cind110_Assignment_03

Ashwin David

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Use RStudio for this assignment. Edit the file `A3_F19_Q.Rmd` and insert your R code where wherever you see the string "#WRITE YOUR ANSWER HERE"

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

This assignment makes use of data that were adapted from: https://www.ted.com/talks

#Install and load required packages (please install if required)

```
#install.packages("tm")
#install.packages("text2vec")
#install.packages("NLP")
#install.packages("SnowballC")
#install.packages("slam")
#install.packages("textstem")
#install.packages("wordcloud")
#install.packages("Matrix")
#install.packages("Rcpp")
library(tm)
```

```
## Warning: package 'tm' was built under R version 4.0.5
```

```
## Loading required package: NLP
```

```
library(SnowballC)
library(NLP)
library(slam)
```

```
## Warning: package 'slam' was built under R version 4.0.5
```

```
library(text2vec)
```

```
## Warning: package 'text2vec' was built under R version 4.0.5
```

```
library(textstem)
```

```
## Warning: package 'textstem' was built under R version 4.0.5
```

```
## Loading required package: koRpus.lang.en

## Warning: package 'koRpus.lang.en' was built under R version 4.0.5

## Loading required package: koRpus

## Warning: package 'koRpus' was built under R version 4.0.5

## Loading required package: sylly

## Warning: package 'sylly' was built under R version 4.0.5

## For information on available language packages for 'koRpus', run
##
##   available.koRpus.lang()
##
## and see ?install.koRpus.lang()

##
## Attaching package: 'koRpus'

## The following object is masked from 'package:tm':
##
##     readTagged
```

```
library(wordcloud)
```

```
## Warning: package 'wordcloud' was built under R version 4.0.5

## Loading required package: RColorBrewer
```

```
library(Matrix)
```

```
## Warning: package 'Matrix' was built under R version 4.0.5
```

Reading the Transcripts

```
data <- read.csv(file = 'Sec-2-IR_Data.csv', header = F, sep = '|')
doc <- 0
for (i in c(2:100)) {doc[i] <- as.character(data$V1[i])}
doc.list <- as.list(doc[2:100])
N.docs <- length(doc.list)
names(doc.list) <- paste0("Doc", c(1:N.docs))
Query <- as.character(data$V1[1])
```

Preparing the Corpus

```
my.docs <- VectorSource(c(doc.list, Query))
my.docs$Names <- c(names(doc.list), "Query")
my.corpus <- Corpus(my.docs)
#my.corpus
```

Cleaning and Preprocessing the text (Cleansing Techniques)

```
#Write your answer here fro Question 1
#Hint: use getTransformations() function in tm Package
#https://cran.r-project.org/web/packages/tm/tm.pdf

#convert numbers to words
for (i in length(my.corpus)){
  my.corpus[[i]]$content <- as.character(textclean::replace_number(my.corpus[[i]]$content))
}
#updated_corpus1 <- my.corpus[[i]]$content

##utilizing a thesaurus
for(i in length(my.corpus)){
  my.corpus[[i]]$content <- textstem::lemmatize_strings(my.corpus[[i]]$content,dictionary = lexicon::has
}

#stemming
my.corpus <- tm::tm_map(my.corpus, stemDocument)
```

```
## Warning in tm_map.SimpleCorpus(my.corpus, stemDocument): transformation drops
## documents
```

```
my.corpus <- tm_map(my.corpus, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(my.corpus, content_transformer(tolower)):
## transformation drops documents
```

```
#removing words of abundance using removeWords function
#my.corpus <- tm_map(my.corpus, removeWords, c("is", "the", "a", "are", "so", "what", "an"))
my.corpus <- tm_map(my.corpus, removeWords, stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(my.corpus, removeWords, stopwords("english")):
## transformation drops documents
```

```
tm_map(my.corpus, removeWords, stopwords("smart"))
```

```
## Warning in tm_map.SimpleCorpus(my.corpus, removeWords, stopwords("smart")):
## transformation drops documents

## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 100
```

```
#remove extra dashes and other punctuation marks that will affect the processing
my.corpus <- tm_map(my.corpus, removePunctuation, ucp = TRUE, preserve_intra_word_contractions = FALSE,
```

```
## Warning in tm_map.SimpleCorpus(my.corpus, removePunctuation, ucp = TRUE, :
## transformation drops documents
```

```
#strip off the white space
my.corpus <- tm_map(my.corpus, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(my.corpus, stripWhitespace): transformation drops
## documents
```

##Creating a uni-gram Term Document Matrix (TDM)

```
#write your answer here for Question 2
#Hint: use TermDocumentMatrix()
#creating a unigram tdm
tdmUnigram <- tm::TermDocumentMatrix(my.corpus)
tm::inspect(tdmUnigram[1:10,1:10])
```

```
## <<TermDocumentMatrix (terms: 10, documents: 10)>>
## Non-/sparse entries: 24/76
## Sparsity           : 76%
## Maximal term length: 8
## Weighting          : term frequency (tf)
## Sample             :
##           Docs
## Terms      1 10 2 3 4 5 6 7 8 9
##   247      1  0 0 0 0 0 0 1 0 0
##   abil     1  0 0 0 0 0 0 0 0 0
##   abl      3  2 0 0 0 2 1 1 0 0
##   absolut  1  2 0 0 0 0 0 0 0 0
##   accident 1  0 0 0 0 0 0 0 0 0
##   actual   6  1 3 0 0 2 3 0 2 0
##   actually 1  0 0 0 2 0 0 0 1 0
##   adam     1  0 0 0 0 0 0 0 0 0
##   admir    1  0 0 0 0 1 0 0 0 0
##   advertis 1  0 0 0 0 0 0 0 0 0
```

```
#use remove Sparse to compute all terms after normalizing and refining the data
tdmUnigram <- tm::removeSparseTerms(tdmUnigram,0.67)
tm::inspect(tdmUnigram[1:10,1:10])
```

```
## <<TermDocumentMatrix (terms: 10, documents: 10)>>
## Non-/sparse entries: 73/27
## Sparsity           : 27%
## Maximal term length: 8
## Weighting          : term frequency (tf)
## Sample             :
##           Docs
## Terms      1 10 2 3 4 5 6 7 8 9
```

```
##   abl       3  2 0 0 0 2 1 1 0 0
##   actual    6  1 3 0 0 2 3 0 2 0
##   ago       3  1 0 2 2 6 0 2 1 0
##   almost    1  0 1 1 0 2 0 0 1 0
##   also      2  8 4 3 1 2 5 2 1 0
##   ani       2  3 1 1 1 1 2 0 0 3
##   applause  1  0 0 2 1 6 4 2 1 1
##   ask       4  4 0 1 2 4 4 1 1 0
##   back      1  3 0 3 4 3 0 4 0 2
##   becaus   16  8 6 1 2 6 8 8 3 3
```

Converting the generated TDM into a matrix and displaying the first 6 rows and the dimensions of the matrix

```
#write your answer here for Question 3
#Hint: use dim to find the dimension
tdmUnigramMat <- as.matrix(tdmUnigram)
dim(tdmUnigramMat)
```

```
## [1] 218 100
```

```
head(tdmUnigramMat)
```

```
##         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
##   abl    3 0 0 0 2 1 1 0 0  2  1  1  0  0  0  4  2  3  0  1  7  3  0  1  3  0
##   actual 6 3 0 0 2 3 0 2 0  1  2  1  0  1  2 11  0  0  0  2  7 12  0  3  7  3
##   ago    3 0 2 2 6 0 2 1 0  1  2  0  0  0  1  0  0  0  0  1  6  2  0  0  0  0
##   almost 1 1 1 0 2 0 0 1 0  0  0  0  0  2  0  0  0  0  0  1  0  1  1  0  0  2
##   also   2 4 3 1 2 5 2 1 0  8  0  1  0  3  0  5  3  2  2  0  7  1  0  0 13  1
##   ani    2 1 1 1 1 2 0 0 3  3  0  0  0  5  4  1  4  0  0  0  3  5  1  1  2  1
##         Docs
## Terms    27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
##   abl     0  0  1  0  3  0  5  0  0  0  0  3  1  0  0  0  0  0  0  2  8  0  0
##   actual  0  1  7  0  0  1 16  1  1 12  2  0  2  2  1  0  6  0  2  3 12  1  4
##   ago     0  0  0  3  5  1  0  0  2  0  4  0  2  0  1  1  1  0  1  2  0  0  0
##   almost  0  1  0  0  0  0  0  0  0  0  3  0  2  1  0  1  2  0  0  0  2  0  3
##   also    0  5  9  4  0  1  2  3  4  3  2  4  4  3  2  1  8  1  4  2 11  1  6
##   ani     0  0  1  0  0  2  0  0  0  2  2  1  2  2  3  0  0  0  1  2  4  2  8
##         Docs
## Terms    50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
##   abl     1  0  9  0  4  0  0  0  0  1  1  0  1  0  1  3  0  0  0  2  5  0  0
##   actual  1  1  9  3  0  1  2 11  2  4  0  0  0  0  1 13  0  1  2  2  5  2  2
##   ago     0  0  1  0  1  1  1  2  0  2  1  2  0  0  5  7  0  4  1  1  5  0  1
##   almost  0  1  0  0  0  0  0  3  0  0  0  0  2  1  0  0  1  0  0  1  0  3  0
##   also    3  3  9  6  6  2  3  5  5  1  5  4  1  3  1  0  5  0  0  7  6  0  2
##   ani     0  2  2  0  0  0  1  1  0  1  1  1  0  1  0  5  1  7  0  1  0  7  1
##         Docs
## Terms    73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
##   abl     0  0  0  3  2  0  1  1  0  3  2  1  0  0  0  0  4  2  1  0  0  0  0
##   actual  1  1  1  2  7  3  3  0  7  0  5  7  1  7  2  0  9  0 14  1  2  2  2
```

```
##   ago     0  0  3  1  0  2  0  0  1  0  4  0  0  1  0  0  0  1  3  3  1  0  3
##   almost  1  0  1  0  2  0  0  0  1  1  2  0  2  0  0  0  0  0  0  0  0  0  0
##   also    5  3  2  3  6  2  2  0  6  2 11  2  5  2  1  0  4  2  0  2  1  4  2
##   ani     0  0  5  0  3  0  1  0  0  9  1  1  2  4  5  0  2  1  8  1  3  2  3
##         Docs
## Terms    96 97 98 99 100
##   abl     0  0  2  2   2
##   actual  0  7  0  4   4
##   ago     0  1  1  1   1
##   almost  0  1  0  0   0
##   also    1  0  1  0   0
##   ani     0  2  0  1   1
```

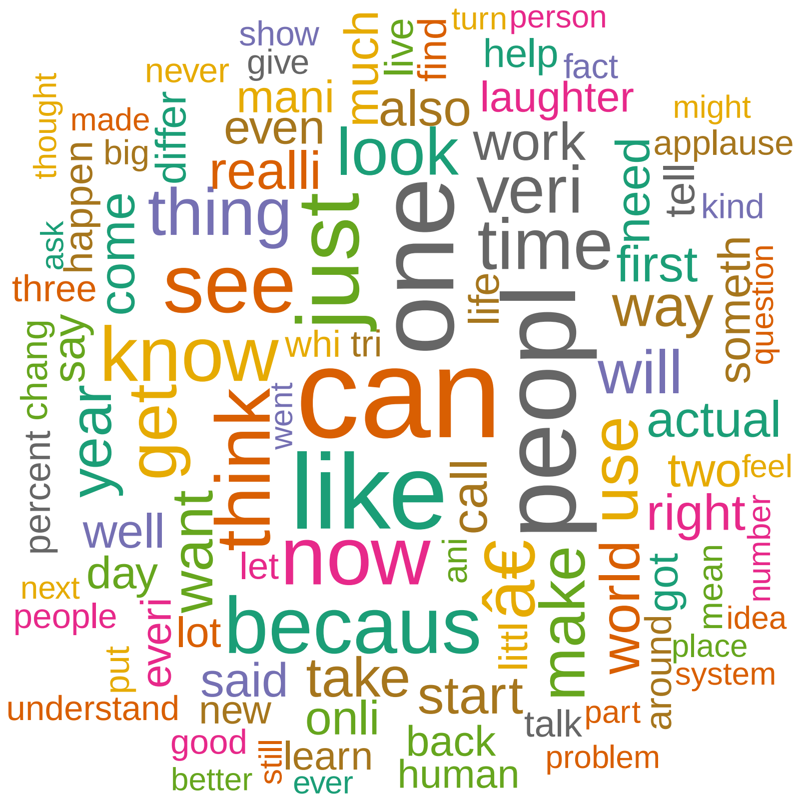
Generate a wordcloud of the most occured 100 words across all transcripts

```
#Write your answer here for Question 4
#Hint: use wordcloud
#set.seed(1)

words = sort(rowSums(tdmUnigramMat), decreasing = TRUE)
df = data.frame(word = names(words), freq = words)
head(df)
```

```
##        word freq
## can     can  923
## like   like  753
## one     one  744
## peopl peopl  719
## just   just  602
## now     now  559
```

```
set.seed(3)
wordcloud(words = df$word, freq = df$freq, min.freq = 1, max.words = 100, colors = brewer.pal(8, "Dark2
```


can
like
peopl
one
now
becaus
know
see
just
think
time
look
thing
veri
use
make
get
year
way
want
will
actual
right
first
realli
take
start
work
also
mani
even
â€
world
well
come
much
need
two
call
day
said
lot
onli
back
human
understand
people
three
something
laughter
help
learn
happen
percent
chang
say
everi
got
new
talk
good
better
differ
made
big
thought
ask
never
show
give
live
find
turn
person
fact
might
applause
tell
kind
question
feel
number
mean
idea
place
system
part
problem
around
littl
ani
let
went
whi
tri
life
next
put
still
ever