how well the given sentiments are distributed across the train dataset we will visualize the data using word clouds. This visualization helps us answer few questions we may have like

- What are the most common words in the entire dataset?
- What are the most common words in the dataset for negative tweets?
- What are the most common words in the dataset positive tweets?

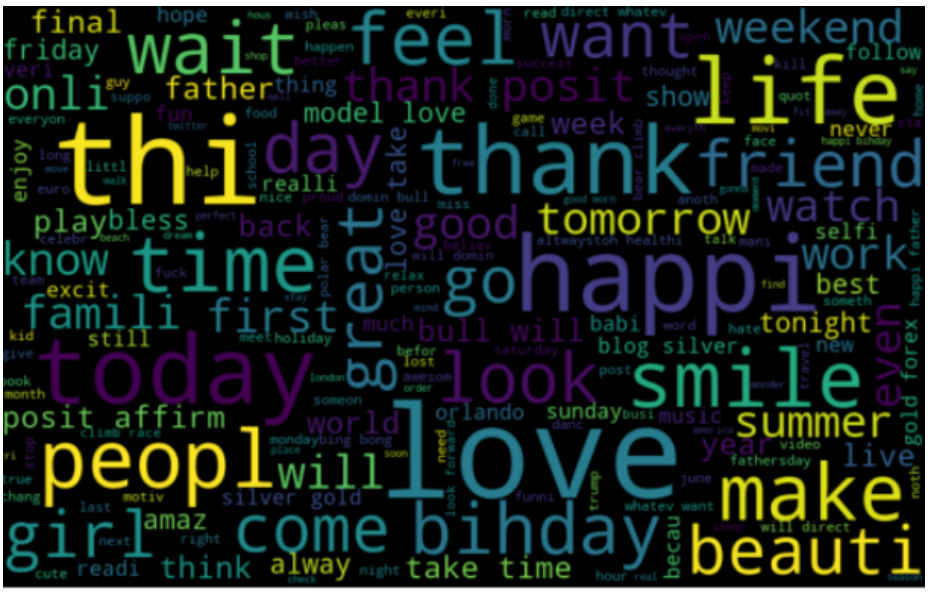
A word cloud is a visualization wherein the most frequent words appear in large size and the less frequent words appear in smaller sizes.

First, we will plot all words in the dataset and the word cloud looks like below

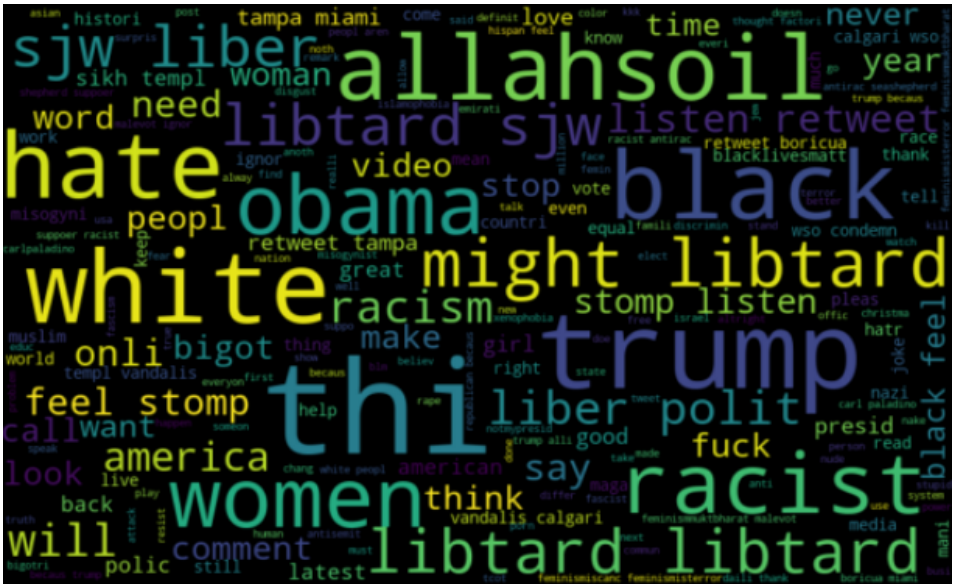


We can see most of the words are positive or neutral. Words like love, great, friend, life are the most frequent ones. It doesn't give us any idea about the words associated with the racist/sexist tweets. Hence, we will plot separate word clouds for both the classes (racist/sexist or not) in our train data.

Words in non-racist/sexist tweets



Similarly, we will plot the word cloud for the other sentiment. Expect to see negative, racist, and sexist terms.



As we can clearly see, most of the words have negative connotations.

2.3. Algorithms and Techniques

We will use supervised learning algorithms for this problem and also few techniques that helps us prepare features for training algorithms

Stemming:
Stemming is a standard technique in NLP. It reduces terms like loves, loving, and lovable to their base word, i.e., 'love'. This reduction to base word helps in removing words with similar meaning so that when we generate features like bow, tfidf, word2vec ect., we don't generate redundant features.

GridSearch Technique:
Grid search technique is an approach to select the best hyper parameters for the model If we have more than one hyper parameter to the model, we can use grid search technique to come up with the best model. Make a table with all the possibilities of the hyper parameters and calculate the score of each combination and pick up the model with best score. For ex: SVM can have two hyper parameters, kernel and C. Make a table with all possibilities of Kernel and C and assign score to them. Pick the kernel, C combination which has the best score.

Naïve Bayes Algorithm:

Bayes Theorem calculates the probability of a certain event happening based on the joint probabilistic distributions of certain other events

It's best to understand this theorem using an example. Let's say you are a member of the Secret Service and you have been deployed to protect the Democratic presidential nominee during one of his/her campaign speeches. Being a public event that is open to all, your job is not easy, and you have to be on the constant lookout for threats. So, one place to start is to put a certain threat-factor for each person. So, based on the features of an individual, like the age, sex, and other smaller factors like whether the person is carrying a bag, looks nervous, etc., you can make a judgment call as to whether that person is a viable threat.

If an individual tick all the boxes up to a level where it crosses a threshold of doubt in your mind, you can take action and remove that person from the

vicinity. Bayes Theorem works in the same way, as we are computing the probability of an event (a person being a threat) based on the probabilities of certain related events (age, sex, presence of bag or not, nervousness of the person, etc.).

One thing to consider is the independence of these features amongst each other. For example, if a child looks nervous at the event then the likelihood of that person being a threat is not as much as say if it was a grown man who was nervous. To break this down a bit further, here there are two features we are considering, age AND nervousness. Say we look at these features individually, we could design a model that flags ALL persons that are nervous as potential threats. However, it is likely that we will have a lot of false positives as there is a strong chance that minors present at the event will be nervous. Hence by considering the age of a person along with the 'nervousness' feature we would definitely get a more accurate result as to who are potential threats and who aren't.

This is the 'Naive' bit of the theorem where it considers each feature to be independent of each other which may not always be the case and hence that can affect the final judgement.

In our case this algorithm classifies racist tweet based on presence of words like say for example hate, white, black etc.

Random Forest Algorithm:

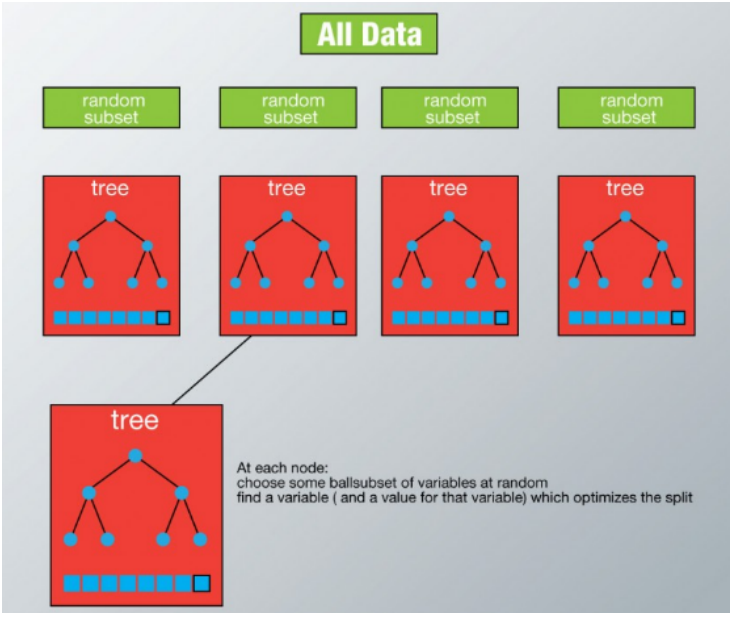
Random Forest is a versatile machine learning algorithm capable of performing both regression and classification tasks. It is a kind of ensemble learning method, where a few weak models combine to form a powerful model. In Random Forest, we grow multiple trees as opposed to a decision single tree. To classify a new object based on attributes, each tree gives a classification and we say the tree “votes” for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

It works in the following manner. Each tree is planted & grown as follows:

1. Assume number of cases in the training set is N. Then, sample of these N cases is taken at random but with replacement. This sample will be the training set for growing the tree.

2. If there are M input variables, a number m (m<M) is specified such that at each node, m variables are selected at random out of the M. The best split on these m variables is used to split the node. The value of m is held constant while we grow the forest.
3. Each tree is grown to the largest extent possible and there is no pruning.
4. Predict new data by aggregating the predictions of the n trees (i.e., majority votes for classification, average for regression).

In our case, Random forest can be assumed, the nodes of the trees are formed by features and decisions are made at each node based on whether a feature is present in particular tweet or not.



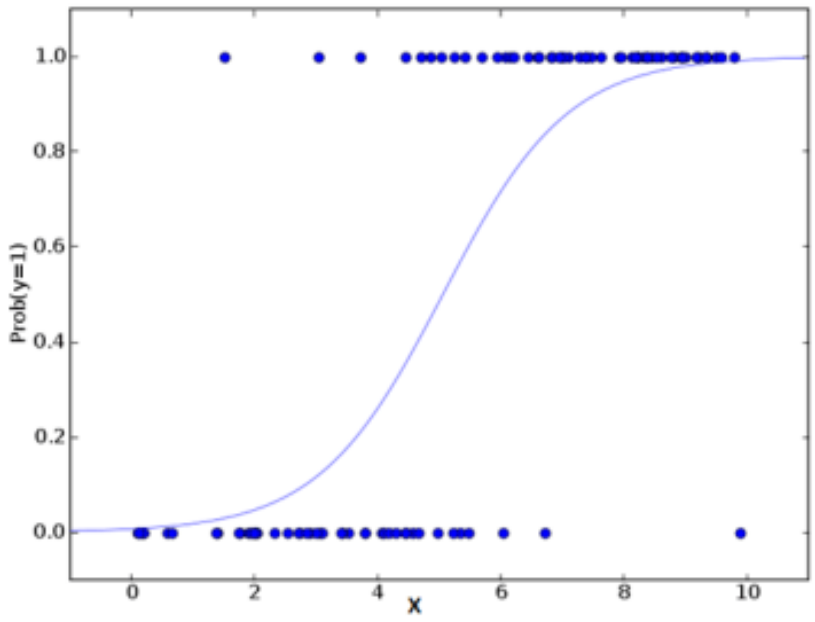
Logistic Regression:

Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. You can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as the dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function.

The following equation is used in Logistic Regression:

$$\log\left(\frac{p}{1-p}\right) = \beta_o + \beta(\text{Age})$$

A typical logistic model plot is shown below. You can see probability never goes below 0 and above 1.



XGBoost Algorithm:

Extreme Gradient Boosting (xgboost) is an advanced implementation of gradient boosting algorithm. It has both linear model solver and tree learning algorithms. Its ability to do parallel computation on a single machine makes it extremely fast. It also has additional features for doing cross validation and finding important variables. There are many parameters which need to be controlled to optimize the model.

Some key benefits of XGBoost are:

1. Regularization - helps in reducing overfitting

2. Parallel Processing - XGBoost implements parallel processing and is blazingly faster as compared to GBM.
3. Handling Missing Values - It has an in-built routine to handle missing values.
4. Built-in Cross-Validation - allows user to run a cross-validation at each iteration of the boosting process

Additional information about packages used in the project is given below.

Package	version	Homepage
numpy	1.15.4	http://www.numpy.org
pandas	0.23.4	http://pandas.pydata.org
gensim	3.7.1	http://radimrehurek.com/gensim
scikit-learn	0.20.2	http://scikit-learn.org
matplotlib	3.0.2	http://matplotlib.org
xgboost	0.82	ttps://github.com/dmlc/xgboost

Please note that default parameters were used for the initial training of supervised learning models. More information on default parameters of algorithms is available in the scikit-learn homepage.

Installing XGBoost requries some prerequisites and can be tricky. Below are the steps to install the package. First install binaries, build and then run pip install. To install xgboost libraries:

```
git clone --recursive https://github.com/dmlc/xgboost.git
cd xgboost
./build.sh
```

Note : gcc compiler is required.

2.4. Benchmark

We will use below paper to benchmark our model:

https://pdfs.semanticscholar.org/a050/90ea0393284e83e961f199ea6cd03d13354f.pdf

Classifier	Training Corpus	Test Corpus	$F1_{pos}$	$F1_{neg}$	$F1_{neutral}$	F1
SVM	SB10k	SB10k	66.16	47.80	81.32	56.98
CNN	SB10k	SB10k	71.46	58.72	81.19	**65.09**
SVM	SB10k	MGS	49.50	38.62	66.41	44.06
CNN	SB10k	MGS	50.41	44.19	71.81	**47.30**
SVM	SB10k	DAI (full)	62.30	61.40	81.22	**61.85**
CNN	SB10k	DAI (full)	62.79	58.43	79.92	60.61
SVM	MGS	SB10k	67.77	53.23	80.20	60.50
CNN	MGS	SB10k	63.94	58.21	70.66	**61.07**
SVM	MGS	MGS	60.34	56.48	69.31	58.41
CNN	MGS	MGS	61.49	58.12	68.62	**59.80**
SVM	MGS	DAI (full)	59.32	56.03	74.83	57.68
CNN	MGS	DAI (full)	61.01	55.74	76.88	**58.38**

We can treat this benchmark as hypothesis as the training data used is different. we will take minimum F1 score 44.06 as benchmark to our model.

3. Methodology

3.1. Data processing

Data is preprocessed to remove unwanted words and characters which are less relevant to find the sentiment of the tweets. We will remove punctuation, special characters, numbers, and terms that don't carry much weightage in context to the text.

Data preprocessing involves:

1. Remove the twitter handles. These twitter handles hardly give any information about the nature of the tweet. Handles here are '@user'.
2. Stemming the data. For example, reducing terms like loves, loving, and lovable to their base word, i.e., 'love' are often used in the same context.
3. Getting rid of the punctuations, numbers and even special characters since they wouldn't help in differentiating different types of tweets except # to identify hashtags
4. Removing short words
5. Removing stop words.

Manually defined google stop words are:

['a','about','above','across','after','again','against','all','almost','alone','along','already','also',
'although','always','among','an','and','another','any','anybody','anyone','anything','anywhere','are','area','areas',
'around','as','ask','asked','asking','asks','at','away','b','back','backed','backing','backs','be','became','because','become',
'becomes','been','before','began','behind','being','beings','best','better','between','big','both','but','by','c','came',
'can','cannot','case','cases','certain','certainly','clear','clearly','come','could','d','did','differ','different','differently',
'do','does','done','down','down','downed','downing','downs','during','e','each','early','either','end','ended','ending',
'ends','enough','even','evenly','ever','every','everybody','everyone','everything','everywhere','f','face','faces','fact',
'facts','far','felt','few','find','finds','first','for','four','from','full','fully','further','furthered','furthering',
'furthers','g','gave','general','generally','get','gets','give','given','gives','go','going','good','goods','got','great','greater',
'greatest','group','grouped','grouping','groups','h','had','has','have','having','he','her','here','herself','high','high',
'high','higher','highest','him','himself','his','how','however','i','if','important','in','interest','interested','interesting','interests',
'into','is','it','its','itself','j','just','k','keep','keeps','kind','knew','know','known','knows','l','large','largely','last',
'later','latest','least','less','let','lets','like','likely','long','longer','longest','m','made','make','making','man','many',
'may','me','member','members','men','might','more','most','mostly','mr','mrs','much','must','my','myself','n','necessary',
'need','needed','needing','needs','never','new','new','newer','newest','who','whole','whose','why','will','with','within',
'without','work','worked','working','works','would','x','y','year','years','yet','you','young','younger','youngest','your',
'yours','z'

All the training and testing samples were transformed to three different feature representations:

Bag of words Features:
The basic idea of Bag of words is to take a piece of text and count the frequency of the words in that text

TF-IDF Features:
This is another method which is based on the frequency method, but it is different to the bag-of-words approach in the sense that it takes into account not just the occurrence of a word in a single document (or tweet) but in the entire corpus.

Word2Vec Features:
Word embeddings are the modern way of representing words as vectors. The objective of word embeddings is to redefine the high dimensional word features into low dimensional feature vectors by preserving the contextual similarity in the corpus. For example, **King -man +woman = Queen**

3.2. Implementation

To implement the pre-processing steps, we used following python packages.

- We used porter Stemmer from gensim package for stemming the data
- We used token_pattern = r'\b[#a-zA-Z]{4,}\b', stop_words=stop_word_list arguments to remove punctuations, short words and stop words while generating bag of words using countvectorizer.
- We used countvectorizer to generate Bow
- We used TfidfVectorizer from sklearn.feature_extraction package to generate TFIDF Features.
- We used gensim package to generate Word2Vec features

3.3. Refinement

We tried generating word2vec features along with bag of words and TFIDF features which gave us following advantages

1. Dimensionality reduction - significant reduction in the no. of features required to build a model.
2. It captures meanings of the words, semantic relationships and the different types of contexts they are used in.

We reduced number of features to 200, whereas in Bag of Words and TF-IDF we had 1000 features.

After running all the models, we noticed that Logistic regression performed well by taking average performance across all features. But this is a simple model with no tuning. We leveraged GridSearch technique which is described in Algorithms and techniques section to pick the best parameters of the model.

The parameters that were tuned are –

- 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]
- 'max_iter':[100,200,300,400,500]

4. Results

4.1. Model Evaluation and Validation

We have tried various classifiers using various features sets. Although RF seemed to perform better on an average across all feature sets, the logistic regression model with Word2Vec features turns out to be the best model with 57.75 F1 score. The results align we with our expectations.

	bow	tfdif	word2vec	avg
rf	52.7	52.7	50.2	51.466667
xg	34.3	34.3	53.8	40.900000
nb	51.5	51.3	0.0	33.466667
lg	49.5	49.5	57.5	47.533333

Note: we have not trained naïve bayes on Word2Vec features.

The parameters tuned for this model are:

1. C Parameter – constant attached to classification error. Large C focuses on classifying correctly.

2. Max_iter Maximum number of iterations taken for the solvers to converge.

The final parameters of the model are

(C=10, class_weight=None, dual=False, fit_intercept=True,intercept_scaling=1, max_iter=100, multi_class='warn', n_jobs=None, penalty='l2', random_state=None, solver='warn', tol=0.0001, verbose=0, warm_start=False)

The models have been trained on various features to be able to better classify the tweets. Given the type of preprocessing techniques we chose and the features we have generated we can be confident that model is robust and generalizes well to any dataset like this.

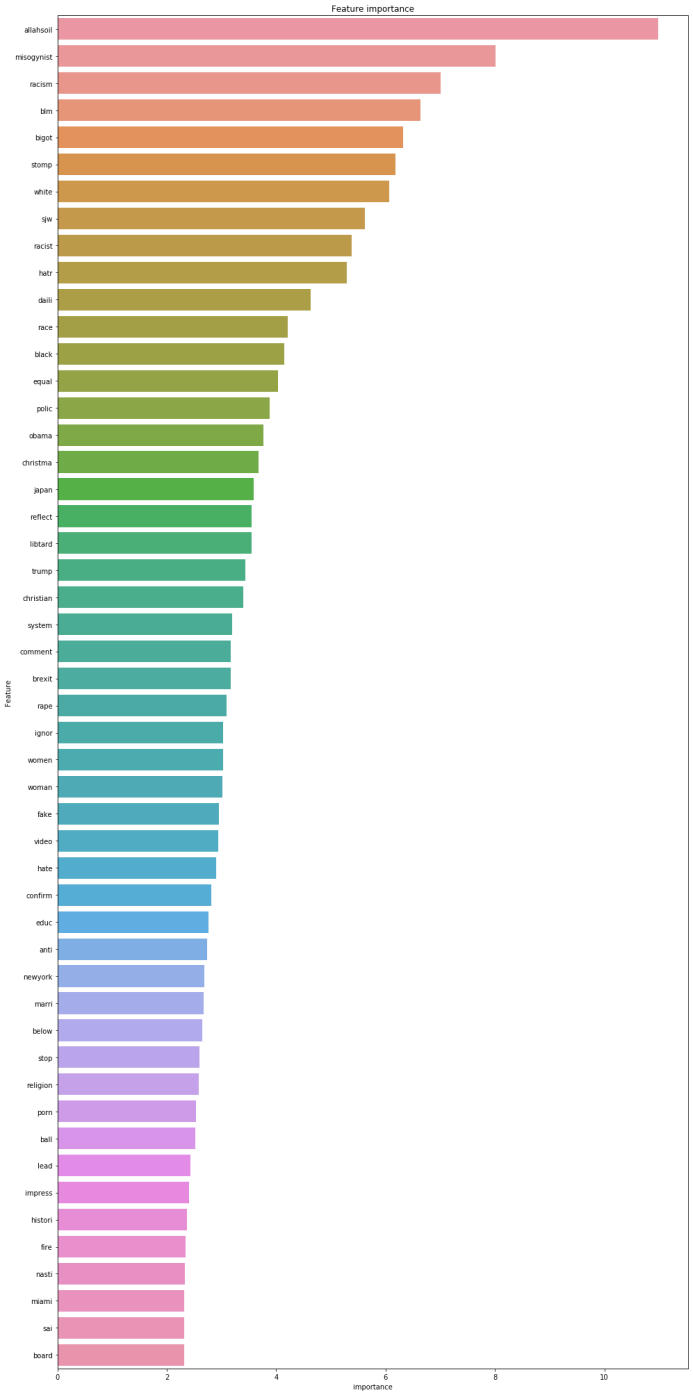
4.2. Justification

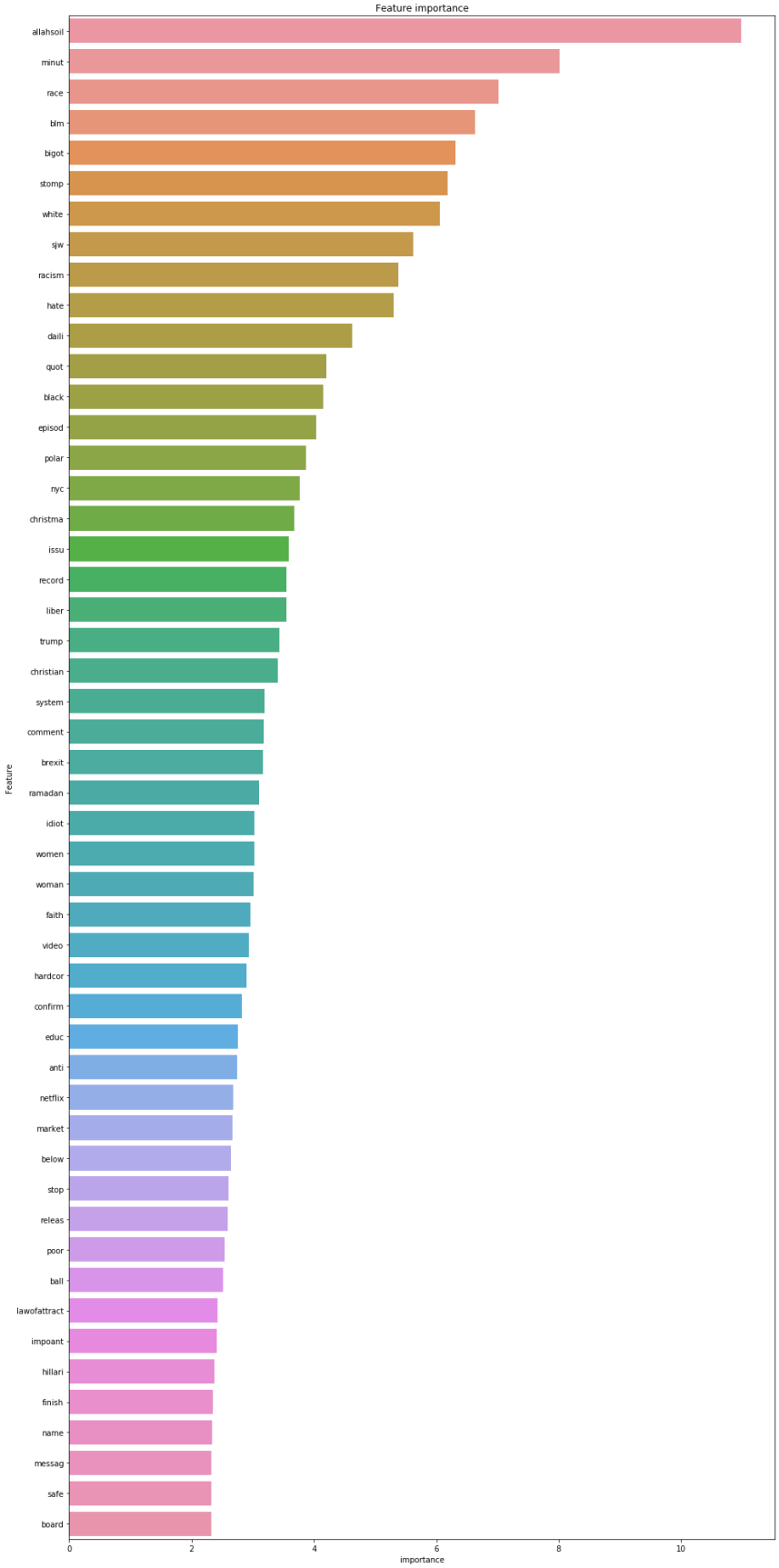
We have taken the benchmark to be as 44.06 F1 score. Our model has achieved an F1 score of 57.5 with precision of 47 and recall of 73. We can see that our model has better chance of identifying racists tweets.

5. Conclusion

5.1. Free form visualization

To get an understanding of the most important words in the tweets that helps us distingiuish racist vs non racist tweets can be seen in the fig below. It turns out that no matter what technique we use to generate features, the important words are more or less the same.


Feature importance
allahsoil
misogynist
racism
blm
bigot
stomp
white
sjw
racist
hatr
dalit
race
black
equal
polic
obama
christma
japan
reflect
libtard
trump
christian
system
comment
brexit
rape
ignor
women
woman
fake
video
hate
confirm
educ
anti
newyork
marri
below
stop
religion
porn
ball
lead
impress
histori
fire
nasti
miami
sai
board
Feature
0
2
4
6
8
10
importance


Feature importance
allahsoil
minut
race
blm
bigot
stomp
white
sjw
racism
hate
daili
quot
black
episod
polar
nyc
christma
issu
record
liber
trump
christian
system
comment
brexit
ramadan
idiot
women
woman
faith
video
hardcor
confirm
educ
anti
netflix
market
below
stop
releas
poor
ball
lawofattract
impoant
hillari
finish
name
messag
safe
board
Feature
0
2
4
6
8
10
importance

5.2. Reflection

In this project we have developed algorithms to determine racist vs non racists tweets. To achieve this, we have done the following:

- Cleansed the data by removing stop words, removing punctuations, and stemming the data.
- We have created features following three techniques:
    - Bag of words
    - TFIDF
    - Word2Vec features
- We have trained various models like Naive Bayes, Random Forest, XGboost and logistic regression using those features and tuned it using GridSearchCV techniques.

Comparing the performance of all trained classifiers with features sets, the best results were achieved using word2Vec features with Logistic Regression model.

The interesting part of the project is to see how Word2Vec features have tremendously improved the performances of models xgboost and logistic regression. The performance of xgboost has improved from 34.3 to 53.8 and logistic regression from 49.5 to 57.5 which is significant.

Given this is social media data it is really a daunting task to clean this data as the data can contain a lot of noise. I would say pre-processing this data is the most difficult part which ultimately decides the features we are going to generate. Unless we remove the noise in the data, we won't be able to get good accuracy.

We have followed some state of art techniques like stemming, used manually defined stop words for social media data and some pretty good preprocessing techniques to clean the data and generate features. Personally, I am satisfied with the performance attained with this model given that there is also not an even distribution of racist vs non racist tweets in the data. We can see that 70% of data contains non racist tweets. We will need more racist tweets and am sure this will help the model to generate more appropriate features and classify appropriately.

5.3. Improvement

Though we have preprocessed data well enough to generate good features, there is still some scope for improvement like

- preprocess data like use lemmatization to get rid of unnecessary words.
- Use part of speech tagging and bi-grams of tri-grams for BOW and TFIDF to generate new features.
- Try model ensembling.
- We can also try some feature selection techniques as prescribed in this paper - http://nmis.isti.cnr.it/sebastiani/Publications/ACMCS02.pdf

References:

[1] Competition: https://datahack.analyticsvidhya.com/contest/practice-problem-twitter-sentiment-analysis/