Entrega do trabalho final

test-text-bbc.txt

Arquivos

Função collocations: collocations.r

Help da função: help-collocations.txt

Help da função

```
collocations              package:none               R Documentation

Extract collocations for a target word from a given raw text.

Description:

  collocations receives a text and a target word and select the
  sentences from the text which contain the target word. From
  those sentences, the co-occurrences between target word and the
  other words which are above a certain threshold will constitue
  the set of collocations.

Usage:

  collocates(thetext, targetword, ncollmax)

Arguments:

  thetext       character. Text given by the user in
  .txt format and UTF-8 encoding.

  targetword        character. Any word the user has chosen
from the text. It will the reference for the extraction of the
collocations.

  ncollmax      numeric. Maximum number of collocates
  to be displayed on the graph generated by the function. In case
  the number of extracted collocates is less than the stipulated
  maximum, then ncollmax will be ignored.

Details:

  The function may not work well depending on the size of
  the text file given even though some optimizations were tried
  such as using environments hash to count faster the words'
```

```
  occurrences.

Value:

  Instead of returning values, collocates generates one text
  file and another file for a barplot in png format. Both are
  saved in the workspace being used to run the function.
Warning:

  Depending on the size of the text file, the function may
  get too slow or not work. As a suggestion, the usar can exeriment
  the function with different text sizes. See Examples for a simple
  teste of the function.

Author:

  Viviane Santos da Silva
  viviane.sds90@gmail.com
  viviane.santos.silva@usp.br

References:

  http://en.wikibooks.org/wiki/R_Programming/Text_Processing Last
access in may 18th 2014.

  About environments and hash argument:
http://adv-r.had.co.nz/Environments.html (There has been created a hash
function to optimize the use of hashes, but it only works for later
versions of R. Read "See Also")

  Download of non-annotaded corpora for testing the function:
http://corpora.informatik.uni-leipzig.de/download.html Last access
in may 15th 2014.
  To understand a little bit more about collocations in a more
intuitive way: http://esl.fis.edu/grammar/easy/colloc.htm

See Also:

  For more information on hash usage in R, see:
http://cran.r-project.org/web/packages/hash/index.html,
http://cran.r-project.org/web/packages/hash/hash.pdf and
http://opendatagroup.wordpress.com/2009/07/26/hash-package-for-r/.

Examples:

  # Download the file "teste-texto-bbc.txt" in
(http://ecologia.ib.usp.br/bie5782/doku.php?id=bie5782:01_curso_atual:alunos
:trabalho_final:
viviane.santos.silva:start) and save it to your R workspace to run this
example.
```

```
  collocates(thetext="test-text-bbc.txt", targetword="fiction", ncollmax=10)
# generates a barplot for the 10 first collocates which co-occur
with the target word "film" in the text given.
```

Código da função

```
##### FUNCTION TO EXTRACT COLLOCATIONS FROM RAW TEXTS #####

collocations <- function(thetext, targetword, ncollmax) {

## Reading the text ##

text <- scan(thetext, character(0), quote = NULL, sep = "\n", allowEscapes =
FALSE, strip.white = TRUE, fileEncoding = "UTF-8") # reads text input in
.txt format
textstring <- tolower(paste(text, collapse = "")) # converts upper-case
characters to lower-case
# textstring

textclean1 <- gsub("[^[:alnum:][:space:]'’\\.]", "", textstring) # removes
from the textstring all ist characters, except alphabetic, spaces,
apostrophes and period
textclean2 <- gsub("[[:space:]]+", ' ', textclean1) # substitutes multiple
spaces that may accidentally appear in the text for simple space character
textclean3 <- gsub("\\. ", '.', textclean2) # substitute "period+space" for
period only (to avoid future problems with the extraction of words from the
begnings of the paragraphs)

textdotsplit <- strsplit(textclean3, split='\\.') # separate the text by its
sentences using the periods to guide this process

txtw <- list() # creating an empty list
# iterating to split the text to make it possible to access its words
individually
for (i in 1:length(textdotsplit[[1]])){
  txtw[i] <- strsplit(textdotsplit[[1]][i], split=' ') # txtw is now a list
of lists, each inner list corresponding to a sentence with indexes to access
its words
}


## Creating a dictionary to save word occurrences ##

wordcounter <- function(sentencelist){ # function to transform a list of
list such as txtw into a dictionary containing words counts
  wordcount <- new.env(hash=TRUE, parent = emptyenv()) # initializing the
dictionary
  for (i in seq(1:length(sentencelist))){
    sentence <- sentencelist[[i]] # this was made to make the code easier to
```

```
read, avoiding excessive '[' notation
    for (j in seq(1:length(sentence))){
      eachword <- sentence[j]  # 'eachword' is accessing each of the words
from the text given
      if (is.null(wordcount[[eachword]])) { # dictionary's new entry
        wordcount[[eachword]] <- list(value=1)
      } else{wordcount[[eachword]]$value <- wordcount[[eachword]]$value + 1
# updates existing entry
      }
    }
  }
  return(wordcount)
}


## Creating a named list from a dictionary ##

ordnamedlist <- function(dict){ # tranform dictionary into list and sort it
by its values
  occlist <- list() # creats an empty list
  i <- 1
  for (w in ls(dict)) { # transform dictionary into a list
    occlist[[i]] <- list(name=w, value=dict[[w]]$value) # extracts words and
their corresponding number of occurrences from the environment hash
    i <- i+1
  }
  occlist[order(as.numeric(sapply(occlist, "[", "value")), decreasing=T)] #
applies the '[' function through the sapply to the named list created
# and extract its values, which will be used by the as.numeric to generate a
simple numeric vector which, in turn, will be sorted by the
# order function to ordenate the values returning their positions that will
be "given" back to occlist and will automatically order it
}

wcount <- wordcounter(txtw) # calling the function that turns a list of
lists into a dictionary
wcountsort <- ordnamedlist(wcount) # transforming the wcount (dictionary)
into an ordered list


## Creating a file containing the words and their counts ##

namefile <- "wordscounts.txt" # just to make it easier to change the file
name if one prefer

write("words\tcounts", file=namefile) # generates a file containing 2
column-headers: words and counts. avoids problems if the user call this
function more than once (function countsfile right below calls a write
function which is using append=T)
```

```
print("A file named 'wordscounts.txt', containing your text words and their
counts has been generated. Feel free to explore it (:") # so the user will
know a file has been generated

countsfile <- function(l){ # function to be used by the lapply to print
properly in the file words and their counts
  write(paste(c(l$name, l$value), collapse="\t"), file=namefile, append=T)
}

trash <- lapply(wcountsort, countsfile) # calling the function to generate a
file. the variable 'trash' is being attributed the lapply output for this
was generating NULL values and, apparently, R has nothing like a procedure
function (a function that returns nothing).


## Finding the co-occurrences ##

hastarget <- function(wlist, target){ # checks if the lists (sentences) in
wlist has the target word
  target %in% wlist # the test is made using the logical function %in%
}

sbin <- which(sapply(txtw, hastarget, targetword)) # function 'which'
returns the positions of the lists from txtw tested by the hastarget
function and that were marked "TRUE"
tsentences <- txtw[sbin] # tsentences receives only those sentences
containing the target word

cooccurrences <- wordcounter(tsentences) # creates a dicitionary with the
counts of the words from the sentences which contains the target word
occlistsort <- ordnamedlist(cooccurrences) # sorts the created dictionary of
co-occurrences


## Calculating cooccurrences words frequencies ##

wtotal <- sum(sapply(txtw, length)) # total number of words in the text
stotal <- sum(sapply(tsentences, length)) # total number of words in the
subtext (sentences which contained the target word)


## Creating a dictionary for the frequencies of the co-occurrences ##

wfreqs <- function(countlist, total){ # countlist is a named list containing
names and associated values and total is the amount of words in the portion
of text from which this countlist was generated
  freqs  <- new.env(hash=TRUE, parent = emptyenv())
  for (i in seq(1:length(countlist))){
    eachword <- countlist[[i]]$name
    freqs[[eachword]]$value <- (countlist[[i]]$value/total) # calculating
the frequencies of the words
```

```
  }
  return(freqs)
}

tfreqs <- wfreqs(wcountsort, wtotal) # calling the functions that creates
the frequency dictionary for the whole text
cfreqs <- wfreqs(occlistsort, stotal) # calling the functions that creates
the frequency dictionary for the sentences of the text that contains the
target


## Creating a dictionary for the frequencies of the relevant words ##

testocc <- function(ftotal, fsent){

  ratio  <- new.env(hash=TRUE, parent = emptyenv())
  for (w in ls(fsent)){
     if(tfreqs[[w]] > 0.0001 & wcount[[w]]$value > 5){ # discarding words
with frequencies in the text below 0.01% or, for a text with less than 5000
words, discarding those words with freqs less than 2
      ratio[[w]]$value <- fsent[[w]]$value/ftotal[[w]]$value # this
important parameter will be used to judge whether or not a word w is forming
a collocation with the target. if this value is 0.5, it means there is no
diference between the distribution of the word w considered alone and its
distribution given the target (and that's the null model). the max of this
ratio will be 'wtotal/stotal'. the minimum will be 1 (when every occurrence
of w meets every occurrence of the target)
     }
  }
  return(ratio)
}

relevfreqs <- testocc(tfreqs, cfreqs) # dictionary containing words and
their ratios
relevfreqslist <- ordnamedlist(relevfreqs) # ordered list od the dictionary
created


## Extracting collocates ##

threshold <- 2 # only words which appear twice as often will be considered
collocates candidates

selectcoll <- function(l){ # will be used to select the collocations
comparing the candidate words' ratios and the chosen threshold
  l$value > threshold
}

collindex <- which(sapply(relevfreqslist, selectcoll)) # selecting the
collocates' indexes
```

```
collocates <- relevfreqslist[collindex] # using the extracted indexes,
collocates access the correct collocates from the relevfreqlist (list of
relevant words and their frequencies)

n <- as.numeric(ncollmax)
if(length(collocates) > n){ # test to check if there are more collocates
than the number the user wants to display on the graph
  collocates <- collocates[1:n]
}

if(length(collocates) == 0) {
  print(paste("Couldn't find any collocate for the '", targetword,"' chosen.
Maybe you can try a more frequent word."))
  return(invisible())
}

## Plotting a barplot displaying the n most probable collocates ##

collnames <- sapply(collocates, function(l) l$name) # generating a vector
containing the words which were selected as being collocates
collratios <- sapply(collocates, function(l) as.numeric(l$value)) #
collocates ratio (or the degree of the collocations extracted)

png("barplot-collocates.png")
barplot(collratios, names.arg = collnames, las=2, main=paste("Collocates
of", targetword), ylab="Collocations ratios", ylim=c(0,
max(collratios)+0.5)) # graph dysplaying the results of the function
dev.off()

}
```