# Homework #10 – Text Mining HW

## R Script (Code)

#

# Course: IST687

# Name: Joyce Woznica

# Homework 10 - Text Mining

# Due Date: 3/19/2019

# Date Submitted:

#

#specify the packages of interest

packages=c("tm","wordcloud", "XML", "tidytext", "syuzhet")

#use this function to check if each package is on the local machine

#if a package is installed, it will be loaded

#if any are not, the missing package(s) will be installed and loaded

package.check <- lapply(packages, FUN = function(x) {

if (!require(x, character.only = TRUE)) {

install.packages(x, dependencies = TRUE)

library(x, character.only = TRUE)

}

})

#verify they are loaded

search()

# MLK speech

mlk<-readLines("MLK.txt")

mlk <- mlk[which(mlk != "")] #remove blank lines

mlk[1:3]

# Create a term matrix

# interprets each element of the "mlk" as a document and create a vector source

words.vec <- VectorSource(mlk)

# create a Corpus, a "Bag of Words"

words.corpus <- Corpus(words.vec)

# first step transformation: make all of the letters in "words.corpus" lowercase

words.corpus <- tm\_map(words.corpus, content\_transformer(tolower))

# second step transformation: remove the punctuation in "words.corpus"

words.corpus <- tm\_map(words.corpus, removePunctuation)

# third step transformation: remove numbers in "words.corpus"

words.corpus <- tm\_map(words.corpus, removeNumbers)

# final step transformation: take out the "stop" words, such as "the", "a" and "at"

words.corpus <- tm\_map(words.corpus, removeWords, stopwords("english"))

# create a term-document matrix "tdm"

tdm <- TermDocumentMatrix(words.corpus)

# view term-document matrix "tdm"

tdm

#

# continue to manipulate the MLK speech, words, terms

# Create a list of counts for each word

# convert tdm into a matrix called "m"

m <- as.matrix(tdm)

m[1:10,]

# create a list of counts for each word named "wordCounts"

wordCounts <- rowSums(m)

wordCounts[1:10]

# sum the total number of words and store the value to "totalWords"

totalWords <- sum(wordCounts)

totalWords

# create a vector "words" that contains all the words in "wordCounts"

words <- names(wordCounts)

head(words)

# sort words in "wordCounts" by frequency

wordCounts <- sort(wordCounts, decreasing=TRUE)

# check the first several items in "wordCounts" to see if it is built correctly

head(wordCounts)

# 1) First read in the AFINN word list. Note that each line is a word and a score

# create two vectors: words and score

sents <- get\_sentiments("afinn")

sents <- as.data.frame(sents)

colnames(sents)<- c("Word", "Score")

# sort words in "wordCounts" by frequency

wordCounts <- sort(wordCounts, decreasing=TRUE)

# check the first several items in "wordCounts" to see if it is built correctly

head(wordCounts)

# Create a df

cloudFrame<-data.frame(word=names(wordCounts),freq=wordCounts)

cloudFrame[1:10,]

# 2) Compute an overall score for the MLK speech

# first merge the mlk df with the AFINN sentiments to create a word, freq and score

mergedTable<-merge(cloudFrame,sents,by.x="word",by.y="Word")

mergedTable[1:10,]

str(mergedTable)

# Calculate the overall score based on summing up the frequency\*score

overallScore<-sum(mergedTable$freq\*mergedTable$Score)

# overallScore

overallScore

sprintf("%1.2f%%", 100\*overallScore/totalWords)

# this gives a different number - just tried for fun.

get\_sentiment(mlk,method="afinn")

Score.MLK <- sum(get\_sentiment(mlk,method="afinn"))

# 3) Compute the senitment score for each quarter (25%)

# create a function to cut the words in chunks in order of the speech

# wc - word corpus

# num.chunks - number of chunks

chunk.up <- function (wc, num.chunks)

{

index <- 1

cutoff <- round(length(wc)/num.chunks)

start <- index

end <- cutoff

# build a new vector of the chunks

while (index <= num.chunks)

{

# grab the chunk

chunked.up <- wc[start:end]

# create term document matrix

tdm <- TermDocumentMatrix(chunked.up)

# make it a matrix

m <- as.matrix(tdm)

word.counts <- rowSums(m)

total.words <- sum(word.counts)

# Create a df

cf<-data.frame(word=names(word.counts),freq=word.counts)

# Compute an score for current chunk

# first merge the mlk df with the AFINN sentiments to create a word, freq and score

mt<-merge(cf,sents,by.x="word",by.y="Word")

# Calculate the overall score based on summing up the frequency\*score

r.scores <- cbind (r.scores,overallScore<-sum(mt$freq\*mt$Score))

# increment up

start <- end + 1

index <- index + 1

end <- end+cutoff+1

}

r.scores

}

quarter.scores <- chunk.up(words.corpus, 4)

barplot(quarter.scores, names.arg = c("1st 25%","2nd 25%","3rd 25%","4th 25%"), main = "AFFIN Sentiment Score for Martin Luther King Speech")

## Console Log (Executed Code)

|  |
| --- |
| > #  > # Course: IST687  > # Name: Joyce Woznica  > # Homework 10 - Text Mining  > # Due Date: 3/19/2019  > # Date Submitted:  > #  > #specify the packages of interest  > packages=c("tm","wordcloud", "XML", "tidytext", "syuzhet")  >  > #use this function to check if each package is on the local machine  > #if a package is installed, it will be loaded  > #if any are not, the missing package(s) will be installed and loaded  > package.check <- lapply(packages, FUN = function(x) {  + if (!require(x, character.only = TRUE)) {  + install.packages(x, dependencies = TRUE)  + library(x, character.only = TRUE)  + }  + })  >  > #verify they are loaded  > search()  [1] ".GlobalEnv" "package:syuzhet" "package:tidytext" "package:XML" "package:wordcloud" "package:RColorBrewer"  [7] "package:tm" "package:NLP" "tools:rstudio" "package:stats" "package:graphics" "package:grDevices"  [13] "package:utils" "package:datasets" "package:methods" "Autoloads" "package:base"  > # MLK speech  > mlk<-readLines("MLK.txt")  > mlk <- mlk[which(mlk != "")] #remove blank lines  > mlk[1:3]  [1] "I am happy to join with you today in what will go down in history as the greatest demonstration for freedom in the history of our nation."  [2] "Five score years ago, a great American, in whose symbolic shadow we stand today, signed the Emancipation Proclamation. This momentous decree came as a great beacon light of hope to millions of Negro slaves who had been seared in the flames of withering injustice. It came as a joyous daybreak to end the long night of their captivity."  [3] "But one hundred years later, the Negro still is not free. One hundred years later, the life of the Negro is still sadly crippled by the manacles of segregation and the chains of discrimination. One hundred years later, the Negro lives on a lonely island of poverty in the midst of a vast ocean of material prosperity. One hundred years later, the Negro is still languishing in the corners of American society and finds himself an exile in his own land. So we have come here today to dramatize a shameful condition."  > # Create a term matrix  > # interprets each element of the "mlk" as a document and create a vector source  > words.vec <- VectorSource(mlk)  > # create a Corpus, a "Bag of Words"  > words.corpus <- Corpus(words.vec)  > # first step transformation: make all of the letters in "words.corpus" lowercase  > words.corpus <- tm\_map(words.corpus, content\_transformer(tolower))  Warning message:  In tm\_map.SimpleCorpus(words.corpus, content\_transformer(tolower)) :  transformation drops documents  > # second step transformation: remove the punctuation in "words.corpus"  > words.corpus <- tm\_map(words.corpus, removePunctuation)  Warning message:  In tm\_map.SimpleCorpus(words.corpus, removePunctuation) :  transformation drops documents  > # third step transformation: remove numbers in "words.corpus"  > words.corpus <- tm\_map(words.corpus, removeNumbers)  Warning message:  In tm\_map.SimpleCorpus(words.corpus, removeNumbers) :  transformation drops documents  > # final step transformation: take out the "stop" words, such as "the", "a" and "at"  > words.corpus <- tm\_map(words.corpus, removeWords, stopwords("english"))  Warning message:  In tm\_map.SimpleCorpus(words.corpus, removeWords, stopwords("english")) :  transformation drops documents  > # create a term-document matrix "tdm"  > tdm <- TermDocumentMatrix(words.corpus)  > # view term-document matrix "tdm"  > tdm  <<TermDocumentMatrix (terms: 463, documents: 29)>>  Non-/sparse entries: 682/12745  Sparsity : 95%  Maximal term length: 14  Weighting : term frequency (tf)  > # continue to manipulate the MLK speech, words, terms  > # Create a list of counts for each word  > # convert tdm into a matrix called "m"  > m <- as.matrix(tdm)  > m[1:10,]  Docs  Terms 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29  demonstration 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  freedom 1 0 0 0 1 1 1 2 0 1 0 0 0 0 1 0 0 0 0 0 1 1 3 1 1 1 1 2 1  greatest 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  happy 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  history 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  join 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1  nation 1 0 0 0 2 3 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0  today 1 1 1 0 1 0 0 1 0 0 0 2 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0  will 1 0 0 0 1 5 0 0 2 0 1 0 1 1 1 2 0 1 0 2 4 2 0 0 0 0 0 0 2  ago 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  > # create a list of counts for each word named "wordCounts"  > wordCounts <- rowSums(m)  > wordCounts[1:10]  demonstration freedom greatest happy history join nation today will ago  1 20 1 1 2 3 10 9 26 1  > # sum the total number of words and store the value to "totalWords"  > totalWords <- sum(wordCounts)  > totalWords  [1] 841  > # create a vector "words" that contains all the words in "wordCounts"  > words <- names(wordCounts)  > head(words)  [1] "demonstration" "freedom" "greatest" "happy" "history" "join"  > # sort words in "wordCounts" by frequency  > wordCounts <- sort(wordCounts, decreasing=TRUE)  > # check the first several items in "wordCounts" to see if it is built correctly  > head(wordCounts)  will freedom negro one let ring  26 20 13 13 13 12  > # 1) First read in the AFINN word list. Note that each line is a word and a score  > # create two vectors: words and score  > sents <- get\_sentiments("afinn")  > sents <- as.data.frame(sents)  > colnames(sents)<- c("Word", "Score")  > # sort words in "wordCounts" by frequency  > wordCounts <- sort(wordCounts, decreasing=TRUE)  > # check the first several items in "wordCounts" to see if it is built correctly  > head(wordCounts)  will freedom negro one let ring  26 20 13 13 13 12  > # Create a df  > cloudFrame<-data.frame(word=names(wordCounts),freq=wordCounts)  > cloudFrame[1:10,]  word freq  will will 26  freedom freedom 20  negro negro 13  one one 13  let let 13  ring ring 12  day day 11  dream dream 11  nation nation 10  come come 10  > # 2) Compute an overall score for the MLK speech  > # first merge the mlk df with the AFINN sentiments to create a word, freq and score  > mergedTable<-merge(cloudFrame,sents,by.x="word",by.y="Word")  > mergedTable[1:10,]  word freq Score  1 allow 2 1  2 alone 1 -2  3 bad 1 -3  4 bankrupt 1 -3  5 beautiful 1 3  6 bright 1 1  7 creative 2 2  8 demand 1 -1  9 demonstration 1 -1  10 despair 2 -3  > str(mergedTable)  'data.frame': 61 obs. of 3 variables:  $ word : Factor w/ 463 levels "able","ago","ahead",..: 6 8 19 21 25 36 74 89 91 93 ...  $ freq : num 2 1 1 1 1 1 2 1 1 2 ...  $ Score: int 1 -2 -3 -3 3 1 2 -1 -1 -3 ...  > # Calculate the overall score based on summing up the frequency\*score  > overallScore<-sum(mergedTable$freq\*mergedTable$Score)  > # overallScore  > overallScore  [1] 113  > sprintf("%1.2f%%", 100\*overallScore/totalWords)  [1] "13.44%"  > # this gives a different number - just tried for fun.  > get\_sentiment(mlk,method="afinn")  [1] 8 6 -6 5 3 8 4 -4 2 7 -3 1 3 1 3 1 1 -3 1 4 5 1 7 2 2 2 2 2 8  > Score.MLK <- sum(get\_sentiment(mlk,method="afinn"))  > # 3) Compute the sentiment score for each quarter (25%)  > # create a function to cut the words in chunks in order of the speech  > # wc - word corpus  > # num.chunks - number of chunks  > chunk.up <- function (wc, num.chunks)  + {  + index <- 1  + cutoff <- round(length(wc)/num.chunks)  + start <- index  + end <- cutoff  + r.scores<-NULL  + # build a new vector of the chunks  + while (index <= num.chunks)  + {  + # grab the chunk  + chunked.up <- wc[start:end]  + # create term document matrix  + tdm <- TermDocumentMatrix(chunked.up)  + # make it a matrix  + m <- as.matrix(tdm)  + word.counts <- rowSums(m)  + total.words <- sum(word.counts)  + # Create a df  + cf<-data.frame(word=names(word.counts),freq=word.counts)  + # Compute an score for current chunk  + # first merge the mlk df with the AFINN sentiments to create a word, freq and score  + mt<-merge(cf,sents,by.x="word",by.y="Word")  + # Calculate the overall score based on summing up the frequency\*score  + r.scores <- cbind (r.scores,overallScore<-sum(mt$freq\*mt$Score))  + # increment up  + start <- end + 1  + index <- index + 1  + end <- end+cutoff+1  + }  + r.scores  + }  > quarter.scores <- chunk.up(words.corpus, 4)  > barplot(quarter.scores, names.arg = c("1st 25%","2nd 25%","3rd 25%","4th 25%"), main = "AFFIN Sentiment Score  for Martin Luther King Speech") |

## Visualizations

