Homework 2 Code

Jared Yu

January 19, 2019

```
library(parallel); library(tm); library(SnowballC); library(NLP) # Load libraries

word_filter <- function(file_df) {
  # The word_filter function will accept a dataframe of a file and sort
  # through the description column, where it will remove non-essential
  # parts of descriptions, and create keywords using NLP-packages that
  # can describe each expense. The function will also return both the
  # keywords, along with an index for removing entries which are either
  # NA or blank.

  # Remove negative expense, punctuation, casing
  file_df = file_df[file_df$total_obligation > 0,]
  file_description <- file_df$description
  removed_punc = gsub('[[:punct:]]', ' ', file_description)
  lower_case = tolower(removed_punc)

  # Remove stopwords, create stem words
  stop_words1 = removeWords(lower_case, stopwords('en'))
  stop_words2 = removeWords(stop_words1, stopwords('SMART'))
  stem_words = stemDocument(stop_words2)

  # Deal with any white space and create removal index
  clean_whitespace = stripWhitespace(stem_words)
  remove_na = clean_whitespace[clean_whitespace != 'NA']
  rm_ind = which(remove_na != '')
  lead_trail = gsub("^\\s+|\\s+$", "", clean_whitespace[rm_ind])
  rm_ind = which(lead_trail != '')

  return(list(lead_trail, rm_ind))
}

weighted_char_vectors <- function(price, character_vector) {
  # The weighted_char_vectors function will take in the price and character
  # vector which have previously been created. It will then split the
  # character vector into the individual keywords of each expense, and then
  # assign the average price to each keyword. These keywords are then combined
  # so that keywords which are matching will also have their weighted prices
  # from across different expenses summed together.

  # Split the keywords from each exepense, and assign them to their average
  # prices from each character vector.
  keywords_list = strsplit(character_vector$keywords, ' ', fixed = TRUE)
  keywords_list_len = sapply(keywords_list, length)
  avg_price = character_vector$weighted_price / keywords_list_len
  weighted_words = Map(cbind, keywords_list, avg_price)
  weighted_words = do.call("rbind", weighted_words)
```

```
  # Turn the keywords into a dataframe, and then group the dataframe
  # by keyword to sum the total weighted average.
  weighted_df = as.data.frame(weighted_words, stringsAsFactors = FALSE)
  names(weighted_df) = c("keywords", "char_weights")
  weighted_df$char_weights = as.numeric(weighted_df$char_weights)
  sum_df = tapply(weighted_df$char_weights, weighted_df$keywords,
                  function(x) sum(x, na.rm = TRUE))

  # Return the top 25 keywords, along with their weighted prices in terms
  # of their proportion.
  top_25 = head(sort(sum_df, decreasing = TRUE), n = 25)
  round(top_25 / sum(top_25), 3)
}

weighted_description <- function(file_df) {
  # The weighted_description function takes in a single file as input and then
  # uses a series of functions from NLP-related packages to filter the description
  # column into keywords which are representative of the file's agency. The function
  # will also create a character vector of these keywords per expense which weighs
  # each keyword based on the price of each expense in the file. The function
  # will then return the top 25 keywords along with their weights.

  # Filter the dataframe's description column, and also create the removal index
  # to index through the corresponding price column.
  filtered_word_list <- word_filter(file_df = file_df)
  lead_trail <- filtered_word_list[[1]]; rm_ind <- filtered_word_list[[2]]

  # Create a data frame that combines the keywords along with the prices of each
  # expense's keywords and the weighted price per expense.
  price <- as.numeric(file_df$total_obligation)
  character_vector <- data.frame(keywords = lead_trail[rm_ind], # df of keywords and prices
                                 weighted_price = price[rm_ind] /
                                   sapply(strsplit(lead_trail[rm_ind], " "), length),
                                 stringsAsFactors = FALSE)

  # Use the price and character vector dataframe to create a combined weighted
  # character vector which returns the top 25 keywords and their proportional
  # weight amongst the top 25.
  weighted_char_vectors(price = price, character_vector = character_vector)
}

# Subset the necessary columns, description, spending which are the 2nd and 4th columns
which_columns = c("NULL", "vector", "NULL", "character", rep("NULL", 3))

descript <- function(file_list, zip_file = 'awards.zip', column_pattern = which_columns) {
  # Read in a file and assign find the top 25 weighted words
  file_df <- read.csv(unz(zip_file, file_list), colClasses = column_pattern, header = TRUE)
  file_df <- file_df[!is.na(file_df$description),] # Remove blank descriptions
  weighted_description(file_df)
}

# Load zip info from path
zip_file_path <- "C:/Users/qizhe/Desktop/STA 141C/hw2/awards.zip"
```

```r
zip_info <- unzip(zip_file_path, list = TRUE)

# Find 91 agencies between 1MB and 50MB, calculate weighted character vector
agencies_91 <- zip_info[zip_info$Length >= 1048576 & zip_info$Length <= 1048576*50,]$Name
weighted_vectors <- lapply(agencies_91, function(x) descript(file_list = x))

# Examine special agencies
special_agencies <- which(agencies_91 %in% c("655.csv", "262.csv", "778.csv", "110.csv"))
sink('special_agencies.txt')
weighted_vectors[[special_agencies[1]]] # 110, 2
weighted_vectors[[special_agencies[2]]] # 262, 40
weighted_vectors[[special_agencies[3]]] # 655, 57
weighted_vectors[[special_agencies[4]]] # 778, 74
sink()

# Time parallel versions
num_cores <- detectCores()
cls <- makeCluster(num_cores, type = "PSOCK")
clusterCall(cls, source, "preprocess_final.R")
clusterCall(cls, ls, envir = globalenv())
clusterEvalQ(cls, source(file = "C:/Users/qizhe/Desktop/STA 141C/hw2/preprocess_final.R"))
parallel_time <- system.time(parLapply(cls, agencies_91, function(x) descript(file_list = x)))
parallel_timeLB <- system.time(clusterApplyLB(cls, agencies_91, function(x) descript(file_list = x)))
lapply_time <- system.time(lapply(agencies_91, function(x) descript(file_list = x)))
stopCluster(cls)
```