Classifying Naughty and Nice Tweets from Twitter

STAT 432 - Project Proposal

Hulya Yigit (hyigit2)

12/11/2018

Project title

Classifying Naughty and Nice Tweets from Twitter

Group Member Info

Hulya Duygu Yigit (hyigit2)

Introduction & Literature Review

Data Overview

In the *R* programming language by @R:2018, the Twitter Data API [@TwitterAPI] can be accessed using the `rtweet` package by @CRAN:rtweet. To fulfill the project's goal of classifying tweets as being positive or negative, data was obtained by searching for politically charged tweets that contained the words "clinton" or "trump". The assembled data set then has with a mixture of original and retweet information across . The first five tweets captured are shown in Appendix Table @ref(tab:preview-tweets) and the accompanying description of each variable can be found in the subsequent Data Codebook section @ref(codebook-info).

Data Introduction & Scientific Goals

Twitter has a notable problem with users writing mean-spirited messages in the form of a "tweet." In 2015, Twitter introduced a "quality filter" that sought to suppress posts that were of "lower-quality content" as described by @WP:QualityFilter:2015. The conversation has further progressed to require more serious intervention on establishing a healthy medium for conversation described in depth by Twitter employees @Twitter:HealthyConversation:2018. As the season of well-tidings is upon us, we seek to provide a classification of textual content provided by the tweets based on whether the sentiment conveyed by tweet is positive or negative.

We searched for 18000 tweets(this limit was put so that it doesn't take too much time to fetch the data(twitter API call limit) and we can perform analysis on the local system) that contains the words Hillary, Clinton and Trump. Our final dataset is a set of tweets that contains the above mentioned keywords.

Sentiment analysis is a major branch of Natural Language Processing(NLP) these days. The usual classification techniques can be used for sentiment classification, only difference being that we can't directly apply it on words. We convert our sentences (in our case, each tweet is a sentence and the collection of all the tweets can be considered a document)into a Document Term Matrix (DTM) before applying any technique we have learnt.

Some existing Analysis on this includes:

- Twitter Sentiment Analysis Using R: This website introduces to sentiment analysis using "Syuzhet" library which has inbuilt functions for sentiment classification. We can firectly apply these functions on text. (http://dataaspirant.com/2018/03/22/twitter-sentiment-analysis-using-r/)
- Clustering on Donald Trump Tweets: This website introduces to basic concepts of NLP(creating a DTM after tokenizing and finding the term frequency) and uses clustering to classify it into positive or negative sentiments. (https://github.com/susanli2016/Data-Analysis-with-R/blob/master/Donald-Trump-Tweets.Rmd)

Proposed Analysis

Sentiment Classification

There are 2 ways to do sentiment classification:

1. Unsupervised Learning

Clustering is a method used when we don't have the training labels (Unsupervised learning), which is generally true for twitter data.

We have used k-means clustering on our DTM to cluster the tweets into positive and negative sentiments.

2. Supervised Learning

Machine Learning techniques like Decision trees, Random Forest, SVM, etc. can be used to classify the tweets data based on the sentiments.

But to use these techniques, we need to have the training labels(sentiments) for a set of tweets as they learn from already existing data and labels. To use these techniques on the tweets we fetched, data was classified into positive and negative sentiment using "Syuzhet" package.

Natural Language Processing (NLP)

Applying mathematical algorithms to (potentially very large) character data sets is a challenging prospect. Thus with some methods, we need to make it numerical.

Tokenization: Given a sentence and a defined document unit, tokenization is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation. What this means is basically that each unique word in the text can be assumed a token. This will benefit later on to calculate the frequencies of each token and convert character data sets to numerical data sets.

Bag of Words: The question we have to answer is that how can we classify documents made up of words when machine learning algorithms work on numerical data. We do this by building a numerical summary of a data set that our algorithms can manipulate. An approach that we commonly use is to identify all possible words in the documents and track the frequencies that each words occurs in data set. But, as it is expected going through this project will give us a very sparse matrix since we have so many tokens, the colums in the matrix, and not all tweets,which are the rows, include all of them at the same time. This concept, where we tokenizes documents to build these sparse matrices is called bag of words. In the bag of words model, each tweet can be mapped into a vector, where the individual elements correspond to the number of times the words appears in the document.

Stop Words: Stop words are common across text documents, their presence in a classification process likely to simply increase the noise, and to give no information in the case of classifcation. As a result, by removing stop words, we likely will produce more accurate classification. Example of stopwords is: words like 'to', 'for', 'this'

TF-IDF: Term frequency–Inverse document frequency. It is the normalized version of our matrix. Simply, we take the count of token occurrence and normalize it over the frequency with which the token occurs in all documents. In this manner, we give higher weight in the classification process to tokens that are more strongly tied to a particular label.

NDSI: Normalized Difference Sentiment Index is the difference of frequencies normalized by their sum. NDSI values are between 0 and 1 with higher values indicating greater correlation with sentiment. This can give us addtional information in terms of classification process. Moreover, after this process to increase the classification accurcy we can clean the data set more. One way to do it is we can penalize infrequent words. If the word "election" occurs just once in a positive review and not at all in any of the negative reviews, we end up with a NDSI value of 1 even though we know it's not a great predictor of sentiment. To prevent this, we add a smoothing term that penalizes infrequent words.

Clustering

We already know that we have to classify the sentiments in 2, positive and negative. Therefore, the number of clusters will be just 2. We are doing clustering analysis just to check what percentage of labels generated from Unsupervised Model match with the labels generated from the inbuilt library.

```
#################Clustering###################
fit.kmeans_tdm = kmeans(tf, centers = 2, nstart = 20)
# fit.kmeans_tdm$betweenss
labels.sentim.kmeans_tdm = as.factor(fit.kmeans_tdm$cluster)
labels.sentim.kmeans_tdm <- ifelse(labels.sentim.kmeans_tdm==1, 2, 1) ##switching the labels as 1 means

acc <- accry(labels.sentim.kmeans_tdm, labels.sentim.pack)

print(1-acc)
```

```
## [1] 0.5842352
```

The clustering analysis was done without much pre-processing as we just wanted to compare what percentage of the labels obtained from unsupervised learning (i.e. no prior information available on labels) are matching with the labels obtained from the package. ~60% of the total labels were the same. One possible reason that the k-means clustering is giving relatively lower matched might be because we have so many tokens that some of them are giving not much information or being sparse. Taking into acount them might hurt our classification.

For the supervised classification analysis, we will use the labels obtained from the package.

Methodolgy

The pre-processing of the text was done by removing the words that won't be much usefuls as tokens. We removed things like hyperlinks, punctuations, digits, unicode symbols & characters, words like RT (which means retweets).

After removing the above, we subset our data set by choosing the columns text, quoted_text, retweet_text, description, quoted_description, retweet_description. Other factors like username, number of friends, etc. are no use for us in this analysis.

We made use of 'get_sentiment' funtion from 'syuzhet' library to create the labels for the tweet data set available with us. This was done so that we get a labelled data set that was further used to train our model using the supervised learning techniques.

To apply Statstical Analysis techniques in our data set, we started by creating text Corpus and using this text Corpus to create a Document Term Matrix. We also removed the stopwords before creating the document term matrix. After creating DTM, we removed words occur very infrequently (kept words that occur in at least 0.1% of tweets) i.e the sparse terms and cast them as a dense matrix for easier analysis.

Next we find the top tokens occuring in negative tweets and positive tweets separately. This helps us identify top keywords occuring in the tweets. This is also used to calculate NDSI(Normalized Difference Sentiment Index) next. We created TF-IDF Document Term Matrix keeping only the top 'n' terms obtained based on their NDSI value(higher the better as this can be assumed to be the corelation of a word with the sentiment). We have calculated the NDSI values and created the TF-IDF matrix on our own, i.e. we didn't use in-built R packages for this. This was done in order to create better understanding of these techniques.

The final matrix/dataframe created after TF-IDF was used in classification using Clustering analysis as well as for the classification using the Supervised Machine Learning methods.

Summary statistics and data visualization

In the present analysis, tweets is used as the data set, and originally they are stored as character vectors. To be able to conduct the analysis, we should convert the data as a numerical representation.Moreover, during the analysis some of the components inside the sentences will not provide us any information such as punctuation, or uni codes. Thus, we need to do some basic pre-processing before going thorough the analysis. Moreover, under this project we will conduct some supervised learning methods thus we need to create labels for each tweets either positive and negative. The table below will give some insight about the raw data (i.e., before pre-progressing) and the data after the prepossessing with the assigned clusters after the sentiment analysis.

Table 1: Example of Five Tweets Before Pre-progressing

text
EXCLUSIVE • Trump Set to Indict Hillary Clinton & Other Deep Staters In ... https://t.co/4H3fPcC0xo via @YouTube
Donald Trump's Phone Call with Hillary Clinton https://t.co/1NeRRANlnU
Trump Takes Personal Control Over Entire US Intelligence Community After Hillary Clinton..: https://t.co/9tQM2eXx6X via @YouTube
Top Trump Official Assassinated After Linking Hillary Clinton To "Fatherland Card" Democrats Plan..: https://t.co/m7p2Yt5uff via
I liked a @YouTube video https://t.co/3YDvfprDZ2 EXCLUSIVE • Trump Set to Indict Hillary Clinton & Other Deep Staters In Coming

Table 2: Example of Five Tweets After Pre-progressing with Labels

	word.df	labels
1	EXCLUSIVETrump Set to Indict Hillary ClintonOther Deep Staters In	1
3	Donald Trumps Phone Call with Hillary Clinton	2
11	Trump Takes Personal Control Over Entire US Intelligence Community After Hillary Clinton	2
12	Top Trump Official Assassinated After Linking Hillary Clinton To Fatherland Card Democrats Planvia	1
13	I liked avideoEXCLUSIVETrump Set to Indict Hillary ClintonOther Deep Staters In Coming	1

The number of tweets with positive and negative sentiments respectively are shown below:

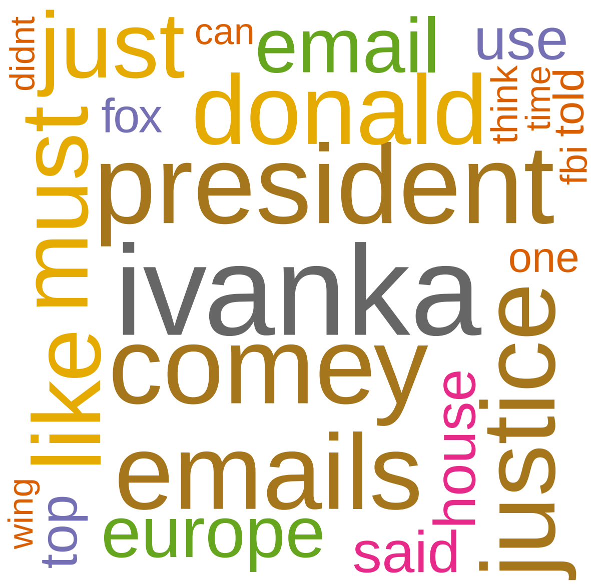
negative	positive
10681	7245

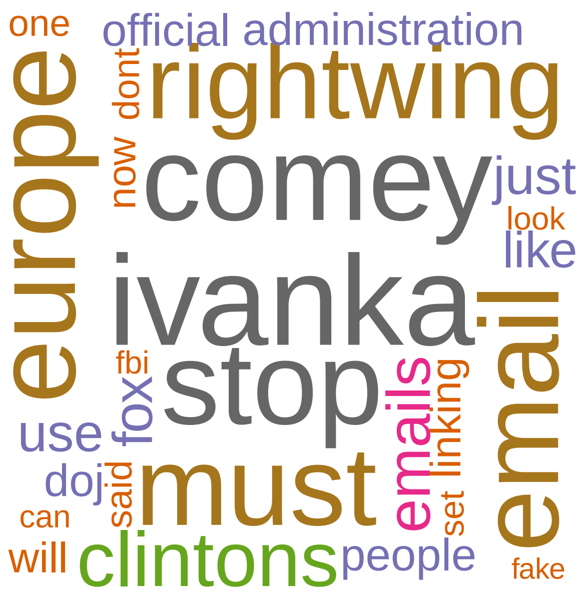
After this process, each word will be considered tokens, and the freqecies of how many time they occur among all the tweets set is recording. The table below demostrates the most 10 frequents words “tokens”.

	word	freq
4	hillary	12917
37	trump	11943
11	clinton	11595
157	ivanka	2432
144	comey	2172
219	must	1947
147	prosecute	1916
241	immigration	1814
112	email	1812
240	europe	1807

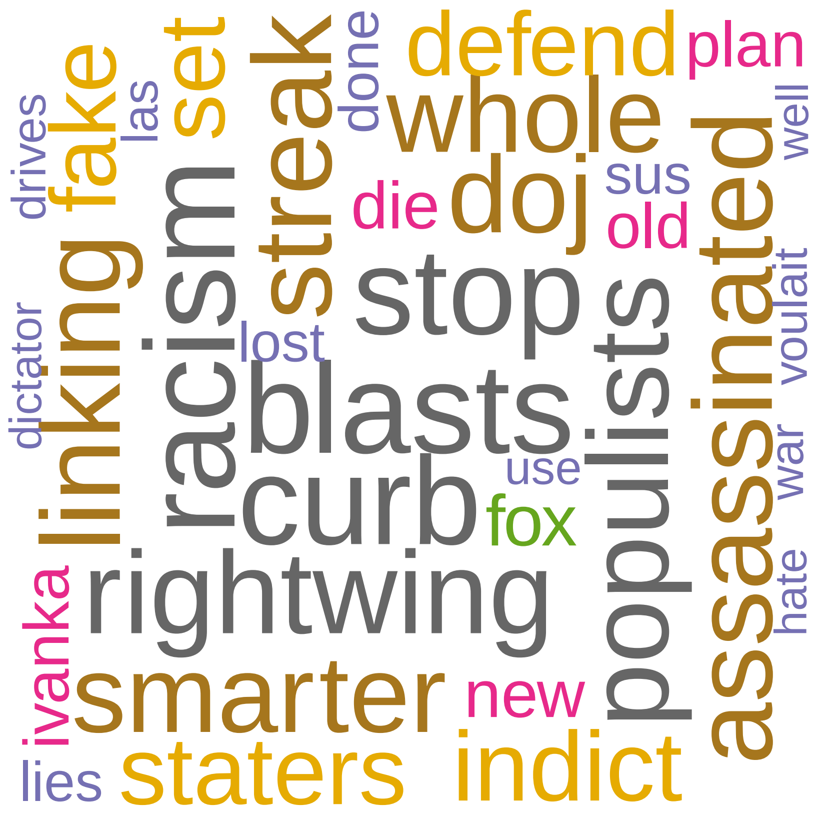
Based on the table above, the most frequent words are names of that we searched for the tweets (i.e., Hillary, Trumps, Clinton). And Ivanka, Comey etc. follows them with a sharp decrease on frequencies. We will see that the top 3 words are not of importance in the analysis based on NDSI scores.

The most commonly occuring words in Positive and Negative tweets are shown below in form of word clouds.





As can be seen from the above 2 figures, the top terms that commonly occur in both positive and negative tweets are similar. Hence we apply NDSI and again find the top important(not necessarily the most frequent) terms.



The important terms coming from NDSI scores shown above seems more relevant in determing the tweets sentiment.

Sentiment Classification

Decision Trees

```
###############trees####################
tree = rpart(labels~.,  method = "class", data = train)
predicted_values_tree <- predict(tree, test)
predicted_values_tree <- ifelse(predicted_values_tree[,1] > 0.5, 1,2)
table(predicted_values_tree,test$labels)
```

```
##
## predicted_values_tree    1    2
##                     1 3522 2107
##                     2   31  315
```

```
acc_tree <- accry(predicted_values_tree,test$labels) # accuracy rate
```

Logistic Regression

```
model_glm = glm(labels~., data=train, family = "binomial")
# saveRDS(model_glm, "glm_model.rds")
# model_glm = readRDS("glm_model.rds")
```

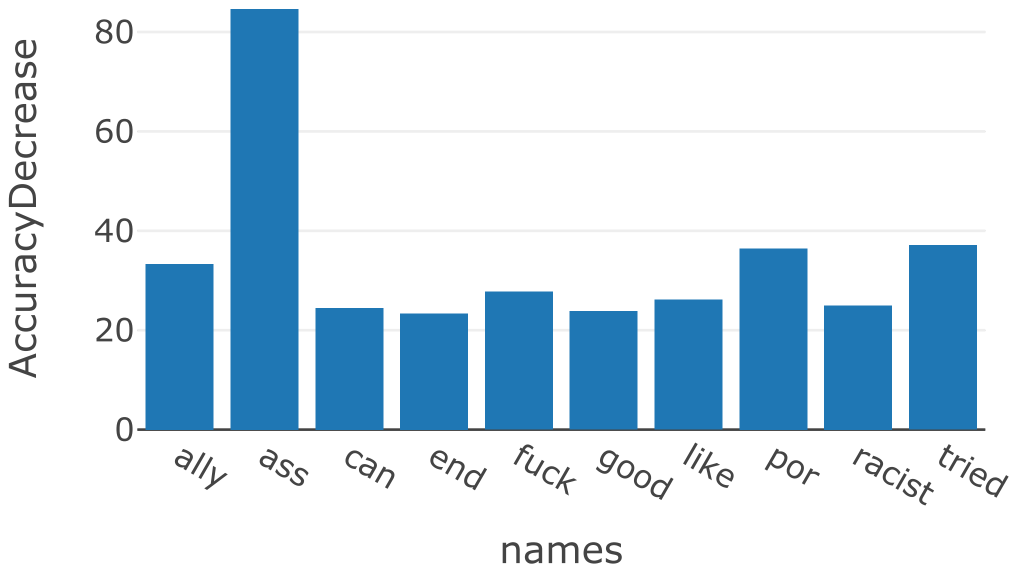
```r
predicted_values_logistic= predict(model_glm, test)
predicted_values_logistic = ifelse(predicted_values_logistic > 0, 2,1)
acc_logit <- accry(predicted_values_logistic,test$labels) # accuracy rate
```

Neural Net Model

```r
reviews.nnet = nnet(labels~., data=train, size=1, maxit=500)
# saveRDS(reviews.nnet, "nnet_model.rds")
# reviews.nnet = readRDS("nnet_model.rds")
prob.nnet= predict(reviews.nnet, test)
prob.nnet = ifelse(prob.nnet > 0.5, 2,1)
acc_nnet <- accry(prob.nnet,test$labels) # accuracy rate
```

Random Forest Model

```r
rf_model2  <- randomForest(labels ~ .,
                           data = train,
                           mtry = 20,
                           do.trace = TRUE,
                           ntree = 200,
                           importance=TRUE,
                           localImp=TRUE,
                           proximity=FALSE)
# saveRDS(rf_model2, "rf_model.rds")
# rf_model <- readRDS("rf.rds")
# predict and write output
ndsi.pred <- predict(rf_model, newdata = test)
acc_rf <- accry(ndsi.pred,test$labels) # accuracy rate
imp <- cbind.data.frame("AccuracyDecrease" = importance(rf_model)[,3][order(-importance(rf_model)[,3])]
imp$names <- row.names(imp)
```



The variable importance plot derived from Random Forest shows that 'ass' was the most important word that was used to classify the sentiments. ‘tried’, ‘por’, ‘ally’

Conclusion and discussion

The major goal of this project was to get introduced to the basic concepts of the Natural Language Processing and use the techniques we learned in the class on the text data. As we can see from the below table of classification accuracies, Neural Network didn’t give the best accuracy as was being expected as we didn’t tune the neural net model. Random Forest gave the accuracy of ~77% as we tuned the RF model. We tried multiple values of mtry and ntree but only the best case is shown. Also the run time required for Random Forest was really high.

Decision Trees	Logistic Regression	Neural Network	Random Forest
0.6421757	0.7491213	0.7474477	0.7682008

Logistic Regression also provides good accuracy of ~75% but the limitations with logistic regression is the sparsity of data. It performs better if the data is not this sparse.

Decision Trees gives us the lowest performance accuracy i.i. ~65%. The possible reason is the huge number of variables that we are using to create the model and sparsity.

In the future, we can also use the bigrams and trigrams in the bag of words for the modelling purposes. We didn’t do it because of my system limitations (the bigrams and trigrams increase the number of features by a lot).

Major Challenges faced during the project was to understand how the numerical analysis of text data is done and getting introduced to concepts like tokenization, tf-idf, vectorization, etc. Pre-processing the text data is always a challenge and I got to learn a lot from this project.

Due to Computationl limititations, we limited our tweets to a decent number and didn’t try variations in more compuation heavy methods like Neural Net. Also, to avoid high compilation time of RMD file, we saved the model results as RDS files and reused it.

Appendix