[DATA-06] Sentiment analysis with Twitter data

Miguel-Angel Canela, IESE Business School

October 7, 2016

Introduction

In a broad sense, **sentiment analysis** aims to determine the attitude of a speaker or a writer, in general, or with respect to a specific topic. In the complex variants of sentiment analysis, this attitude may be judgment, affective state or intended emotional communication. There are many good papers and resources describing methods to estimate sentiment. Some of these methods involve very complex algorithms.

Sentiment is a multidimensional construct, so it can be measured in many ways. Business applications are usually restricted to **polarity** (positive/negative). There are many ways of measuring polarity, all of them based on counting positive and negative terms.

Examples of positive and negative terms are:

- Positive terms: love, best, cool, great, good, amazing.
- Negative terms: hate, worst, sucks, awful, nightmare.

The sentiment analyst works on a corpus of documents, counting the occurrences of the terms of both lists in every document of the corpus. From these counts, different polarity can be calculated. In the examplke of this lecture, I explore how a company can measure customer sentiment using **Twitter data**. The 140-character limit imposed to tweets makes them manageable, and no normalization for the document length is needed. Moreover, although tweets may look as poor writing (they are), the length limit forces the writer to use a simple style which facilitates sentiment analysis.

The Twitter API

Most social networks, including Twitter, have an **application programming interface** (API). The API allows us to access the network database in a programmatic way (see `https://dev.twitter.com/overview/api` for an overview of the Twitter API). As in many other networks, the Twitter API requires the user to be authenticated. The **authentication** is carried out by means of a standard testing procedure called **OAuth**.

OAuth, defined as *an open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications* (see `http://oauth.net`), actually in version 2.0, is currently the leading choice for API developers. In practice, the procedure is simple: an authorization server issues a set of passwords, called **tokens**, to the API client, so he/she can use these tokens to access the resources hosted by a resource server.

In the case of Twitter, you have to make two steps:

- Register in Twitter, if you are not already registered. In this step, you create an account under a Twitter username, such as @miquelcanela.
- Create an application in the Twitter Applications website (`https://apps.twitter.com`). As a result, you get a set of four passwords: the **Consumer Key**, the **Consumer Secret**, the **Access Token** and the **Access Token Secret**. These four passwords are needed to authenticate you before starting a tweet search.

Strictly speaking, the Twitter API is not an API, but a set of API's. More specifically, there is the **REST API**, on which we perform downloads one-at-a-time, and the **Streaming API**, which involves a continuous connection and is used to monitor or process tweets in real-time. In this lecture, I use the **Twitter Search API**, which is an element of the REST API that allows queries by content.

Capturing Twitter data

There are many ways to download tweets. I list some of them.

- Directly from the API website, using the browser.
- Using Java, as explained in Kumar *et al.* (2013).
- Using Python, as explained in Russell (2013).
- Using PHP, as explained in Peri (2011).
- Using R, with the package `twitteR`.

In the example, I use `twitteR`, which is based on the Twitter Search API. As you will see, the function `searchTwitter` is quite easy to use, allowing us to specify an atomic **search string**, such as '@Delta', but also to combine several terms. The Twitter data consists in the text of the tweets plus **metadata**. Depending on the extraction process, the metadata can have more or less attributes.

I start with Twitter setup. The tokens are associated to a Twitter identity created especially for this course.

```
library(twitteR)
setup_twitter_oauth("9PQLhrb4ZUvte5Cb2VZRa7wce",
  "Xxkae5otxok1j30dwPNLgScJKzvLqSKJcvU77D13azYCDmoO1D",
  "2795394087-drbIfGPKwFZTfiNIKYXUqUrUvg1ypC2A7DfdXal",
  "FfN5Up4WtArKoLlitGUC2ukXKElmyA7paMpYphVxGQhlP")
```

```
## [1] "Using direct authentication"
```

If there is no error message, I have been authenticated, so I can start downloading tweets. There are various ways to select the tweets. I do it here with four arguments, which specify the search string, the maximum number of tweets to download (the default is 100) and the time interval.

I cannot dowload the data used in the example, since the Twitter Search API only offers free access to recent tweets (one-week), but I describe here the data collection process, using `searchString="@Delta"`, and the actual date. First, I set the range of dates as one week before today, which in R is `Sys.Date()`. Note that `Sys.Date()` comes in a date format, so I can subtract 7 days, although I transform it into a string to use it in `searchTwitter`.

```
date1 <- as.character(Sys.Date() - 7)
date2 <- as.character(Sys.Date())
twlist = searchTwitter(searchString="@Delta", n=100, since=date1, until=date2)
```

This creates a list of `status` objects. I skip these objects by transforming the list into a data frame with the function `twListToDF`.

```
twdf <- twListToDF(twlist)
```

Now, `twdf` is an ordinary data frame, has 100 rows and 16 columns, so I can work on it, leaving aside the package `twitteR` and its objects. I check the contents of this data frame (be careful with `str` here, it can give you trouble, due to the presence of special characters in the tweets).

```
dim(twdf)
```

```
## [1] 100  16
```

```
names(twdf)
```

```
##  [1] "text"          "favorited"     "favoriteCount" "replyToSN"
##  [5] "created"       "truncated"     "replyToSID"    "id"
##  [9] "replyToUID"    "statusSource"  "screenName"    "retweetCount"
## [13] "isRetweet"     "retweeted"     "longitude"     "latitude"
```

The lexicons

Many attempts have been done to develop **lexicons** for various purposes, in particular the lexicons of positive and negative words that are used in sentiment analysis. Those used in the example have been prepared by M Hu and B Liu (see Hu & Liu, 2004). They come in two text files, `positive-words.txt` and `negative-words.txt`. The lexicon of positive words contains 2,006 terms, starting with 'a+', 'abound', 'abounds', ... The lexicon of negative words contains 4,783 terms, starting with '2-faced', '2-faces', 'abnormal', ... I downloaded these files from `http://www.cs.uic.edu~liub/FBS/sentiment-analysis`.

Other choices are:

- **SentiWordNet** stems from WordNet, a lexical database of English developed at the University of Princeton. The first version of SentiWordNet was published in 2006. The current version (see Bacianella *et al.*, 2010) is 3.0. In SentiWordNet, each term comes with three numerical scores, positive, negative and neutral. The scores are non-negative and the sum of the scores is 1.
- The **Harvard General Inquirer** is a lexicon with syntactic, semantic and pragmatic information of the words included. It comes in spreadsheet format, and can be downloaded from `http://www.wjh.harvard.edu/~inquirer`. A list of 1,915 positive words and a list of 2,291 negative words can be easily extracted from the spreadsheet.
- The **Opinion Finder Subjectivity Lexicon** is a list of 8,8221 words scored for polarity (positive or negative). It is part of the Multi-Perspective Question Answering (MPQA) project, hosted by the University of Pittsburg. See Wilson *et al.* (2005) for details about its development.
- **AFINN** has been developed for sentiment analysis of microblogs (see Nielsen, 2011). It contains 2,477 words, scored from -5 to 5.
- **SentiSense** is a recent addition, also based on WordNet, which scores categories, not words. So, words have been first grouped into emotional categories. There is a version for Spanish.

Example: Consumer attitude towards airlines

Airlines rank very low among the different industries in the American Customer Satisfaction Index (ACSI). But there is no uniformity within the airline industry. At the beginning of 2015, among the six principal companies in the United States, JetBlue and Southwest was on top, followed by Delta. In this example, inspired in a talk of J Breen at the Boston Predictive Analytics MeetUp in 2011, I present an approach to measuring customer attitude towards airlines, based on sentiment analysis of Twitter data.

I describe with detail the analysis for Delta, which was replicated with other companies. In total, it involved six companies: American Airlines (AA) Delta Airlines (DL), JetBlue Airways (B6), Southwest Airlines (WN), United Airlines (UA) and US Airways (US).

As search strings, I used the Twitter names of the six companies: @AmericanAir, @Delta, @JetBlue, @SouthwestAir, @united and @USAirways. Omitting the '@' symbol could create trouble here, since some companies, like JetBlue, are easily identified, but 'Delta' could refer to other things (such as the delta of the Nile). The Twitter names allow an equal treatment of the six companies.

I limited the search to the week starting January 5 (Monday) and ending January 11 (Sunday), obtaining 22,344 tweets mentioning @AmericanAir, 8,075 for @Delta, 6,125 for @JetBlue, 8,189 for @SouthwestAir, 16,487 for @united and 6,069 for @USAirways. The data are stored in `RData` files, which contain a single object, a data frame called `twdf`.

The `twdf` data sets have 16 columns. The six relevant fields are:

- The text of the tweet (`text`). Example:
  `THANK YOU @Delta FOR THE FLIGHT DELAY. no, seriously. i'm going to need more time. #raretweet`.
- The screen name of the user who posted the tweet (`screenName`). Example: `thedressjunkie`.
- The ID of the tweet (`id`). Example: `551890878965768193`.
- When the tweet was posted (`created`). Example: `2015-01-05 00:00:13 UTC`.
- If this tweet is a retweet (`isRetweet`).
- The number of times this tweet has been retweeted (`retweetCount`).

The positive lexicon comes in the file `positive-words.txt`. The list of terms is preceded by some lines containing miscellaneous information about the lexicon. The lines containing comments start with semicolon (`;`), so they can be skipped when importing the list. The negative lexicon is in the file `negative-words.txt`.

I start importing the tweet data with the `load` function

```
load("Delta.RData")
```

Alternatively, the data can be imported by double-clicking the icon of the `RData` file. This would open the R console and load the objects contained in the file, in this case a single object, the data frame `twdf`. This can be checked with the function `ls`.

```
ls()
```

```
## [1] "twdf"
```

```
str(twdf)
```

```
## 'data.frame':    8075 obs. of  16 variables:
##  $ text         : chr  "Tonight's @delta flight from  got a toe-tapper / armrest rapper be
tter than a farter #businesstravel" "#fixitjesus apparently @delta can't" "I'm talking about
 you @delta" "@KLM @Delta ah thanks for the tip! We gaan er naar kijken. ^^ het moet idd een
 mooie trip worden" ...
##  $ favorited    : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ favoriteCount: num  0 0 5 0 0 0 0 7 0 0 ...
##  $ replyToSN    : chr  NA NA "_alligat0r_" "KLM" ...
##  $ created      : POSIXct, format: "2015-01-11 23:59:51" "2015-01-11 23:59:18" ...
##  $ truncated    : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ replyToSID   : chr  NA NA "554426967786029056" "554408110568534016" ...
##  $ id           : chr  "554427499225317376" "554427363660791808" "554427216646656001" "554
427155527254017" ...
##  $ replyToUID   : chr  NA NA "445222393" "56377143" ...
##  $ statusSource : chr  "<a href=\"http://twitter.com/download/android\" rel=\"nofollow\">T
witter for Android</a>" "<a href=\"http://twitter.com/download/iphone\" rel=\"nofollow\">Twit
ter for iPhone</a>" "<a href=\"http://twitter.com/download/iphone\" rel=\"nofollow\">Twitter
 for iPhone</a>" "<a href=\"http://twitter.com/download/iphone\" rel=\"nofollow\">Twitter for
 iPhone</a>" ...
##  $ screenName   : chr  "JK13" "jmcrouch77" "_alligat0r_" "Rdaalwill" ...
##  $ retweetCount : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ isRetweet    : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ retweeted    : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ longitude    : chr  NA NA NA NA ...
##  $ latitude     : chr  NA NA NA NA ...
```

I work on the first column of this data frame, `twdf$text` .

```
twdf$text[1:10]
```

```
##  [1] "Tonight's @delta flight from  got a toe-tapper / armrest rapper better than a farter
 #businesstravel"
##  [2] "#fixitjesus apparently @delta can't"

##  [3] "I'm talking about you @delta"

##  [4] "@KLM @Delta ah thanks for the tip! We gaan er naar kijken. ^^ het moet idd een mooie
 trip worden"
##  [5] "@Delta I love your sky magazine articles- especially this month's feature on @Americ
anExpress ceo Kenneth Chenault! http://t.co/U2SXuB0aEU"
##  [6] "@Delta wait, I bring a checked bag for quick weekend trips to not wait at bag clai
m...now there's no room for my one bag? I don't understand"
##  [7] "So glad the Captain of the plane flying from SJO --&gt; ATL told us the @packers wo
n! @Delta #GoPackGo @nfl #Playoffs"
##  [8] "It is refreshing to have a nice flight attendant. Thank you @Delta #GoodPeople #delt
a"
##  [9] "@Delta double booked my seat on existing reservation, forced off plane, but no sky c
lub pass although departure is 12 hrs later?  not cool"
## [10] "I want to talk to whoever is in charge of making the planes for @Delta, and let them
 know the planes are too small."
```

Preprocessing

I present next some transformations performed before counting positive and negative words. Some of these transformations are specific of this case. Some are not needed here, since the objective of the analysis is to assess the polarity of the messages, but they are typical of text mining, and help to have a clean picture of

the Twitter data. In most cases, I add extra white space to prevent terms merging.

I start by removing all **control characters**, such as `\n`, which indicates a line break. This can occur in Twitter data, since users are allowed to break lines within the tweet (see tweet 7,988). Note that, if a positive term, such as `good`, occurred after a line break, we would find `\ngood`, which would not be counted as positive. So, we need to take care of this.

```
twdf$text <- gsub(twdf$text, pattern="[[:cntrl:]]", replacement=" ")
```

Emoticons, such as `:)`, reflect positive or negative sentiment. Here, I transform `:)` and `:-)` into `smiley`, and `:(` and `:((` into `weeping`. I use regular expressions to shorten the code. The question mark in the first line means *zero or more times* and the plus sign in the second line *one or more times*. The square brackets are not needed in the first line, because a parenthesis can be closed without having been opened previously in a regular expression, but they are needed in the second line.

```
twdf$text <- gsub(twdf$text, pattern=":-?)", replacement=" smiley ")
twdf$text <- gsub(twdf$text, pattern=":[(]+", replacement=" weeping ")
```

As usual in this type of analysis, I drop apostrophes.

```
twdf$text <- gsub(twdf$text, pattern="'", replacement="")
```

The text can be contaminated with **HTML conventions**, such as `&` which is a **character entity** which stands for the ampersand symbol. Other character entities are `<` and `>`, which mean `<` and `>`, respectively. The second one appears frequently in these tweets, sometimes before an URL, sometimes to indicate a route, as in `ATL -> LAX`. I remove all this in one shot, again with a regular expression. The vertical bar means 'or', and the parenthesis is used to group strings.

```
twdf$text <- gsub(twdf$text, pattern="&([lg]t|amp);", replacement=" ")
```

I remove all the URL's, such as `http://t.co/ZRRj7H9eH9` in tweet 7. Note that this is not the real URL written in that tweet, but a fictitious one provided by Twitter to keep things short.

```
twdf$text <- gsub(twdf$text, pattern="http://[/a-zA-Z0-9.]+", replacement=" ")
```

I remove all the Twitter usernames, such as `@Delta` in tweet 1.

```
twdf$text <- gsub(twdf$text, pattern="@([A-Za-z]+[A-Za-z0-9]+)", replacement=" ")
```

I remove the dollar (`$`) and the **hash** (`#`) symbols.

```
twdf$text <- gsub(twdf$text, pattern="[$#])", replacement=" ")
```

I remove all the words starting with numbers.

```
twdf$text <- gsub(twdf$text, pattern="([0-9]+[A-Za-z0-9]+)", replacement=" ")
```

I remove punctuation.

```
twdf$text <- gsub(twdf$text, pattern="[[:punct:]]", replacement=" ")
```

It is typical to get trouble when processing Twitter data, due to the inclusion in the tweets of special characters and emojis. These characters would not affect my polarity calculations, but some of them can stop the conversion to lower case, which I cannot skip. Special characters may look differently in the R console in different computers. For instance, in my Windows IESE computer, I find in tweet 84 something presented as `\u0083`. At home, the same thing appears in my Macintosh as `\x83`. You can find it as `<U+0083>` somewhere else. It is a symbol called *no break here*, which can be depicted as a calligraphic *f*. So, I remove everything which is not letters or white space.

```
twdf$text <- gsub(twdf$text, pattern="[^a-zA-Z ]", replacement=" ")
```

I can perform now the conversion to lower case,

```
twdf$text <- tolower(twdf$text)
```

Finally, I strip white space.

```
twdf$text <- gsub(twdf$text, pattern=" +", replacement=" ")
twdf$text <- gsub(twdf$text, pattern="^ | $", replacement="")
twdf$text[1:10]
```

```
##  [1] "tonights flight from got a toe tapper armrest rapper better than a farter businesstr
avel"
##  [2] "fixitjesus apparently cant"

##  [3] "im talking about you"

##  [4] "ah thanks for the tip we gaan er naar kijken het moet idd een mooie trip worden"

##  [5] "i love your sky magazine articles especially this months feature on ceo kenneth chen
ault"
##  [6] "wait i bring a checked bag for quick weekend trips to not wait at bag claim now ther
es no room for my one bag i dont understand"
##  [7] "so glad the captain of the plane flying from sjo atl told us the won gopackgo playof
fs"
##  [8] "it is refreshing to have a nice flight attendant thank you goodpeople delta"

##  [9] "double booked my seat on existing reservation forced off plane but no sky club pass
 although departure is hrs later not cool"
## [10] "i want to talk to whoever is in charge of making the planes for and let them know th
e planes are too small"
```

Polarity measurement

Now comes my first attempt to evaluate the polarity of the tweets. I use the function `scan` to import the lexicons, because it allows input files with comment lines, which it controls with the argument `comment.char`.

```
pos.terms = scan("positive-words.txt", what="character", comment.char=";")
neg.terms = scan("negative-words.txt", what="character", comment.char=";")
```

This creates two character vectors of lengths 2,006 and 4,783, respectively. Breen suggests adding ‘upgrade’ to the positive terms, and I also added ‘smiley’, explained above. To the list of negative words, I add some suggested by Breen, some on my own (you can probably do better).

```
pos.terms = c(pos.terms, "smiley", "upgrade")
neg.terms = c(neg.terms, "cancel", "cancelled", "fix", "weeping", "fucked", "mechanical",
  "mistreat", "mistreated", "piss", "pissed", "shitty", "stranded", "wait", "waiting", "wtf")
```

I define two functions that count the matches with the positive and negative lexicons, respectively.

```
posCount <- function(x) sum(x %in% pos.terms)
negCount <- function(x) sum(x %in% neg.terms)
```

To apply these functions to the Delta corpus, I create a list of length 8,075, whose terms are bags of words for the tweets, splitting the messages.

```
term.list = strsplit(twdf$text, " ")
```

Now, I apply my counting functions to every element of `term.list` with `lapply`. The output of `lapply` is a list, which I transform into a vector with `unlist`.

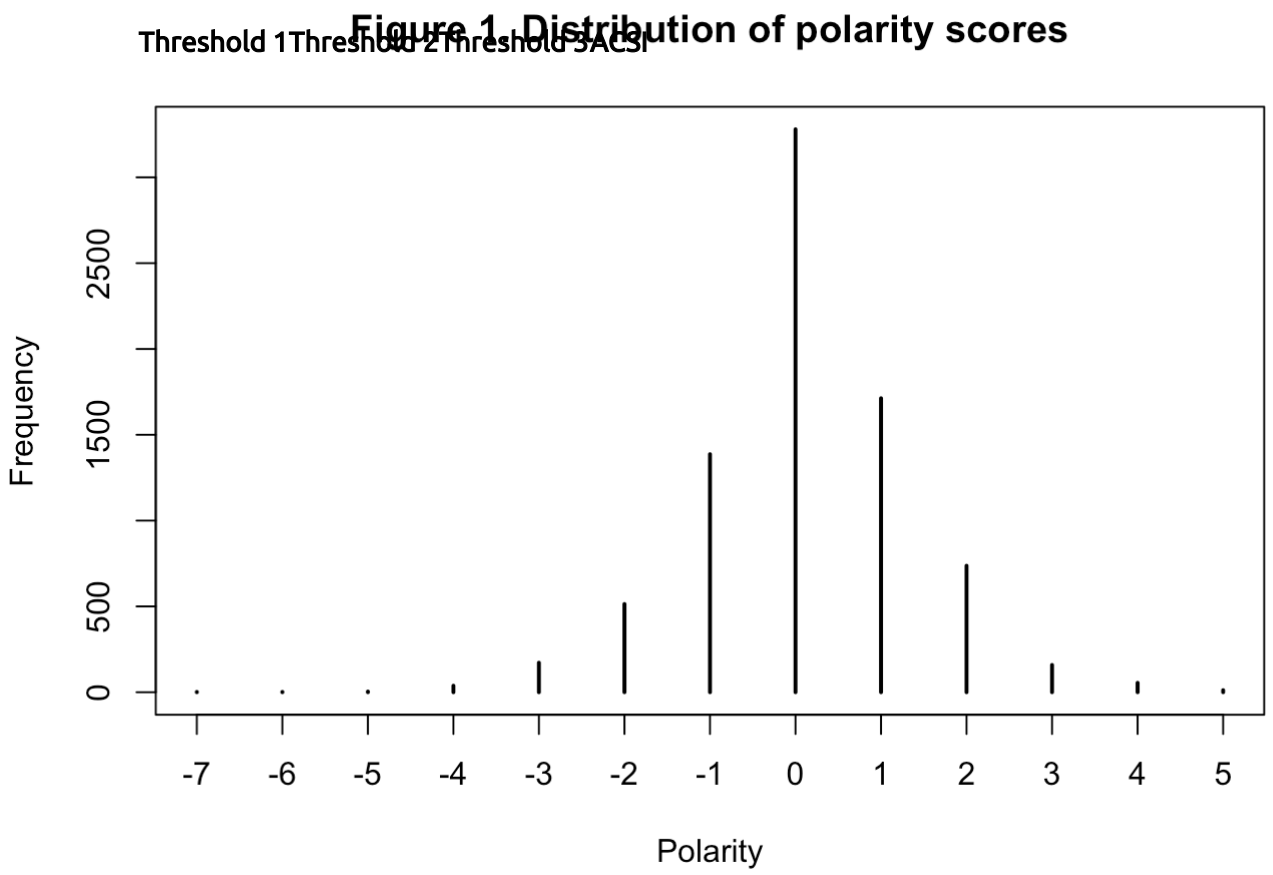
```
positive <- unlist(lapply(term.list, posCount))
negative <- unlist(lapply(term.list, negCount))
```

This leaves me with two numeric vectors of length 8,075. The polarity score is obtained as the difference.

```
polarity <- positive - negative
```

This type of sentiment measure is usually normalized, dividing by the size of the document. This accounts for the different size of the documents that integrate the corpus. Such normalization is not needed here, since one of the advantages of Twitter data is that the 140-character restriction grants a reasonable uniformity in the size of the tweets. The distribution of the polarity scores can be seen in Figure 1.

```
plot(table(polarity), type="h", main="Figure 1. Distribution of polarity scores",
  xlab="Polarity", ylab="Frequency")
```



In my first analysis, I follow a suggestion of Breen, calculating a sentiment measure for each company as the ratio of the number of tweets with polarity 2 or more by the number of tweets with polarity -2 or less. For Delta, this gives us 1.316.

I have collected in Table 1, column 3, the results of this approach, for the six companies. The table also shows the results of changing the threshold to 1 and 3. I have posted in column 4 the ACSI scores for year 2014. The picture is consistent except for American Airlines.

```
sum(polarity>=1)/sum(polarity<=-1)
```

```
## [1] 1.26286
```

```
sum(polarity>=2)/sum(polarity<=-2)
```

```
## [1] 1.315574
```

```
sum(polarity>=3)/sum(polarity<=-3)
```

```
## [1] 1.036697
```

TABLE 1. Results for the six airlines

	Threshold 1	Threshold 2	Threshold 3	ACSI
American	1.085	1.772	3.907	66
Delta	1.263	1.316	1.037	71

JetBlue	1.872	3.170	5.667	79
Southwest	1.932	1.340	0.909	48
United	0.494	0.340	0.238	60
US Airways	0.523	0.465	0.346	66

Prop. non-retweet Threshold 1 Threshold 2 Threshold 3

Discarding retweets

As you probably guess, tweets starting with 'RT' are **retweets**, that is, redundant information. So, some analysts ignore them, restricting the analysis to the non-retweets (you may disagree). In our data set, this is easy to manage with the field `isRetweet` For our six companies, the percentage of retweet ranges between 20 and 40%.

To get this, I define a filter based on `isRetweet` .

```
filter <- twdf$isRetweet==F
sum(filter)/nrow(twdf)
```

```
## [1] 0.7442724
```

I recalculate the sentiment measure.

```
sum(polarity>=1 & filter)/sum(polarity<=-1 & filter)
```

```
## [1] 1.018847
```

```
sum(polarity>=2 & filter)/sum(polarity<=-2 & filter)
```

```
## [1] 0.9196429
```

```
sum(polarity>=3 & filter)/sum(polarity<=-3 & filter)
```

```
## [1] 0.8932039
```

The results for this restricted corpus are presented in Table 2. The picture is a bit more consistent here than in Table 1. The ratios are lower here, suggesting that the negative tweets are less retweeted.

TABLE 2. Results for the six airlines (only non-retweets)

	Prop. non-retweet	Threshold 1	Threshold 2	Threshold 3
American	0.629	0.686	0.538	0.432
Delta	0.744	1.019	0.920	0.893
JetBlue	0.701	1.533	1.944	2.652
Southwest	0.814	1.737	1.436	0.909
United	0.748	0.544	0.380	0.321
US Airways	0.628	0.572	0.463	0.373

Alternative polarity measure

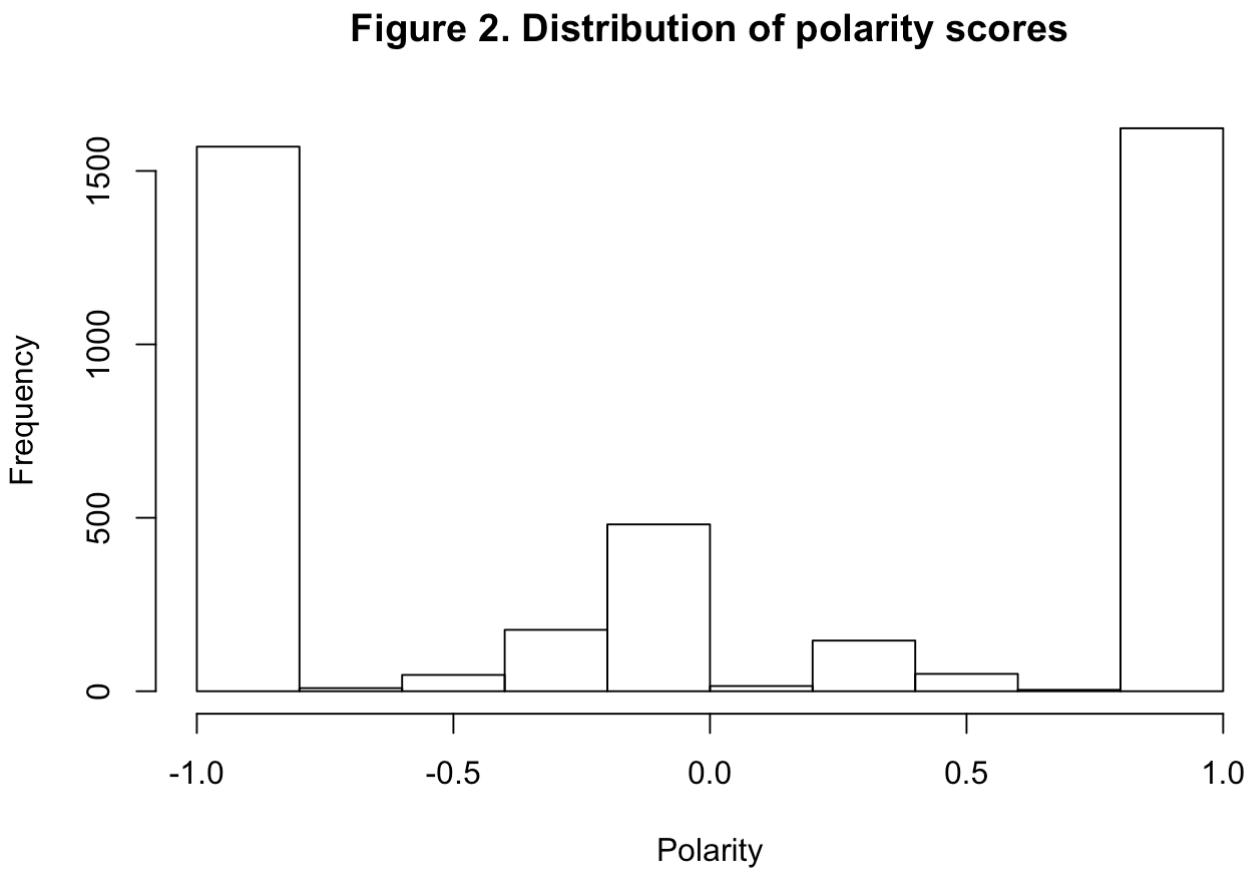
Finally, my own approach to polarity, based on readings. For every tweet, I define a polarity score as the following ratio.

```
polarity <- (positive - negative)/(positive + negative)
```

Prop. neutralAverage polarity

By definition, this measure is already normalized, taking values from -1 to 1. Note that, when a tweet does not contain positive nor negative words, this is 0/0, so R will produce a `NaN` value for these tweets. The distribution of this polarity measure in the Delta corpus (not including the retweets and `NaN` values) is seen in Figure 2, obtained as follows.

```
hist(polarity[filter], main='Figure 2. Distribution of polarity scores', xlab='Polarity')
```



Note that most of the tweets fall on the extremes, which is convenient for my analysis. Table 3 shows the average polarity for the six companies. The proportion of neutral tweets (not containing positive nor negative words) is quite uniform across companies. Sorting the six companies by the average polarity is consistent with the ACSI ranking. The calculations were performed as follows.

```
sum(positive==negative & filter)/sum(filter)
```

```
## [1] 0.39401
```

```
mean(polarity[filter], na.rm=T)
```

```
## [1] 0.01129017
```

TABLE 3. Polarity analysis

Prop. neutralAverage polarity

American	0.430	-0.151
Delta	0.394	0.011
JetBlue	0.426	0.189
Southwest	0.348	0.239
United	0.388	-0.245
US Airways	0.381	-0.221

Prop. neutral Average polarity

References

[1] S Baccianella, A Esuli & F Sebastiani (2010), SentiWordNet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining, *Proceedings of the Seventh Conference on International Language Resources and Evaluation*, 2200-2204.

[2] R Boyd (2012), *Getting Started with OAuth 2.0*, O'Reilly.

[3] `http://www.cs.uic.edu~liub/FBS/sentiment-analysis` .

[4] `http://www.wjh.harvard.edu/$\sim$inquirer` .

[5] M Hu & B Liu (2004), Mining and Summarizing Customer Reviews, *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004)*, Seattle (Washington, USA).

[6] S Kumar (2013), *Twitter Data Analytics*, Springer.

[7] FA Nielsen (2011), A new ANEW: Evaluation of a word list for sentiment analysis in microblogs, arXiv:1103.2903v1.

[8] MA Russell (2013), *Mining the Social Web*, O'Reilly.

[9] T Wilson, J Wiebe & P Hoffmann (2005), Recognizing contextual polarity in phrase-level sentiment analysis, *Proceedings of HLT-EMNLP-2005*.