Using APIs with R

Steve Pittard

December 10, 2015

This document discusses ways to judge the sentiment of text and how to interact with various APIs such as those offered by Twitter. This is not meant to be an in-depth discussion on text mining.Some helpful webpages you might refer to include:

- http://bit.ly/1RddcOa
- http://www.r-bloggers.com/create-twitter-wordcloud-with-sentiments/
- http://www.datumbox.com/machine-learning-api/
- https://sites.google.com/site/miningtwitter/questions/sentiment/viralheat
- http://blog.mashape.com/list-of-20-sentiment-analysis-apis/

Prerequisites

In order for this document to work you will first need to install the following packages. There could be some difficulties with this based on the version of R you have. The **sentiment** package relies on a package on OmegaHat called **RStem**. It can be downloaded from http://www.omegahat.org/Rstem/ You might have to install this from the R command line.

Loading the necessary packages

```
library(twitteR)
```

```
##
## Attaching package: 'twitteR'
##
## The following objects are masked from 'package:dplyr':
##
##     id, location
```

```
library(ROAuth)
library(RCurl)
```

```
## Loading required package: bitops
```

```
library(sentiment)
```

```
## Loading required package: tm
## Loading required package: NLP
##
## Attaching package: 'NLP'
##
## The following object is masked from 'package:ggplot2':
##
```

```
##     annotate
##
## Loading required package: Rstem
```

```
library(dplyr)
library(RJSONIO)
library(stringr)
```

How are you Feeling today ?

Wouldn't it be nice to have a way to classify negative or positive emotions expressed in text - like Tweets from the Twitter service ? Or just random sentences found in a book. This way we could figure out if someone was happy or sad about something without having to read it ourselves. Here is an example that demonstrates this. There is a package for R called **sentiment** which can help us examine text to predict the overall mood of the piece.

```
documents <- c("I hate Delta Airlines. The are always late and the gate agents are incredily rude.",
               "I enjoyed my flight from New York to Paris. Cabin service was quite good.",
               "Gee Delta. How nice of you to lose my luggage.")

# The sentiment library will let us do this right inside of R

classify_emotion(documents)
```

```
##      ANGER              DISGUST            FEAR
## [1,] "7.34083555412328" "3.09234031207392" "2.06783599555953"
## [2,] "1.46871776464786" "3.09234031207392" "2.06783599555953"
## [3,] "1.46871776464786" "3.09234031207392" "2.06783599555953"
##      JOY                SADNESS           SURPRISE           BEST_FIT
## [1,] "1.02547755260094" "1.7277074477352" "2.78695866252273" "anger"
## [2,] "7.34083555412328" "1.7277074477352" "2.78695866252273" "joy"
## [3,] "1.02547755260094" "1.7277074477352" "2.78695866252273" NA
```

```
classify_polarity(documents)
```

```
##      POS                NEG                POS/NEG              BEST_FIT
## [1,] "1.03127774142571" "18.5054868578024" "0.0557282145209216" "negative"
## [2,] "8.78232285939751" "9.47547003995745" "0.926848253686942"  "negative"
## [3,] "9.47547003995745" "17.8123396772424" "0.531961000724885"  "negative"
```

```
# It might be useful to try a different algorithm

classify_emotion(documents,algorithm="voter")
```

```
##      ANGER      DISGUST FEAR    JOY        SADNESS SURPRISE BEST_FIT
## [1,] "1.000001" "1e-06" "1e-06" "1e-06"    "1e-06" "1e-06"  "anger"
## [2,] "1e-06"    "1e-06" "1e-06" "1.000001" "1e-06" "1e-06"  "joy"
## [3,] "1e-06"    "1e-06" "1e-06" "1e-06"    "1e-06" "1e-06"  "anger"
```

```
classify_polarity(documents,algorithm="voter")
```

```
##      POS        NEG        POS/NEG             BEST_FIT
## [1,] "1e-06"    "1.000001" "9.99999000001e-07" "negative"
## [2,] "1.000001" "0.500001" "1.999998000004"    "neutral"
## [3,] "0.500001" "1.500001" "0.333333777777482" "negative"
```

It would be better to create a function that we could call to process arbitrary numbers of sentences. In R we like to create functions anyway since it helps us to easily reproduce our work.

```
myFeelings <- function(docs,algorithm="bayes") {
  len <- length(docs)

  # Create a data frame for returning to the user

  retdf <- data.frame(text=docs,emotion=rep("",len),polarity=rep(",len"),
                      stringsAsFactors = FALSE)

  retdf$emotion <- classify_emotion(docs,algorithm = algorithm)[,7]

  # Here we capture the output of the polarity classification column
  retdf$polarity <- classify_polarity(docs,algorithm = algorithm)[,4]

  # If no emotion was predicted then replace it with a value of unknown
  retdf[is.na(retdf)] = "unknown"
  return(retdf)
}

myFeelings(documents)
```

```
##                                                                                text
## 1 I hate Delta Airlines. The are always late and the gate agents are incredily rude.
## 2          I enjoyed my flight from New York to Paris. Cabin service was quite good.
## 3                                       Gee Delta. How nice of you to lose my luggage.
##   emotion polarity
## 1   anger negative
## 2     joy negative
## 3 unknown negative
```

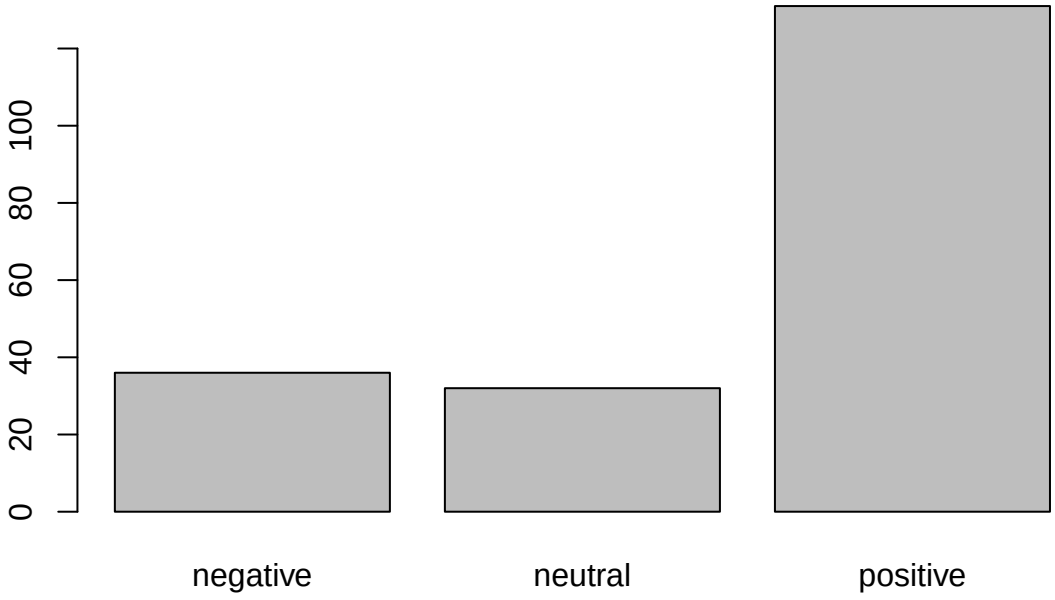
Working with some actual data

Let's download some actual tweets having to do with Delta Airlines. This is a list with 200 tweets that have been cleaned up. This is far more interesting than just processing some made up sentences.

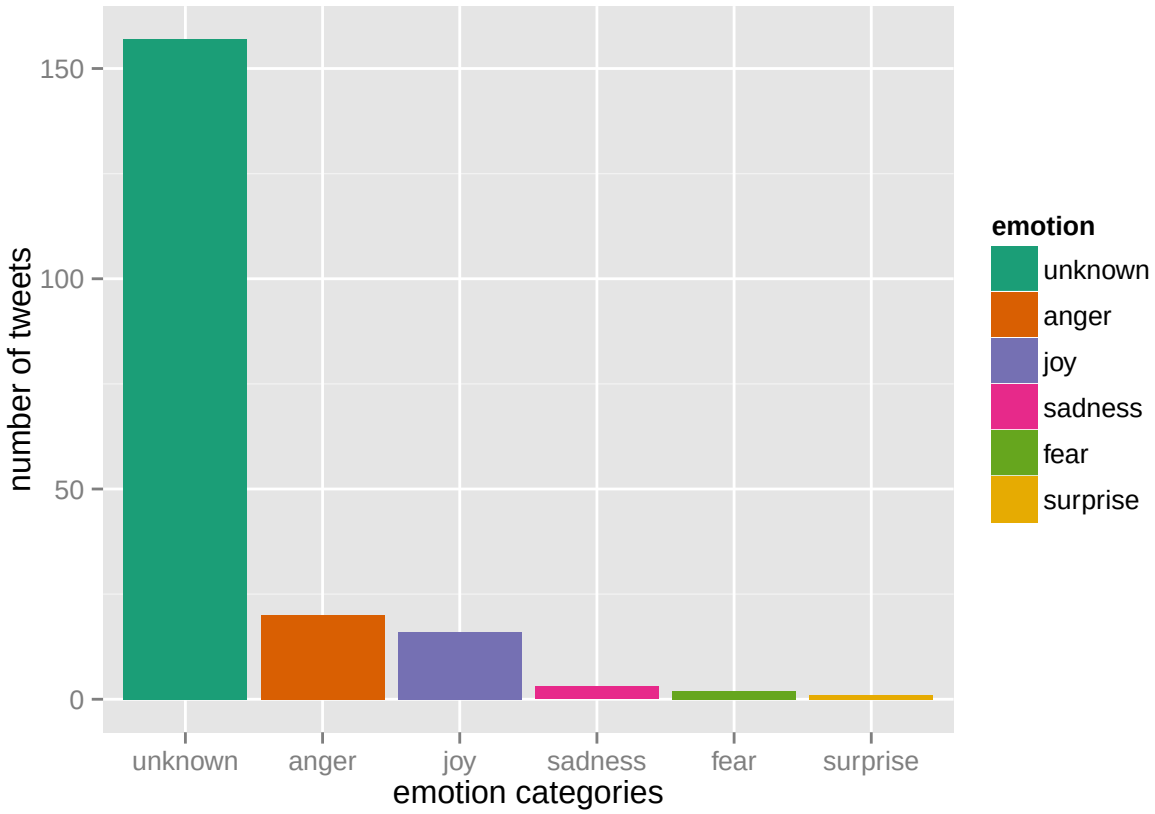
```
load(url("http://steviep42.bitbucket.org/YOUTUBE.DIR/tweets.RData"))

delta_tweets

(sentiment_tweets <- myFeelings(delta_tweets))
```

Plotting the results

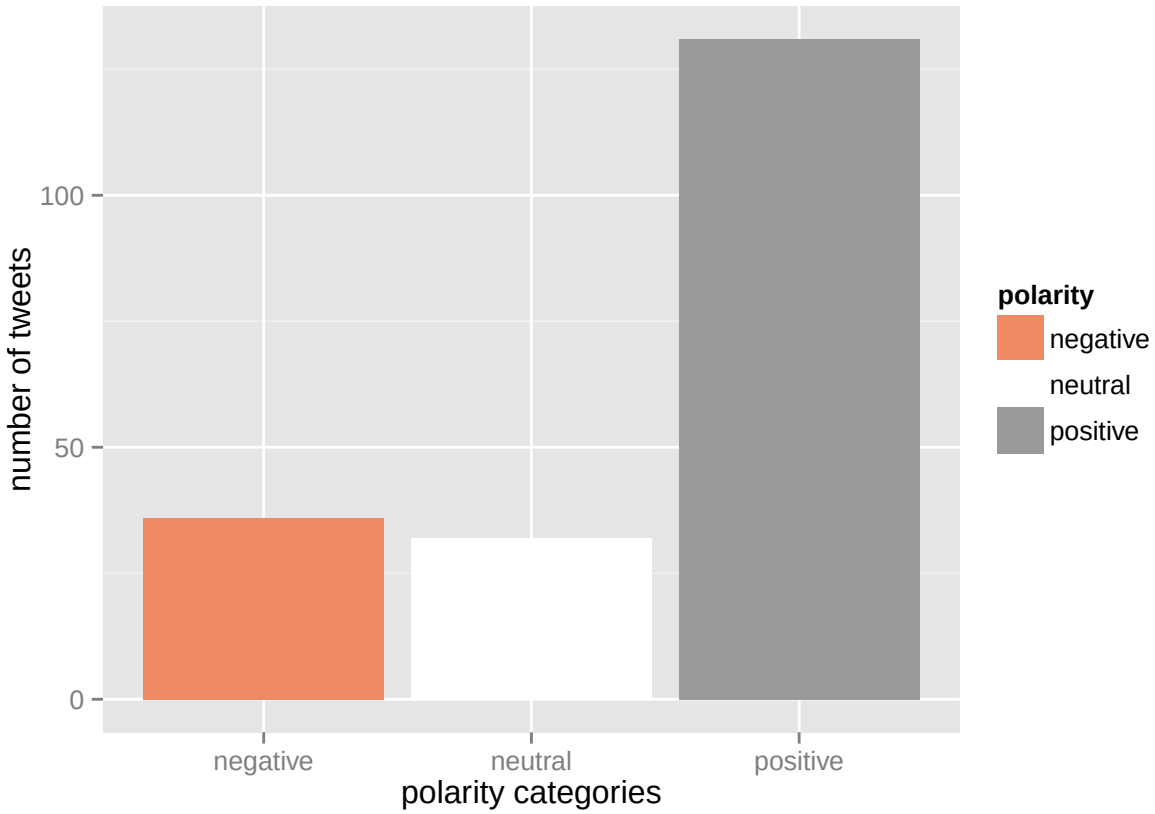
```
load(url("http://steviep42.bitbucket.org/YOUTUBE.DIR/tweets.RData"))
emodf <- data.frame(classify_emotion(delta_tweets,algorithm="bayes",
                                     prior=1.0),stringsAsFactors = FALSE)
poldf <- data.frame(classify_polarity(delta_tweets,algorithm="bayes"),
                                     stringsAsFactors = FALSE)

barplot(table(poldf$BEST_FIT))
```




```
emotion <- emodf[,7]
# substitute NA's by "unknown"
emotion[is.na(emotion)] = "unknown"

polarity <- poldf[,4]

sent_df = data.frame(text=delta_tweets, emotion=emotion,
                     polarity=polarity, stringsAsFactors=FALSE)

# sort data frame
sent_df = within(sent_df,
                 emotion <- factor(emotion, levels=names(sort(table(emotion), decreasing=TRUE))))

ggplot(sent_df, aes(x=emotion)) +
  geom_bar(aes(y=..count.., fill=emotion)) +
  scale_fill_brewer(palette="Dark2") +
  labs(x="emotion categories", y="number of tweets")
```




```
ggplot(sent_df, aes(x=polarity)) +
  geom_bar(aes(y=..count.., fill=polarity)) +
  scale_fill_brewer(palette="RdGy") +
  labs(x="polarity categories", y="number of tweets")
```



Other ways to assess sentiment

It's important to note that there are more involved and sophisticated services that allow us to determine the sentiment or emotive content of a body of text or, in our case, a collection of tweets. Some of these are standalones programs whereas others are online services that can be accessed via APIs. There are at least 20 different services on the web that we could use to judge the sentiment of text. Some cost money and some are free (or mostly so within limits). We'll use the one at Datumbox to see how it rates the above text. You have to sign up for a free account and then get an API key.

In this first example we'll process the third document/sentence which is ""Gee Delta. How nice of you to lose my luggage." Recall that this sentence expresses sarcasm which is notoriously difficult for sentiment detection systems to pick up on.

```
key <- "74f660f74f09dacc1f795533a70ed94f"
url <- "http://api.datumbox.com/1.0/TwitterSentimentAnalysis.json?api_key="

data <- getURL(paste(url, key, "&text=",documents[3], sep=""))
js <- fromJSON(data,asText=TRUE)
sentiment <- js$output$result

# Lets do this for Subjectivity

url <- "http://api.datumbox.com/1.0/SubjectivityAnalysis.json?api_key="

data <- getURL(paste(url, key, "&text=",documents[3], sep=""))
js <- fromJSON(data,asText=TRUE)
subject <- js$output$result
```

```
# Predict the gender of the person

url <- "http://api.datumbox.com/1.0/GenderDetection.json?api_key="
data <- getURL(paste(url, key, "&text=",documents[3], sep=""))
js <- fromJSON(data,asText=TRUE)
gender <- js$output$result

c(sentiment,subject,gender)
```

```
## [1] "negative"   "subjective" "male"
```

So we could run our tweets through this also. But first it might be useful to put this into a function.

```
getSentiment <- function(text, key){
  text <- URLencode(text)

  # The API limits the string to 360 characters
  if (str_length(text) > 360){
    text <- substr(text, 0, 359)
  }

  # Predict the emotion/sentiment
  data <- getURL(paste("http://api.datumbox.com/1.0/TwitterSentimentAnalysis.json?api_key=",
                       key, "&text=",text, sep=""))
  js <- fromJSON(data, asText=TRUE)
  sentiment = js$output$result

  # Predict the degree of subjectivity
  data <- getURL(paste("http://api.datumbox.com/1.0/SubjectivityAnalysis.json?api_key=",
                       key, "&text=",text, sep=""))
  js <- fromJSON(data, asText=TRUE)
  subject <- js$output$result

  # Predict the topic
  data <- getURL(paste("http://api.datumbox.com/1.0/TopicClassification.json?api_key=",
                       key, "&text=",text, sep=""))
  js <- fromJSON(data, asText=TRUE)
  topic <- js$output$result

  # Predict the gender
  data <- getURL(paste("http://api.datumbox.com/1.0/GenderDetection.json?api_key=", key, "&text=",text,
  js <- fromJSON(data, asText=TRUE)
  gender <- js$output$result

  return(list(sentiment=sentiment,subject=subject,topic=topic,gender=gender))
}


# Let's process the documents list with the three sentences against DatumBox solution

doc_num <- length(documents)
doc_df <- data.frame(text=documents, sentiment=rep("", doc_num),
```

```
                        subject=1:doc_num, topic=1:doc_num, gender=1:doc_num, stringsAsFactors=FALSE)

sentiment <- rep(0, doc_num)
for (i in 1:doc_num)
{

  tmp <- getSentiment(documents[i], "74f660f74f09dacc1f795533a70ed94f")
  doc_df$sentiment[i] <- tmp$sentiment

  doc_df$subject[i] <- tmp$subject
  doc_df$topic[i] <- tmp$topic
  doc_df$gender[i] <- tmp$gender
}

doc_df
```

```
##                                                                                text
## 1 I hate Delta Airlines. The are always late and the gate agents are incredily rude.
## 2          I enjoyed my flight from New York to Paris. Cabin service was quite good.
## 3                                      Gee Delta. How nice of you to lose my luggage.
##   sentiment    subject                   topic gender
## 1  negative subjective                    News female
## 2  positive subjective Recreation & Activities female
## 3  positive subjective                Shopping female
```

Now Let's do this with a subset of the 200 tweets. The reason I'm doing it with a subset is because it can take a while to process the tweets over the internet. Plus the provider of the API might "throttle" accesses since it probably reserves some capacity for the premium/paid services. Because of this I sometimes put in a call to the R function **Sys.sleep()** which briefly pauses the execution of the loop so as not to "beat up" the server that is offering the setiment service. So let's process 50 of the tweets. This won't be drastically different from what just did above with the smaller document set.

```
load(url("http://steviep42.bitbucket.org/YOUTUBE.DIR/tweets.RData"))
documents <- delta_tweets[1:50]   # Get 50 of the tweets
doc_num <- length(documents)
doc_df <- data.frame(text=documents, sentiment=rep("", doc_num),
                        subject=1:doc_num, topic=1:doc_num, gender=1:doc_num, stringsAsFactors=FALSE)

sentiment <- rep(0, doc_num)
for (i in 1:doc_num)
{

  tmp <- getSentiment(documents[i], "74f660f74f09dacc1f795533a70ed94f")
  doc_df$sentiment[i] <- tmp$sentiment

  doc_df$subject[i] <- tmp$subject
  doc_df$topic[i] <- tmp$topic
  doc_df$gender[i] <- tmp$gender

  # Pause for 0.5 seconds

  Sys.sleep(0.5)
}
```

```
doc_df
```