**I. Load and Explore a Corpus**

**I.1. Install packages:**

## wordcloud -- plot a word cloud

## biclust -- Data preprocessing (normalization and discretisation),

## visualisation, and validation of bicluster

## tm -- Framework for text mining.

## SnowballC -- Provides wordStem() for stemming.

## dplyr -- Data preparation and pipes.

## RColorBrewer -- Generate palette of colours for plots.

## ggplot2 -- Plot word frequencies.

## scales -- Include commas in numbers.

## Rgraphviz -- Correlation plots.

## fpc - Flexible Procedures for Clustering

require('wordcloud')

require('biclust')

require('cluster')

require('igraph')

require('dplyr')

require('scales')

require('SnowballC')

require('RColorBrewer')

require('ggplot2')

require('tm')

require('Rgraphviz')

require('fpc')

**I.2. Load the Corpus**

getSources()

getReaders()

# Windows

mypwd = file.path('C:','MENGQIAN LU','Columbia Stat Club',

'StatFest\_2015','Data')

mypwd

## Mac

mypwd = file.path('~','Desktop','Columbia Stat Club','StatFest\_2015','Data')

mypwd

## Check what is under your present working directory

dir(mypwd)

## To load a Corpus of text documents

docs = Corpus(DirSource(mypwd))

**I.3. Explore the Corpus**

class(docs)

class(docs[[1]])

inspect(docs[1])

**II. Pre-processing your text documents, the 'Corpus'**

***tm\_map* -- Ensure all transformation are applied to all documents within the Corpus**

**II.1. Simple replacement**

We will be using the *'crude'* data for the following demonstration.

data('crude')

docs = crude

docs

toSpace = content\_transformer(function(yourdata, target) gsub(target, ' ', yourdata))

docs = tm\_map(docs, toSpace, "/")

docs = tm\_map(docs, toSpace, "@")

inspect(docs[1])

**Note: search regular expression in R for more information on how to define a search pattern for your *target* value**

**II.2. To Lowercase**

docs <- tm\_map(docs, content\_transformer(tolower))

**II.3. Remove numbers**

docs = tm\_map(docs, removeNumbers)

**II.4. Remove punctuation**

docs = tm\_map(docs, removePunctuation)

**II.5. Remove stopwords**

words like **'for','very','of'** are common stop words.

length(stopwords('english'))

stopwords('english')

docs = tm\_map(docs, removeWords, stopwords("english"))

**II.6. Remove own stop words**

This step is **optional** for pre-processing.

docs = tm\_map(docs, removeWords,c('school','department'))

**II.7. Define your acronym**

toString = content\_transformer(function(x,from,to) gsub(from,to,x))

docs = tm\_map(docs, toString, 'Columbia Statistics Club','CSC')

docs = tm\_map(docs, toString,'Earth and Environmental Engineering', 'EAEE')

**II.8. "Stemming" your documents**

A word can then be recognizable to the computer, despite whether or not it may have a variety of possible endings, such as "ing", "es" or "s"

docs = tm\_map(docs,stemDocument)

**II.9. Stripping the whitespace due to previous "Stemming" process**

docs = tm\_map(docs,stripWhitespace)

**II.10. Tell R your docs are ready as "Plain Text Documents"**

Once you have completed your pre-processing of the text documents, or your Corpus. Get the text ready as **plain text documents**. This step is crucial, and it is the end of the pre-processing stage.

docs = tm\_map(docs,PlainTextDocument)

**END OF PREPROCESSING**

**III. Explore the data**

First create a document-term matrix and a term-document matrix, which describes the frequency of terms that occur in a collection of documents. These are what you will be using from this point on. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms.

dtm = DocumentTermMatrix(docs)

dtm

dim(dtm) # display the dimensions

dimnames(dtm)[1]$Docs = paste('doc',1:20,sep ='')

inspect(dtm[1:5,1:20])

#Avoiding fill up your memory, inspect a subset of the document-term matrix. This subset views first 5 docs & first 20 terms

A term-document matrix is the transpose of the document-term matrix.

tdm = TermDocumentMatrix(docs)

tdm

dim(tdm)

dimnames(tdm)[2]$Docs = paste('doc',1:20,sep='')

inspect(tdm[1:20,1:5]) #compare to your dtm

# save dtm or tdm as .csv for future use if you like

write.csv(as.matrix(dtm),file='dtm.csv')

write.csv(as.matrix(tdm),file='tdm.csv')

**III.1. Individual level: Frequencies and counts**

III.1.(a) Organize terms by their frequencies and explore by plots

mfreq = colSums(as.matrix(dtm))

length(mfreq)

p1 = ggplot(data.frame(word=names(mfreq),freq=mfreq),aes(word,freq))+ geom\_bar(stat='identity') + theme(axis.text.x=element\_text(angle=45,hjust=0.5))

p1

**Too busy? let's just view some of them with a threshold**

p1 = ggplot(subset(data.frame(word=names(mfreq),freq=mfreq),freq>20),aes(word,freq))+ geom\_bar(stat='identity') + theme(axis.text.x=element\_text(size=12,color='red',fac='bold.italic',angle=45,hjust=0.5))

p1

III.1.(b) Check most and least frequently occuring words

mord = order(mfreq, decreasing=TRUE) #increasing order as default

mfreq[head(mord,20)]

mfreq[tail(mord,10)]

# Visualize it if you like

set.seed(100) # if you'd like to make the configuration of the layout consistent each time

wordcloud(names(mfreq),mfreq,

random.color=FALSE,

# colors chosen randomly or based on the frequency

random.order=FALSE) # plot in an dreasing frequency

set.seed(100)

wordcloud(names(mfreq),mfreq,

min.freq=5, # plot words apprear 10+ times

scale=c(4,0.5), # make it bigger with argument "scale"

colors=brewer.pal(8, "Dark2"), # use color palettes

random.color=FALSE,

random.order=FALSE)

III.1.(3) Check the frequency of frequencies, insight on the overall distribution of the frequencies of the bags of words

head(table(mfreq),20)

#The top number is the frequency with which words appear and the bottom number reflects how many words appear that frequently.

tail(table(mfreq), 20)

**Further removing sparse terms**

dtm

inspect(removeSparseTerms(dtm, 0.6)) #better inspect first

# Normally this reduces the matrix dramatically without losing significant relations inherent to the matrix:

dtmc = removeSparseTerms(dtm,sparse=0.6)

tdmc = removeSparseTerms(tdm,sparse=0.6)

# argument "sparse" is a numeric for the maximal allowed sparsity in the range [0 1].

**Now the new wordcloud looks like this:**

mfreq = colSums(as.matrix(dtmc))

set.seed(100)

wordcloud(names(mfreq),mfreq,

min.freq=5, # plot words apprear 10+ times

scale=c(4,0.5), # make it bigger with argument "scale"

colors=brewer.pal(8, "Dark2"), # use color palettes

random.color=FALSE,

# colors chosen randomly or based on the frequency

random.order=FALSE) # plot in an dreasing frequency

dim(dtmc)

inspect(dtmc)

III.1.(4) Find terms with frequency bounds

mfreq = colSums(as.matrix(dtm))

findFreqTerms(dtm, lowfreq = 30, highfreq = 90)

# or tabulate it first and find by your eyes

bagwords = data.frame(word=names(mfreq),freq=as.vector(mfreq))

head(bagwords,20)

head(sort(bagwords[,2],decreasing = TRUE),10)

**III.2.Between words: association and clustering**

**III.2.(1) Words association -- equals to 1 when words always appear together; equals to 0 when they never do**

# package: Rgraphviz

plot(dtm,terms=findFreqTerms(dtm,lowfreq = 20),corThreshold = 0.5)

plot(dtm,terms=findFreqTerms(dtm,lowfreq = 10),corThreshold = 0.5)

find associations with "oil" and "speech" with chosen limits 0.7 and 0.8 respectively

cor1 = findAssocs(dtm,terms=c('oil','waste'),corlimit=c(0.7,0.8))

cor1

qplot(names(cor1$oil),cor1$oil,stat='identity',decreasing=T,main="\"oil\" Word Associations",geom='bar',xlab='Terms',ylab='Association (>0.7)') + coord\_flip()

**III.2.(2) Clustering by term similarity**

**Types of Clustering**

**Hierachical clustering (e.g., agglomerative, divisive clustering)**

* Partitions can be visualized using a tree structure (dendrogram)
* Does not need the number of clusters as input
* Possible to view partitions at different levels of granularities using different K

**Flat or Partitional clustering (e.g., K-means, Gaussian mixture models and etc.)**

* Partitions are independent of each other

**III.2.(2).a Hierarchal Clustering**

mdist = dist(tdmc,method='euclidian')

mfit = hclust(d=mdist,method='ward.D2')

mfit

plot(mfit,hang=-1)

# draw dendrogram with red borders that identify groups

rect.hclust(mfit,k=5,border='red')

# you can also cut the tree into groups of data

grps = cutree(mfit,k=c(3,5,7)) # specify No. of clusters you want

# compare different groupings

table(grp2=grps[,'3'],grp4=grps[,'5'])

**III.2.(2).b K-means Clustering**

cluster words into groups to minimize the total sum of the squared distance of every point to its corresponding cluster centroid. Particitions are independent of each other.

mkm = kmeans(mdist,3)

clusplot(as.matrix(mdist),mkm$cluster,color=T,shade=T,labels=2,lines=0)

**NOTES:**

**Hierarchical Clustering vs. K-means**

* K-means clustering produces a single partitioning
* Hierarchical clustering can give diﬀerent partitionings depending on the level-of-resolution we are looking at
* K-means clustering needs the number of clusters to be speciﬁed
* Hierarchical clustering doesn’t need the number of clusters to be speciﬁed
* K-means clustering is usually more eﬃcient run-time wise
* Hierarchical clustering can be slow (has to make several merge/split decisions)
* No clear consensus on which of the two produces better clustering

**IV. Topic Modeling**

require(topicmodels)

dtm2 = as.DocumentTermMatrix(tdmc)

mlda = LDA(dtm2,k=3) #find k topics

mterms = terms(mlda,4) # find the first 4 terms of each topic

mterms

mterms = apply(mterms,MARGIN=2,paste,collapse=', ')

# First topic identified for every document

mtopic = topics(mlda,1)

mtopics = data.frame(doc=1:20,topic1=mtopic)

qplot(doc,..count..,data=mtopics,geom='density',

fill=mterms[mtopic],position='stack')