Data Examination

The first step that was taken in this assignment was to examine the data. I dropped a number of columns in order to make the data more manageable. Almost immediately I decided to concentrate on examining account types. I created a bar graph that can be seen in Figure 2 to examine the number of different tweets associated with the various account types.

```
In [1]: import pandas as pd
df = pd.read_csv("russian_troll_tweets/IRAhandle_tweets_1.csv")
df.drop(['external_author_id','language','publish_date','harvested_date'],axis=1,inplace=True)
df.drop(['tweet_id','article_url','tco1_step1','tco2_step1','tco3_step1'],axis=1,inplace=True)
df.drop(['updates','retweet','new_june_2018','alt_external_id'],axis=1,inplace=True)
df.tail()
```

Out[1]:

	author	content	region	following	followers	post_type	account_type	account_category
243886	AUSTINLOVESBEER	BREAKING: Killer avalanche sweeps three skiers...	United States	41	34	RETWEET	Right	RightTroll
243887	AUSTINLOVESBEER	Why men should support International Women's D...	United States	41	34	RETWEET	Right	RightTroll
243888	AUSTINLOVESBEER	How we can rebuild trust in a UK divided by in...	United States	41	34	RETWEET	Right	RightTroll
243889	AUSTINLOVESBEER	John Humphrys accused of patronising Angela Ra...	United States	41	34	RETWEET	Right	RightTroll
243890	AUSTINLOVESBEER	Fossilized poop found in 180-million-year-old ...	United States	41	34	RETWEET	Right	RightTroll

Figure 1: Data frame immediately after loading data and dropping columns.

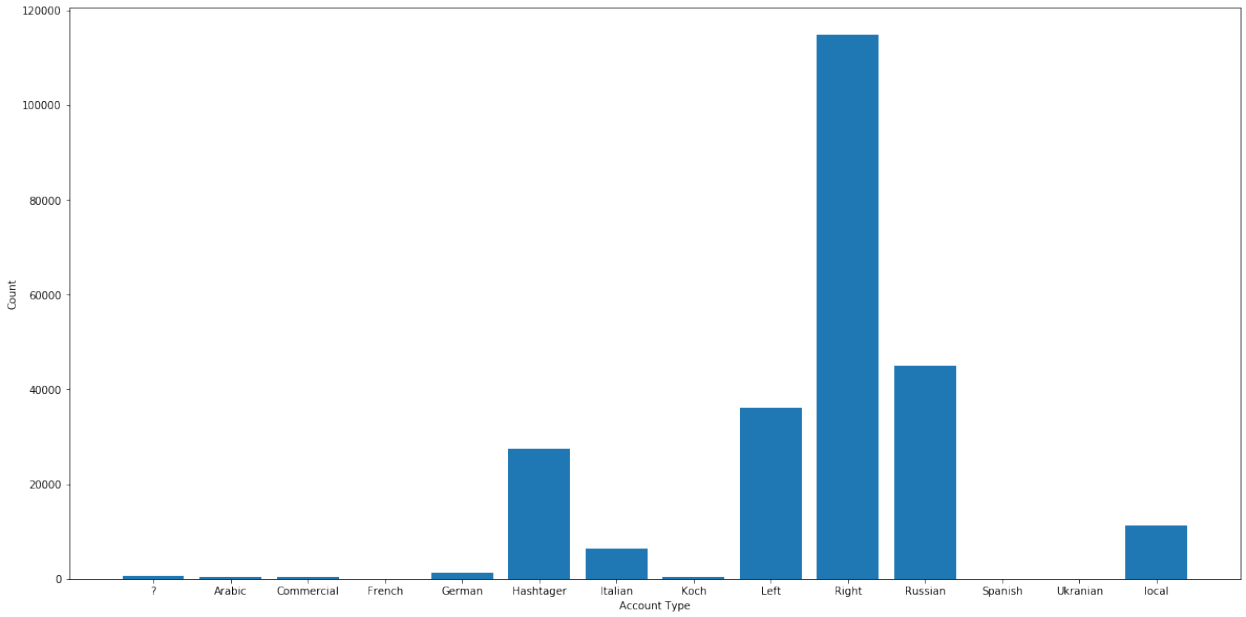

Figure 2: Bar graph examining the number of tweets associated with each account type.

Data Cleaning

These are the steps I employed associated with cleaning the data:

1. HTML Decoding – HTML decoding is used because sometimes HTML encoding is not converted to text and ends up in the text field. This will look like '&','&quot', etc. To do this I used Beautiful Soup.

2. Remove @mention – I removed the @ mention tags.
   re.sub(r'@[A-Za-z0-9]+',",df.text[343])

3. URL links – I removed all the URL links from the content column.
   re.sub('https?://[A-Za-z0-9./]+',",df.text[0])

4. UTF- Byte Order Mark – This refers to a sequence of bytes that identify a file as being encoded in UTF-8.

5. Remove hashtags – In any hashtag the '#' was removed while the text was left in the content column.
   re.sub("[^a-zA-Z]", " ", df.text[175])

6. Lower case – Capital letters were removed.
   lower_case = stripped.lower()

7. Handling negations – A negations dictionary was used. Example: {"isn't":"is not", "aren't":"are not", "wasn't":"was not", "weren't":"were not" … }

8. Removing numbers and special characters:
   neg_handled = neg_pattern.sub(**lambda** x: negations_dic[x.group()], lower_case)
   letters_only = re.sub("[^a-zA-Z]", " ", neg_handled)

9. Tokenizing and joining - This was done because white spaces were created by some of the other steps.
   words = [x for x in tok.tokenize(letters_only) if len(x) > 1] return (" ".join(words)).strip()

10. Remove null values – There were null entries that were removed.

Afterwards, I saved the file as 'clean_tweet_texts.csv' which can be found in the github repository linked with this assignment. Figure 3 is what the data frame looks like now.

```
In [29]: clean_df = pd.DataFrame(clean_tweet_texts,columns=['text'])
         clean_df['account_type'] = df.account_type
         clean_df.head()
```

Out[29]:

	text	account_type
0	we have sitting democrat us senator on trial f...	Right
1	marshawn lynch arrives to game in anti trump s...	Right
2	daughter of fallen navy sailor delivers powerf...	Right
3	just in president trump dedicates presidents c...	Right
4	respecting our national anthem standforouranthem	Right

Figure 3: clean_tweet_texts.csv

For visualization purposes I created a wordcloud. This was also done to help me understand the data. To do this, I appended all of the content together from each row. With a wordcloud you can automatically remove stop words but I chose not to for this visualization.

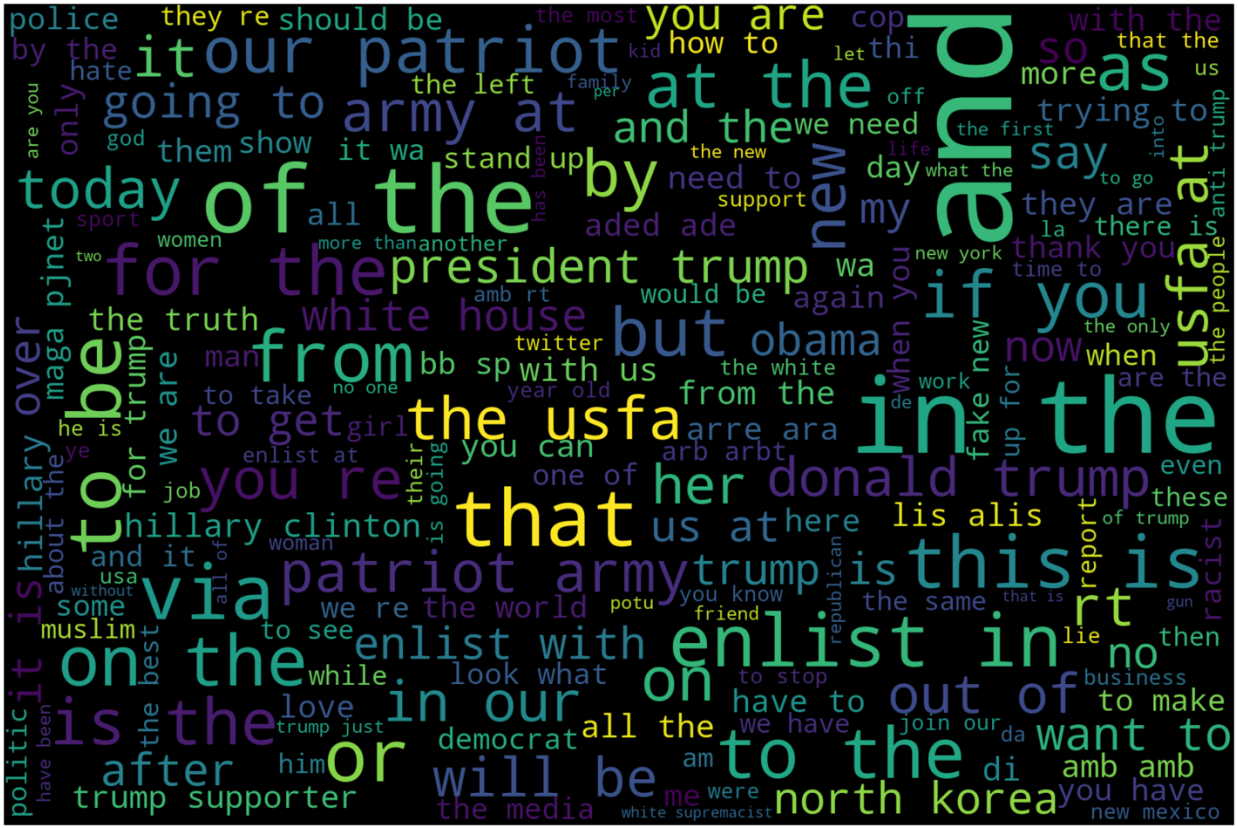

Figure 4: Wordcloud without stopwords removed.

Data Preparation 1

1. Tokenization - I wanted to compare word counts so I tokenized all of the words from the list that I created for the above wordcloud.

   ```
   from nltk.tokenize import word_tokenize
   tokenized_word=word_tokenize(all_string)
   ```

2. Remove stopwords – I used ntlk to remove stopwords. I also added a list of my own words that I felt were not important:
   ```
   stop_words = ["https", "co", "RT","ap","photo","video","do","not","need","like",
   "make", "one","via","would","want", "today","see","think","go","get", "man","take",
   "know", "say","new","let", "thank", "back","come","call","may","says","good","going",
   "really","way","still","another","got","day","news","could","year","got","even",
   "keep","show","people","thing","great","better", "without","never","first","real",
   "girl","give", "never","first", "time","now","will"] + list(STOPWORDS)
   ```

At this point I created wordclouds organized around account types. This first one show below in Figure 5 is from account types labelled left while the second one, Figure 6, is from account types labelled right.

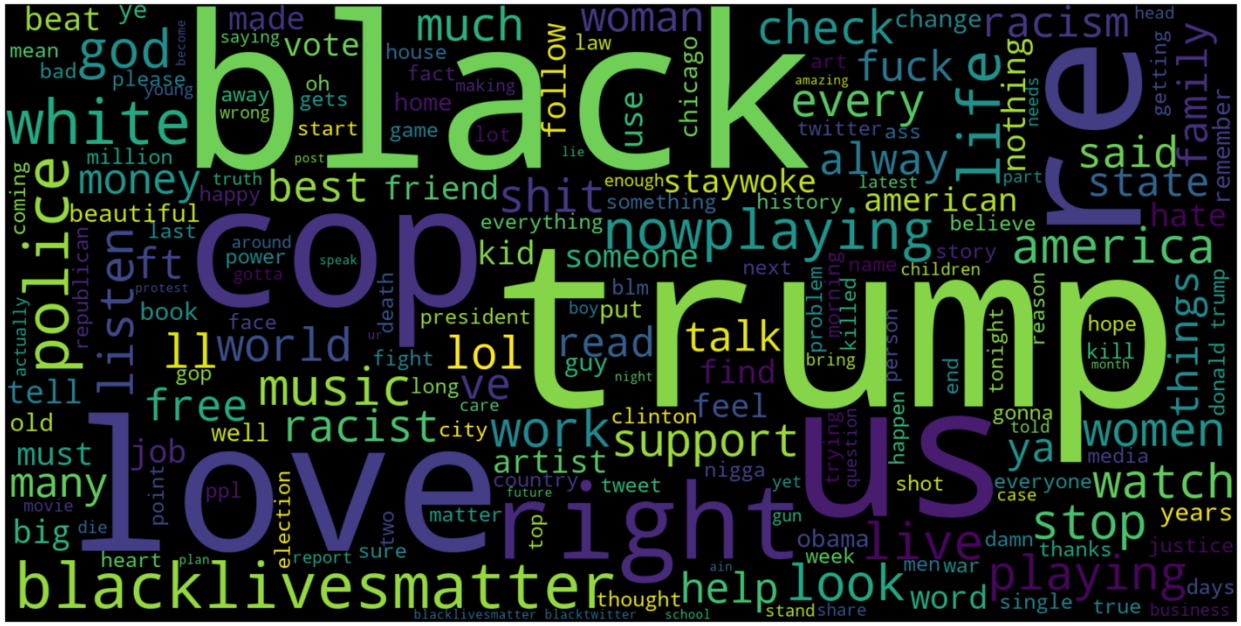

Figure 5: Wordcloud for account_type = ‘left’.

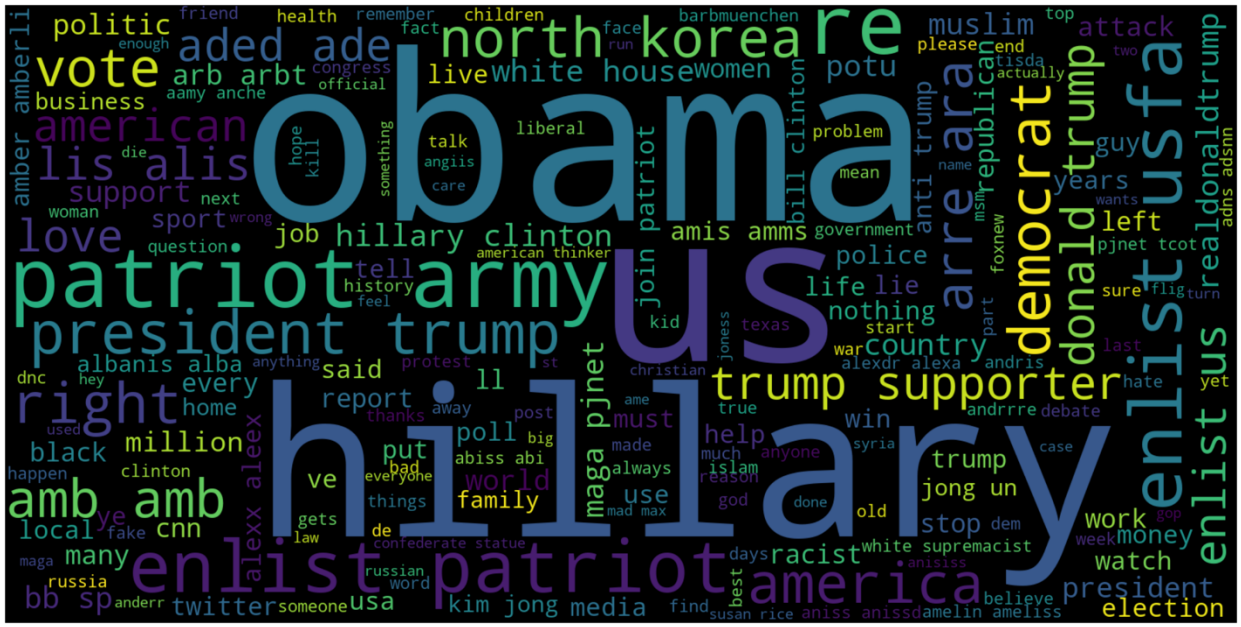

Figure 6: Wordcloud for account_type = ‘right’.

Data Preparation 2

1. Stemming – I used ntlk to remove suffixes and prefixes associated with words.
2. Finding percentages – I assembled columns for the different account types that I was looking at: right, left, Russian, hashtager, and total. I converted the total count values to percentages.

	right	left	russian	hashtager	total
trump	17.310012	4.898393	1.288245	2.227841	11.919438
rt	16.768963	2.762694	1.610306	0.340772	10.911062
obama	4.492479	0.766948	0.391074	0.527646	2.990973
hillary	4.127936	0.551419	0.184035	0.795134	2.756228
maga	3.065352	0.033589	0.023004	0.043971	1.887723
america	2.819662	1.164418	0.218542	0.630244	2.052533
president	2.801923	0.744556	0.379572	1.014987	2.024884
clinton	2.651139	0.629793	0.483092	0.465355	1.834052
media	2.558894	0.702570	0.460087	0.190539	1.750020
white	2.095899	2.507977	0.126524	0.813455	1.893145

Figure 7: Data frame showing the chance of word occurring for different account types by percent.

Finally, I created graphs based on this table sorted by different account types. Figure 8 shows word frequencies sorted by tweets labelled ‘right’. Interestingly all account types show a high amount of interest in Trump with Russia accounts showing the lowest. Figure 9 shows data for accounts labelled Russian. The words ‘iphone’, ‘apple’, and ‘google’ are all amongst the top 5 most frequent.

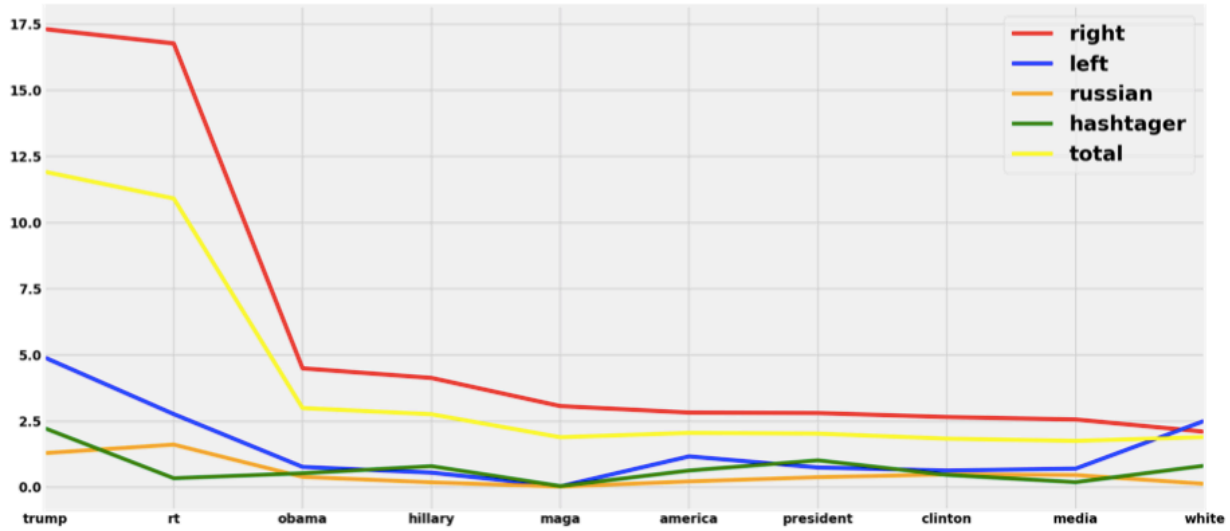

Figure 8: Top 10 word frequencies for different words sorted by account_type = ‘right’.

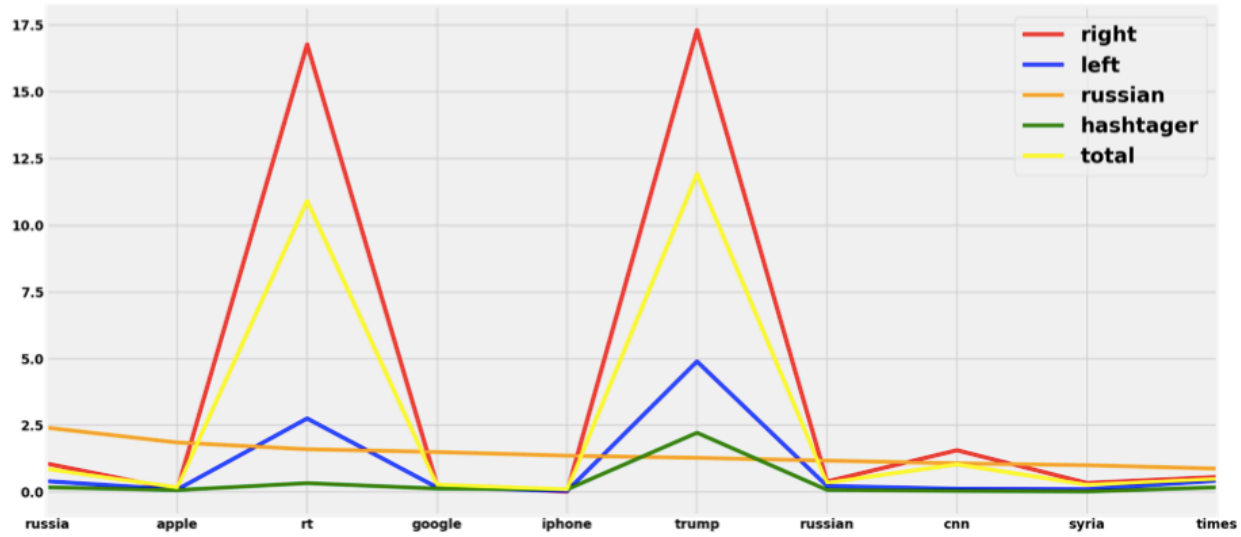

Figure 9: Top 10 word frequencies for different words sorted by account_type = 'russian'.

Word Embeddings

For this section I used the clean_tweet_texts.csv file shown in Figure 3. 'X' values are taken from the text column while 'y' values are taken from the account_type. I used Doc2Vec which is different from Word2Vec because it uses sentence vectors as well as word vectors. Word2Vec only uses word vectors. Logistic regression was then used to get scores for different Doc2Vec methods.

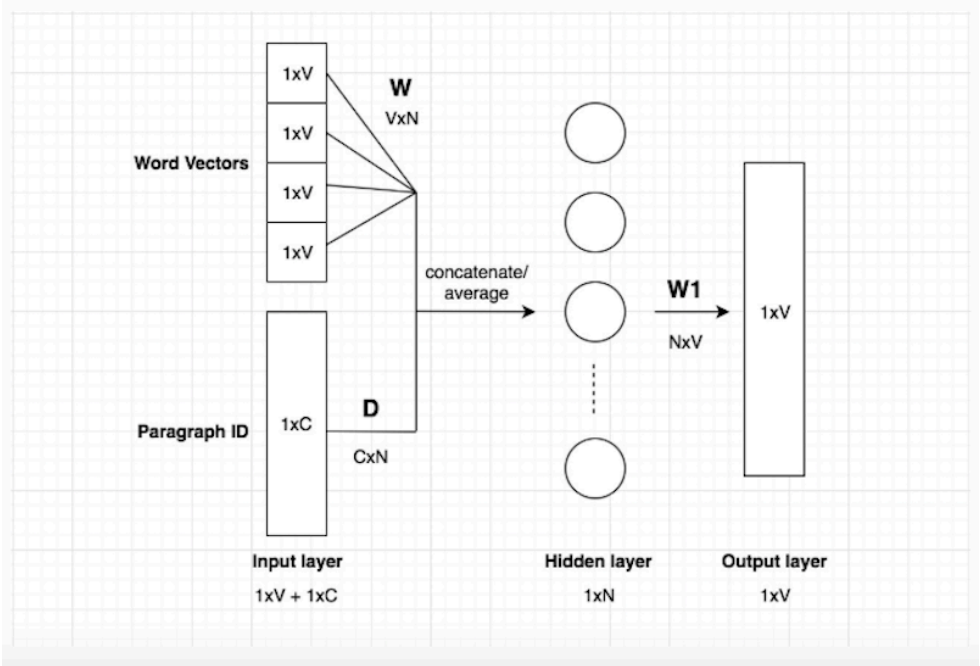

Figure 10: Doc2Vec adds one more vector to Word2Vec which is a sentence or paragraph vector.

1. Distributed Bag of Words: A score of 72.8% was recorded for this method.
2. Distributed Memory Concatenated (dmc): A score of only 56.6% was recorded for this method. I do not yet know why this score was so low.

```
In [26]: model_ug_dmc.wv.most_similar('trump')

Out[26]: [('bepositive', 0.6405110359191895),
          ('detractors', 0.6370944380760193),
          ('austerity', 0.5803809762001038),
          ('trumps', 0.5713320970535278),
          ('rainews', 0.5609710216522217),
          ('whacko', 0.5527499318122864),
          ('bure', 0.5491324663162231),
          ('kyei', 0.5484522581100464),
          ('realdonaldt', 0.547576367855072),
          ('gagop', 0.5339516997337341)]
```

Figure 11: Most similar scores collected from dmc for the word 'trump'.

3. Distributed Memory Mean (dmm): 65.2% was the score recorded.

```
cores = multiprocessing.cpu_count()
model_ug_dmm = Doc2Vec(dm=1, dm_mean=1, size=100, window=4, negative=5, min_count=2, workers=cores, alpha=0.065,
model_ug_dmm.build_vocab([x for x in tqdm(all_x_w2v)])

for epoch in range(30):
    model_ug_dmm.train(utils.shuffle([x for x in tqdm(all_x_w2v)]), total_examples=len(all_x_w2v), epochs=1)
    model_ug_dmm.alpha -= 0.002
    model_ug_dmm.min_alpha = model_ug_dmm.alpha

train_vecs_dmm = get_vectors(model_ug_dmm, x_train, 100)
validation_vecs_dmm = get_vectors(model_ug_dmm, x_validation, 100)

clf = LogisticRegression()
clf.fit(train_vecs_dmm, y_train)
clf.score(validation_vecs_dmm, y_validation)
```

Figure 12: Python code used for the dmm Doc2Vec model to get the logistic regression score.

Pre-trained vs. Custom models

For this section I imported the 'glove-twitter-25' model. I chose this model because it is taken from Twitter like the data we are using for this assignment. I realize that it is not a fully effective dataset for this report as I used Doc2Vec and the 'glove-twitter-25' dataset is built using Glove which is a newer version of Word2Vec that does not utilize sentence vectors.

```
import gensim.downloader as api
model = api.load("glove-twitter-25")
```

The benefits of using pre-trained models are fairly numerous. They are much easier to use, they are often built from very large and extensive datasets, and because of these reasons and others they are very useful for many machine learning like tasks. Transfer learning is a concept that encompasses the use of pre-trained models. Some models may take a very extensive amount of time to create and it can make a lot more sense to use one that has already been developed for the same purpose. A pre-trained model may also be more effective because it has been effectively scrutinized and checked for errors.

A drawback to pre-trained models is that it is very easy to be working with data that does not accurately fit a pre-trained model. Before I worked with the ‘glove-twitter-25’ dataset model I was working with a dataset model based on Wikipedia called ‘text8’. Figure 13 was created with the ‘text8’ data. I was not getting useful results from this model for various reasons so I found the ‘glove-twitter-25’ one shown in Figure 14.

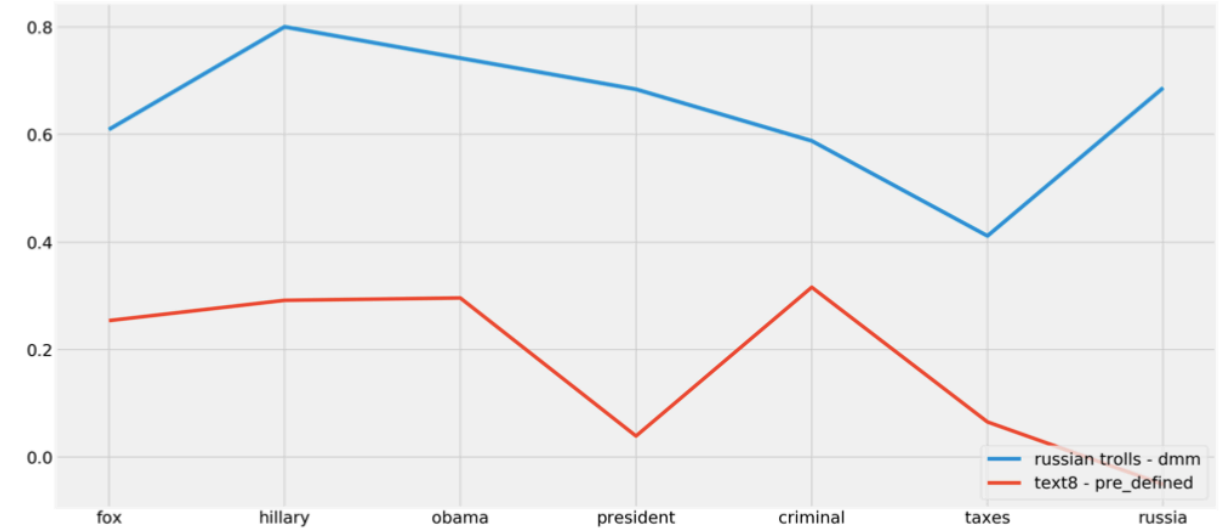

Figure 13: Similarity scores for various words when compared to ‘trump’. ‘text8’ is a model derived from Wikipedia.

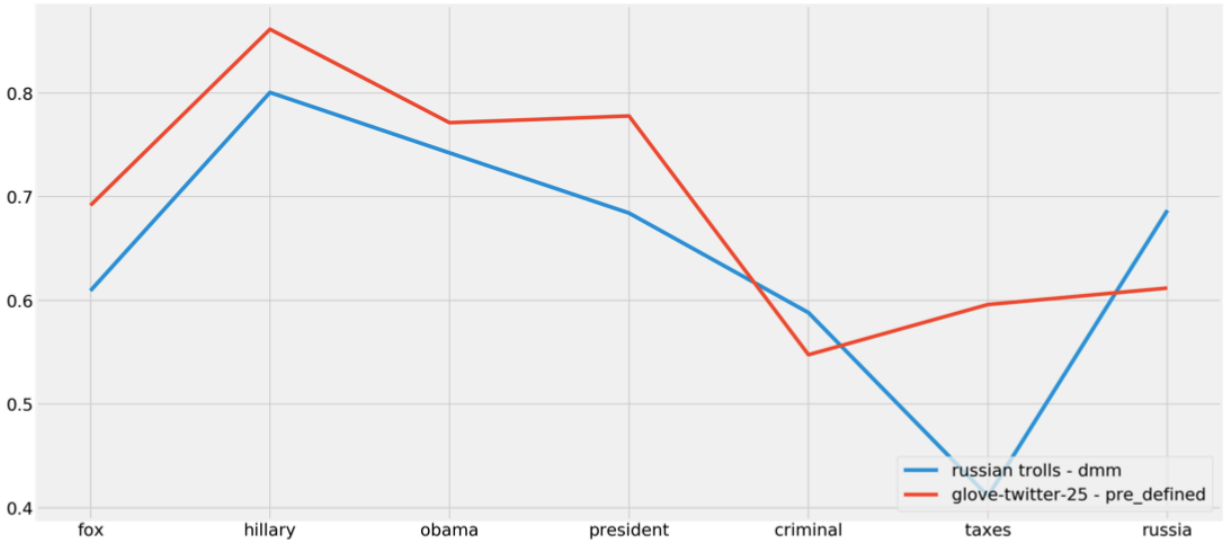

Figure 14: Similarity scores for various words when compared to ‘trump’. The blue line shows scores recorded from the Russian_trolls dataset using Doc2Vec and the distributed memory mean method. The red line is from the ‘glove-twitter-25’ pre-trained model. It could look like Russian trolls are less interested in the fact that Trump has not paid taxes in years and more interested in the connection between Trump and Russia. This is a purely hypothetical statement and would need a lot more study.

Bias

Identifying bias in a dataset can be extremely challenging. Two types of bias that are identified in the paper, “Mitigating Gender Bias in Natural Language Processing: A Literature Review” [1] are allocation and representation bias. Allocation bias refers to a situation when resources are unfairly allocated to certain groups at the expense of others. Representation bias on the other hand occurs when the social identity and representation of certain groups is being diminished.

Data should be able to be tested to detect some common biases. The bias that I am interested in studying as it relates to the Russian troll dataset that we were given for this report could fall into the category of ‘Allocation Bias’. Resources – controversial tweets – are potentially being over assigned to Russian actors. This is maybe bias that is only inherent in the dataset as it is presented and not in the data itself. Representation bias could potentially be attributed to groups and people concerned about racism. Saying that anger surrounding these issues is created by foreign actors would represent representation bias.

My hypothesis is that Russian trolls associated with a certain think tank called, ‘The Internet Research Agency’ are being unfairly connected to the vast majority of the tweets in the Russian Trolls dataset. Social media companies under a huge amount of pressure from various politicians, government agencies and traditional media companies released data after the 2016 US election relating to the issue of Russians apparently hacking the election. Social media platforms themselves were essentially blamed for allowing Trump to win which came as a huge surprise to virtually every election pundit.

The major problem with the Russian Trolls dataset and others like it is that no provenance information was given with the data. As noted in a report for the United States Senate Select Committee on Intelligence (SSCI) on the issue; “platforms didn’t include methodology for identifying the accounts; we are assuming the provenance and attribution is sound for the purposes of this analysis.” [2] Basically, we have no idea how the accounts and opinions in these datasets have been connected to Russian actors. This is a huge problem. Massive censorship is starting to take place of legitimate ideas and opinions carried out under the guise of preventing Russian or Iranian influence. Data scientists themselves are taking a huge leap in making the huge assumption that there is no bias in these datasets.

Before we start to examine this data and publish results we need to look at the data provenance chain and determine whether these various accounts can be 100% linked to Russian Trolls. I think that the fact that social media platforms refused to release this information is telling. People who question the United States, whether it has to do with foreign policy, minority rights, the right to free speech, or corporate accountability are now at risk of being silenced because their viewpoints match up with what algorithms and the general public think constitutes proof that they are either not who they say they are or they are only expressing opinions that they have bee tricked into holding.

I would propose comparing the Russian Trolls dataset to regular election-related Twitter data from the same period. In Figure 13, I attempted a start at this approach with the limited amount

of time that I have had. I personally believe that Twitter handed over random election related data and that they would be unable to prove that these tweets actually come from a Russian backed agency namely the IRA. A full investigation into issues surrounding allegations made against foreign actors is sorely needed and it can only be started with complete transparency.

I believe that Americans on the right and the left are extremely frustrated by the way that their country is being run and the anger and angst that we can detect on Twitter is a natural and organic phenomenon resulting from real things like political and corporate corruption. Remember that approximately half of all Americans refuse to vote. Many Americans are struggling with inflation and a diminishing quality of life. I think that we have seen a turning point in society which signals the end of the dominance of traditional media. Elites are struggling with this and are looking for ways to control the exchange of free ideas that has been taking place on the internet.

Looking at other types of bias with this data could also be interesting. Why for instance are some accounts labelled left while others are labelled right? I know people who don't fit neatly in either of these categories. Looking for this type of bias would entail spending a lot of time with this data. Many people who voted for Obama for two straight elections ended up voting for Trump in 2016. Why is this? Blaming Russians or blaming bias against women does not tell the whole story. Knowing exactly how much influence Russians had would help. No one can begin to do this without data provenance and being able to fully understand the allegations. With better understanding people would be better equipped to find real solutions.

References

[1] Sun, Tony, et al. "Mitigating Gender Bias in Natural Language Processing: Literature Review." *arXiv preprint arXiv:1906.08976* (2019).

[2] DiResta, Renee, et al. "The tactics & tropes of the Internet Research Agency." (2019).

CSC501 Assignment 4 – Jonathan Healy