cind110\_Assignment\_03

Ajay Herod

## 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

#install.packages("tm") #Please Install if required  
#install.packages("text2vec") #Please Install if required  
library(tm)

## Loading required package: NLP

library(text2vec)

## Reading the Transcripts

data <- read.csv(file = 'transcripts.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

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

## Cleaning and Preprocessing the text (Cleansing Techniques)

#Write your answer here for Question 1  
#Hint: use getTransformations() function in tm Package  
#https://cran.r-project.org/web/packages/tm/tm.pdf  
my.corpus <- tm\_map(my.corpus, removeWords, stopwords("en"))

## Warning in tm\_map.SimpleCorpus(my.corpus, removeWords, stopwords("en")):  
## transformation drops documents

#By applying a stop word removal algorithm to our text pre=processing we are going to remove words that do not drive the analysis and free up space, which helps processing.   
my.corpus <- tm\_map(my.corpus, removePunctuation)

## Warning in tm\_map.SimpleCorpus(my.corpus, removePunctuation): transformation  
## drops documents

#Similarly to the stop word text pre-processing techniques we are removing noise that does not contribute to the meaning of the sentence with the punctionation removal algorithm.  
#install.packages(SnowballC)  
library(SnowballC)  
my.corpus <- tm\_map(my.corpus, stemDocument, language="english")

## Warning in tm\_map.SimpleCorpus(my.corpus, stemDocument, language = "english"):  
## transformation drops documents

#By applying a stemming algorithm we are trimming the suffix and prefix of words, which helps text pre-processing by simplifying the words.

##Creating a uni-gram Term Document Matrix

term.doc.matrix <- TermDocumentMatrix(my.corpus)  
inspect(term.doc.matrix[1:10,1:10])

## <<TermDocumentMatrix (terms: 10, documents: 10)>>  
## Non-/sparse entries: 25/75  
## Sparsity : 75%  
## Maximal term length: 9  
## 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 1 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 7 1 3 0 1 2 3 0 3 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 1 0 0 0 0  
## affection 1 0 0 0 0 0 0 0 0 0

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

term.doc.matrix <- as.matrix(term.doc.matrix)  
head(term.doc.matrix)

## 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  
## 247 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## abil 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1  
## 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  
## absolut 1 0 0 0 0 0 0 0 0 2 0 2 0 0 0 1 0 0 0 0 1 0 0 0 1 0  
## accident 1 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0  
## actual 7 3 0 1 2 3 0 3 0 1 2 1 0 1 3 13 0 0 0 2 8 13 0 3 10 3  
## 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  
## 247 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## abil 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 3 0  
## abl 0 0 1 0 3 0 5 0 0 0 0 3 1 0 0 0 0 0 0 2 8 0 0  
## absolut 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0  
## accident 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## actual 0 1 7 1 0 1 16 1 1 14 2 0 2 2 1 0 7 0 2 3 13 1 7  
## 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  
## 247 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## abil 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0 0  
## abl 1 0 9 0 4 0 0 0 0 1 1 0 1 0 1 3 0 0 0 2 5 0 0  
## absolut 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0  
## accident 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## actual 1 2 9 3 0 1 2 11 2 4 0 0 0 0 1 13 0 1 2 2 5 2 3  
## 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  
## 247 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## abil 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0  
## abl 0 0 0 3 2 0 1 1 0 3 2 1 0 0 0 0 4 2 1 0 0 0 0  
## absolut 0 0 0 0 0 0 0 0 0 0 1 0 2 1 0 0 0 0 0 1 0 0 0  
## accident 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## actual 1 1 1 2 10 3 3 0 7 0 5 7 1 7 2 1 11 0 14 1 2 2 2  
## Docs  
## Terms 96 97 98 99 100  
## 247 0 0 0 0 0  
## abil 0 1 2 1 1  
## abl 0 0 2 2 2  
## absolut 0 1 0 0 0  
## accident 0 0 0 0 0  
## actual 0 7 0 4 4

dim(term.doc.matrix)

## [1] 9877 100

## Declaring weights (TF-IDF)

get.tf.idf.weights <- function(tf.vec) {  
 # Computes the tfidf weights from the term frequency vector  
 n.docs <- length(tf.vec)  
 doc.frequency <- length(tf.vec[tf.vec > 0])  
 weights <- rep(0, length(tf.vec))  
 relative.frequency <- tf.vec[tf.vec > 0] / sum(tf.vec[tf.vec > 0])  
 weights[tf.vec > 0] <- relative.frequency \* log(n.docs/doc.frequency)  
 return(weights)  
}

## Declaring weights (TF-IDF variants)

#First Varient   
get.tf.idf.weights1 <- function(tf.vec) {  
 # Computes the tfidf weights from the term frequency vector  
 n.docs <- length(tf.vec)  
 doc.frequency <- length(tf.vec[tf.vec > 0])  
 weights <- rep(0, length(tf.vec))  
 relative.frequency <- tf.vec[tf.vec > 0] / sum(tf.vec[tf.vec > 0])  
 weights[tf.vec > 0] <- relative.frequency \* 1  
 return(weights)  
}  
#Second Varient   
get.tf.idf.weights2 <- function(tf.vec) {  
 # Computes the tfidf weights from the term frequency vector  
 n.docs <- length(tf.vec)  
 doc.frequency <- length(tf.vec[tf.vec > 0])  
 weights <- rep(0, length(tf.vec))  
 relative.frequency <- 1 + log(tf.vec[tf.vec > 0])  
 weights[tf.vec > 0] <- relative.frequency \* log(1+(n.docs/doc.frequency))  
 return(weights)  
}  
#Third Varient   
get.tf.idf.weights3 <- function(tf.vec) {  
 # Computes the tfidf weights from the term frequency vector  
 n.docs <- length(tf.vec)  
 doc.frequency <- length(tf.vec[tf.vec > 0])  
 weights <- rep(0, length(tf.vec))  
 relative.frequency <- tf.vec[tf.vec > 0]  
 weights[tf.vec > 0] <- relative.frequency \* log(n.docs/doc.frequency)  
 return(weights)  
}

###Computing Cosine Similarity and Displaying a heatmap

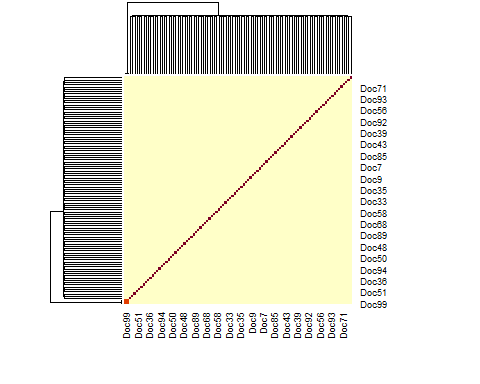
tfidf.matrix <- t(apply(term.doc.matrix, 1,  
 FUN = function(row) {get.tf.idf.weights(row)}))  
  
colnames(tfidf.matrix) <- my.docs$Names  
  
head(tfidf.matrix)

##   
## Terms Doc1 Doc2 Doc3 Doc4 Doc5 Doc6  
## 247 1.956011503 0.000000000 0 0.0000000000 0.000000000 0.000000000  
## abil 0.056525999 0.000000000 0 0.0000000000 0.000000000 0.056525999  
## abl 0.021388599 0.000000000 0 0.0000000000 0.014259066 0.007129533  
## absolut 0.081657068 0.000000000 0 0.0000000000 0.000000000 0.000000000  
## accident 1.304007668 0.000000000 0 0.0000000000 0.000000000 0.000000000  
## actual 0.005947548 0.002548949 0 0.0008496497 0.001699299 0.002548949  
##   
## Terms Doc7 Doc8 Doc9 Doc10 Doc11 Doc12  
## 247 1.956011503 0.000000000 0 0.0000000000 0.000000000 0.0000000000  
## abil 0.000000000 0.000000000 0 0.0000000000 0.000000000 0.0565259988  
## abl 0.007129533 0.000000000 0 0.0142590660 0.007129533 0.0071295330  
## absolut 0.000000000 0.000000000 0 0.1633141360 0.000000000 0.1633141360  
## accident 0.000000000 0.000000000 0 0.0000000000 0.000000000 0.0000000000  
## actual 0.000000000 0.002548949 0 0.0008496497 0.001699299 0.0008496497  
##   
## Terms Doc13 Doc14 Doc15 Doc16 Doc17 Doc18  
## 247 0 0.0000000000 0.000000000 0.00000000 0.00000000 0.0000000  
## abil 0 0.0000000000 0.056525999 0.00000000 0.00000000 0.0000000  
## abl 0 0.0000000000 0.000000000 0.02851813 0.01425907 0.0213886  
## absolut 0 0.0000000000 0.000000000 0.08165707 0.00000000 0.0000000  
## accident 0 0.0000000000 2.608015337 0.00000000 0.00000000 0.0000000  
## actual 0 0.0008496497 0.002548949 0.01104545 0.00000000 0.0000000  
##   
## Terms Doc19 Doc20 Doc21 Doc22 Doc23 Doc24  
## 247 0.000000 0.000000000 0.000000000 0.00000000 0 0.000000000  
## abil 0.056526 0.000000000 0.000000000 0.00000000 0 0.000000000  
## abl 0.000000 0.007129533 0.049906731 0.02138860 0 0.007129533  
## absolut 0.000000 0.000000000 0.081657068 0.00000000 0 0.000000000  
## accident 0.000000 0.000000000 0.000000000 0.00000000 0 0.000000000  
## actual 0.000000 0.001699299 0.006797197 0.01104545 0 0.002548949  
##   
## Terms Doc25 Doc26 Doc27 Doc28 Doc29 Doc30  
## 247 0.000000000 0.000000000 0 0.0000000000 0.000000000 0.0000000000  
## abil 0.000000000 0.056525999 0 0.0000000000 0.056525999 0.0000000000  
## abl 0.021388599 0.000000000 0 0.0000000000 0.007129533 0.0000000000  
## absolut 0.081657068 0.000000000 0 0.0816570680 0.000000000 0.0000000000  
## accident 0.000000000 0.000000000 0 0.0000000000 0.000000000 0.0000000000  
## actual 0.008496497 0.002548949 0 0.0008496497 0.005947548 0.0008496497  
##   
## Terms Doc31 Doc32 Doc33 Doc34 Doc35  
## 247 0.0000000 0.0000000000 0.00000000 0.0000000000 0.0000000000  
## abil 0.0000000 0.0000000000 0.00000000 0.0565259988 0.0000000000  
## abl 0.0213886 0.0000000000 0.03564767 0.0000000000 0.0000000000  
## absolut 0.0000000 0.0816570680 0.00000000 0.0000000000 0.0000000000  
## accident 0.0000000 0.0000000000 0.00000000 0.0000000000 0.0000000000  
## actual 0.0000000 0.0008496497 0.01359439 0.0008496497 0.0008496497  
##   
## Terms Doc36 Doc37 Doc38 Doc39 Doc40 Doc41  
## 247 0.0000000 0.000000000 0.0000000 0.000000000 0.000000000 0.0000000000  
## abil 0.0000000 0.000000000 0.0000000 0.000000000 0.000000000 0.0000000000  
## abl 0.0000000 0.000000000 0.0213886 0.007129533 0.000000000 0.0000000000  
## absolut 0.0000000 0.000000000 0.0000000 0.081657068 0.000000000 0.0000000000  
## accident 0.0000000 0.000000000 0.0000000 0.000000000 0.000000000 0.0000000000  
## actual 0.0118951 0.001699299 0.0000000 0.001699299 0.001699299 0.0008496497  
##   
## Terms Doc42 Doc43 Doc44 Doc45 Doc46 Doc47  
## 247 0 0.000000000 0 0.000000000 0.000000000 0.00000000  
## abil 0 0.000000000 0 0.056525999 0.000000000 0.00000000  
## abl 0 0.000000000 0 0.000000000 0.014259066 0.05703626  
## absolut 0 0.000000000 0 0.000000000 0.000000000 0.00000000  
## accident 0 0.000000000 0 0.000000000 0.000000000 0.00000000  
## actual 0 0.005947548 0 0.001699299 0.002548949 0.01104545  
##   
## Terms Doc48 Doc49 Doc50 Doc51 Doc52  
## 247 0.0000000000 0.000000000 0.0000000000 0.000000000 0.000000000  
## abil 0.1695779965 0.000000000 0.0000000000 0.000000000 0.000000000  
## abl 0.0000000000 0.000000000 0.0071295330 0.000000000 0.064165797  
## absolut 0.0000000000 0.000000000 0.0000000000 0.000000000 0.000000000  
## accident 0.0000000000 0.000000000 0.0000000000 0.000000000 0.000000000  
## actual 0.0008496497 0.005947548 0.0008496497 0.001699299 0.007646847  
##   
## Terms Doc53 Doc54 Doc55 Doc56 Doc57  
## 247 0.000000000 0.00000000 0.0000000000 0.000000000 0.000000000  
## abil 0.056525999 0.00000000 0.0000000000 0.000000000 0.000000000  
## abl 0.000000000 0.02851813 0.0000000000 0.000000000 0.000000000  
## absolut 0.000000000 0.00000000 0.0000000000 0.000000000 0.000000000  
## accident 0.000000000 0.00000000 0.0000000000 0.000000000 0.000000000  
## actual 0.002548949 0.00000000 0.0008496497 0.001699299 0.009346146  
##   
## Terms Doc58 Doc59 Doc60 Doc61 Doc62 Doc63  
## 247 0.000000000 0.000000000 0.000000000 0 0.000000000 0.00000000  
## abil 0.056525999 0.000000000 0.000000000 0 0.000000000 0.05652600  
## abl 0.000000000 0.007129533 0.007129533 0 0.007129533 0.00000000  
## absolut 0.000000000 0.000000000 0.081657068 0 0.000000000 0.08165707  
## accident 0.000000000 0.000000000 0.000000000 0 0.000000000 0.00000000  
## actual 0.001699299 0.003398599 0.000000000 0 0.000000000 0.00000000  
##   
## Terms Doc64 Doc65 Doc66 Doc67 Doc68 Doc69  
## 247 0.0000000000 0.00000000 0 0.0000000000 0.000000000 0.000000000  
## abil 0.0000000000 0.05652600 0 0.0000000000 0.000000000 0.000000000  
## abl 0.0071295330 0.02138860 0 0.0000000000 0.000000000 0.014259066  
## absolut 0.0000000000 0.00000000 0 0.0000000000 0.081657068 0.000000000  
## accident 0.0000000000 0.00000000 0 0.0000000000 0.000000000 0.000000000  
## actual 0.0008496497 0.01104545 0 0.0008496497 0.001699299 0.001699299  
##   
## Terms Doc70 Doc71 Doc72 Doc73 Doc74  
## 247 0.000000000 0.000000000 0.000000000 0.0000000000 0.0000000000  
## abil 0.056525999 0.000000000 0.000000000 0.0000000000 0.0565259988  
## abl 0.035647665 0.000000000 0.000000000 0.0000000000 0.0000000000  
## absolut 0.081657068 0.000000000 0.000000000 0.0000000000 0.0000000000  
## accident 0.000000000 0.000000000 0.000000000 0.0000000000 0.0000000000  
## actual 0.004248248 0.001699299 0.002548949 0.0008496497 0.0008496497  
##   
## Terms Doc75 Doc76 Doc77 Doc78 Doc79  
## 247 0.0000000000 0.000000000 0.000000000 0.000000000 0.000000000  
## abil 0.0000000000 0.000000000 0.000000000 0.000000000 0.056525999  
## abl 0.0000000000 0.021388599 0.014259066 0.000000000 0.007129533  
## absolut 0.0000000000 0.000000000 0.000000000 0.000000000 0.000000000  
## accident 0.0000000000 0.000000000 0.000000000 0.000000000 0.000000000  
## actual 0.0008496497 0.001699299 0.008496497 0.002548949 0.002548949  
##   
## Terms Doc80 Doc81 Doc82 Doc83 Doc84  
## 247 0.000000000 0.000000000 0.0000000 0.000000000 0.000000000  
## abil 0.000000000 0.000000000 0.0000000 0.000000000 0.000000000  
## abl 0.007129533 0.000000000 0.0213886 0.014259066 0.007129533  
## absolut 0.000000000 0.000000000 0.0000000 0.081657068 0.000000000  
## accident 0.000000000 0.000000000 0.0000000 0.000000000 0.000000000  
## actual 0.000000000 0.005947548 0.0000000 0.004248248 0.005947548  
##   
## Terms Doc85 Doc86 Doc87 Doc88 Doc89  
## 247 0.0000000000 0.000000000 0.000000000 0.0000000000 0.000000000  
## abil 0.0000000000 0.056525999 0.000000000 0.0000000000 0.000000000  
## abl 0.0000000000 0.000000000 0.000000000 0.0000000000 0.028518132  
## absolut 0.1633141360 0.081657068 0.000000000 0.0000000000 0.000000000  
## accident 0.0000000000 0.000000000 0.000000000 0.0000000000 0.000000000  
## actual 0.0008496497 0.005947548 0.001699299 0.0008496497 0.009346146  
##   
## Terms Doc90 Doc91 Doc92 Doc93 Doc94  
## 247 0.00000000 0.000000000 0.0000000000 0.000000000 0.000000000  
## abil 0.00000000 0.000000000 0.0565259988 0.000000000 0.000000000  
## abl 0.01425907 0.007129533 0.0000000000 0.000000000 0.000000000  
## absolut 0.00000000 0.000000000 0.0816570680 0.000000000 0.000000000  
## accident 0.00000000 0.000000000 0.0000000000 0.000000000 0.000000000  
## actual 0.00000000 0.011895095 0.0008496497 0.001699299 0.001699299  
##   
## Terms Doc95 Doc96 Doc97 Doc98 Doc99 Query  
## 247 0.000000000 0 0.000000000 0.00000000 0.000000000 0.000000000  
## abil 0.000000000 0 0.056525999 0.11305200 0.056525999 0.056525999  
## abl 0.000000000 0 0.000000000 0.01425907 0.014259066 0.014259066  
## absolut 0.000000000 0 0.081657068 0.00000000 0.000000000 0.000000000  
## accident 0.000000000 0 0.000000000 0.00000000 0.000000000 0.000000000  
## actual 0.001699299 0 0.005947548 0.00000000 0.003398599 0.003398599

dim(tfidf.matrix)

## [1] 9877 100

similarity.matrix <- sim2(t(tfidf.matrix), method = 'cosine')  
heatmap(similarity.matrix)



##Showing the Results

sort(similarity.matrix["Query", ], decreasing = TRUE)[1:10]

## Doc99 Query Doc47 Doc58 Doc3 Doc97   
## 1.000000000 1.000000000 0.012579770 0.008470972 0.005469461 0.005464656   
## Doc91 Doc36 Doc19 Doc2   
## 0.005388431 0.005308758 0.005298112 0.005174653

## Use the following chunck to comment and conclude after conducting your comparative analyses

#The first TF-IDF variant is similar to the original variant in the aspects of ordered doc weights. We can see that they share 9 out of the 10 highest weighted docs with similar order. The first variant differs from the original in the in individual weight, we can see that the first variants weights are more than doubled for each doc not including the matching doc 99 and the query.  
#The second TF-IDF variant is also similar to the original variant by ordered doc weights. We can see that they share 8 out of 10 highest weighted docs, but differ in the ordering. The weights of each doc in the second variant is drastically increased compared to the original with the 10th highest weighted doc being more than the third highest in the original.  
#The final TF-IDF variant is also similar to the original variant by ordered doc weights. They share 7 out of 10 highest weighted docs and differ in order. The weights of each doc in the second variant is drastically increased compared to the original.   
#Overall we can see more change in each variant from the original. This tells me as we change the TF in the function it increases variance in the weight order. Whereas change in IDF shows more increase in weigth value.

## Use the following chunck to answer Question 4

#Terms with two adjacent words is only worth pursuing if the term's meaning differs from the term when both words are separated. These are referred to as an oxymoron, an example is "living dead"; which greatly varies in meaning when separate or together. The term frequency would need to be weighted higher in the TF-IDF.