

Karthiga Thangavelu
V00925048
CSC 501

Content:

- 1.1 Datas
- 1.2. Preprocessing
- 1.3. Bias in Language
- 1.4. Clustering the tweet content based on labels
- 1.5. Analyzing the hashtag and finding similarity based on hashtag
- Appendix

1.1. Dataset:

The given dataset is twitter dataset which was alleged that Russian “toll factory”, the Internet Research Agency(IRA) had deliberately sought to sow political dissatisfaction in the US with inflammatory social media content. The tweets produced by these twitter accounts are handles by IRA. There are 13 csv files which has all the tweets, content, publishes date, language, tweet ids, region, author, etc in total of 21columns. These data has account_category which is righttroll and lefttroll. RightTroll represents the Republican party and Left Troll represents the Democratic party. The given are not clean dataset in order to take out useful information we have to sanitize the dataset because it contain emoji’s, url, hashtag, etc.

1.2. Pre-Processing:

Prior to answering the questions, I have sanitized the dataset for text modeling to achieve better result and efficiency of the analysis. I have combined all the data file and filtered only the english tweets.

Lower case:

In lower case all the content are converted to lower case if they are in upper case. Since the text data are sensitive in programming language it is important to convert the word to lower case. When unsanitized data are tokenized it would consider lower case and upper case as different words. For example ‘**Word**’ and ‘**word**’ are not consider as same. Changing to lower case will change the ‘**Word**’ to ‘**word**’

URL:

We can often see links in the tweets which are URL. Tokenizing the URL will not give any useful information for text modeling. So URL can be removed while sanitizing by removing words with 'http\S+' and using regular expression 're'.

Punctuation:

The punctuation won't give any useful information for modeling. So the punctuation can be removed and also it facilitated the tokenizing using the function `re.sub(r'[^\w\s]', '', i)`.

Stop words:

I have used stop word library from `nlk.corpus` to remove stop words in english like (on, the, us, from, etc) which are commonly used word in sentence. Removing the stop word from content will improve the computational and space efficiency.

Stemming:

Stemming the content will remove the multiple similar words. For example, a sentence can have words '**playing**' and '**played**'. Stemming the sentence will change both '**played**' and '**playing**' to play therefore decreasing the length of the token list. For stemming I have used `PorterStemmer()` from `nlk.stem` library

Lemmatization:

Lemmatization which inflects the words used in the sentence for better efficiency. For example, if there is a word like '**penni**' using lemmatization will change it to '**penny**'. I have used `WordNetLemmatizer` from `nlk.stem` library.

Token:

After removing punctuation and converting to lower case I have tokenized the words. For which I have used `nlk.word_tokenize` from `nlk` library. The sentence has to be tokenized for finding tfidf or for word embedding.

1.3. Questions 1:

How much bias can be identified in the language used in the tweets

In order to identify biases in the given twitter dataset, we have to sanitize the data and extract only the main words and tokenize required for modeling which is explained above. Here, I have taken the whole dataset which was around which was around

5892415. From this I have filtered only the English tweets and righttroll and lefttroll to time and space complexity.

To search for bias in the language, I have used word2vec to train the data and found the similarity in the data given.

The Gensim library which has word2vec model to train the dataset and find similarity between words. Word2vec can be represented in two ways one is CBOW (continuous bags of words) and skipgram. CBOW, predicts the target word based on all its neighborhood whereas skipgram predicts the target word based on only the neighbor word. For train the dataset I have used CBOW since the training time is less compared to skipgram. Skipgram works well with small dataset and CBOW can represent the words well and it is faster. So for modeling and analyzing the given dataset I have used CBOW. The other parameter consider here are size and minimum count. The minimum count of the word is 1 and the size taken for each iteration is 150. I have passed the sanitized data into the word2vec model.

Below shown is the word cloud for which I have used word cloud library to represent the important words in the dataset.



From the given word cloud diagram, since we have taken righttroll and lefttroll data we can see that the words like hillary clinton, trump, obama, white house are present.

In this we can also see words like woman, man which are the gender based words. Race related words like white people, black people, racist, white supremacy. We can also find political words like voters fraud, trump supporter

Insight:

Calculating the similarity:

After training the model with the sanitized data, we can find if there is any bias in the language.

Here, I have used the similarity function from the word2vec. Using the function `model.wv.most_similar('woman')` the words in the document similar to the word woman are:

```
[('men', 0.6932409405708313), ('man', 0.6367347836494446), ('girl', 0.6133425831794739), ('female', 0.6106760501861572), ('people', 0.6010871529579163), ('bw', 0.5691353678703308), ('feminist', 0.5631645917892456), ('child', 0.5381681323051453), ('poc', 0.5334669351577759), ('couple', 0.532174289226532)]
```

From the above figure, we can see the words related to woman are girl, man, female, people, couple, etc. And also the similarity between the word woman and other words is also shown.

Few other examples are:

The word asian, `model.wv.most_similar('asian')`

From the below figure, the word asian are more similar to african, black, femalewhite, native which are mostly related to race words.

```
[('african', 0.5412837266921997), ('black', 0.5218251347541809), ('femalewhite', 0.5149621367454529), ('poc', 0.4892129600048065), ('native', 0.487309068441391), ('average', 0.479128897190094), ('mostly', 0.4764682948589325), ('diversifies', 0.4749071002006531), ('profiled', 0.4747272729873657), ('bisexual', 0.47423771023750305)]
```

The word corrupt, `model.wv.most_similar('corrupt')`

```
[('corrupted', 0.6490533351898193), ('incompetent', 0.6010700464248657), ('corruption', 0.590725302696228), ('crook', 0.5385136008262634), ('voteantidemocratic', 0.5173730254173279), ('lying', 0.5111324191093445), ('crooked', 0.5107511281967163), ('revolutionrequired', 0.4988127648830414), ('criminal', 0.49276742339134216), ('deceitful', 0.49201250076293945)]
```

From the above picture the word corrupt are mostly related to revolutionrequired, criminal which are belong to political words.

The bias found in this are gender, political, and race.

Gender bias:

Analyzing the similarity between the words like engineering, woman, and man we can see the similarity between engineering and woman is **0.14** and the similarity between engineering and man is **0.04** we can say engineering is more biased toward woman than man.

Analyzing the other words like abuse, woman, and man the similarity between abuse and woman is **0.35** and the similarity between abuse and man is **0.20**. Here, the word abuse is more toward woman.

Race bias:

For analyzing race bias, I have taken the similarity between the word crime and white which is **0.18** and the similarity between the words crime and black is **0.2**. So the word crime is more toward black than white.

Political analysis:

Even though we have seen words like corrupt, revolutionrequired, criminal the similarity between the word trump and corrupt and the similarity between the word obama and corrupt are **0.26** which are very similar.

1.4. Question 2:

Does grouping the tweets by content similarity align with the labels (e.g., "RightTroll") that have been provided with the dataset?

For grouping the content similarity align with labels I have filtered righttrolls and lefttrolls. Also, filtered only the english words.

For grouping the tweet contents I have used tfidf vector to find the vector of the given words. After finding tfidf, the similarity should be found by taking inner product of the two row content vector. Here, I have used k-means algorithm to group the tweets based on the vector from the tfidf and converting it into array. Also, used PCA to visualize the grouping of the tweets. Since the matrix conversion required large memory space I have taken few content to show the cluster visualization in PCA.

The parameter I have used for finding the vector are preprocessor which are to sanitize the data, token and stop_words.

```
vec = TfidfVectorizer(preprocessor=preprocessing,tokenizer= textblob_tokenizer,
stop_words = 'english',norm='l1', use_idf=True)
```

In tfidf calculation, the idf is calculated as,

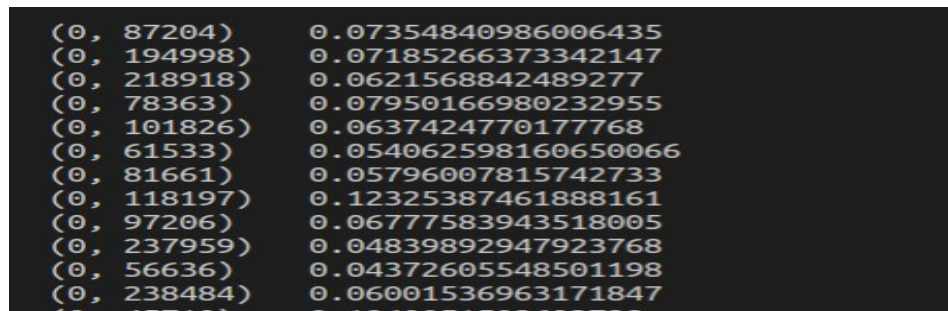
$idf = \log(\text{number of documents} / \text{number of docs contains a particular word})$. For example, Let us say the number of document is 5 and the word 'play' appears in 3 document then the idf is $\log(5/3)$.

Tf in tfidf is calculated as number of words in a document. For example, word 'play' appears twice in first document then, $tf(\text{play}, d_0) = 2$.

Now, the tfidf is calculated as $tf * idf$, therefore $2 * \log(5/3)$

The similarity for these document is calculated as $\langle d_0 \rangle * \langle d_1 \rangle$.

For the given entire dataset the calculated tfidf is,



| | |
|-------------|----------------------|
| (0, 87204) | 0.07354840986006435 |
| (0, 194998) | 0.07185266373342147 |
| (0, 218918) | 0.0621568842489277 |
| (0, 78363) | 0.07950166980232955 |
| (0, 101826) | 0.0637424770177768 |
| (0, 61533) | 0.054062598160650066 |
| (0, 81661) | 0.05796007815742733 |
| (0, 118197) | 0.12325387461888161 |
| (0, 97206) | 0.06777583943518005 |
| (0, 237959) | 0.04839892947923768 |
| (0, 56636) | 0.04372605548501198 |
| (0, 238484) | 0.06001536963171847 |

From the above figure, (0, 87204) - 0 represents the document which is the content in dataset and 87204 represents the word in the list of tokens. The float number 0.073 for (0, 87204) represents the tfidf of the single documents with particular word.

After finding the tfidf, I have calculated the similarity between the content. For this I have used cosine_similarity function. The cosine_similarity function calculate the inner dot product of the two document's tfidf vectors value and gives out the similarity value. The matrix from the cosine similarity is n^2 since each document in the dataset is multiplied with every other document to produce the similarity. In order to cluster the tweets by content I have used k means clustering. Using kmeans function and passing the similarity matrix the content are clustered according to the RightTroll and LeftTroll. The kmeans works in such a way that it calculate the centroid for the two similar values based on the number of components. Here, the components is 2 which represents

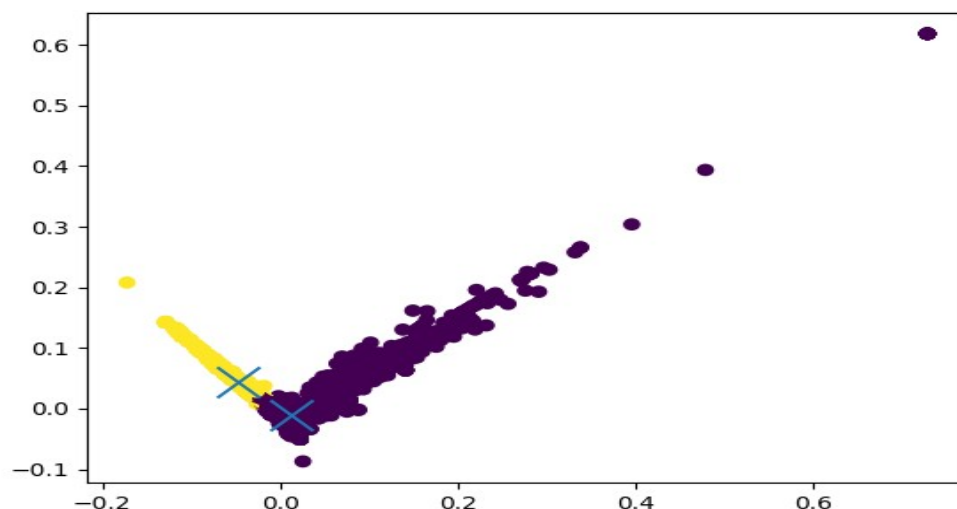
RightTroll and LeftTroll. Based on the centroid value it group the RightTrolls and LeftTrolls value that are nearer to their respective centroid.

To reduce the dimension and to visualize the grouped content I have used PCA. The PCA would reduce the dimension and plot the labels.

```
(1999, 6554) 0.13892377019907932
[[1.  0.  0.  ... 0.  0.  0.  ]
 [0.  1.  0.  ... 0.  0.  0.  ]
 [0.  0.  1.  ... 0.  0.  0.  ]
 ...
 [0.  0.  0.  ... 1.  0.04460521 0.05915469]
 [0.  0.  0.  ... 0.04460521 1.  0.18183572]
 [0.  0.  0.  ... 0.05915469 0.18183572 1.  ]]
```

The above figure shows the result of the similarity matrix.

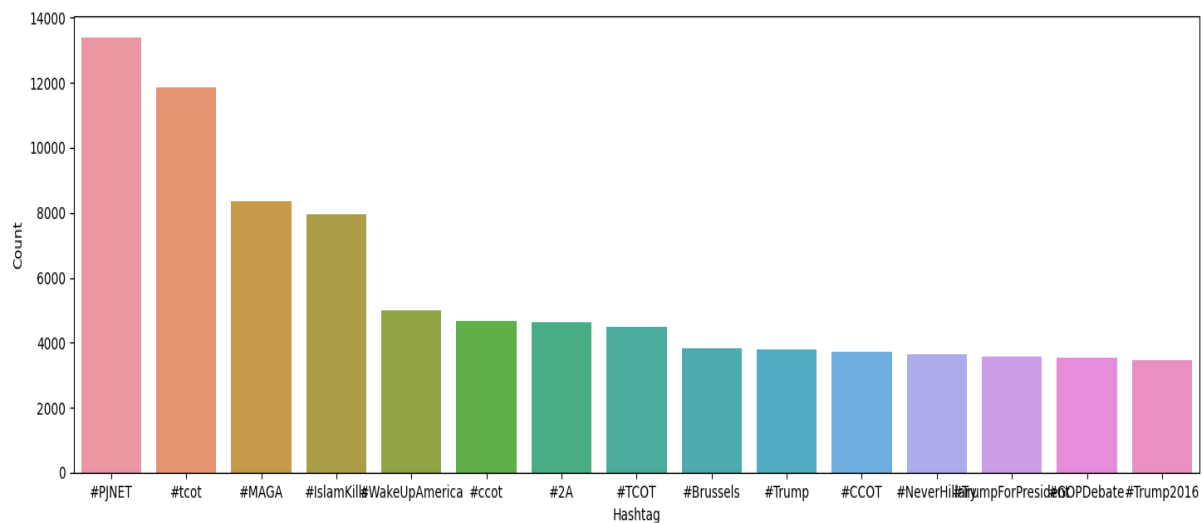
The below plot represents the clustering of RightTroll and LeftTrolls with respect to their center means which is marks in blue X.



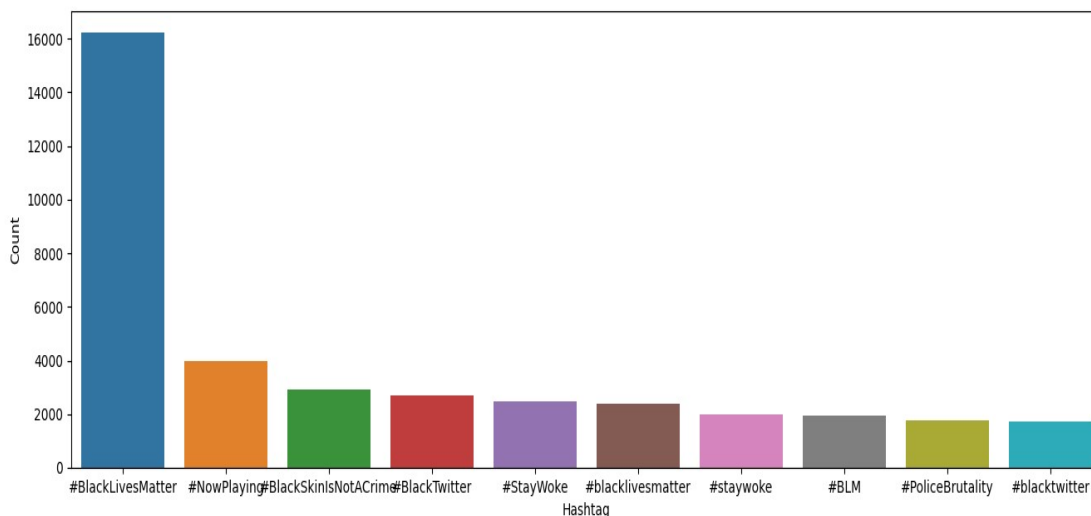
1.5. Questions 3:

Popular hashtag during 2016 election in RightTroll and LeftTroll and find the similarity between the hashtag content with the labels (RightTroll and LefttTroll).

For finding the hashtags from the content I have filtered the english content and filtered RightTroll for popular hashtag from RightTrolls content. Similarly, filtered LeftTroll for popular hashtag from LeftTroll. I have taken harvested date for analyzing popular hashtag during 2016 election. Using the function `re.findall(r"#(\w+)", I)` I have removed the hashtag from the content. Created a dictionary with index of the row and assigned the hashtag to the content index from which hashtag has been extracted. Using `nlTK.FreqDist()` function taken frequency of the hashtag and plotted the visualization between hashtag and frequency of the hashtag.



From the above visualization we can see that hashtag #MAGA (Make America Great Again) is the one of the popular hashtag. Also, we can see the hashtag that related to trump are positive like #Neverhillary, Trump2016, #WakeUpAmerica. From this we can say that make america great again hashtag could have been used to influence the people to support Trump in his election.



From above visualization we can see that hashtags like #BlackLivesMatter, #PoliceBrutality are the positive words to the lefttrolls.

During extracting the hashtag, the Id belongs to the extracted hashtag also extracted from the dataset. Created a dictionary with the hashtag and Id.

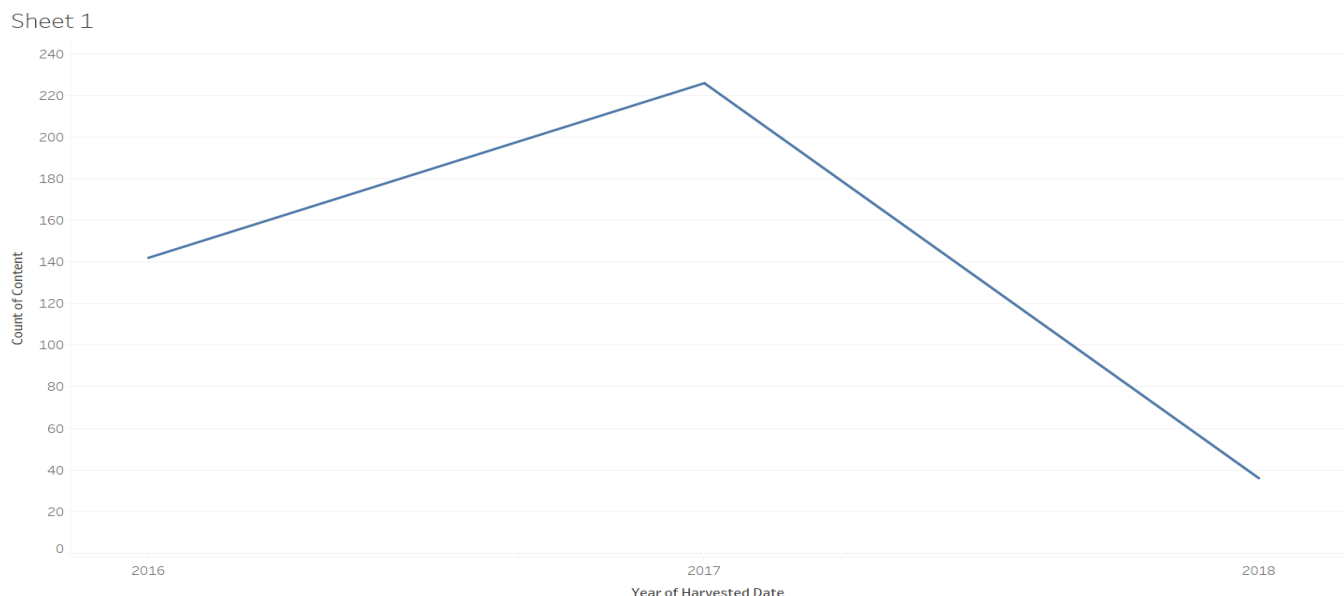
Below image shows the Id and their hashtags.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash
e'], 242878: ['#WaterDemocratized'], 242883: ['#DubNation'], 242887: ['#VirtualJustice'], 242888: ['#RobKardashian', '#DreamKardashian', '#BlacChyna'], 242892: ['#LegendsofCH'], 242895: ['#AfroTech16', '#Panel', '#Speakers', '#Blavity', '#QandA'], 242910: ['#Insecure'], 242916: ['#BlackPrivilege'], 242921: ['#VotingMatters'], 242923: ['#TheCruzShow'], 242926: ['#BlavityLituation', '#BlavityGetLit', '#BlavityFan', '#Blavity'], 242935: ['#TraceeEllisRoss', '#GlamourMOTV'], 242936: ['#CALICHRISTMAS'], 242937: ['#NASAVgUE', '#NASA'], 242938: ['#JourneyToMars'], 242940: ['#BlackAmericaPBS'], 242943: ['#EventsatHowardU', '#IraAldridgeTheatre'], 242945: ['#DXB', '#rollsoneverycorner'], 242948: ['#ThankYouObama'], 242951: ['#Supermoon'], 242955: ['#HiddenFiguresatSpelman'], 242962: ['#QueenSugar', '#GimmeSugar'], 242983: ['#Kanye'], 242990: ['#FlintWaterCrisis'], 242997: ['#harlen', '#deleontequila', '#nextlevel', '#NoMercy', '#tryit'], 243003: ['#NowPlaying'], 243004: ['#California', '#YesOn64', '#Vote'], 243005: ['#YesOn64'], 243007: ['#CaliChristmas'], 243019: ['#BlackGirlMagic'], 243024: ['#NowPlaying'], 243037: ['#MikeBrown'], 243042: ['#RevolthHonorsNas'], 243044: ['#NIpolitics', '#NIunion'], 243050: ['#watchthis', '#thenextlevel', '#tryit'], 243055: ['#MedalofFreedom', '#STEM', '#CS4All'], 243071: ['#1', '#BlackFriday'], 243075: ['#Prometheus'], 243077: ['#NowPlaying'], 243083: ['#NoDAPL'], 243085: ['#amwriting'], 243100: ['#ThanksObama'], 243106: ['#TeyanaTaylor'], 243111: ['#YoungCaliforniaRadio'], 243120: ['#realitych...]
```

From the dictionary of Id and hashtag, I have extracted Id of the the popular hashtag #maga and appended it to a list. Converted the list o Ids to dataframe and joined the dataframe of Ids to the original dataframe using inner join. Now the RightTroll which has only the #maga has be extracted.

Temporal analysis:

From the below visualization we can see that during the year 2017 the #maga was more popular than during the 2016 election.



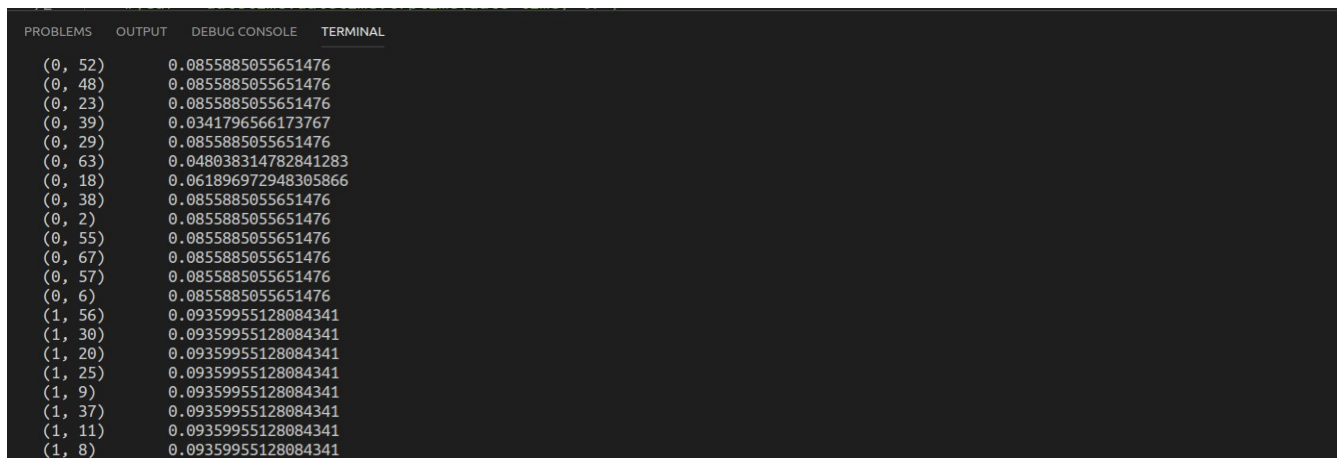
Insight:

Finding similarity between the tweets content has hashtag #maga and the original RightTroll given to find whether the hashtag tweets are belong to righttrolls :

From the extracted data, I have analyzed similarity between the righttroll. Followed the same procedure for the lefttroll.

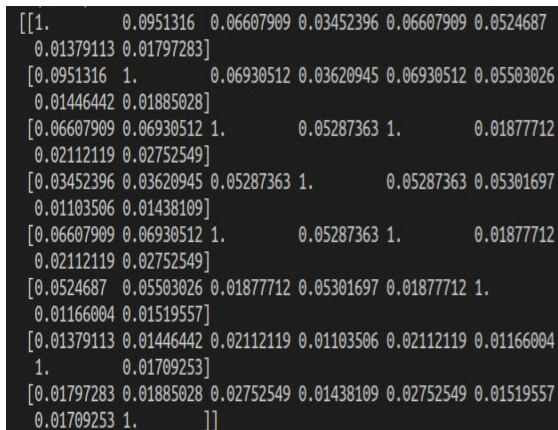
For finding similarity between original righttroll and the righttroll with #maga I have used Tfidf vector and cosine_similarity.

The below image shows the tfidf of the original righttroll and the righttroll extracted using #maga.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
(0, 52) 0.0855885055651476
(0, 48) 0.0855885055651476
(0, 23) 0.0855885055651476
(0, 39) 0.0341796566173767
(0, 29) 0.0855885055651476
(0, 63) 0.048038314782841283
(0, 18) 0.061896972948305866
(0, 38) 0.0855885055651476
(0, 2) 0.0855885055651476
(0, 55) 0.0855885055651476
(0, 67) 0.0855885055651476
(0, 57) 0.0855885055651476
(0, 6) 0.0855885055651476
(1, 56) 0.09359955128084341
(1, 30) 0.09359955128084341
(1, 20) 0.09359955128084341
(1, 25) 0.09359955128084341
(1, 9) 0.09359955128084341
(1, 37) 0.09359955128084341
(1, 11) 0.09359955128084341
(1, 8) 0.09359955128084341
```

The below image shows the similarity.



```
[[1. 0.0951316 0.06607909 0.03452396 0.06607909 0.0524687
 0.01379113 0.01797283]
 [0.0951316 1. 0.06930512 0.03620945 0.06930512 0.05503026
 0.01446442 0.01885028]
 [0.06607909 0.06930512 1. 0.05287363 1. 0.01877712
 0.02112119 0.02752549]
 [0.03452396 0.03620945 0.05287363 1. 0.05287363 0.05301697
 0.01103506 0.01438109]
 [0.06607909 0.06930512 1. 0.05287363 1. 0.01877712
 0.02112119 0.02752549]
 [0.0524687 0.05503026 0.01877712 0.05301697 0.01877712 1.
 0.01166004 0.01519557]
 [0.01379113 0.01446442 0.02112119 0.01103506 0.02112119 0.01166004
 1. 0.01709253]
 [0.01797283 0.01885028 0.02752549 0.01438109 0.02752549 0.01519557
 0.01709253 1. ]]]
```

Appendix:

Libraries used:

- Genism library used for word2Vec to analyses bias in the dataset. Used CBOW model to train the dataset
- Used nltk (Natural Language Toolkit) for Stopwords, Lemmatization, Stemming.
- Wordcloud library to visualize the data
- TfidfVectorizer from sklearn library for finding tfidf of the content
- TextBlob library for processing textual data

Code:

pre-processing:

#lowercase

```
def lower_case(sanitize):  
    data_details_lowercase = sanitize.lower()  
    return data_details_lowercase
```

#punctuation

```
def punctuation(sanitize):  
    data_details_punct = re.sub(r'^\w\s', "", sanitize)  
    return data_details_punct
```

#url

```
def url(sanitize):  
    data_details_url = re.sub(r"http\S+", "", sanitize)  
    return data_details_url
```

#semitization

```
def semitaization(sanitize):  
    ps =PorterStemmer()  
    rootWord=ps.stem(sanitize)  
    return rootWord
```

lemitization

```
def lemitization(sanitize):
    lemiti = WordNetLemmatizer()
    lem = lemiti.lemmatize(sanitize)
    return lem
```

word2vec:

```
model = Word2Vec(min_count=1, size=150, iter=5)
model.build_vocab(lemitize)
model.train(lemitize,total_examples=model.corpus_count, epochs=model.epochs)
```

```
model.wv.most_similar('woman')
model.similarity('abuse','woman')
```

Tfidf:

```
vec = TfidfVectorizer(preprocessor=preprocessing,tokenizer= textblob_tokenizer,
stop_words = 'english',norm='l1', use_idf=True)
print('*****')
print(type(makeChunks))
print('*****')
matrix = vec.fit_transform(makeChunks)
kmeans = KMeans(n_clusters=2)
y_means = kmeans.fit(matrix)
prediction = y_means.predict(matrix)
```

```
pca = PCA(n_components=2)
features = pca.fit_transform(matrix.todense())
cluster_centers = pca.transform(kmeans.cluster_centers_)
```

```
plt.scatter(features[:,0], features[:,1],c=prediction)
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], s=300, marker='x', label =
'Centroids')
plt.show()
```

Hashtag extraction:

```
def hashtag_extract(x):
    hashtags = []
    index = []
    for i,j in zip(x['content'],x['Id']):
```

```

ht = re.findall(r"#(\w+)", i)
if ht:
    ht_1 = ["#" + term for term in ht]
    hashtags.extend(ht_1)
    if apphashtag.get(j, None) is None:
        apphashtag[j] = ht_1
    else:
        apphashtag[j].extend(ht_1)

return hashtags, apphashtag

```

Creating Dictionary with Id and hashtag:

```

maga_id_list = []
for c_id, hashtag_list in app.items():
    if '#MAGA' in hashtag_list:
        maga_id_list.append(c_id)
print(maga_id_list)

df = pd.DataFrame(maga_id_list, columns=['Id'])
print(df.shape)

data_details_1 = pd.merge(data_details, df, on='Id')
print(data_details_1.shape)
makeChunks = data_details_1['content']

```

References:

- <http://arno.uvt.nl/show.cgi?fid=134009>
- <https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089>
- <https://www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysis-dataset-python/>
- <https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa>
- <https://github.com/fivethirtyeight/russian-troll-tweets>
- Sample assignment posted in brightspace