Sentiment Analysis of Election 2016 with Twitter Data

Ana Laguna Pradas

Disclaimer: This demo is intended to show the techniques that can be used for Sentiment Analysis based on Twitter data. The outcomes are driven by the data and not intended to express support on behalf of (or in opposition to) any candidate.

Before we start any kind of Data Analysis, the first step is to obtain the data. For this sentiment analysis, I obtained tweets related to declared presidential candidates through Twitter Search API.

The candidates from both Republican and Democrats are:

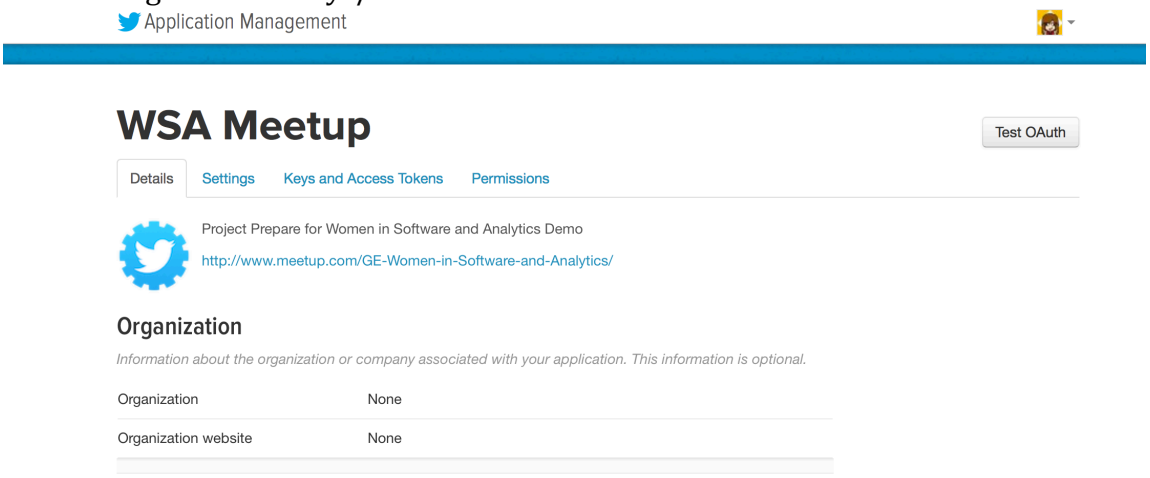
```
dems <- c("Chafee, Lincoln", "Clinton, Hillary", "O'Malley, Martin", "Sanders, Bernie", "Webb, Jim")
repub <- c("Bush, Jeb", "Carson, Ben", "Christie, Chris", "Cruz, Ted", "Fiorina, Carly", "Gilmore, Jim", "Graham, Lindsey", "Huckabee, Mike", "Jindal, Bobby", "Kasich, John", "Pataki, George", "Paul, Rand", "Perry, Rick", "Rubio, Marco", "Santorum, Rick", "Trump, Donald", "Walker, Scott")
```

There is one female candidate from each party.

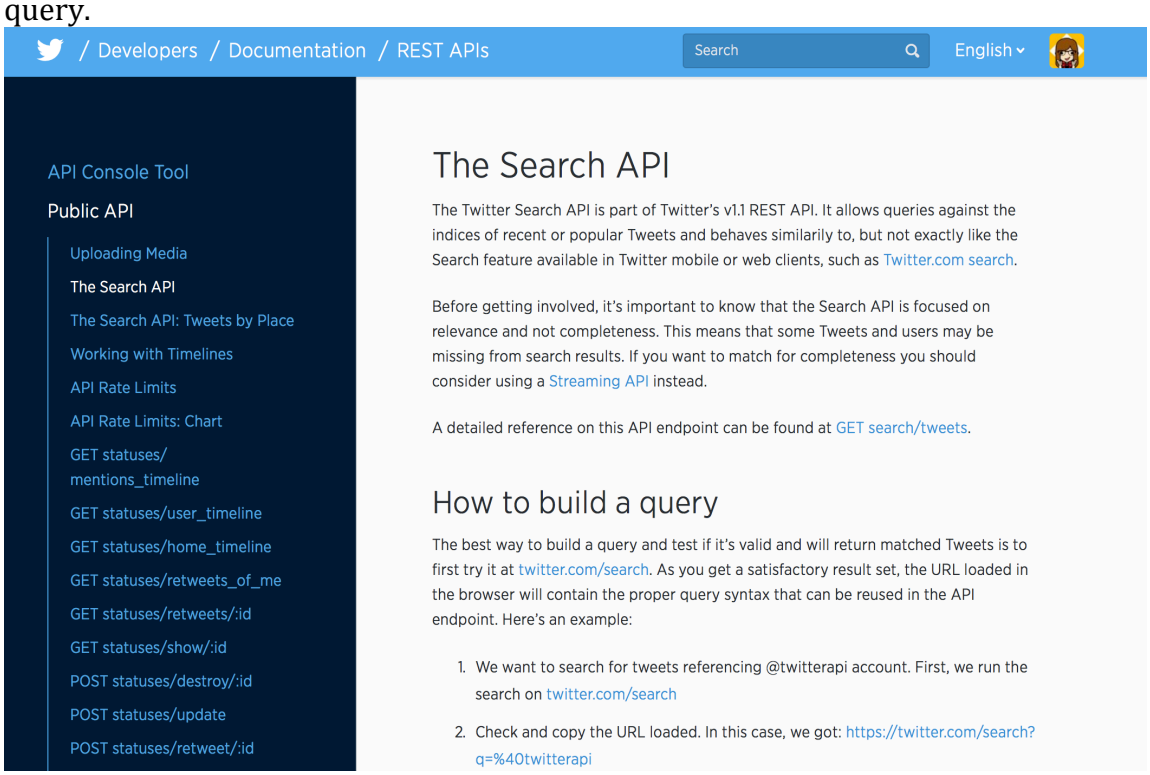
```
dems_candidate <- "Clinton, Hillary"
repub_candidate <- "Fiorina, Carly"
```

Step 1: Access to Twitter Search API

By navigating to Twitter application web page, I created a new application for this demo and generated Keys/Access Tokens of the API.



Documentation of Search API addresses more details about how to build the search query.



Step 2: Gather the Data

Time Range: "2015-08-01" to Today ("2015-08-13")

Number of Record per Candidate: 10,000

```
# R packages for Twitter Search API
library(ROAuth)
library(twitteR)

# parameters for the connection
consumer_key <- '***************'
consumer_secret <- '***************'
access_token <- '***************'
access_secret <- '***************'

# create you OAuth search credential
searchCred <- setup_twitter_oauth(consumer_key, consumer_secret, access_t
oken, access_secret)

# a function to pull data for each candidate from twitter
gatherData <- function(candidate, count) {
  # search tweets relate to a particular content
  obj <- searchTwitter(searchString=candidate, n=count, lang="en", since=
"2015-08-01", until=str(Sys.Date()))
```

```
  # parses the tweets
  df <- do.call("rbind", lapply(obj, as.data.frame))
  # write output to a csv file
  candidate <- gsub(", ", "_", candidate)
  fname <- paste(candidate, ".csv", sep="")
  write.csv(df, file=fname, row.names=FALSE)
}

# pull data one example
gatherData(repub_candidate, 10000)
```

Step 3: Review the Data

```
# a function to read in data from a csv file
readData <- function(candidate) {
  # csv file name
  candidate <- gsub(", ", "_", candidate)
  fname <- paste(candidate, ".csv", sep="")
  df <- read.csv(fname, stringsAsFactors=FALSE)
  return(df)
}

# read in the data from .csv file
setwd("/Users/alaguna/Desktop/Ana/Scripts_machinelearning_statistics/Sentiment-Analysis-of-Election-2016-with-Twitter-Data-master/data")
df_dems <- readData(dems_candidate)
df_repub <- readData(repub_candidate)

# review the data
head(df_dems)
```

```
##
text
## 1         Whoever "through gross negligence permits [it] to be removed
from its proper place of custody" like to private server http://t.co/2CZW
DImkhN
## 2                 Watch @RepMikePompeo on @TheMalzbergShow abt Hillary
Clinton turning over her servers to the DOJ at 7:00PM ET http://t.co/iWzw
6bJ5zd
## 3 RT @ezraklein: March poll:\nHillary Clinton: 44%\nBernie Sanders: 8%
\n\nAugust poll:\nHillary Clinton: 37%\nBernie Sanders: 44%\n\nWow. http:
//t.co…
## 4                                                  RT @MAKERSwomen: Watch Hi
llary Clinton's first election ad: http://t.co/EsBlMDS0ZK http://t.co/t7K
Etd892A
## 5                  RT @FoxNews: .@ChuckLane1: "The politics... are get
ting worse and worse for Hillary Clinton." #SpecialReport http://t.co/B5x
1ko8hRd
## 6         RT @RealJamesWoods: Hillary Clinton turns in ‘blank’ email s
erver to investigators | "The information had been migrated" - actual quo
```

```
te! ht…
##   favorited favoriteCount replyToSN             created truncated
## 1     FALSE             0      <NA> 2015-08-13 22:45:19     FALSE
## 2     FALSE             0      <NA> 2015-08-13 22:45:18     FALSE
## 3     FALSE             0      <NA> 2015-08-13 22:45:15     FALSE
## 4     FALSE             0      <NA> 2015-08-13 22:45:14     FALSE
## 5     FALSE             0      <NA> 2015-08-13 22:45:12     FALSE
## 6     FALSE             0      <NA> 2015-08-13 22:45:11     FALSE
##   replyToSID           id replyToUID
## 1         NA 6.319597e+17         NA
## 2         NA 6.319597e+17         NA
## 3         NA 6.319597e+17         NA
## 4         NA 6.319597e+17         NA
## 5         NA 6.319597e+17         NA
## 6         NA 6.319597e+17         NA
##
statusSource
## 1                   <a href="http://twitter.com" rel="nofollow">Twitte
r Web Client</a>
## 2                      <a href="http://www.hootsuite.com" rel="nofollo
w">Hootsuite</a>
## 3                   <a href="http://twitter.com" rel="nofollow">Twitte
r Web Client</a>
## 4   <a href="http://twitter.com/download/iphone" rel="nofollow">Twitte
r for iPhone</a>
## 5              <a href="https://mobile.twitter.com" rel="nofollow">Mob
ile Web (M5)</a>
## 6 <a href="http://twitter.com/download/android" rel="nofollow">Twitter
for Android</a>
##        screenName retweetCount isRetweet retweeted longitude latitude
## 1   _MatthewClark            0     FALSE     FALSE        NA       NA
## 2       NewsmaxTV            0     FALSE     FALSE        NA       NA
## 3   allanholloway         6647      TRUE     FALSE        NA       NA
## 4 VermontSongbird            5      TRUE     FALSE        NA       NA
## 5         kumerle            4      TRUE     FALSE        NA       NA
## 6     SantaKlausH          325      TRUE     FALSE        NA       NA
```

Step 4: Text Cleaning

```
# R package for text mining
library(stringr)
library(tm)

## Loading required package: NLP

# get the tweets and creation time
df_dems <- df_dems[1:10000, c("text", "created")]
df_repub <- df_repub[1:10000, c("text", "created")]

# a function to clean text by converting text to lower cases and removing
RT, @, puncutations, numbers, links and etc.
```

```r
cleanText <- function(df) {
  # get the text
  tweet <- df$text
  # remove retweet entities
  tweet <- gsub("(RT|via)((?:\\b\\W*@\\w+)+)", " ", tweet)
  # remove html links
  tweet <- gsub("http\\S+", " ", tweet)
  # remove at people
  tweet <- gsub("@\\S+", " ", tweet)
  # remove hashtags
  tweet <- gsub("#\\S+", " ", tweet)
  # remove punctuation
  tweet <- gsub("[[:punct:]]", " ", tweet)
  # remove numbers
  tweet <- gsub("[[:digit:]]", " ", tweet)
  # define "tolower error handling" function
  tryTolower <- function(x)
  {
    # create missing value
    y <- NA
    # tryCatch error
    try_error <- tryCatch(tolower(x), error=function(e) e)
    # if not an error
    if (!inherits(try_error, "error"))
      y <- tolower(x)
    # result
    return(y)
  }
  # lower case using tryTolower with sapply
  tweet <- sapply(tweet, tryTolower)
  # remove English stop words
  tweet <- removeWords(tweet, stopwords("english"))
  # remove words less than 2 characters
  tweet <- gsub("(\\b)?\\w{1,2}(\\b)?", " ", tweet)
  # remove unnecessary spaces
  tweet <- gsub("[ \t]{2,}", " ", tweet)
  tweet <- gsub("^\\s+|\\s+$", "", tweet)
  # remove \n, \t and etc
  tweet <- gsub("\n|\t", "", tweet)
  # remove NAs in tweet
  index <- !is.na(tweet)
  tweet <- tweet[index]
  names(tweet) <- NULL
  df <- data.frame(cbind(text_c=tweet, text=df$text[index], created=df$cr
eated[index]))
  return(df)
}
df_dems_c <- cleanText(df_dems)
df_repub_c <- cleanText(df_repub)
```

```
# review the data
head(df_dems_c)

##
text_c
## 1        whoever gross negligence permits removed proper place custody
like private server
## 2                                                watch abt hillary clinton
turning servers doj
## 3 march poll hillary clinton bernie sanders august poll hillary clinto
n bernie sanders wow
## 4                                                          watch hillary cl
inton first election
## 5                                                 politics getting worse w
orse hillary clinton
## 6 hillary clinton turns blank email server investigators information m
igrated actual quote
##
text
## 1         Whoever "through gross negligence permits [it] to be removed
from its proper place of custody" like to private server http://t.co/2CZW
DImkhN
## 2                 Watch @RepMikePompeo on @TheMalzbergShow abt Hillary
Clinton turning over her servers to the DOJ at 7:00PM ET http://t.co/iWzw
6bJ5zd
## 3 RT @ezraklein: March poll:\nHillary Clinton: 44%\nBernie Sanders: 8%
\n\nAugust poll:\nHillary Clinton: 37%\nBernie Sanders: 44%\n\nWow. http:
//t.co…
## 4                                                RT @MAKERSwomen: Watch Hi
llary Clinton's first election ad: http://t.co/EsBlMDS0ZK http://t.co/t7K
Etd892A
## 5                  RT @FoxNews: .@ChuckLane1: "The politics... are get
ting worse and worse for Hillary Clinton." #SpecialReport http://t.co/B5x
1ko8hRd
## 6         RT @RealJamesWoods: Hillary Clinton turns in ‘blank’ email s
erver to investigators | "The information had been migrated" - actual quo
te! ht…
##               created
## 1 2015-08-13 22:45:19
## 2 2015-08-13 22:45:18
## 3 2015-08-13 22:45:15
## 4 2015-08-13 22:45:14
## 5 2015-08-13 22:45:12
## 6 2015-08-13 22:45:11
```

Step 5: Build a Classification Model using Naive Bayes Algorithm

In order to classify some text as positive or negative, the classification was done by using a Naive Bayes algorithm trained on Janyce Wiebe’s subjectivity lexicon

Naïve Bayes Theory:

Given a set of variables, $X = \{x_1, x_2, x_3, \dots, x_n\}$, the algorithm is to construct the posterior probability for the event C_j among a set of possible outcomes

$$C = \{c_1, c_2, c_3, \dots, c_k\}$$

Using Bayes' rule:

$$p(C_k|x_1, x_2, \dots, x_n) = \frac{p(C_k)p(x_1, x_2, \dots, x_n|C_k)}{p(x_1, x_2, x_3, \dots, x_n)} = \frac{\text{Prior} \times \text{Likelihood}}{\text{Evidence}}$$

Using chain rule of conditional probability:

$$p(C_k|x_1, x_2, \dots, x_n) \propto p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

The classifier:

$$C_k = \underset{k \in \{1,\dots,K\}}{\text{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

The Lexicon dataset contains words that were manually labeled *Positive* or *Negative*, so the prior probabilities of positive and negative words are:

$$prior(positive) = \frac{\#\ of\ positive\ words}{total\ \#\ of\ words}$$

$$prior(negative) = \frac{\#\ of\ negative\ words}{total\ \#\ of\ words}$$

In this Sentiment Analysis, the outcome set is

$$\{\text{Sentiment}_{\text{positive}}, \text{Sentiment}_{\text{negative}}, \text{Sentiment}_{\text{neutral}}\}$$

The algorithm is based on the assumption that the appearances of words are independent of each other.

The probabilities of the sentiment of a tweet being positive/negative can be expressed as below

$$posterior(positive) = \frac{Prior(positive)\ P(amazing|positive)P(hate|positive)P(smart|positive)}{P(amazing, hate, smart)}$$

$$posterior(negative) = \frac{Prior(negative)\ P(amazing|negative)P(hate|negative)P(smart|negative)}{P(amazing, hate, smart)}$$

Since the denominators are the same, the classification can be made by simply comparing the numerators. Also, let's take the natural log of both sides.

$$\ln posterior(positive) = \ln Prior(positive) + \ln P(amazing|positive) + \ln P(hate|positive) + \ln P(smart|positive)$$
$$\ln posterior(negative) = \ln Prior(negative) + \ln P(amazing|negative) + \ln P(hate|negative) + \ln P(smart|negative)$$

Decision Rules:

$$Prior\ Ratio = \frac{|\ln prior(positive)|}{|\ln prior(negative)|}$$

$$Posterior\ Ratio = \frac{|\ln posterior(positive)|}{|\ln posterior(negative)|}$$

when *Posterior Ratio* > *Prior Ratio*: Positive;
when *Posterior Ratio* = *Prior Ratio*: Neutral;
When *Posterior Ratio* > *Prior Ratio*: Negative.

```
lexicon <- read.csv('/Users/alaguna/Desktop/Ana/Scripts_machinelearning_statistics/Sentiment-Analysis-of-Election-2016-with-Twitter-Data-master/data/lexicon.csv')
head(lexicon)
```

```
##         Words Strong.Weak Polarity
## 1   abandoned    weaksubj negative
## 2 abandonment    weaksubj negative
## 3     abandon    weaksubj negative
## 4       abase  strongsubj negative
## 5   abasement  strongsubj negative
## 6       abash  strongsubj negative
```

```
# R package for creating a text matrix
library(sentiment)
library(RTextTools)
```

```
## Loading required package: SparseM
```

```
##
## Attaching package: 'SparseM'
```

```
## The following object is masked from 'package:base':
##
##     backsolve
```

```
# a function for the Naive Bayes model
classifyNB <- function (textColumns, algorithm = "bayes", pstrong = 0.5,
pweak = 1,
          prior = 1)
{
  matrix <- create_matrix(textColumns)
  lexicon <- read.csv('/Users/alaguna/Desktop/Ana/Scripts_machinelearning
_statistics/Sentiment-Analysis-of-Election-2016-with-Twitter-Data-master/
data/lexicon.csv', header = FALSE)
  counts <- list(positive = length(which(lexicon[, 3] == "positive")),
                 negative = length(which(lexicon[, 3] == "negative")),
                 total = nrow(lexicon))
  documents <- c()
  # determine the scores for each document
  for (i in 1:nrow(matrix)) {
    scores <- list(positive = 0, negative = 0)
    doc <- matrix[i, ]
    words <- findFreqTerms(doc, lowfreq = 1)
    # match each word with Lexiton words to determine the scores for posi
tive and negative
    for (word in words) {
      index <- pmatch(word, lexicon[, 1], nomatch = 0)
      if (index > 0) {
        entry <- lexicon[index, ]
        polarity <- as.character(entry[[2]])
        category <- as.character(entry[[3]])
        count <- counts[[category]]
        score <- pweak
        if (polarity == "strongsubj") {score <- pstrong}
        if (algorithm == "bayes") {score <- abs(log(score * prior/count))
```

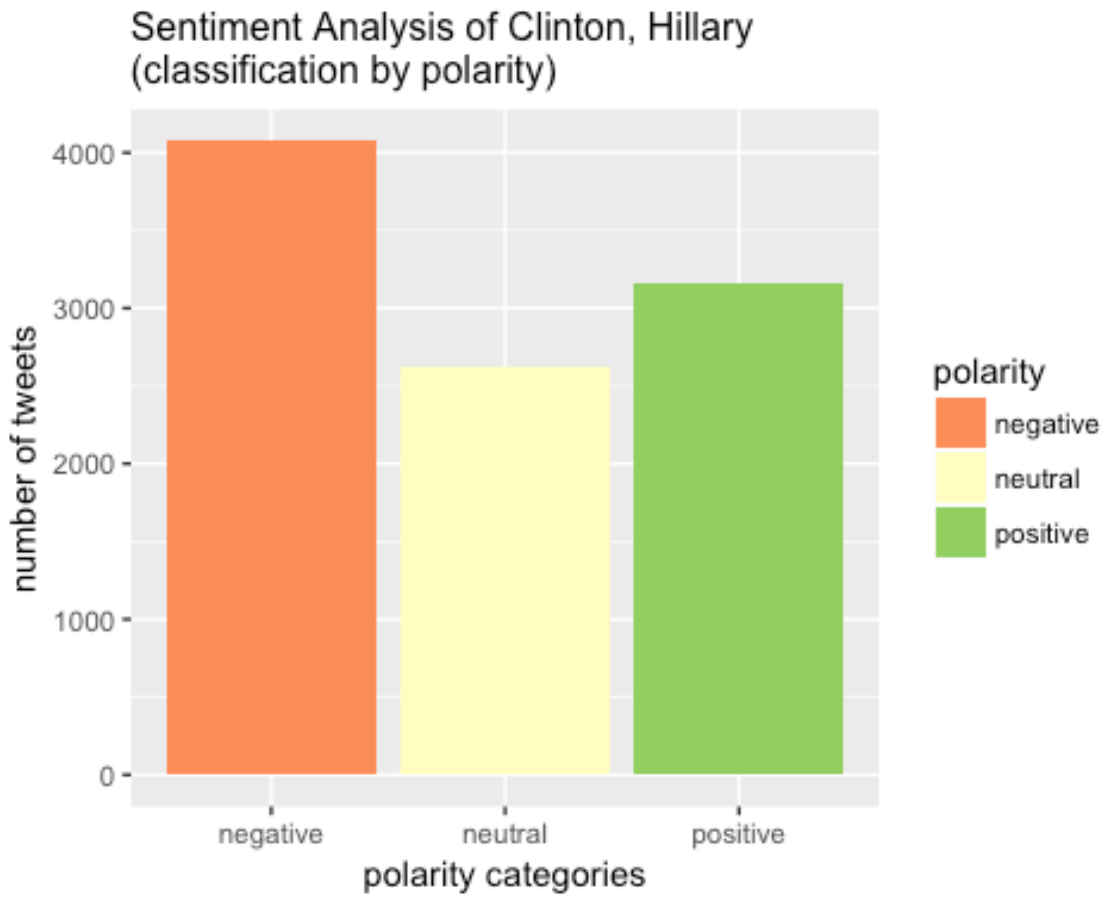
```
}
        scores[[category]] <- scores[[category]] + score
      }
    }
    # if no word matches the Lexicon, then the scores will based on the p
rior probability of positive words and negative words
    for (key in names(scores)) {
      count <- counts[[key]]
      total <- counts[["total"]]
      score <- abs(log(count/total))
      scores[[key]] <- scores[[key]] + score
    }

    ratio <- abs(scores$positive/scores$negative) # ratio of positive/neg
ative scores
    prior_ratio <- abs(log(counts$positive/counts$total))/abs(log(counts$
negative/counts$total)) # prior probability ratio
    # determine the best fit
    if (ratio == prior_ratio) {best_fit <- "neutral"}
    else if (ratio > prior_ratio) {best_fit <- "positive"}
    else if (ratio < prior_ratio) {best_fit <- "negative"}
    documents <- rbind(documents, c(scores$positive, scores$negative,
                                    abs(scores$positive/scores$negative),
                                    abs(log(counts$positive/counts$total)
)/abs(log(counts$negative/counts$total)),
                                    best_fit))
  }
  colnames(documents) <- c("POS", "NEG", "POS/NEG", "PRIOR RATIO", "BEST_
FIT")
  return(documents)
}

# test on a few examples
classifyNB("This meetup is amazing!!", algorithm="bayes")

##      POS                 NEG                  POS/NEG
## [1,] "9.47562344948448" "0.445367139362065" "21.2759824693335"
##      PRIOR RATIO        BEST_FIT
## [1,] "2.31687846937282" "positive"

classifyNB("I hate being stuck in traffic.", algorithm="bayes")

##      POS                 NEG                 POS/NEG
## [1,] "1.03186153615413" "9.47562344948448" "0.108896426884742"
##      PRIOR RATIO        BEST_FIT
## [1,] "2.31687846937282" "negative"

classifyNB("I went for a run.", algorithm="bayes")

##      POS                 NEG                  POS/NEG
## [1,] "1.03186153615413" "0.445367139362065" "2.31687846937282"
```

```
##      PRIOR RATIO        BEST_FIT
## [1,] "2.31687846937282" "neutral"
```

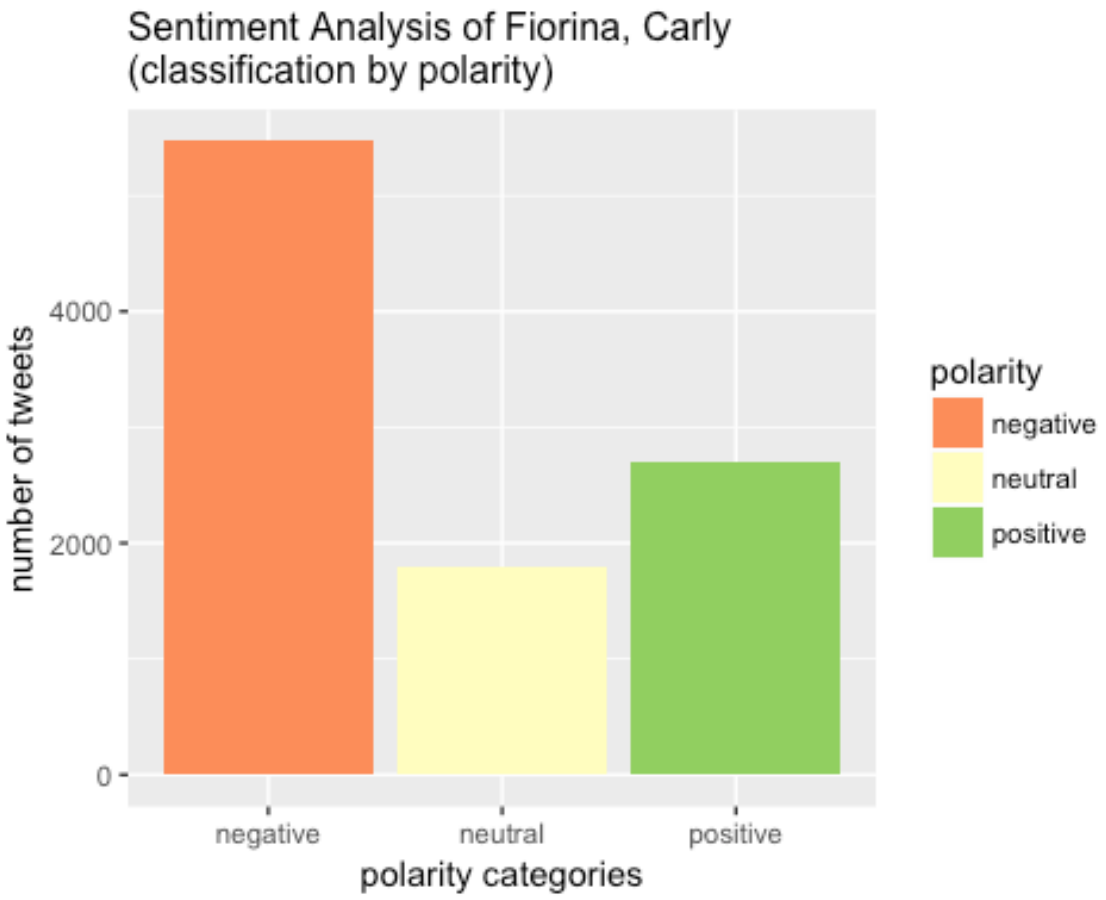
Step 6: Data Visualization

```
# R packages for data visualization
library(ggplot2)

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:NLP':
##
##     annotate

library(wordcloud)

## Loading required package: RColorBrewer

# a function to build the model and generate output dataset for visualiza
tion
prepOut <- function(df) {
  # build the model
  classifier <- classifyNB(df$text_c, algorithm="bayes")

  # prepare the results for visualization
  df_out <- data.frame(candidate= "Trump, Donald",
                       tweet=df$text,
                       tweet_c=df$text_c,
                       polarity=classifier[,5],
                       creation_time=df$created, stringsAsFactors=FALSE)
  return(df_out)
}

# output dataset for Republican candidate
df_out_repub <- prepOut(df_repub_c)

# output dataset for Democrats candidate
df_out_dems <- prepOut(df_dems_c)

# plot distribution of polarity
barPlot <- function(df, candidate) {
  plot <- ggplot(df, aes(x=polarity)) +
          geom_bar(aes(y=..count.., fill=polarity)) +
          scale_fill_brewer(palette="RdYlGn") +
          labs(x="polarity categories", y="number of tweets",
               title = paste("Sentiment Analysis of", candidate, "\n(clas
sification by polarity)")) +
          theme(plot.title = element_text(size=12))
  return(plot)
}
```

```
# bar chart for Democrats candidate
barPlot(df_out_dems, dems_candidate)
```

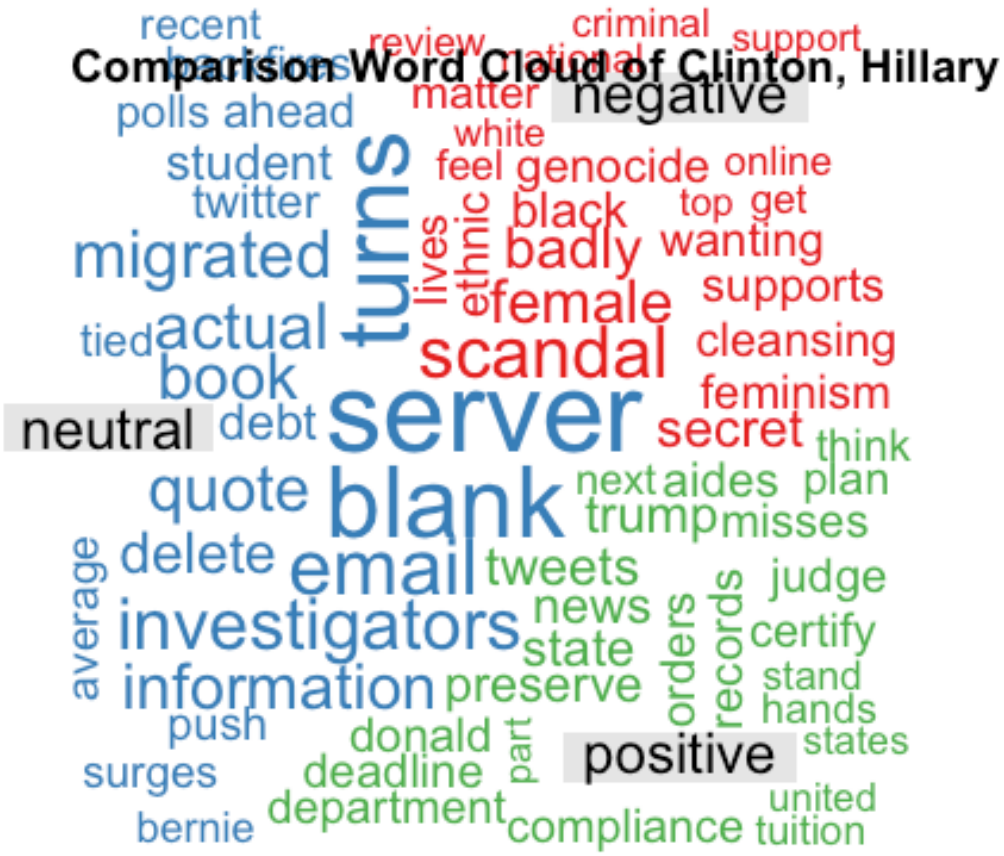



```
# bar chart for Republican candidate
barPlot(df_out_repub, repub_candidate)
```

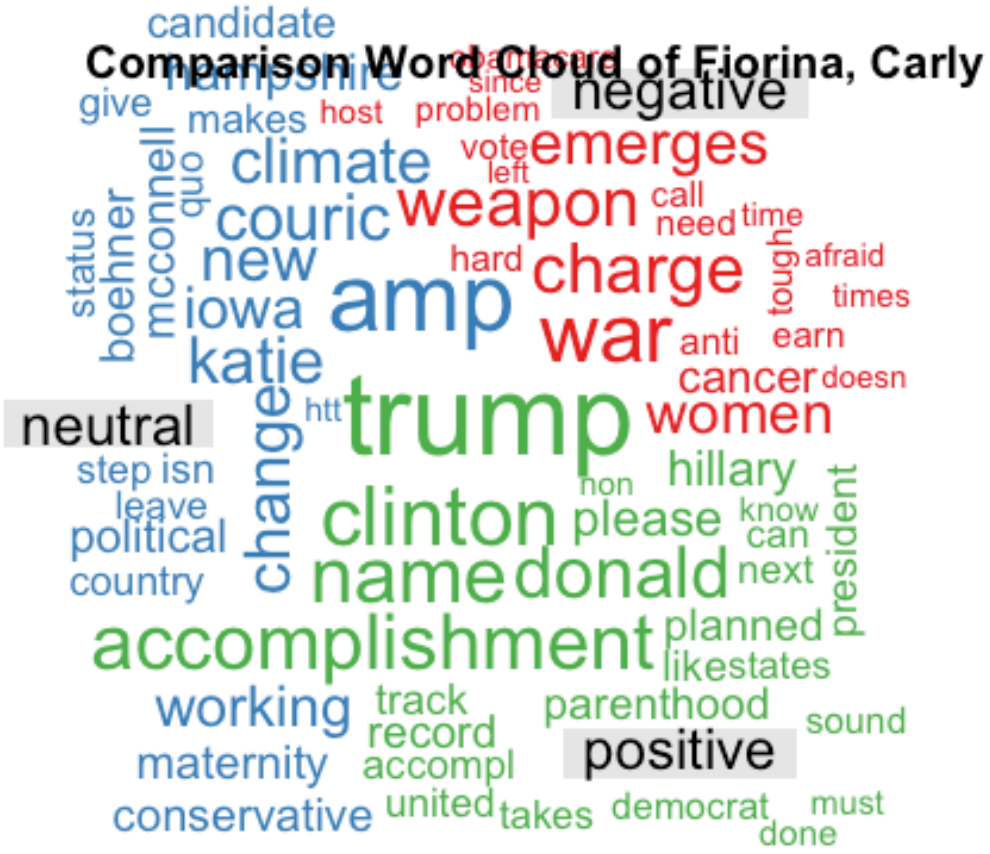



```r
# plot comparison word cloud
wordCloud <- function(df, candidate) {

  # separating text by polarity
  pol <- levels(factor(df$polarity))
  npol <- length(pol)
  pol.docs <- rep("", npol)
  for (i in 1:npol)
  {
    text <- df$tweet_c
    text <- gsub(tolower(gsub(", ", "|", candidate)), " ", text)
    tmp <- text[df$polarity == pol[i]]
    pol.docs[i] <- paste(tmp, collapse=" ")
  }

  # create corpus
  pol.corpus <- Corpus(VectorSource(pol.docs))
  tdm <- as.matrix(TermDocumentMatrix(pol.corpus))
  termFrequency <- rowSums(tdm)
  tdm_sub <- subset(tdm, termFrequency>=100)
  colnames(tdm_sub) <- pol
```

```
  # comparison word cloud
  comparison.cloud(tdm_sub, colors=brewer.pal(npol, "Set1"),
                   scale = c(3,.5), random.order = FALSE,
                   title.size = 1.5, max.words=80)
  title(paste("Comparison Word Cloud of", candidate), font.main=2.5)
}

options(warn=-1)
# comparison word cloud for Democrats candidate
wordCloud(df_out_dems, dems_candidate)
```




```
# comparison word cloud for Republican candidate
wordCloud(df_out_repub, repub_candidate)
```



This is the demo I prepared for the Women in Software and Analytics Meetup on September 2nd, 2015. All R packages used in this script were obtained from The Comprehensive R Archive Network (CRAN).