Title: Assignment #3: (*Analysis of Text Mining*)

Purpose: *This assignment is to study the text mining process in R. It helps us to identify the important similarity metrics that can be used for computing the relevant documents for clustering.*

Dataset(s): *The dataset used is the Assignment Corpus database.*

Approach:

- *First, the dataset is loaded, checked if there are any NA's and omitted if any. (This dataset had none)*
- *The required libraries are loaded such as arules, arulesViz and tm.*
- *A volatile corpus of the dataset is stored in a variable.*
- *Stopword removal, punctuation removal, stemming, converting words to lower, converting the document to a plain text, stripping whitespace and removing numbers is done.*
- *It is observed that the document corpus is sensistive to the order in which the process is done, ie; after making the document a plain text is when I am able to obtain the term matrix for the corpus*
- *Document Term Matrix for the corpus is computed. It can be computed in four different control settings such as:*
    1. *WeightTfIdf*
    2. *WeightTf*
    3. *WeightBin*
    4. *WeightSmart*
- *The document term matrix is supposed to be having a sparsity of 98% and it does not differ much with respect to the control metric for the term matrix.*
- *However, when converting a dense matrix with a sparse control limit of 0.1%, the document term matrix of WeightTfIdf is observed to have sparse 5% and the rest of the metrics gave results of sparse 1%.*
- *Switching the control can also change the threshold of sparsity that can be allowed.*
- *Now the distance matrix is computed using different methods such as:*
    1. *Eucledian*
    2. *Manhattan*
    - *It is computed for both dense and sparse matrix.*
- *K-Means Clustering is applied on both the dense and sparse matrix also using both the eucledian and the manhattan distance*

Eucledian : *Dense Matrix : Kmeans (betweenss/totss) – 94%* *Sparse Matrix: Kmeans (betweenss/totss) – 71%*
Manhattan:

Dense Matrix: Kmeans (betweenss/totss) – 85%
Sparse Matrix: Kmeans (betweenss/totss) -25%

- *Hierarchical clustering is applied on the dataset using method "ward.D2" and it is plotted.*

Summary:

Text mining completely depends on how the words are classified based on the similarity and clustered together. So it is trivial to do the pre-processing. Else if stopwords are not removed or the root word is not obtained by stemming, the clustering of documents will be significantly erroring out. Moreover, the denser matrix provides more accurate results for clustering than the sparse matrix. It is also observed that TfIDf provides better results in terms of document terms being compared based on similarity. And eucledian distance metric provides better results compared to manhattan distance metric for distance metric.

Appendix :

This includes the R code that I have done to obtain the analysis for the text mining:

```
library(arules)
library(arulesViz)
library(tm)
library(ggplot2)

#path to the Assignment corpus <- get from user - but I embeded it here
cname <- file.path("/Users","sasrinivasan","Downloads","Education","Spring 17","R for Data Scientists","Assignments","AssignmentCorpus")
#dir(cname) #just making sure that right file is present

docs <- VCorpus(DirSource(cname))
#summary(docs)

#pre-process the documents - punctuation, numbers, special characters, convert to lower, remove sto
docs <- tm_map(docs,content_transformer(PlainTextDocument))
docs <- tm_map(docs,removePunctuation)
docs <- tm_map(docs,removeNumbers)
docs = tm_map(docs, content_transformer(tolower) )
docs <- tm_map(docs,removeWords,stopwords("english"))
docs <- tm_map(docs,stripWhitespace)
docs <- tm_map(docs,stemDocument)
docs <- tm_map(docs,content_transformer(PlainTextDocument))

#create a document term matrix out of it
```

```
dtm <- DocumentTermMatrix(docs,control = list(weighting = weightTfIdf))
dtm
inspect(dtm[1:5,1:20])

#create using TFWeight
dtm.tf <- DocumentTermMatrix(docs,control = list(weighting = weightTf))
dtm.tf
inspect(dtm.tf[1:5,1:20])

#create using WeightBin
dtm.bin <- DocumentTermMatrix(docs,control = list(weighting = weightBin))
dtm.bin
inspect(dtm.bin[1:5,1:20])

#create weightSmart
dtm.smart <- DocumentTermMatrix(docs,control = list(weighting = weightSMART))
dtm.smart
inspect(dtm.smart[1:5,1:20])

#organize the terms by frequency
freq <- colSums(as.matrix(dtm))
length(freq)
ord <- order(freq)


#create a dense matrix by removing the sparse words
dtms <- removeSparseTerms(dtm,0.1)
inspect(dtms)

#least frequent terms
freq[head(ord)]

#most frequent terms
freq[tail(ord)]

#table of frequencies
head(table(freq),20)

tail(table(freq),20)

#less frequent dtm, dtms

freq <- colSums(as.matrix(dtms))
freq
```

```
findFreqTerms(dtm,lowfreq = 20)

#another approach for frequent terms
wf <- data.frame(word = names(freq), freq= freq)
head(wf)

#plot words appear atleast 20 times
p <- ggplot(subset(wf,freq>20),aes(word,freq))
p <- p + geom_bar(stat="identity")
p <- p + theme(axis.text.x=element_text(angle=45, hjust=1))
p

#how often terms appear together
#findAssocs(dtm, c("piano" , "cartoon"), corlimit=0.98)

#h cluster on the package - on terms
d <- dist(t(dtms),method = "euclidian")
fit <- hclust(d=d,method="ward.D2")
fit

plot(fit)

#cluster on the docs
d <- dist(dtms,method="euclidian")
fit <- hclust(d=d, method = "ward.D2")
fit
plot(fit)


#kmeans
d <- dist(dtm,method="manhattan")

kc <- kmeans(d,4)
kc$betweenss/kc$totss


d <- dist(dtms,method="manhattan")

kc <- kmeans(d,4)
kc$betweenss/kc$totss

d <- dist(dtm,method="euclidian")
```

```
kc <- kmeans(d,4)
kc$betweenss/kc$totss

d <- dist(dtms,method="euclidian")

kc <- kmeans(d,4)
kc$betweenss/kc$totss
```