Assignment for application Data Science track in Information Studies

Konstantinos Papastamos

28 February 2017

Sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic. The attitude may be his or her judgment or evaluation, or an intended emotional communication. Sentiment analysis is widely applied to reviews and social media for a variety of applications, ranging from marketing to customer service. The provided dataset (imdb-reviews.tsv) is a tab-separated file and consists of 1000 IMDB movie reviews. Data fields in the file are:

- id - Unique ID of each review
- sentiment - Sentiment of the review; 1 for positive reviews and 0 for negative reviews
- review - Text of the review The sentiment of reviews is binary, meaning the IMDB rating < 5 results in a sentiment score of 0, and rating $>=7$ have a sentiment score of 1.

The Assignment:

Design and implement a class that can preprocess and store the IMDB reviews. Specifically, the methods of your class should take as an input the IMDB file of reviews, and extract each review. For each polarity class (i.e. positive or negative) construct a vector of size equal to the entire vocabulary of words used in all the IMDB reviews, vec_positive, and vec_negative. Each position in the two vectors should correspond to the same unique word, that is vec_positive[33] and vec_negative[33] should correspond to the same word. Populate the two vectors with the number of times each word appears in the positive and negative reviews.

Q1) Write a method that plot the two histograms; plot the two histograms and discuss your findings (in three lines of text).

Q2) Write a method that implements a statistical test, to test whether the two histograms (or distributions) are statistically significantly different or not. Run your method and discuss your findings (in three lines of text).

Q3) Write a method that finds the top-5 most frequent words in the positive reviews and the top-5 most frequent words in the negative reviews. Run your method and discuss your findings (in three lines of text). Can you suggest any way you could find the most discriminative words in each class of polarity?

My Solution:

The assignment will be completed using R and the RStudio platform.

First we need to download and take a look at the given Data.

```
if (!("stringr" %in% rownames(installed.packages()))){

  install.packages("stringr")
  library("stringr")

} else {

  library("stringr")
```

```
}


link = paste("http://gss.uva.nl/binaries/content",
             "/assets/programmas/information-studies",
             "/ds/imdb-reviews---assigment-for-application-2017",
             "-2018-data-science---uva.txt?2951705059991",sep="")

if(!file.exists("reviews.txt")) {

  download.file(link,  destfile = "reviews.txt")

}
cc =c("character","factor","character")

reviews = read.table("reviews.txt", header = T, colClasses=cc)

names(reviews) = c("id","sentiment","review")

dim(reviews)
```

```
[1] 1000    3
```

```
levels(reviews$sentiment)
```

```
[1] "0" "1"
```

So we have a dataset of 1000 rows and 3 columns, the first column is a unique ID of type character, the second is a factor that takes the values 1 symbolizing a positive sentiment and 0 symbolizing a negative one. Finally there is the review column that contains the actual text of the review.

As requested, my first task is for each polarity class (i.e. positive or negative) to construct a vector of size equal to the entire vocabulary of words used in all the IMDB reviews, vec_positive and vec_negative.

In order to accomplish the task we will create two supporting vectors, one containing all the words used in positive reviews and one containing the words used in negative reviews.

```
pos_words = reviews[which(reviews[,2]==1),3]
neg_words = reviews[which(reviews[,2]==0),3]
```

In these two vectors we won't remove duplicate values so that we know how many times a word appears in a positive or negative review.

Our next step is to create a function that will preprocess our data, remove unwanted values and make any necessary transformations

```
if (!("tm" %in% rownames(installed.packages()))){

  install.packages("tm")
  library("tm")

} else {

  library("tm")

}

text_preproc <- function(data_vector){
```

```
  # Here I start by pasting all the different reviews, to create one text containing
  # all the words used.
  words = paste( data_vector, collapse = ' ')

  # Switch everything to lowercase
  words = tolower(words)

  # Remove anything that is not a letter (I assume that numbers and punctuation
  # don't contain valuable information for our analysis).
  words = str_replace_all(words,"[^a-zA-Z]", " ")

  # Reduce the whitespace down to one so I can effectively split the words later.
  words = gsub("\\s+"," ",words)

  # At this point we are left with a character vector containing
  # all the reviews in a single text value.
  # Every word is separated from the next by a whitespace,
  # so now I am going to split them using the whitespace
  # as the separator. Then I use the unlist function to
  # change the returned list to a character vector.

  words = unlist( strsplit( words, " "))

  #Remove any remaining whitespace and empty strings
  words = gsub(" ","", words)
  words=words[words!=""]

  # Here I am going to take an initiative and remove all stopwords from the vector.
  # Stopwords are words that do not provide us with any information about the sentiment
  # of the review (i.e. "and","the","I","you")
  # This will be done using the text mining package 'tm'
  # In an effort to reduce the noise in the dataset, I will also remove one-letter
  # values and the two letter value "br" used in the html tags many times in the reviews

  words=words[words!="br"]
  words=words[nchar(words)!=1]

  '%nin%' <- Negate('%in%')
  stopWords <- stopwords("en")
  words = words[words %nin% stopWords]

  return(words)

}
```

Now let's use the function we created to clean the two vectors and the table() function to get the frequency of each word:

```
pos_words_clean = table( text_preproc(pos_words))
neg_words_clean = table( text_preproc(neg_words))

 # Let's take a look at the new vector
sample(pos_words_clean, size=10, replace=FALSE)
```

```
performance        read       royal         cal      fenech  wondrously
         82          34           3           1           1           1
 abdicating        shin    incident      doomed
          1           2           4           3
```

Finally lets take the entire vocabulary of words and clean it using the same method:

```
vocabulary = text_preproc(reviews$review)

  # Now let's remove duplicate values and sort the vector.
vocabulary = unique(sort(vocabulary))
```

Next I will create vec_positive and vec_negative. The two vectors will be of type numeric and have the same size as the vocabulary vector. In this way the position of each value in the two vectors will correspond to the same word, found in the same position in the vocabulary vector. Then I will populate each one of the two vectors with the number of occurences of each word in the positive and negative reviews respectively.

```
 # Initialize the two vectors
vec_positive = rep(0,length(vocabulary))
vec_negative = rep(0,length(vocabulary))

for (i in 1:length(vocabulary)){
    vec_positive[i]=as.integer(pos_words_clean[vocabulary[i]])
}

for (i in 1:length(vocabulary)){
    vec_negative[i]=as.integer(neg_words_clean[vocabulary[i]])
}

vec_positive[is.na(vec_positive)]=0
vec_negative[is.na(vec_negative)]=0

names(vec_positive) = vocabulary
names(vec_negative) = vocabulary

 # Let's take a look at the positive vector
sample(vec_positive, 10, replace = FALSE)
```

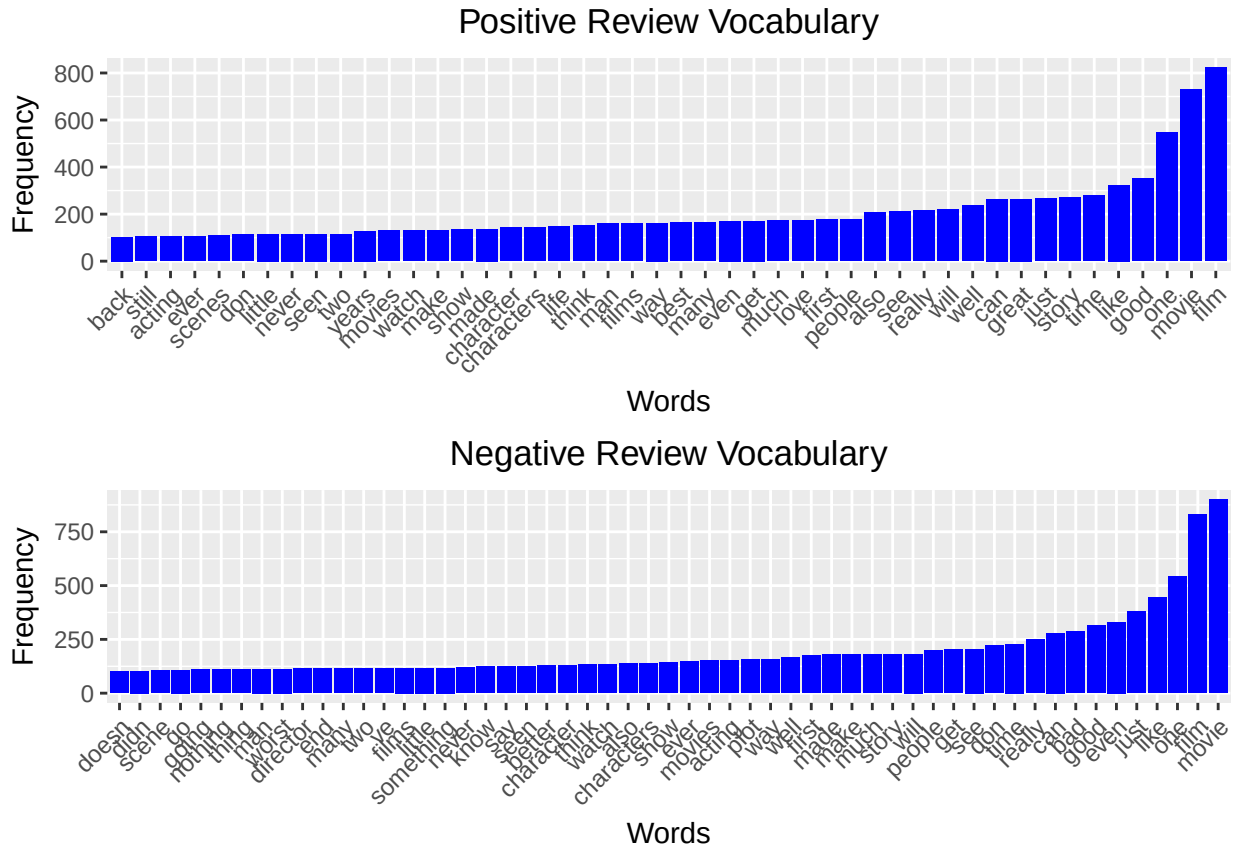
```
     virgin     victims        tsui    happened      carved   breakfast
          8           2           3          19           1           2
     stairs        dyan maintenance     dresser
          2           2           0           1
```

Q1

Now I will proceed to Q1 and create a method that takes the two vectors as an input and creates the requested plots. I will use the ggplot2 package to create the plots and the gridExtra package to arrange them. I will also create the histogram for words with frequency higher that 100, or else it will be too dense to be interpreted.

```
create_plots <- function(pos_vector, neg_vector){

  if (!("ggplot2" %in% rownames(installed.packages()))){

  install.packages("ggplot2")
  library("ggplot2")
```

```
} else {

  library("ggplot2")

}

if (!("gridExtra" %in% rownames(installed.packages()))){

  install.packages("gridExtra")
  library("gridExtra")

} else {

  library("gridExtra")

}
  #In order to create the plots we will convert the two vectors to data frames
  #with one column being the word and the second the frequency.

pos_df = data.frame(word = names(pos_vector), freq = pos_vector)
neg_df = data.frame(word = names(neg_vector), freq = neg_vector)

pos_df$word = factor(pos_df$word, levels = pos_df$word[order(pos_df$freq)])
neg_df$word = factor(neg_df$word, levels = neg_df$word[order(neg_df$freq)])

pos_histogram = ggplot(subset(pos_df, freq>100), aes(word, freq))
pos_histogram = pos_histogram + xlab("Words") + ylab("Frequency")
pos_histogram = pos_histogram + geom_bar(stat="identity", fill="blue")
pos_histogram = pos_histogram + ggtitle("Positive Review Vocabulary")
pos_histogram = pos_histogram + theme(plot.title = element_text(hjust = 0.5))
pos_histogram = pos_histogram + theme(axis.text.x=element_text(angle=45, hjust=1))

neg_histogram = ggplot(subset(neg_df, freq>100), aes(word, freq))
neg_histogram = neg_histogram + xlab("Words") + ylab("Frequency")
neg_histogram = neg_histogram + geom_bar(stat="identity", fill="blue")
neg_histogram = neg_histogram + ggtitle("Negative Review Vocabulary")
neg_histogram = neg_histogram + theme(plot.title = element_text(hjust = 0.5))
neg_histogram = neg_histogram + theme(axis.text.x=element_text(angle=45, hjust=1))

grid.arrange(pos_histogram, neg_histogram, ncol=1)
}

create_plots(vec_positive,vec_negative)
```



Q1 Results

After plotting the two vectors, it is clear that many of the most frequent words are common in both classes of polarity. In lower frequencies we can see that there are some discriminative words (e.g. best, worst) but still most of the words are neutral.

Q2

For Q2 I will create a method that performs a paired t test to check if the two distributions are statistically significantly different. The null hypothesis is that the true mean difference is zero while the alternative hypothesis is that the true difference is not equal to zero.

```
t_test <- function(x,y){

  sum_of_dif = sum(x-y)
  sum_of_sqrt_dif = sum((x-y)^2)

  size = length(x)

  t = (sum_of_dif/size) / sqrt((sum_of_sqrt_dif - (sum_of_dif^2)/size)/ ( size*(size-1) ))
  t = abs(t)
  t = round(t,4)

  p_val = 2*pt(t, df=size-1, lower=FALSE)
  p_val = round(p_val, 4)
```

```
  m = mean(x-y)
  error = qt(0.975,df = size-1)*sd(x-y)/sqrt(size)
  ci_right = round(m + error, 5)
  ci_left = round(m - error, 5)

  results = paste("t = ", t, "\nP Value = ", p_val, "\nDegrees of Freedom =", size-1)
  results = paste(results,"\nConfidence Interval: ",ci_left,"< mean < ",ci_right)
  return(results)
}

cat(t_test(vec_negative,vec_positive))
```

```
t =  1.7408
P Value =  0.0817
Degrees of Freedom = 17974
Confidence Interval:  -0.15411 < mean <  0.00913
```

Q2 Results

For significance level= 0.05 we can see that p>0.05. Also the calculated confidence interval states that 95% of the time, the true mean lies between the values -0.15411 and 0.00913. Thus we fail to reject the null hypothesis and we conclude that the two distributions are not significantly different.

Q3

Finally for Q3 I will create a short method that finds the 5 most frequent words in the positive and the 5 most frequent words in the negative reviews.

```
top_five <- function(positive,negative){

  positive = sort(positive, decreasing = TRUE)
  negative = sort(negative, decreasing = TRUE)

  col1 = paste(names(positive)[1:5],positive[1:5],sep=" ")
  col2 = paste(names(negative)[1:5],negative[1:5],sep=" ")

  results = data.frame(col1, col2)
  names(results) = c("Top Five Positive Words","Top Five Negative Words")

  print(results)
}

top_five(vec_positive,vec_negative)
```

```
  Top Five Positive Words Top Five Negative Words
1               film 823              movie 899
2              movie 731               film 830
3                one 547                one 541
4               good 350               like 445
5               like 324               just 380
```

Q3 Results

The top five words in both classes of polarity are very similar with a small difference in frequency. However there are almost no words that express a positive or negative sentiment in both cases.

Most Discriminative Words

A very simple way to get the most discriminative words in each class of polarity is to take the frequency difference for each word between the two vectors, and only keep the words that have a frequency difference over a certain threshold.

```
freq_diff = vec_positive - vec_negative

size = length(freq_diff)

freq_diff = sort(freq_diff, decreasing = TRUE)

col1 = names(freq_diff)[1:20]
col2 = names(freq_diff)[size:(size-19)]

cnames=c("Most Discriminative Positive Words","Most Discriminative Negative Words")

df = data.frame(col1, col2)
names(df) = cnames

df
```

```
   Most Discriminative Positive Words Most Discriminative Negative Words
1                               great                                bad
2                                love                              movie
3                               story                               even
4                                best                               like
5                                also                               just
6                                well                                don
7                                life                              worst
8                         performance                               plot
9                           excellent                            nothing
10                               many                             better
11                               time                              awful
12                                man                               poor
13                              years                              waste
14                               role                            minutes
15                             always                              thing
16                              films                              point
17                              young                               make
18                          beautiful                              worse
19                             family                             acting
20                               real                               didn
```

In the new dataframe, even though some neutral words still exist, it is clear that we have extracted the most discriminative words for each class of polarity.