STATS133 Final Project

Nick Rodriguez, Navya Putta, Anna Yea Jung and Bret Hart

December 12, 2015

A Sentiment Analysis of Twitter Text

Purpose of the Project

The question we wanted to answer is simple. How do people feel about certain fast food establishments? We attempted to answer this question by taking a look at the tweets that people messaged to certain resturants' Twitter accounts. The restaurants we looked at were **In-n-Out Burger**, **McDonald's**, **Ben & Jerry's** and **Wendy's**.

Text has always been a rich source of data, and we feel that all analysts should become familiar with it if they are not already. So, this project was fantastic practice to tackle text analysis and basics mining methods.

In order to answer our question, we had to discover how to measure people's sentiments. The team ended up utilizing useful packages in order to perform the desired analyses.

Data Extraction

The data came from many sources that were essentially pipelined through the **twitteR** package. We collected the tweets that were messaged to a certain restaurant's Twitter account.

Here is an example query that we would send in a Twitter session in R:

```
> McDsample <- searchTwitter(searchString = "to:McDonalds" , n = 200, lang = "en")
```

However, to interact with Twitter, you'll need to go through the authentication process. There are directions to do this in the *DataAcquisition.R* file which can be found in the subdirectory, **DataAcquisition**, of the **RawData** directory. When it comes to the actual queries that someone may wish to submit, we found the Twitter Search API page very helpful in creating our own queries.

A Side Note

If you are interested in Twitter data, but would like to obtain it by other means then we suggest you do it through Python. UC Berkeley's D-Lab possesses a great walkthrough on how to interact with the Twitter API in Python. However, either method you choose, you will need to set up a Twitter Developer's account.

Cleaning the Raw Data

Our data came in the form of character vectors. With that being said, we worked closely with the regular expressions in order to clean the tweets. We had to remove retweet instances, punctuation and other unneeded strings like emoticons. The process of trimming emoticons was interesting becaused we used the *iconv()* function to convert a character vector's encoding from "latin1" to "ASCII". This conversion as a result trimmed away the syntax.

Furthermore, we took advantage of R's tendency to vectorize data, which makes our cleaning script very powerful for Twitter text data. With the help of a regular expression cheat sheet found online, we created a function that deleted both leading and trailing white spaces that were of excess.

However, there is still a challenge that taunts our script. We have yet to address words that possess repeating characters such as *sweeeeeeeeet*. The difficult part is because some words are correctly spelled with repeating characters like *free*. There is also a problem of misspelled words and the annoying tendency for people to put many words together.

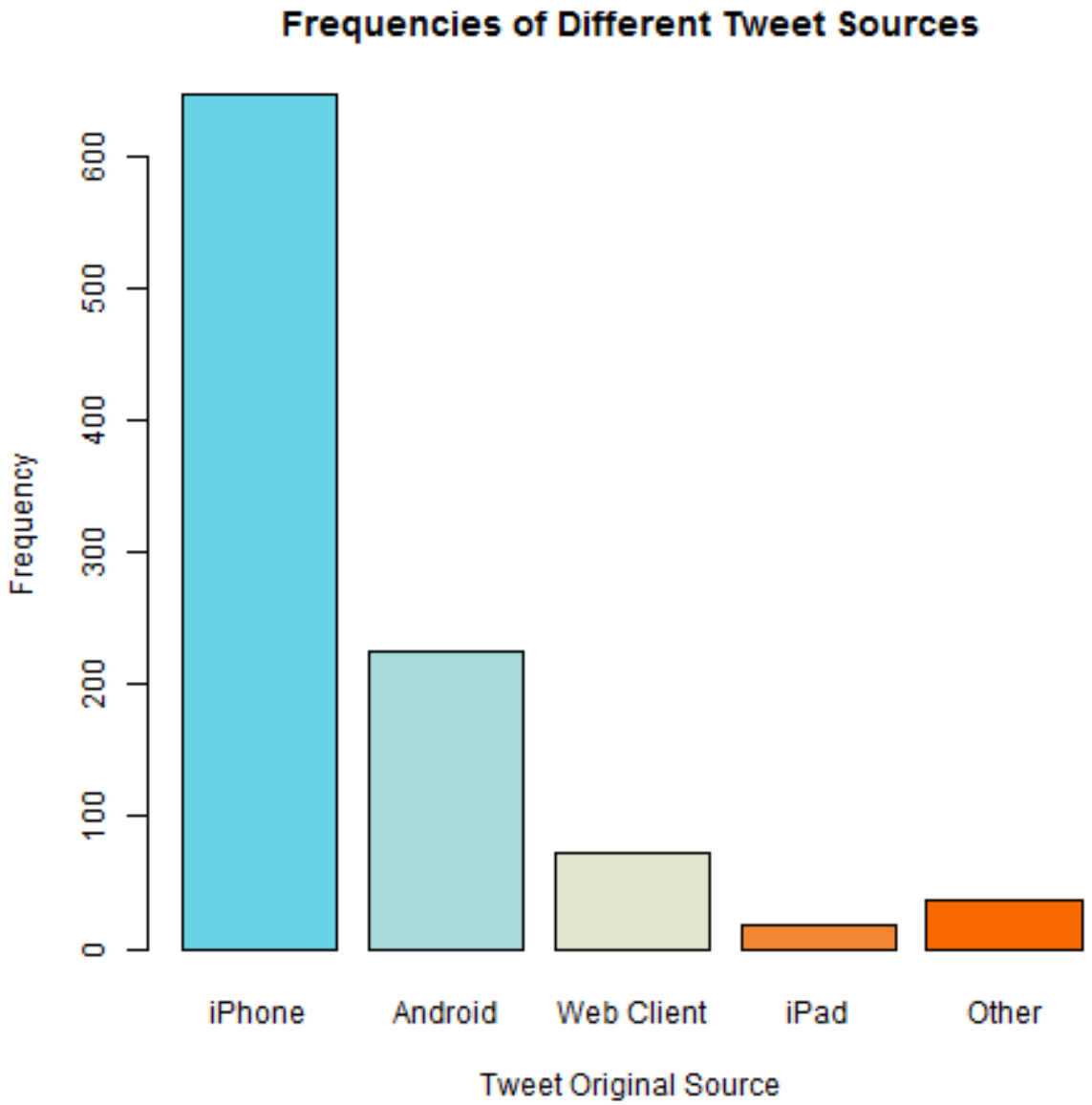
Transforming the Data Further

After the cleanup, we possessed vectors of character type data. We needed to conduct sentiment analysis to actually answer our question. The **sentiment** package provided us with the methods we needed to get this data. However, the **sentiment** package is no longer supported on CRAN, so you must download the source code. Here are the commands in order to achieve this.

```
> install_url("http://cran.r-project.org/src/contrib/Archive/Rstem/Rstem_0.4-1.tar.gz")
> install_url("http://cran.r-project.org/src/contrib/Archive/sentiment/sentiment_0.2.tar.gz")
```

A Quick Analysis Before We Go On

If you avert your attention to the **WendysExploratory.Rmd** in the **Scripts** directory, at the top of the script, we took a look at the mobile devices that tweets were sent from. A cool fact we found out, that actually shouldn't be of too much surprise, is that most devices used are iPhones. Follow up on the **Wendy's** related scripts for further details. Here is a nice plot that shows this. Again, this is only for **Wendy's** tweets.



Classifying Tweets By Emotion

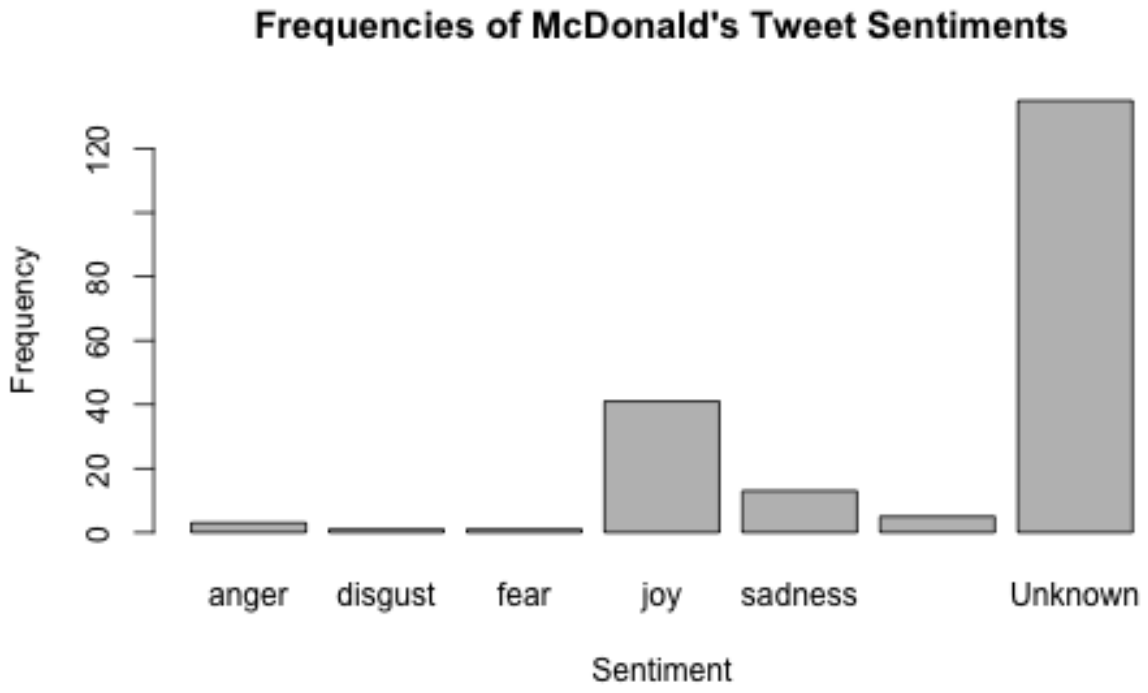
In order to classify tweets under a certain sentiment, we used the *classify_emotion()* function. The function essentially uses a naive Bayes classifier that is trained on an emotion lexicon from Carlo Strapparava and Alessandro Valitutti. The emotions are the following: **anger**, **disgust**, **joy**, **sadness**, **surprise** and **fear**. If a string cannot be classified, we classified it as **Unknown**. So, first we would load a text file that carried some clean data. Next we would implement the method of on all the tweets in that file.

```
> McDtext <- readLines("./CleanData/CleanMcDTweets.txt")
> McDemotions <- classify_emotion(McDtext, algorithm = "bayes", prior = 1)
> head(McDemotions)

   ANGER              DISGUST            FEAR               JOY
```

```
[1,] "1.46871776464786" "3.09234031207392" "2.06783599555953" "1.02547755260094"
[2,] "1.46871776464786" "3.09234031207392" "2.06783599555953" "1.02547755260094"
[3,] "1.46871776464786" "3.09234031207392" "2.06783599555953" "1.02547755260094"
[4,] "1.46871776464786" "3.09234031207392" "2.06783599555953" "1.02547755260094"
[5,] "1.46871776464786" "3.09234031207392" "2.06783599555953" "1.02547755260094"
[6,] "1.46871776464786" "3.09234031207392" "2.06783599555953" "1.02547755260094"
     SADNESS           SURPRISE          BEST_FIT
[1,] "1.7277074477352" "2.78695866252273" NA
[2,] "1.7277074477352" "2.78695866252273" NA
[3,] "1.7277074477352" "2.78695866252273" NA
[4,] "1.7277074477352" "2.78695866252273" NA
[5,] "1.7277074477352" "2.78695866252273" NA
[6,] "1.7277074477352" "2.78695866252273" NA
```

If you take a look at the output of *classify_emotion()*, you can see that it creates a data frame. The values represent absolulte log likelihoods, which signifies that the higher the value then the greater likelihood that tweet is classified in that sentiment.

```
> McDEmobestfit <- McDemotions[ , 7]
> McDEmobestfit[is.na(McDEmobestfit)] <- "Unknown"
> McDemofreq <- table(McDEmobestfit)
> barplot(McDemofreq)
```



We also took a closer look at all frequency of sentiments for a given restaruant's set of tweets. The plot above depicts the barplot of sentiment frequencies for McDonald's.

Classifying Tweets by Polarity

The **sentiments** package also provided us with a *classify_polarity()* function that will categorize a string as **positive**, **negative** or **neutral**. The process is the same as we did previously to get the emotions.

```
> McDPolar <- classify_polarity(McDtext, algorithm = "bayes")
> McDPolbestfit <- McDPolar[ , 4]
> McDSentiments <- data.frame(text = McDtext,
                              Emotion = McDEmobestfit,
                              Polarity = McDPolbestfit,
                              stringsAsFactors = FALSE)
> tail(McDSentiments)
```

```
                                                    text Emotion Polarity
194                          youre breakfasts are heart eyes     joy positive
195 single fuc it by  prod by  cutitfreestyle listen retweet Unknown positive
196                                          i got school Unknown positive
197                                            yes please     joy positive
198                                strahan cardigan shadyside Unknown positive
199   youre so welcome  your chicken biscuits are to die for Unknown positive
```

After creating a data frame, that holds both Emotion and Polarity classifications and the corresponding text, we could ask some basic questions.

```
#How many tweets are classified with a joyful emotion?
> length(McDSentiments$Emotion[McDSentiments$Emotion == "joy"])
[1] 41

#How many entries are classified with a positive polarity?
> length(McDSentiments$Polarity[McDSentiments$Polarity == "positive"])
[1] 119
```

More Exploratory Data Analysis

Next, we wanted to be able to label words of a tweet with its corresponding parts of speech (POS). So installed and loaded the **openNLP** package and cite the tagPOS function. We had to search for this function because it might not have been a built-in function of the natural language processing packages.

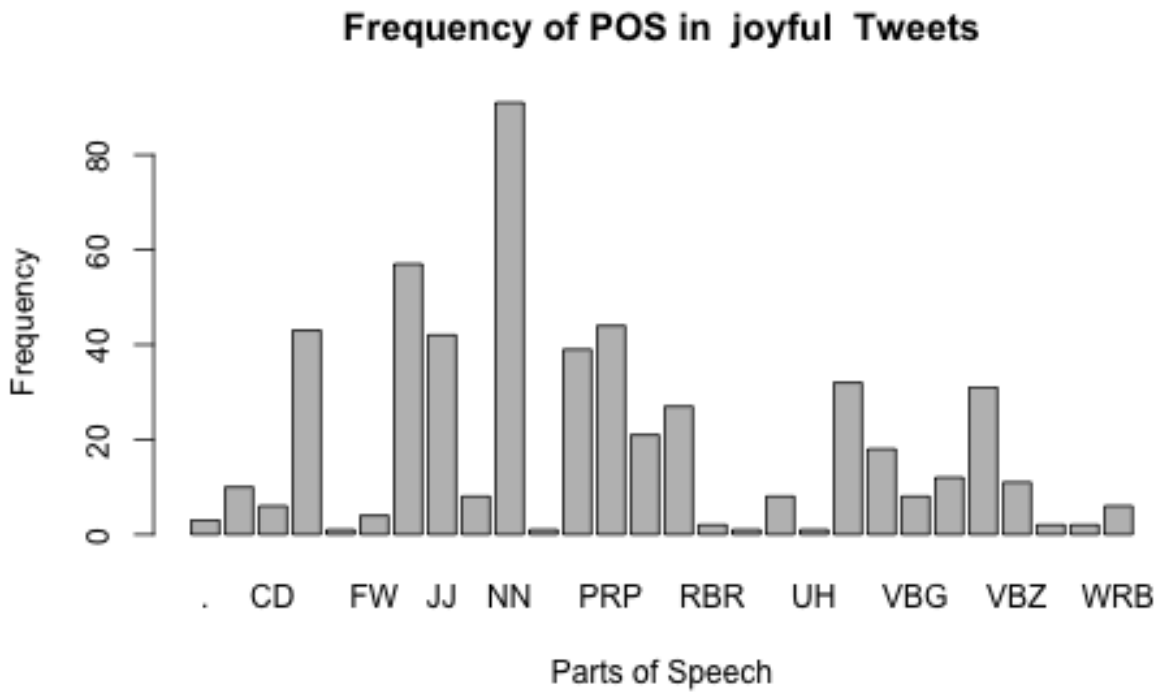
This part of the analysis was especially annoying because applying the *tagPOS()* function to many character vectors was hard on the memory. I recommend to the following if you want to try out this type of analysis: (1) try relieving the R environment of unneeded values and data (2) segment the character vector in to smaller chunks like in this example below.

WARNING: We had another issue with the *tagPOS()* function, where we it would fall prey to a *cannot coerce into data frame* error. There are some actions that may have brought about this error. I suggest you only load the following packages and only once in one R session: **sentiment**, **openNLP**, and **stringr**. The packages were also loaded in that order. Keep in mind that **sentiment** also loads the **NLP** packages which will be needed. You may want to try loading the **ggplot2** package before you load the three previously mentioned packages. If **ggplot2** is loaded after, the "annotate" object is masked from **NLP** which may be the source of the problem.

However, if successfully implemented, *tagPOS()* will create a data frame with two rows and the number of columns depends on the length of your character vector. The row that was of interested is labeled as "POStags" which attaches the POS to a word in the string. We created a function, *posFreqPlot()*, that would

(1) parse out the POS which were separated by a backslash character, (2) accumulate all the POS and (3) plot the frequency of POS. The input for this function is actually the data frame that we would receive from *tagPOS()*.

```
> joyfulTweets <- McDSentiments$text[McDSentiments$Emotion == "joy"]
> joyfulTags1to20 <- sapply(joyfulTweets[1:20], tagPOS)
> joyfulTags21to41 <- sapply(joyfulTweets[21:41], tagPOS)
> joyfulTagsDF <- cbind(joyfulTags1to20, joyfulTags21to41)
> posFreqPlot(joyfulTagsDF)
```



If you're interested, you can look at what the POS codes signify by going to this link.

Taking a Closer Look at the Words

Other analyses we dove into were to plot frequencies of actual words. The process was inspired by Basics of Text Mining from RStudio. First, we tried to create the actual frequency table.

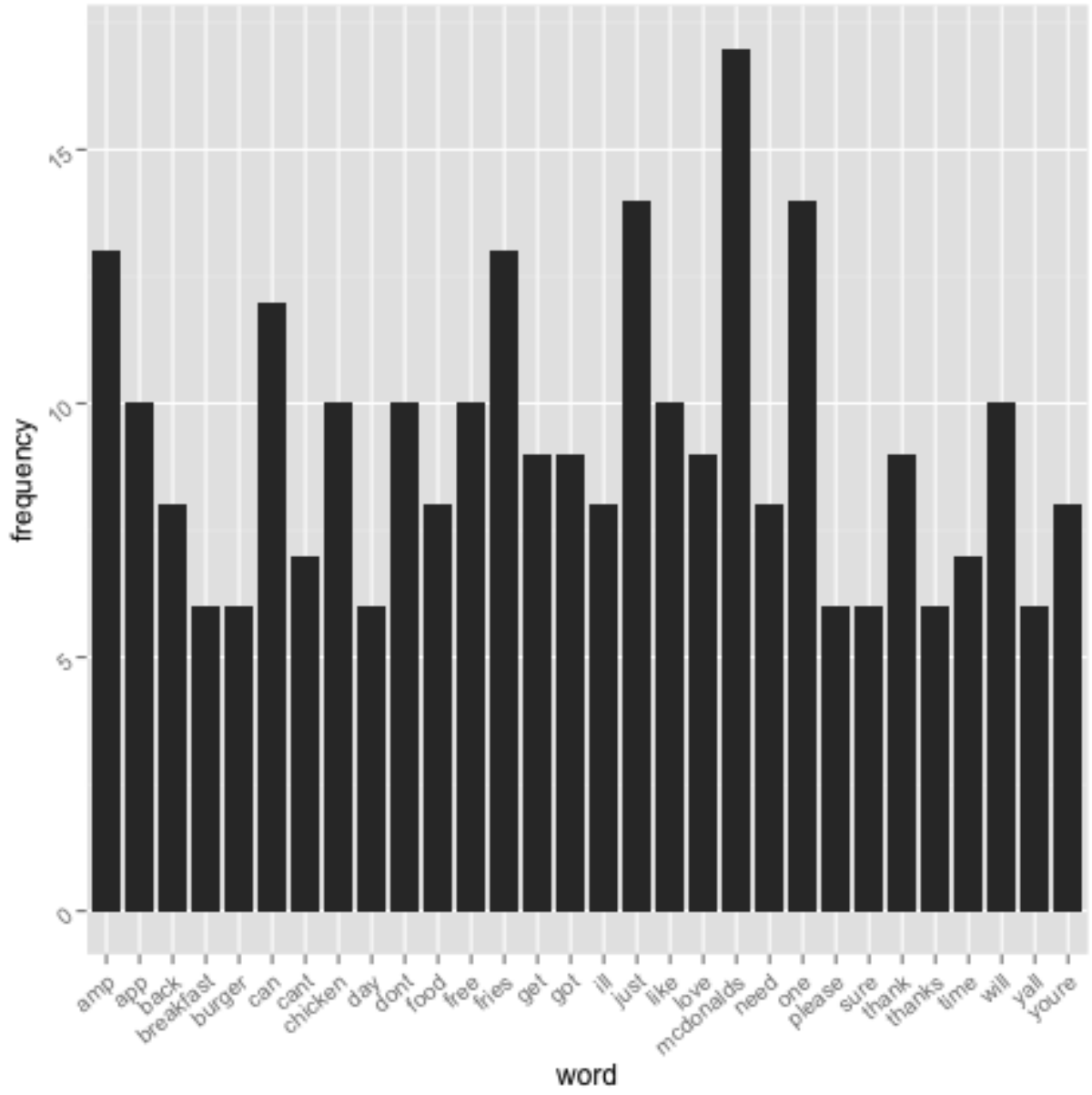
Note: the **sentiment** package already loads the **tm** package for you.

```
#In order to do this, I want to create a list of the files I have in a certain #directory. Of course, w
> McDCorpus <- Corpus(DirSource(dirname(path = "./CleanData/CleanMcDTweets.txt")))

#I attempt to get rid of frequent words that are not of interest (ie. "and", "a")
> McDCorpus <- tm_map(McDCorpus, removeWords, stopwords("english"))

#Now I create a Document Term Matrix of only part of the list I created earlier.
#I use the first index only because that corresponds to the CleanMcDTweets.txt #file.
> McDDtm <- DocumentTermMatrix(McDCorpus[1])
```

```
#Next, I create a table of word frequencies
> freqDF <- colSums(as.matrix(McDDtm))
> head(freqDF)
mcdonalds      just       one       amp     fries       can
       17        14        14        13        13        12

> tail(freqDF)
                 yorktown                    youll                    young
                        1                        1                        1
                youngest yourcommericalsfoolnoone                    youve
                        1                        1                        1

#Here are the least frequent words
> freqDF[head(order(freqDF))]
 accents accident     ache      act  address     addy
       1        1        1        1        1        1

#The most frequent words
> freqDF[tail(order(freqDF))]
      can       amp     fries      just       one mcdonalds
       12        13        13        14        14        17
```

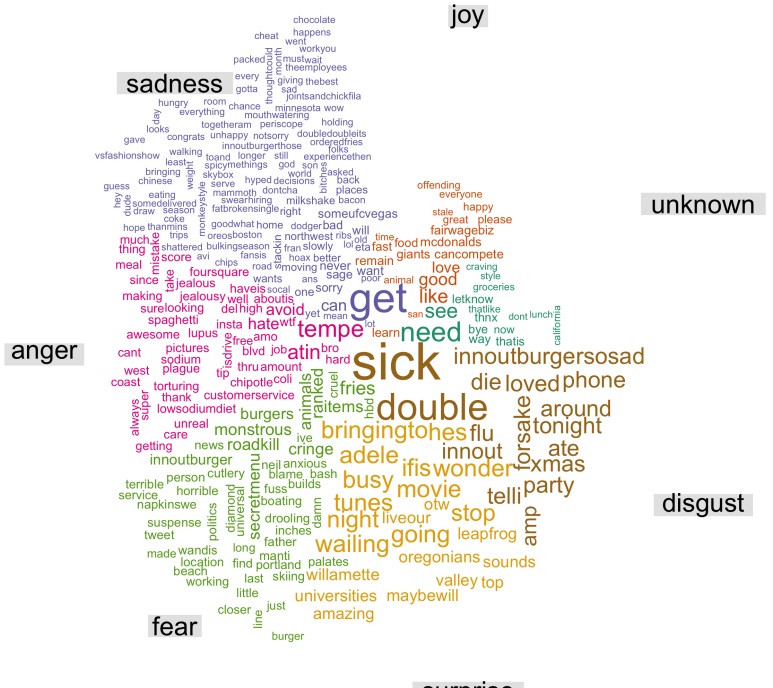
Now, we attempt to plot the frequencies. We will plot words that have a frequency of at least 5.

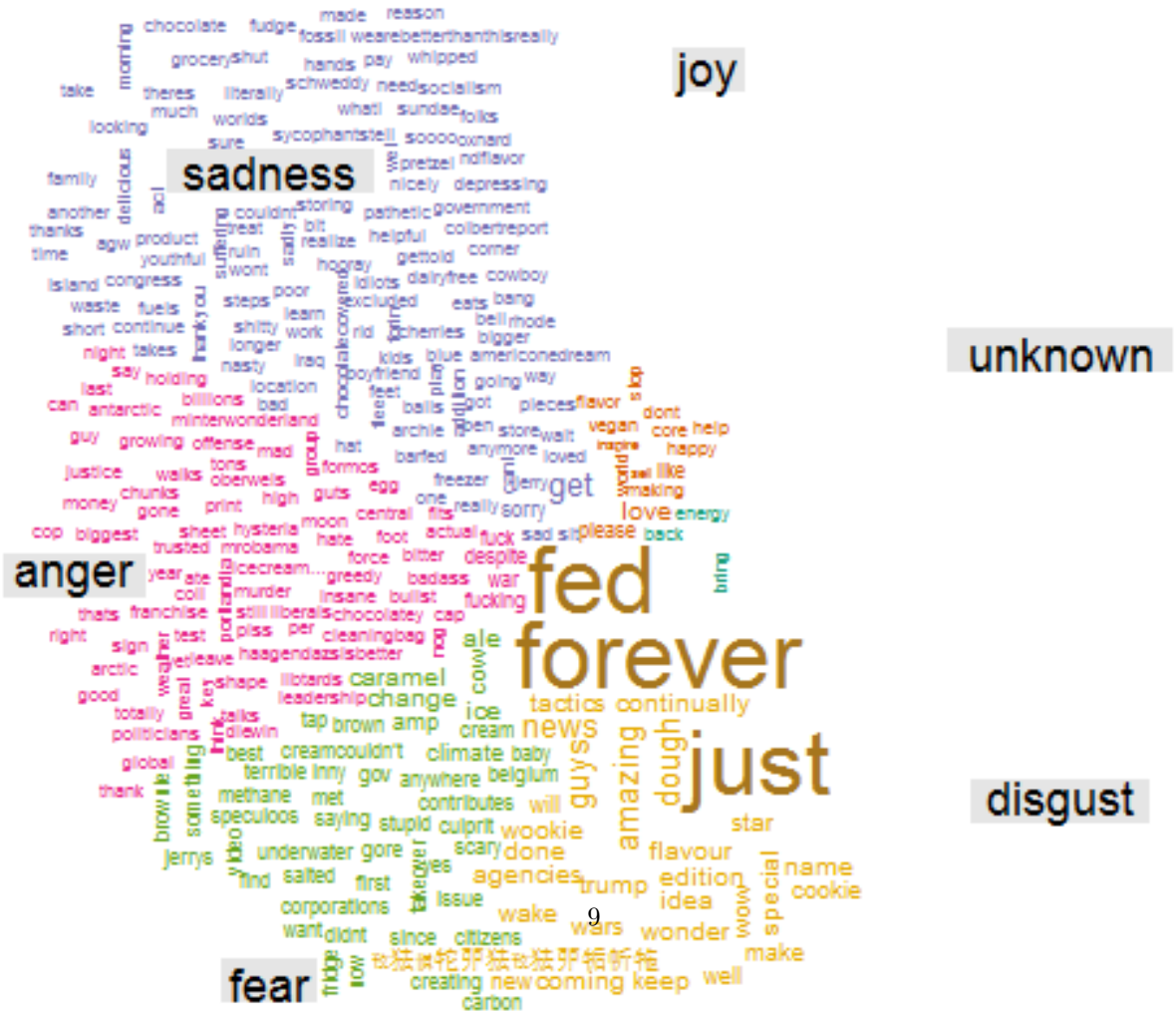
```
#First prepare the data frame for ggplot
> freqDF <- sort(colSums(as.matrix(McDDtm)), decreasing = TRUE)
> plotFreqDF <- data.frame(word = names(freqDF), frequency = freqDF)
> head(plotFreqDF)
               word frequency
mcdonalds mcdonalds        17
just           just        14
one             one        14
amp             amp        13
fries         fries        13
can             can        12

> McDWordplot <- ggplot(subset(plotFreqDF, frequency > 5), aes(word, frequency))
> McDWordplot <- McDWordplot + geom_bar(stat = "identity")
> McDWordplot <- McDWordplot + theme(axis.text =
                                     element_text(angle = 40, hjust = 1))
> McDWordplot
```



Visualizations

Another exciting part of the report was to attempt creating word clouds. The aim was to separate words by sentiments and plot them in groups. As a result, we have a nice overview of the words and are able to compare. Below you will find word clouds of **In-n-Out Burger** and **Ben & Jerry's**, respectively.

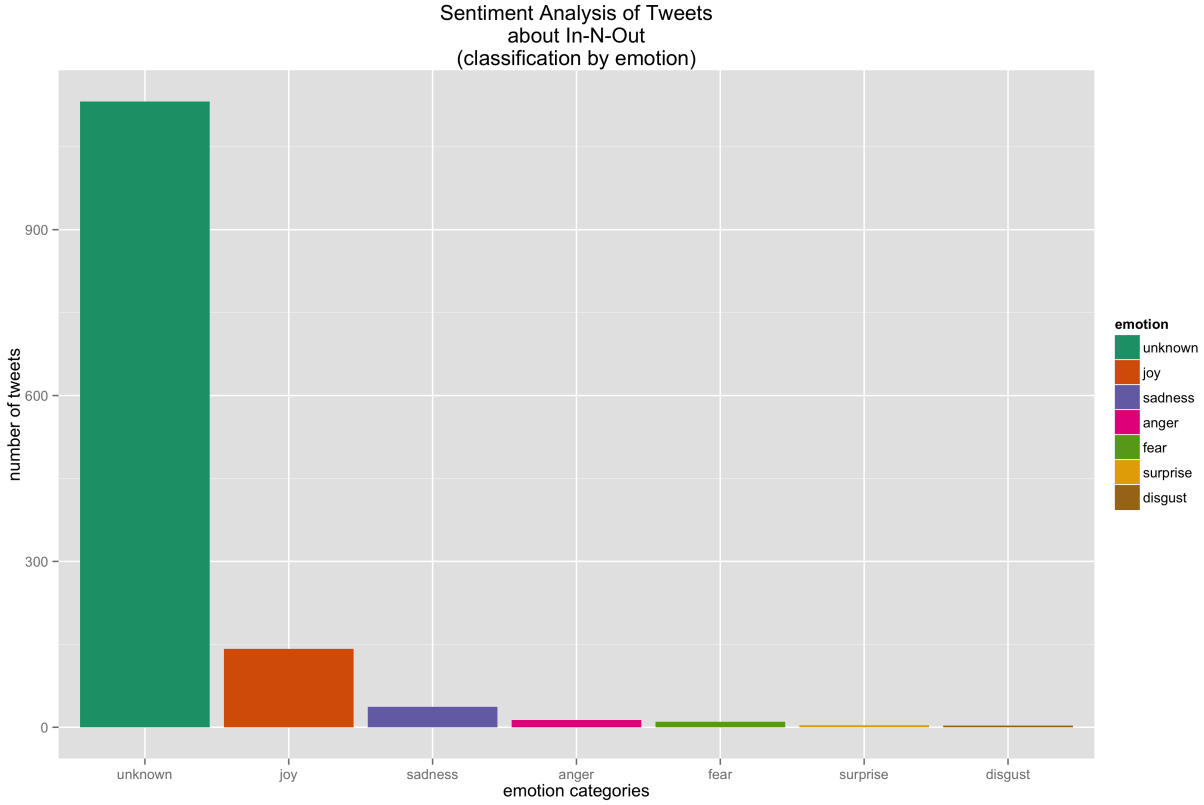

joy
sadness
unknown
anger
disgust
fear
surprise

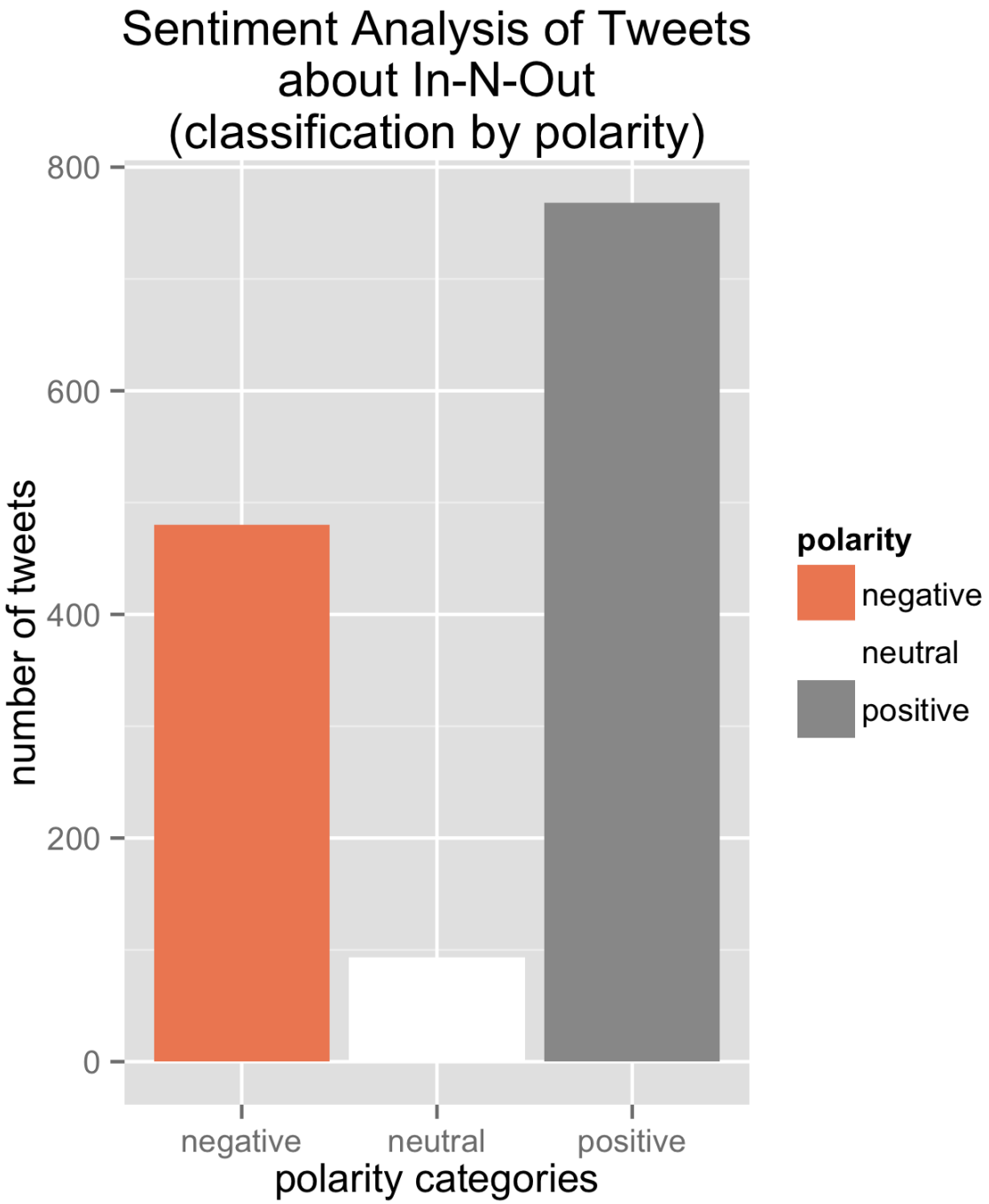

joy
sadness
unknown
anger
disgust
fear
surprise

In case you are unfamiliar with word clouds, the idea is that the larger a text is then the more frequent it was in some larger body of text. If we take a look at the word cloud for **In-n-Out Burger**, we notice some interesting words in some of the sentiment groupings. For example, in the *joy* sentiment we can see that the word *mcdonalds* was mentioned. This can perhaps signify some tweeters' preferences for McDonald's over In-n-Out. Perhaps tweeters are actually comparing the two in favor of In-n-Out. These visualizations only scratch the surface for answering our question. However, we are gaining a good sense of what people are talking about when tweeting about these restaurants.

Conclusions

To answer our question of what are people's sentiments of these restaurants (as reflected in tweets) seems to show the same trend as we evaluate our data and plots. There seems to be a huge amount of tweets that are classified as *Unknown*, but *joy* tweets are the second largest population. As for polarity, the amount of *positive* tweets triumph over *negative* and *neutral*. These two trends were reflected in our data. Here are some other plots that show this behavior.





An explanation for this conclusion could be that people do not generally tweet sentimental postings about restaurants. Another possibility is to blame the Bayes classifier. It will take much more research to study the latter possibility. Next major steps would be to work more with the quantitative data, so perhaps plot change of log likelihoods across restaurants. Overall, we can perhaps argue that people generally tweet positive things about fast food establishments because perhaps people look forward to eating. Furthermore, they may be either thinking of its food, at the restaurant or on the way to visiting a restaurant.

We would like cite the following sources:

Twitter Search API

Regular Expressions Cheat Sheet

Sentiment for Consumer Analysis

Basics of Text Mining