Trabalho Final da Júlia Raíces

Primeiro seguem os arquivos...

1. da função: funcao_final.R
2. do help: help_funcao_final.txt
3. e os arquivos de entrada possíveis...
    - a lista de palavras e scores: pode ser baixado aqui
    - um texto de verdade para ser analisado: stoya.txt
    - um texto de mentirinha para ser analisado ("texto" criado a partir de palavras com score e que não faz sentido): test.txt

A Função

```
########## Function to assert a sentiment analysis of a given text
##########
####### Author: Julia Raices - noUSP: 6802291 #######

sentiment.analysis <- function(your.text,
word.list.and.scores="WordScores.txt"){ # this line asserts the function
name and files it will use.
    unclean.text <- scan(your.text, character(0)) #separate each word from
the file (accordin to R wiki ^^) # and it actually does so! and in text[n]
is stored the Nth word from the text =)
    text <- gsub("[^[:alpha:][:space:]'']", "", unclean.text) # #Vivi helped
me on thi one... because I had to remove all punctuation except for the
apostrophes. So here I only keep the alphabet characters, the spaces and the
apostophes....
    text <- tolower(text) #assures every thing is lower case, so there is no
mismacth because of case
    score.text=0 # equals the text score to zero, so there is no mistake
when we start the function
    positive=0 # equals the number of positive words to zero, to prevent
future mistakes
    negative=0 # equals the number of negative words to zero, to prevent
future mistakes
    neutral=0 # equals the number of neutral words to zero to prevent future
mistakes
    leng <- length(text)#gets and stores the size (in words) of your file
    LIST <- read.table(word.list.and.scores, sep="\t", header=F,
strip.white=T, blank.lines.skip=T, col.names=c("words", "scores"), as.is=T)
#.. opens your score file if you gave one, and opens the default file, if
you didn't gave one
    len <- length(LIST$words)# gets the length (number of words) in the word
score file
    LIST$word <- tolower(LIST$words)# also makes sure the words in the list
are all lower case, so there is no mismacth because of the case.
    for(i in 1:len){ # for the length of the score file do the following:
```

```
        for(j in 1:leng){# for the length of the text file do the following:
            if(text[j]==LIST$word[i]){ #if you can find the word from the
text in the list of words...
                two.space <- paste(LIST$word[i], LIST$word[i+1], sep=" ") #
creates a string with the word that was not in the list and the next one,
separate them with a space
                two.dash <- paste(LIST$word[i], LIST$word[i+1], sep="-") #
creates a string with the word that was not in the list and the next one,
separate them with a dash
                three.space <- paste(LIST$word[i-1], LIST$word[i],
LIST$word[i+1], sep=" ") # creates a string with the word that was not in
the list and the next two, separate them with a space
                three.dash <- paste(LIST$word[i-1], LIST$word[i],
LIST$word[i+1], sep="-") # creates a string with the word that was not in
the list and the next two, separate them with a dash
                four.space <- paste(LIST$word[i], LIST$word[i+1],
LIST$word[i+2], LIST$word[i+3], sep=" ") # creates a string with the word
that was not in the list and the next three, separate them with a space
                four.dash <- paste(LIST$word[i-1], LIST$word[i],
LIST$word[i+1], LIST$word[i+3], sep="-") # creates a string with the word
that was not in the list and the next three, separate them with a dash
                if(four.space==LIST$word[i]){ # checks if the new string
created with the two consecutive words
                    score.text <- score.text + LIST$scores[i] # increase the
text score by the word score.
                    if(LIST$scores[i] > 0){ # checks if the word is positive
or negative
                        positive <- positive + 1 # if it's positive, add one
to the positive-words couter
                    }# closes bracket from if the word is positive
                    else if(LIST$scores[i] < 0){ # checks if the word is
positive or negative
                        negative <- negative + 1 # if it's negative, add one
to the negative-words counter
                    }# closes bracket from if the word is negative
                } # close brackets for the created string
                else if(four.dash==LIST$word[i]){ # checks if the new string
created with the two consecutive words
                    score.text <- score.text + LIST$scores[i] # increase the
text score by the word score.
                    if(LIST$scores[i] > 0){ # checks if the word is positive
or negative
                        positive <- positive + 1 # if it's positive, add one
to the positive-words couter
                    }# closes bracket from if the word is positive
                    else if(LIST$scores[i] < 0){ # checks if the word is
positive or negative
                        negative <- negative + 1 # if it's negative, add one
to the negative-words counter
                    }# closes bracket from if the word is negative
```

```
                } # close brackets for the created string
                else if(three.space==LIST$word[i]){ # checks if the new
string created with the two consecutive words
                    score.text <- score.text + LIST$scores[i] # increase the
text score by the word score.
                    if(LIST$scores[i] > 0){ # checks if the word is positive
or negative
                        positive <- positive + 1 # if it's positive, add one
to the positive-words couter
                    }# closes bracket from if the word is positive
                    else if(LIST$scores[i] < 0){ # checks if the word is
positive or negative
                        negative <- negative + 1 # if it's negative, add one
to the negative-words counter
                    }# closes bracket from if the word is negative
                } # close brackets for the created string
                else if(three.dash==LIST$word[i]){ # checks if the new
string created with the two consecutive words
                    score.text <- score.text + LIST$scores[i] # increase the
text score by the word score.
                    if(LIST$scores[i] > 0){ # checks if the word is positive
or negative
                        positive <- positive + 1 # if it's positive, add one
to the positive-words couter
                    }# closes bracket from if the word is positive
                    else if(LIST$scores[i] < 0){ # checks if the word is
positive or negative
                        negative <- negative + 1 # if it's negative, add one
to the negative-words counter
                    }# closes bracket from if the word is negative
                } # close brackets for the created string
                else if(two.space==LIST$word[i]){ # checks if the new string
created with the two consecutive words is in the list
                    score.text <- score.text + LIST$scores[i] # increase the
text score by the word score.
                    if(LIST$scores[i] > 0){ # checks if the word is positive
or negative
                        positive <- positive + 1 # if it's positive, add one
to the positive-words couter
                    }# closes bracket from if the word is positive
                    else if(LIST$scores[i] < 0){ # checks if the word is
positive or negative
                        negative <- negative + 1 # if it's negative, add one
to the negative-words counter
                    }# closes bracket from if the word is negative
                } # close brackets for the created string
                else if(two.dash==LIST$word[i]){ # checks if the new string
created with the two consecutive words is in the list
                    score.text <- score.text + LIST$scores[i] # increase the
text score by the word score.
                    if(LIST$scores[i] > 0){ # checks if the word is positive
```

```
or negative
                        positive <- positive + 1 # if it's positive, add one
to the positive-words couter
                    }# closes bracket from if the word is positive
                    else if(LIST$scores[i] < 0){ # checks if the word is
positive or negative
                        negative <- negative + 1 # if it's negative, add one
to the negative-words counter
                    }# closes bracket from if the word is negative
                } # close brackets for the created string
                else { # else for if there is no grouping of the word on the
list that was also on the list
                    score.text <- score.text + LIST$scores[i] # increase the
text score by the word score.
                    if(LIST$scores[i] > 0){ # checks if the word is positive
or negative
                        positive <- positive + 1 # if it's positive, add one
to the positive-words couter
                    }# closes bracket from if the word is positive
                    else if(LIST$scores[i] < 0){ # checks if the word is
positive or negative
                        negative <- negative + 1 # if it's negative, add one
to the negative-words counter
                    }# closes bracket from if the word is negative
                } # close else from if there was no grouping of the word on
the list that was also on the list
            }#close brackets from the "if the word from the text is on the
list"
            else{ # if the single word is not in the list, we try
combination of the word with the following one and two, to assert
expressions
                two.space <- paste(LIST$word[i], LIST$word[i+1], sep=" ") #
creates a string with the word that was not in the list and the next one,
separate them with a space
                two.dash <- paste(LIST$word[i], LIST$word[i+1], sep="-") #
creates a string with the word that was not in the list and the next one,
separate them with a dash
                three.space <- paste(LIST$word[i-1], LIST$word[i],
LIST$word[i+1], sep=" ") # creates a string with the word that was not in
the list and the next two, separate them with a space
                three.dash <- paste(LIST$word[i-1], LIST$word[i],
LIST$word[i+1], sep="-") # creates a string with the word that was not in
the list and the next two, separate them with a dash
                four.space <- paste(LIST$word[i], LIST$word[i+1],
LIST$word[i+2], LIST$word[i+3], sep=" ") # creates a string with the word
that was not in the list and the next three, separate them with a space
                four.dash <- paste(LIST$word[i-1], LIST$word[i],
LIST$word[i+1], LIST$word[i+3], sep="-") # creates a string with the word
that was not in the list and the next three, separate them with a dash
                if(four.space==LIST$word[i]){ # checks if the new string
```

```
created with the two consecutive words
                    score.text <- score.text + LIST$scores[i] # increase the
text score by the word score.
                    if(LIST$scores[i] > 0){ # checks if the word is positive
or negative
                        positive <- positive + 1 # if it's positive, add one
to the positive-words couter
                    }# closes bracket from if the word is positive
                    else if(LIST$scores[i] < 0){ # checks if the word is
positive or negative
                        negative <- negative + 1 # if it's negative, add one
to the negative-words counter
                    }# closes bracket from if the word is negative
                } # close brackets for the created string
                else if(four.dash==LIST$word[i]){ # checks if the new string
created with the two consecutive words
                    score.text <- score.text + LIST$scores[i] # increase the
text score by the word score.
                    if(LIST$scores[i] > 0){ # checks if the word is positive
or negative
                        positive <- positive + 1 # if it's positive, add one
to the positive-words couter
                    }# closes bracket from if the word is positive
                    else if(LIST$scores[i] < 0){ # checks if the word is
positive or negative
                        negative <- negative + 1 # if it's negative, add one
to the negative-words counter
                    }# closes bracket from if the word is negative
                } # close brackets for the created string
                else if(three.space==LIST$word[i]){ # checks if the new
string created with the two consecutive words
                    score.text <- score.text + LIST$scores[i] # increase the
text score by the word score.
                    if(LIST$scores[i] > 0){ # checks if the word is positive
or negative
                        positive <- positive + 1 # if it's positive, add one
to the positive-words couter
                    }# closes bracket from if the word is positive
                    else if(LIST$scores[i] < 0){ # checks if the word is
positive or negative
                        negative <- negative + 1 # if it's negative, add one
to the negative-words counter
                    }# closes bracket from if the word is negative
                } # close brackets for the created string
                else if(three.dash==LIST$word[i]){ # checks if the new
string created with the two consecutive words
                    score.text <- score.text + LIST$scores[i] # increase the
text score by the word score.
                    if(LIST$scores[i] > 0){ # checks if the word is positive
or negative
                        positive <- positive + 1 # if it's positive, add one
```

```
to the positive-words couter
                    }# closes bracket from if the word is positive
                    else if(LIST$scores[i] < 0){ # checks if the word is
positive or negative
                        negative <- negative + 1 # if it's negative, add one
to the negative-words counter
                    }# closes bracket from if the word is negative
                } # close brackets for the created string
                else if(two.space==LIST$word[i]){ # checks if the new string
created with the two consecutive words is in the list
                    score.text <- score.text + LIST$scores[i] # increase the
text score by the word score.
                    if(LIST$scores[i] > 0){ # checks if the word is positive
or negative
                        positive <- positive + 1 # if it's positive, add one
to the positive-words couter
                    }# closes bracket from if the word is positive
                    else if(LIST$scores[i] < 0){ # checks if the word is
positive or negative
                        negative <- negative + 1 # if it's negative, add one
to the negative-words counter
                    }# closes bracket from if the word is negative
                } # close brackets for the created string
                else if(two.dash==LIST$word[i]){ # checks if the new string
created with the two consecutive words is in the list
                    score.text <- score.text + LIST$scores[i] # increase the
text score by the word score.
                    if(LIST$scores[i] > 0){ # checks if the word is positive
or negative
                        positive <- positive + 1 # if it's positive, add one
to the positive-words couter
                    }# closes bracket from if the word is positive
                    else if(LIST$scores[i] < 0){ # checks if the word is
positive or negative
                        negative <- negative + 1 # if it's negative, add one
to the negative-words counter
                    }# closes bracket from if the word is negative
                } # close brackets for the created string
            } # closes the brackets from the else, case in which the word
from the text was not in the list
        }# close brackets from the "for the length of the text"
    }# close brackets from the "for the length of the word score file"
    vetor <- c(rep("positive", positive), rep("negative", negative))#creates
a vector with the positive and negative counts
    vetor <- factor(vetor, levels=c("negative", "positive")) # makes this
vector into a factor vector
    dev.new() # opens new graphic device
    barplot(prop.table(table(vetor)), main="Relative frequency of
categorized\n words in each sentiment category", xlab="Sentiment category",
ylab="Relative frequency", col=c("indianred1", "lightskyblue3")) # creates a
```

```
barplot with the frequencies of each category (positive or negative)
    neutral <- leng - (positive+negative) # since all the words that are
neither positive nor negative are neutral (or uncategorized, which will be
treated as neutral), here we have all the neutral words =)
    vetorn <- c(rep("positive", positive), rep("negative", negative),
rep("neutral", neutral))#creates a vector with the positive, negative and
neutral counts
    vetorn <- factor(vetorn, levels=c("negative", "neutral", "positive")) #
makes this vector into a factor vector
    dev.new() #opens new graphic device
    barplot(prop.table(table(vetorn)), main="Relative frequency of all
words\n in each sentiment category", xlab="Sentiment category",
ylab="Relative frequency", col=c("indianred1", "lightgreen",
"lightskyblue3")) # creates a barplot with the frequencies of each category
(positive or negative or neutral)
    return(score.text) # returns the sum of the scores from the words of the
text
} # closes the brackets of the function
```

O Texto de Ajuda

```
sentiment.analysis      package:none        R Documentation

Gives a sentiment score to a text that is the sum of the score of all
 scored words/expressions in the text.

Description:
    sentiment.analysis receives a text and a table of words
    and their scores (if none is provided, the function will
    search for the "WordScores.txt" file provided with this
    function). Each word and expression of up to 4 words from
    the text is searched in the table data base and the scores
    of all words/expressions is added to get the text score.

Usage:
    sentiment.analysis(your.text, word.list.and.scores)
    ## Default:
    sentiment.analysis(your.text, word.list.and.scores="WordScores.txt")

Arguments:
    your.text:    character. A text given by the user in
            UTF-8 encoding, in english language,
            usually a .txt file, but internet sites
            can also be given here.

    word.list.and.scores:    table. A table with the words/
                expressions in the fist column,
                and their score in the second
                column. The separator must be
```

```
                a tab ("\t") and their should
                be no header in the file.

Value:

    The function returns the value of the text score (sum of all
    word/expression scores) and two graphics: one of the relative
    positive and negative words (in the universe of all categorized
    words) and one of the relative positive, negative and neutral
    words (in the universe of all words from the text, were un-
    categorized words are considered neutral).

Warning:
    The function is not really fast, the bigger the text and
    you word list and score the more it will take. With the
    default word list even small texts may take a while to be
    processed. Remember that R demands quite a bit from your
    RAM memory, so it mey be a good ideia to make a camomile
    tea while you wait for the function to run, mainly if you
    are using large data.
    Also, in the case of low RAM memory and large texts there
    may appear a few warnings, but the program usually works
    out just fine. Just don't forget the camomile tea.

Author:

    Júlia Beck Raíces
    nºUSP: 6802291
    julia.raices@gmail.com
    juliar@riseup.net
    fingerprint: BF75 AF9A 1232 DFF6 0189 5D72 7877 3E81 1433 5F11

Thanks:

    Special thanks to Viviane Santos who helped me with the ideia
    of the function and with the references and to Chalom, who
    always helps me with the constant despair of computer programming.

References:

    - The words list and scores was obtainned (and slightly modified) from:
    " Lars Kai Hansen, Adam Arvidsson, Finn Årup Nielsen, Elanor Colleoni,
    Michael Etter, "Good Friends, Bad News - Affect and Virality in
    Twitter", The 2011 International Workshop on Social Computing,
    Network, and Services (SocialComNet 2011). "

    - The "stoya.txt" test archive is the text "Sigh" from Stoya.
    Obtained from her blog at: http://graphicdescriptions.com/11-sigh

See Also:
```

```
    - Bo Pang and Lillian Lee "Opinion Mining and Sentiment Analysis",
    Foundations and Trands on Information Retrieval, Vol 2 (2008).

    - SentiWordNet ( http://sentiwordnet.isti.cnr.it/ )
    - Stanford's Sentiment Analysis website
    ( http://nlp.stanford.edu/sentiment/ )

Examples:

    # Download both the "WordScores.txt" file and the "stoya.txt"
    ## file at http://tinyurl.com/q353stg
    sentiment.analysis("stoya.txt", "WordScores.txt")
    # gives the text score and the graphs

    # Download both the files "WordScores.txt" and "test.txt"
    ##at http://tinyurl.com/q353stg

    sentiment analysis("test.txt")
    # gives the score and graphics for another text (in the case
    ## a made-up text of scored words). Notice that when
    ## word.list.and.scores is not given the function automatticaly
    ## uses the "WordScores.txt" file.
```