**SEARCH for keywords- Equivalent to CONTENT :D**

* **Process**
* **Master**
* **Scraping**
* **Scraper**
* **Grams**
* **K-means**
* **Bayes**
* **SVM**
* **Wordcloud**
* **IDF**
* **Topic**
* **Recommender**
* **Sentiment1**
* **Sentiment2**
* **Project**
* **Senti**
* **Assignment**

**Scrape + Pre process**

library(dplyr)

library(rvest)

library(stringr)

article\_html <- read\_html("https://www.thehindu.com/opinion/editorial/none-gains-the-hindu-editorial-on-us-withdrawal-from-who/article32035308.ece")

article\_body <- html\_nodes(article\_html,xpath='//\*[contains(concat( " ", @class, " " ), concat( " ", "drop-caps", " " ))]')

article\_body

article\_body\_text <- html\_text(article\_body)

article\_body\_text

textdata = data.frame(article\_body\_text)

names(textdata) = "text"

#View(textdata)

str(textdata)

# textdata <- data.frame(textdata)

# create corpus

library(tm)

myCorpus<-VCorpus(VectorSource(textdata$text))

writeLines(as.character(myCorpus[[1]]))

# tokenization

# install.packages("tidytext")

library(tidytext)

library(dplyr)

# textdata <- mutate(textdata, text = as.character(text))

textdata %>%

unnest\_tokens(word,text) %>%

count(word,sort=TRUE)

## data preprocessing

# convert to lower case

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

#myCorpus <- tm\_map(myCorpus, tolower)

# remove punctuation

myCorpus <- tm\_map(myCorpus, removePunctuation)

# remove numbers

myCorpus <- tm\_map(myCorpus, removeNumbers)

# remove stopwords from corpus with default in built stopwords

stopwords()

myCorpus <- tm\_map(myCorpus,removeWords,stopwords("english"))

# add extra stop words: 'fy' within the inbuilt list

myStopwords <- c(stopwords("english"), "fy")

# remove stopwords from corpus

myCorpus <- tm\_map(myCorpus, removeWords, myStopwords)

# Remove urls

removeURL <- function(x) gsub("http[[:alnum:]]\*", "", x)

myCorpus <- tm\_map(myCorpus, content\_transformer(removeURL))

# Removing white spaces (extra spaces)

myCorpus <- tm\_map(myCorpus,stripWhitespace)

writeLines(as.character(myCorpus[[1]]))

dtm <- DocumentTermMatrix(myCorpus)

dtm.matrix <- as.matrix(dtm)

wordcount <- colSums(dtm.matrix)

topten <- head(sort(wordcount, decreasing=TRUE), 10)

topten # Top ten words with counts

# Plotting the data -

# For top 10 -

dtm <- DocumentTermMatrix(myCorpus)

dtm.matrix <- as.matrix(dtm)

wordcount <- colSums(dtm.matrix)

topten <- head(sort(wordcount, decreasing=TRUE), 10)

library(reshape2)

library(ggplot2)

dfplot <- as.data.frame(melt(topten))

dfplot$word <- dimnames(dfplot)[[1]]

dfplot$word <- factor(dfplot$word,

levels=dfplot$word[order(dfplot$value,

decreasing=TRUE)])

fig <- ggplot(dfplot, aes(x=word, y=value)) + geom\_bar(stat="identity")

fig <- fig + xlab("Word in Corpus")

fig <- fig + ylab("Count")

print(fig)

**Pre process MASTER**

# Session 3 - 10/07/2020 - 2:30 pm

# Text pre processing

library(tm)

library(NLP)

library(textstem)

library(SnowballC)

library(reshape2)

library(ggplot2)

textdata = data.frame(readLines("D:/Analytics/R/TA/textdata2.txt"))

names(textdata) = "text"

View(textdata)

str(textdata)

# textdata <- data.frame(textdata)

# create corpus

myCorpus<-VCorpus(VectorSource(textdata$text))

for(i in 1:4){

writeLines(as.character(myCorpus[[i]]))

}

# tokenization

# install.packages("tidytext")

library(tidytext)

library(dplyr)

# textdata <- mutate(textdata, text = as.character(text))

textdata %>%

unnest\_tokens(word,text) %>%

count(word,sort=TRUE)

## data preprocessing

# convert to lower case

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

#myCorpus <- tm\_map(myCorpus, tolower)

# remove punctuation

myCorpus <- tm\_map(myCorpus, removePunctuation)

# remove numbers

myCorpus <- tm\_map(myCorpus, removeNumbers)

# remove stopwords from corpus with default in built stopwords

stopwords()

myCorpus <- tm\_map(myCorpus,removeWords,stopwords("english"))

# add extra stop words: 'fy' within the inbuilt list

myStopwords <- c(stopwords("english"), "fy")

# remove stopwords from corpus

myCorpus <- tm\_map(myCorpus, removeWords, myStopwords)

# Remove urls

removeURL <- function(x) gsub("http[[:alnum:]]\*", "", x)

myCorpus <- tm\_map(myCorpus, content\_transformer(removeURL))

# Removing white spaces (extra spaces)

myCorpus <- tm\_map(myCorpus,stripWhitespace)

# Stemming

library(SnowballC)

myCorpus <- tm\_map(myCorpus, stemDocument)

# Lemmatization

library(textstem)

lemmatize\_strings(myCorpus)

# Replacing specific pattern

myCorpus <- tm\_map(myCorpus, content\_transformer(gsub), pattern = c("indias"), replacement = c("india"))

myCorpus <- tm\_map(myCorpus, content\_transformer(gsub), pattern = c("chinas"), replacement = c("china"))

# Basic pre processing ends here -

# The document term matrix ----- Data exploration (Top 10)

dtm <- DocumentTermMatrix(myCorpus)

dtm.matrix <- as.matrix(dtm)

wordcount <- colSums(dtm.matrix)

topten <- head(sort(wordcount, decreasing=TRUE), 10)

topten # Top ten words with counts

# Plotting the data -

# For top 10 -

dtm <- DocumentTermMatrix(myCorpus)

dtm.matrix <- as.matrix(dtm)

wordcount <- colSums(dtm.matrix)

topten <- head(sort(wordcount, decreasing=TRUE), 10)

library(reshape2)

library(ggplot2)

dfplot <- as.data.frame(melt(topten))

dfplot$word <- dimnames(dfplot)[[1]]

dfplot$word <- factor(dfplot$word,

levels=dfplot$word[order(dfplot$value,

decreasing=TRUE)])

fig <- ggplot(dfplot, aes(x=word, y=value)) + geom\_bar(stat="identity")

fig <- fig + xlab("Word in Corpus")

fig <- fig + ylab("Count")

print(fig)

### For all the data

# data Exploration

dtm <- DocumentTermMatrix(myCorpus)

dtm.matrix <- as.matrix(dtm)

wordcount <- colSums(dtm.matrix)

topten <- sort(wordcount, decreasing=TRUE)

library(reshape2)

library(ggplot2)

dfplot <- as.data.frame(melt(topten))

dfplot$word <- dimnames(dfplot)[[1]]

dfplot$word <- factor(dfplot$word,

levels=dfplot$word[order(dfplot$value,

decreasing=TRUE)])

fig <- ggplot(dfplot, aes(x=value, y=word)) + geom\_bar(stat="identity")

fig <- fig + xlab("Count")

fig <- fig + ylab("Word in Corpus")

print(fig)

**Scraping basics**

# Session 2 - Date - 05/07/2020 - 6:00pm-7:30pm

#install.packages("rvest")

library(dplyr)

library(rvest)

library(stringr)

google <- read\_html("https://news.google.com")

headline\_all <- google %>% html\_nodes("article") %>% html\_text("span") %>%

str\_split("(<=[a-z0-9!?\\.])(?=[A-Z])")

headline\_all <- sapply(headline\_all, function(x) x[1])

headline\_all[1:10]

# Scrapping news article

article\_html <- read\_html("https://zeenews.india.com/india/iafs-sukhoi-su-30mkis-apache-patrol-lac-in-ladakh-send-a-strong-message-to-china-2293693.html")

article\_body <- html\_nodes(article\_html,xpath="//p")

# xpath can be fetched using selectorgadget extension in chrome

article\_body # this is in html format which is having <p> for paragraphs

article\_body\_text <- html\_text(article\_body) # Filtering only text part removing <p> part

article\_body\_text

article\_html <- read\_html("https://in.finance.yahoo.com/quote/%5ENSEI/history?p=%5ENSEI")

article\_body <- html\_nodes(article\_html,xpath='//\*[contains(concat( " ", @class, " " ), concat( " ", "smartphone\_Px\(20px\)", " " ))]')

# Session 3 - 10/07/2020 - 2:30 pm

# Extracting table data - using inspect

library(xml2)

url<-"https://en.wikipedia.org/wiki/2016\_Summer\_Olympics\_medal\_table"

medal\_tally <-url %>% read\_html() %>%

html\_nodes(xpath='//\*[@id="mw-content-text"]/div/table[2]')%>% html\_table(fill=TRUE)

medal\_tally <- medal\_tally[[1]]

head(medal\_tally)

**Amazon -Scraper**

# Session 6 - Amazon review

library(tidyverse)

library(rvest)

scrape\_amazon <- function(ASIN, page\_num){

url\_reviews <- paste0("https://www.amazon.in/product-reviews/",ASIN,"/?pageNumber=",page\_num)

doc <- read\_html(url\_reviews) # Assign results to `doc`

trim <- function (x) gsub("^\\s+|\\s+$", "", x)

# Review Title

doc %>%

html\_nodes("[class='a-size-base a-link-normal review-title a-color-base review-title-content a-text-bold']") %>%

html\_text() %>% gsub("\n", "", .) %>% trim() -> review\_title

# Review Text

doc %>%

html\_nodes("[class='a-size-base review-text review-text-content']") %>%

html\_text() %>% gsub("\n", "", .) %>% trim()-> review\_text

# Number of stars in review

doc %>%

html\_nodes("[data-hook='review-star-rating']") %>%

html\_text() -> review\_star

# Return a tibble

tibble(review\_title,

review\_text,

review\_star,

page = page\_num) %>% return()

}

scrape\_amazon(ASIN = "B07VT1Z7DV", page\_num = 1) ->outputlist1

scrape\_amazon(ASIN = "B07VT1Z7DV", page\_num = 2) ->outputlist2

#head()

outputlist1$review\_text

outputlist1$review\_title

review\_data<-as.data.frame(rbind(outputlist1,outputlist2))

View(review\_data)

write.csv(review\_data,"review.csv")

**N-Grams**

# Session 6 - n-grams

library(SnowballC)

library(textstem)

library(tm)

library(NLP)

library(reshape2)

library(ggplot2)

library(tidyr)

library(igraph)

library(ggraph)

library(tidytext)

library(wordcloud)

textdata <- data.frame(readLines("Zara Review.txt"))

names(textdata) = "text"

View(textdata)

str(textdata)

# tokenization

library(tidytext)

library(dplyr)

uni\_gram<-as.data.frame(textdata%>%

unnest\_tokens(word,text) %>%

count(word,sort=TRUE))

uni\_gram

str(uni\_gram)

data\_bigrams <- textdata %>%

unnest\_tokens(bigram, text, token = "ngrams", n = 2)%>%

count(bigram,sort=TRUE)

data\_bigrams

str(data\_bigrams)

### deleting stop words

bigrams\_separated <- data\_bigrams %>%

separate(bigram, c("word1", "word2"), sep = " ")

bigrams\_filtered <- bigrams\_separated %>%

filter(!word1 %in% stop\_words$word) %>%

filter(!word2 %in% stop\_words$word)

# new bigram counts:

bigram\_counts <- bigrams\_filtered %>%

count(word1, word2, sort = TRUE)

bigram\_counts

######################## for any error -

# textdata <- mutate(textdata, text = as.character(text))

# Tri-grams

tri\_grams<-textdata %>%

unnest\_tokens(trigram, text, token = "ngrams", n = 3) %>%

separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%

filter(!word1 %in% stop\_words$word,

!word2 %in% stop\_words$word,

!word3 %in% stop\_words$word) %>%

count(word1, word2, word3, sort = TRUE)

tri\_grams

## Visualization

library(igraph)

# original counts

bigram\_counts

# filter for only relatively common combinations

bigram\_graph <- bigram\_counts %>%

filter(n > 0) %>%

graph\_from\_data\_frame()

bigram\_graph

library(ggraph)

set.seed(2017)

ggraph(bigram\_graph, layout = "fr") +

geom\_edge\_link() +

geom\_node\_point() +

geom\_node\_text(aes(label = name), vjust = 1, hjust = 1)

## combine words

bigrams\_united <- bigrams\_filtered %>%

unite(bigram, word1, word2, sep = " ")

bigrams\_united

##bar chart

ggplot(head(bigrams\_united,20), aes(reorder(bigram,n), n)) +

geom\_bar(stat = "identity") + coord\_flip() +

xlab("Bigrams") + ylab("Frequency") +

ggtitle("Most frequent bigrams")

**Unsupervised – K-means clustering**

# Unsupervised learning - 31/07/2020

library(tm)

#Import data

sms\_raw<-read.csv("spam2.csv")

# Create corpus

sms\_corpus <- VCorpus(VectorSource(sms\_raw$V2))

# print corpus

inspect(sms\_corpus)

corpus\_clean <- sms\_corpus

# Remove Numbers

corpus\_clean <- tm\_map(x = corpus\_clean, FUN = removeNumbers)

# Remove punctuation

corpus\_clean <- tm\_map(x = corpus\_clean, FUN = removePunctuation)

# Remove stop words

corpus\_clean <- tm\_map(x = corpus\_clean, FUN = removeWords, stopwords("english"))

# Remove extra white spaces

corpus\_clean <- tm\_map(x = corpus\_clean, FUN = stripWhitespace)

# Create Document Term Matrix

DTM <- DocumentTermMatrix(x = corpus\_clean)

inspect(DTM[1:10, 1001:1010])

#TF-IDF

DTM\_tfidf<-weightTfIdf(na.omit(DTM))

inspect(DTM\_tfidf[1:10, 1001:1010])

# Clustering

m<-as.matrix(na.omit(DTM\_tfidf))

rownames(m)<-1:nrow(m)

norm\_eucl<-function(m)

m/apply(m, 1, function(x) sum(x^2)^.5)

m\_norm<-norm\_eucl(m)

results<-kmeans(na.omit(m\_norm),20)

results$size

results$withinss

results$betweenss

clusters <- 1:20

for (i in clusters) {

cat("Cluster " , i, ":", findFreqTerms(DTM\_tfidf[results[['cluster']]==i,], 2),"\n\n")

}

**Supervised – Naïve Bayes**

# Session 7 - Date- 24/07/2020

# Supervised learning in TEXT

library(tm)

library(SnowballC)

library(wordcloud)

library(RColorBrewer)

library(e1071)

library(caret)

#Import data

sms\_raw <- read.csv("spam.csv")

#View the first few lines of the dataset

head(sms\_raw)

#Select & rename appropriate columns of the dataset

sms\_raw <- sms\_raw[,1:2]

colnames(sms\_raw) <- c("Type", "Text")

sms\_raw$Type <- factor(sms\_raw$Type)

str(sms\_raw)

# Check the number of spam and ham messages

table(sms\_raw$Type)

prop.table(table(sms\_raw$Type))

# load the tm package

library(tm)

# Create corpus

sms\_corpus <- VCorpus(x = VectorSource(sms\_raw$Text))

# Print corpus

sms\_corpus

# Check the text in some messages and their type

lapply(sms\_corpus[5:8], as.character)

sms\_raw$Type[5:8]

corpus\_clean <- sms\_corpus

# Remove Numbers

corpus\_clean <- tm\_map(x = corpus\_clean, FUN = removeNumbers)

# Transform all letters to lower case

#corpus\_clean <- tm\_map(x = corpus\_clean, FUN = tolower)

# Remove punctuation

corpus\_clean <- tm\_map(x = corpus\_clean, FUN = removePunctuation)

# Remove stop words

corpus\_clean <- tm\_map(x = corpus\_clean, FUN = removeWords, stopwords())

library(SnowballC)

# Stem words in corpus

corpus\_clean <- tm\_map(x = corpus\_clean, FUN = stemDocument)

# Remove extra white spaces

corpus\_clean <- tm\_map(x = corpus\_clean, FUN = stripWhitespace)

# Create Document Term Matrix

DTM <- DocumentTermMatrix(x = corpus\_clean)

DTM

## Create training and test set

# Create Training Set

DTM\_train <- DTM[1:round(nrow(DTM)\*0.80, 0), ]

# Create Test Set

DTM\_test <- DTM[(round(nrow(DTM)\*0.80, 0)+1):nrow(DTM), ]

# Create vectors with labels for the training and test set

train\_labels <- sms\_raw[1:round(nrow(sms\_raw)\*0.80, 0), ]$Type

test\_labels <- sms\_raw[(round(nrow(sms\_raw)\*0.80, 0)+1):nrow(DTM), ]$Type

# Check proportion of ham and spam is similar on the training and test set

prop.table(table(train\_labels))

prop.table(table(test\_labels))

library(wordcloud)

# Create wordcloud for the whole dataset

wordcloud(words = corpus\_clean,

min.freq = 100, # minimum number of times a word must be present before it appears

random.order = FALSE, # Arrange most frequent words to be in the center of the word cloud

color = (colors = c("#4575b4","#74add1","#abd9e9","#e0f3f8","#fee090","#fdae61","#f46d43","#d73027")) # Colour gradient for the font

)

##

threshold <- 0.1

min\_freq = round(DTM$nrow\*(threshold/100),0) # calculate minimum frequency

min\_freq

# Create vector of most frequent words

frequent\_words <- findFreqTerms(x = DTM, lowfreq = min\_freq)

str(frequent\_words)

# Filter DTM to only have most frequent words

DTM\_train\_most\_frequent <- DTM\_train[, frequent\_words]

DTM\_test\_most\_frequent <- DTM\_test[, frequent\_words]

# Check dimension of DTM

dim(DTM\_train\_most\_frequent)

# Create function that converts numeric values to "Yes" or "No" if word is present or absent in document

is\_present <- function(x) {

x <- ifelse(test = x > 0, yes = "Yes", no = "No")

}

# Apply is\_present() function to training and test DTM

DTM\_train\_most\_frequent <- apply(X = DTM\_train\_most\_frequent,

MARGIN = 2, # Apply function to columns

FUN = is\_present) # Specify function to be used

DTM\_test\_most\_frequent <- apply(X = DTM\_test\_most\_frequent,

MARGIN = 2, # Apply function to columns

FUN = is\_present) # Specify function to be used

library(e1071)

# Create model from the training dataset

spam\_classifier <- naiveBayes(x = DTM\_train\_most\_frequent, y = train\_labels)

train\_predictions <- predict(object = spam\_classifier, newdata = DTM\_train\_most\_frequent)

# Create confusion matrix

confusionMatrix(data = train\_predictions, reference = train\_labels, positive = "spam", dnn = c("Prediction", "Actual"))

## Make predictions on test set

test\_predictions <- predict(object = spam\_classifier, newdata = DTM\_test\_most\_frequent)

## Create confusion matrix

# install.packages("caret")

library(caret)

# Create confusion matrix

confusionMatrix(data = test\_predictions, reference = test\_labels, positive = "spam", dnn = c("Prediction", "Actual"))

**Supervised – SVM**

library(tm)

library(SnowballC)

library(wordcloud)

library(RColorBrewer)

library(e1071)

library(caret)

library(readr)

#Import data

sms\_raw <- read.csv("spam.csv")

#View the first few lines of the dataset

head(sms\_raw)

#Select & rename appropriate columns of the dataset

sms\_raw <- sms\_raw[,1:2]

colnames(sms\_raw) <- c("Type", "Text")

sms\_raw$Type <- factor(sms\_raw$Type)

str(sms\_raw)

# Check the number of spam and ham messages

table(sms\_raw$Type)

prop.table(table(sms\_raw$Type))

# Create corpus

sms\_corpus <- VCorpus(x = VectorSource(sms\_raw$Text))

# Print corpus

sms\_corpus

# Check the text in some messages and their type

lapply(sms\_corpus[5:8], as.character)

sms\_raw$Type[5:8]

corpus\_clean <- sms\_corpus

# Remove Numbers

corpus\_clean <- tm\_map(x = corpus\_clean, FUN = removeNumbers)

# Remove punctuation

corpus\_clean <- tm\_map(x = corpus\_clean, FUN = removePunctuation)

# Remove stop words

corpus\_clean <- tm\_map(x = corpus\_clean, FUN = removeWords, stopwords())

# Remove extra white spaces

corpus\_clean <- tm\_map(x = corpus\_clean, FUN = stripWhitespace)

# Create Document Term Matrix

DTM <- DocumentTermMatrix(x = corpus\_clean)

## Create training and test set

# Create Training Set

DTM\_train <- DTM[1:round(nrow(DTM)\*0.80, 0), ]

# Create Test Set

DTM\_test <- DTM[(round(nrow(DTM)\*0.80, 0)+1):nrow(DTM), ]

# Create vectors with labels for the training and test set

train\_labels <- sms\_raw[1:round(nrow(sms\_raw)\*0.80, 0), ]$Type

test\_labels <- sms\_raw[(round(nrow(sms\_raw)\*0.80, 0)+1):nrow(DTM), ]$Type

# Check proportion of ham and spam is similar on the training and test set

prop.table(table(train\_labels))

prop.table(table(test\_labels))

# Create wordcloud for the whole dataset

wordcloud(words = corpus\_clean,

min.freq = 50, # minimum number of times a word must be present before it appears

random.order = FALSE, # Arrange most frequent words to be in the center of the word cloud

color = (colors = c("#4575b4","#74add1","#abd9e9","#e0f3f8","#fee090","#fdae61","#f46d43","#d73027")) # Colour gradient for the font

)

##

threshold <- 0.1

min\_freq = round(DTM$nrow\*(threshold/100),0) # calculate minimum frequency

min\_freq

# Create vector of most frequent words

frequent\_words <- findFreqTerms(x = DTM, lowfreq = min\_freq)

# Filter DTM to only have most frequent words

DTM\_train\_most\_frequent <- DTM\_train[, frequent\_words]

DTM\_test\_most\_frequent <- DTM\_test[, frequent\_words]

# Check dimension of DTM

dim(DTM\_train\_most\_frequent)

# Create function that converts numeric values to either 1 or 0 if word is present or absent in document

is\_present <- function(x) {

x <- ifelse(test = x > 0, 1, 0)

}

# Apply is\_present() function to training and test DTM

DTM\_train\_most\_frequent <- apply(X = DTM\_train\_most\_frequent,

MARGIN = 2, # Apply function to columns

FUN = is\_present) # Specify function to be used

DTM\_test\_most\_frequent <- apply(X = DTM\_test\_most\_frequent,

MARGIN = 2, # Apply function to columns

FUN = is\_present) # Specify function to be used

library(e1071)

#SVM

svm\_model<-svm(formula=as.factor(train\_labels)~.,data=DTM\_train\_most\_frequent)

svm\_model

svm\_train\_predictions <- predict(object = svm\_model, newdata = DTM\_train\_most\_frequent)

confusionMatrix(data = svm\_train\_predictions, reference = as.factor(train\_labels), positive = "spam", dnn = c("Prediction", "Actual"))

svm\_test\_predictions <- predict(object = svm\_model, newdata = DTM\_test\_most\_frequent)

confusionMatrix(data = svm\_test\_predictions, reference = as.factor(test\_labels), positive = "spam", dnn = c("Prediction", "Actual"))

**Wordcloud**

# Session - 6 - Zara Review

library(SnowballC)

library(textstem)

library(tm)

library(NLP)

library(reshape2)

library(ggplot2)

library(tidyr)

library(igraph)

library(ggraph)

library(tidytext)

library(wordcloud)

textdata <- data.frame(readLines("Zara Review.txt"))

names(textdata) = "text"

View(textdata)

str(textdata)

# Load the data as a corpus

docs <- VCorpus(VectorSource(textdata$text))

View(docs)

#Text transformation

toSpace <- content\_transformer(function (x , pattern ) gsub(pattern, " ", x))

docs <- tm\_map(docs, toSpace, "/")

docs <- tm\_map(docs, toSpace, "@")

docs <- tm\_map(docs, toSpace, "\\|")

#Cleaning the text

# Convert the text to lower case

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

# Remove numbers

docs <- tm\_map(docs, removeNumbers)

# Remove english common stopwords

#docs <- tm\_map(docs, removeWor

docs <- tm\_map(docs, removeWords, stopwords("english"))

# Remove punctuations

docs <- tm\_map(docs, removePunctuation)

# Eliminate extra white spaces

docs <- tm\_map(docs, stripWhitespace)

# Text stemming

# docs <- tm\_map(docs, stemDocument)

#Build a term-document matrix

dtm <- TermDocumentMatrix(docs)

m <- as.matrix(dtm)

v <- sort(rowSums(m),decreasing=TRUE)

d <- data.frame(word = names(v),freq=v)

head(d, 10)

#Generate the Word cloud

set.seed(1234)

wordcloud(words = d$word, freq = d$freq, min.freq = 1,

max.words=200, random.order=FALSE, rot.per=0.35,

colors=brewer.pal(8, "Dark2"))

**TF-IDF**

# Session 5 : Date - 17/07/2020 -- 2:30 pm

# TF-IDF ------

library(tm)

library(proxy)

library(dplyr)

doc <- c( "The sky is blue.", "The Sun is bright today.",

"The sun in the sky is bright.", "We can see the shining sun, the bright sun." )

# create term frequency matrix using functions from tm library

doc\_corpus <- Corpus( VectorSource(doc) )

control\_list <- list(removePunctuation = TRUE, stopwords = TRUE, tolower = TRUE)

tdm <- TermDocumentMatrix(doc\_corpus, control = control\_list)

stopwords()

# print

( tf <- as.matrix(tdm) )

# idf

( idf <- log( ncol(tf) / ( 1 + rowSums(tf != 0) ) ) )

# diagonal matrix

( idf <- diag(idf) )

tf\_idf <- crossprod(tf, idf)

colnames(tf\_idf) <- rownames(tf)

View(tf\_idf)

# Note that normalization is computed "row-wise"

tf\_idf / sqrt( rowSums( tf\_idf^2 ) )

doc

**Topic Modelling**

setwd("D:/Analytics/R/TA/Webhose")

textdata <- read.csv("combined.csv")

library(dplyr)

library(rvest)

library(stringr)

library(tm)

library(tidytext)

library(dplyr)

colnames(textdata) <- "text"

head(textdata)

myCorpus <- VCorpus(VectorSource(textdata$text))

writeLines(as.character(myCorpus[[1]]))

# Tokenization

# textdata <- mutate(textdata, text = as.character(text))

textdata %>%

unnest\_tokens(word,text) %>%

count(word,sort=TRUE)

# Data pre-processing

# Convert to lower case

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

# Remove punctuation

myCorpus <- tm\_map(myCorpus, removePunctuation)

# Remove numbers

myCorpus <- tm\_map(myCorpus, removeNumbers)

# Remove stop-words from corpus with default in built stopwords

stopwords()

myCorpus <- tm\_map(myCorpus,removeWords,stopwords("english"))

# add extra stop words: 'fy' within the inbuilt list

myStopwords <- c(stopwords("english"), "also", "can", "will", "said", "like", "one")

# remove stopwords from corpus

myCorpus <- tm\_map(myCorpus, removeWords, myStopwords)

# Remove urls

removeURL <- function(x) gsub("http[[:alnum:]]\*", "", x)

myCorpus <- tm\_map(myCorpus, content\_transformer(removeURL))

# Removing white spaces (extra spaces)

myCorpus <- tm\_map(myCorpus,stripWhitespace)

writeLines(as.character(myCorpus[[1]]))

# Creating Document Term Matrix

Dtm <- DocumentTermMatrix(myCorpus)

inspect(Dtm)

Dtm2 <- removeSparseTerms(Dtm, sparse=0.98)

inspect(Dtm2)

#load topic models library

#install.packages("topicmodels")

library(topicmodels)

#Set parameters for Gibbs sampling

#set burn in

burnin <- 4000

#set iterations

iter <- 2000

#thin the spaces between samples

thin <- 500

#use random integers as seed

seed <-list(2003,5,63,100001,765)

#set random starts at 5

nstart <- 5

# return the highest probability as the result

best <- TRUE

#Number of topics

k <- 5

#Run LDA using Gibbs sampling

ldaOut <-LDA(Dtm2,k, method="Gibbs", control=list(nstart=nstart, seed = seed, best=best, burnin = burnin, iter = iter, thin=thin))

#view the top 6 terms for each of the 5 topics, create a matrix and write to csv

terms(ldaOut,6)

ldaOut.terms <- as.matrix(terms(ldaOut,6))

#view the topic assignment for each document

topics(ldaOut)

##################

#write out results

#docs to topics

ldaOut.topics <- as.matrix(topics(ldaOut))

setwd("D:/Analytics/R/TA/Webhose/Outputs")

write.csv(ldaOut.topics,file=paste("LDAGibbs",k,"DocsToTopics.csv"))

#top 6 terms in each topic

ldaOut.terms <- as.matrix(terms(ldaOut,6))

write.csv(ldaOut.terms,file=paste("LDAGibbs",k,"TopicsToTerms.csv"))

#probabilities associated with each topic assignment

topicProbabilities <- as.data.frame(ldaOut@gamma)

write.csv(topicProbabilities,file=paste("LDAGibbs",k,"TopicProbabilities.csv"))

**Recommender**

#The dataset includes "5000 users from the anonymous rating data from the Jester Online Joke Recommender System."

#The ratings contained in the data frame range between -10.00 and 10.00.

#Each user included has rated at least 36 jokes.

#The dataset is stored as an item of class realRatingMatrix

library(recommenderlab)

data(Jester5k)

typeof(Jester5k)

jester <- as.vector(Jester5k@data)

jester <- matrix(data = jester, nrow = Jester5k@data@Dim[1], ncol = Jester5k@data@Dim[2])

jester <- as.data.frame(jester)

dim(jester)

head(jester)

rownames(jester) <- Jester5k@data@Dimnames[[1]]

names(jester) <- Jester5k@data@Dimnames[[2]]

#The goal of the recommender system will be to provide 5 recommended jokes per user for the first twenty users in the dataset.

#Based on the information available and the dimensions of the dataset.

#Creating the Model

premade\_model <- Recommender(data = Jester5k, method = "UBCF", parameter = list(method = "Cosine", k = 20))

#Applying the Model

#The model is then applied to the dataset to obtain five recommendations each for the first twenty users.

#Here, the dataset is again subsetted to exclude those with fewer than 20 unrated jokes.

Jester5k\_rec <- Jester5k[100 - rowCounts(Jester5k) > 20]

premade\_allrecs <- predict(object = premade\_model, newdata = Jester5k\_rec, n = 5)

#Results

premade\_recs <- data.frame(matrix(nrow = 20, ncol = 5))

rownames(premade\_recs) <- names(premade\_allrecs@items)[1:20]

for(i in 1:20) {

for (j in 1:5) {

premade\_recs[i, j] <- paste0("j", premade\_allrecs@items[[i]][j])

}

}

names(premade\_recs) <- as.character(1:5)

premade\_recs

**Sentiment1 – 1st Guest Lecture**

# Date - 21/08/2020 - 2:30 pm - Guest Lecture - Sentiment Analysis

setwd("D:/Analytics/R/TA/Sentiment")

# Load library

library (tm)

## Loading required package: NLP

library (stringr)

library (rvest)

## Loading required package: xml2

library (SnowballC)

trip = read.csv ("tripadvisor\_reviews.csv")

N

# sentiment, echo=FALSE}

# Sentiment Analysis of Hotel Reviews on Trip Advisor - JW Marriott

hotel\_reviews <- as.character (trip $ review)

# Load the positive and negative lexicon data and explore

positive\_lexicon <- read.csv ("positive-lexicon.txt")

negative\_lexicon <- read.csv ("negative-lexicon.txt")

# Load the stop words text file and explore

stop\_words <- read.csv ("stopwords\_en.txt")

vector\_source\_reviews = VectorSource (hotel\_reviews)

reviews\_corpus <- Corpus (vector\_source\_reviews)

#inspect(reviews\_corpus)

reviews\_corpus[[2]] $ content

# Remove stop words

filtered\_corpus\_no\_stopwords <- tm\_map (reviews\_corpus, removeWords, stopwords ('english'))

## Warning in tm\_map.SimpleCorpus(reviews\_corpus, removeWords,

## stopwords("english")): transformation drops documents

#inspect(filtered\_corpus\_no\_stopwords)

# Remove Punctuation

filtered\_corpus\_no\_puncts <- tm\_map (filtered\_corpus\_no\_stopwords, removePunctuation)

## Warning in tm\_map.SimpleCorpus(filtered\_corpus\_no\_stopwords, removePunctuation):

## transformation drops documents

#inspect(filtered\_corpus\_no\_puncts)

# Remove Numbers

filtered\_corpus\_no\_numbers <- tm\_map (filtered\_corpus\_no\_puncts, removeNumbers)

## Warning in tm\_map.SimpleCorpus(filtered\_corpus\_no\_puncts, removeNumbers):

## transformation drops documents

#inspect(filtered\_corpus\_no\_numbers)

# Remove unwanted white spaces

filtered\_corpus\_no\_whitespace <- tm\_map (filtered\_corpus\_no\_numbers, stripWhitespace)

## Warning in tm\_map.SimpleCorpus(filtered\_corpus\_no\_numbers, stripWhitespace):

## transformation drops documents

#inspect(filtered\_corpus\_no\_whitespace)

# Make all words to lowercase

filtered\_corpus\_to\_lower <- tm\_map (filtered\_corpus\_no\_whitespace, content\_transformer (tolower))

## Warning in tm\_map.SimpleCorpus(filtered\_corpus\_no\_whitespace,

## content\_transformer(tolower)): transformation drops documents

#inspect(filtered\_corpus\_to\_lower)

# Remove stop words of the external file from the corpus and whitespaces again and inspect

stopwords\_vec <- as.data.frame (stop\_words)

final\_corpus\_no\_stopwords <- tm\_map (filtered\_corpus\_to\_lower, removeWords, stopwords\_vec[,1])

## Warning in tm\_map.SimpleCorpus(filtered\_corpus\_to\_lower, removeWords,

## stopwords\_vec[, : transformation drops documents

#inspect(final\_corpus\_no\_stopwords)

final\_corpus <- tm\_map (final\_corpus\_no\_stopwords, stripWhitespace)

## Warning in tm\_map.SimpleCorpus(final\_corpus\_no\_stopwords, stripWhitespace):

## transformation drops documents

#inspect(final\_corpus)

# Character representation of the corpus of first review

final\_corpus[[1]] $ content

## [1] " significant travel challenge day visit hotel hotel handled staff members chris jennifer expectations chris welcoming remembered time jennifer handled difficult situation professionalism thoroughness hotel lovely common"

hotel\_reviews[1]

## [1] "I had a significant travel challenge during my 5 day visit to this hotel and the hotel could not have handled it better. Staff members, Chris and Jennifer went beyond my expectations. Chris was especially welcoming and remembered my name each time I saw him. Jennifer handled my difficult situation with professionalism and thoroughness.\nThe hotel itself was lovely. Common..."

# Stem the words to their root of all reviews present in the corpus

stemmed\_corpus <- tm\_map (final\_corpus, stemDocument)

## Warning in tm\_map.SimpleCorpus(final\_corpus, stemDocument): transformation drops

## documents

stemmed\_corpus[[1]] $ content

## [1] "signific travel challeng day visit hotel hotel handl staff member chris jennif expect chris welcom rememb time jennif handl difficult situat profession thorough hotel love common"

TDM\_corpus <- TermDocumentMatrix (stemmed\_corpus)

# terms occurring with a minimum frequency of 5

findFreqTerms (TDM\_corpus, 5)

total\_pos\_count <- 0

total\_neg\_count <- 0

pos\_count\_vector <- c ()

neg\_count\_vector <- c ()

size <- length (stemmed\_corpus)

for (i in 1 : size){

corpus\_words<- list ( strsplit (stemmed\_corpus[[i]] $ content, split = " "))

#print(intersect(unlist(corpus\_words), unlist(positive\_lexicon))) ## positive words in current review

pos\_count <- length ( intersect ( unlist (corpus\_words), unlist (positive\_lexicon)))

#print(intersect(unlist(corpus\_words), unlist(negative\_lexicon))) ## negative words in current review

neg\_count <- length ( intersect ( unlist (corpus\_words), unlist (negative\_lexicon)))

if (pos\_count > neg\_count){

#print("It's a positive review")

} else {

#print("It's a negative review")

}

total\_count\_for\_current\_review <- pos\_count + neg\_count ## current positive and negative count

pos\_percentage <- (pos\_count \* 100) / total\_count\_for\_current\_review

neg\_percentage <- (neg\_count \* 100) / total\_count\_for\_current\_review

#print(pos\_percentage) ## current positive percentage

#print(neg\_percentage) ## current negtive percentage

total\_pos\_count <- total\_pos\_count + pos\_count ## overall positive count

total\_neg\_count <- total\_neg\_count + neg\_count ## overall negative count

pos\_count\_vector <- append (pos\_count\_vector, pos\_count)

neg\_count\_vector <- append (neg\_count\_vector, neg\_count)

}

print (pos\_percentage) ## current positive percentage

## [1] 71.42857

print (neg\_percentage) ## current negtive percentage

## [1] 28.57143

# Sentiment score of each review and visualizing using boxplot

counts <- data.frame (pos\_count\_vector, neg\_count\_vector)

sentiment <- data.frame ( c (1 : size),(pos\_count\_vector - neg\_count\_vector) / (pos\_count\_vector + neg\_count\_vector))

names (sentiment)= c ('review\_id','SentimentScore')

boxplot (sentiment $ SentimentScore[0 : 5] ~ sentiment $ review\_id[0 : 5])

# Visualiztion of positive and negative count of single review

singe\_review <- c (counts $ pos\_count\_vector[8], counts $ neg\_count\_vector[8])

barplot ( t ( as.data.frame (singe\_review)), ylab = "Count", xlab = "Positve v/s Negative", main = "Positive")

# Calculating overall percentage of positive and negative words of all the reviews

total\_pos\_count ## overall positive count

total\_count <- total\_pos\_count + total\_neg\_count

overall\_positive\_percentage <- (total\_pos\_count \* 100) / total\_count

overall\_negative\_percentage <- (total\_neg\_count \* 100) / total\_count

overall\_positive\_percentage ## overall positive percentage

overall\_negative\_percentage ## overall negative percentage

# Visualization of positive and negative word count for all the reviews

review\_count\_frame <- data.frame ( matrix ( c (pos\_count\_vector, neg\_count\_vector), nrow = 100, ncol = 2))

colnames (review\_count\_frame) <- c ("Positive Word Count", "Negative Word Count")

barplot (review\_count\_frame $ `Positive Word Count`, ylab = "Positive Word Count", xlab = "Reviews from 1 ")

# Visualization of Overall positive and negative reviews

percent\_vec <- c (overall\_positive\_percentage, overall\_negative\_percentage)

percent\_frame <- as.data.frame (percent\_vec)

rownames (percent\_frame) <- c ("Positive Reviews","Negative Reviews")

colnames (percent\_frame) <- c ("Percentage")

percentage <- t (percent\_frame)

barplot (percentage, ylab = "Percentage", main = "Sentiment Analysis of JW Marriot Reviews on TripAdvisor")

library (wordcloud)

## Loading required package: RColorBrewer

set.seed (1234) # for reproducibility

textM = as.matrix (TDM\_corpus)

#textM[0:10]

textFreqs <- rowSums (textM)

freqTab <- data.frame (term = names (textFreqs), num = textFreqs)

write.csv (freqTab, "freq.csv", row.names = FALSE)

wordcloud (freqTab $ term, freqTab $ num, max.words = 500, color = "green")

wordcloud (words = freqTab $ term, freq = freqTab $ num, min.freq = 4,

max.words=500, random.order=FALSE, rot.per=0.35,

colors= brewer.pal (8, "Dark2"))

freqTab = read.csv ('freq.csv', stringsAsFactors = FALSE)

dim (freqTab)

head (freqTab)

library (wordcloud2)

#wordcloud2(data=freqTab)

wordcloud2 (data=freqTab[0 : 100,], size=1.6, color='random-dark')

wordcloud2 (data=freqTab[100 : 200,], size = 0.7, shape = 'pentagon')

**Sentiment2 – 2nd Guest Lecture**

# Guest lecture - Date - 28/08/2020 - 2:30pm

#install.packages("tidyverse")

#install.packages("tidytext")

library(tidyverse)

setwd("D:/Analytics/R/TA/Sentiment")

roomba\_reviews <- read.csv('Roomba Reviews.csv', stringsAsFactors=FALSE)

str(roomba\_reviews)

View(roomba\_reviews)

roomba\_reviews %>% group\_by((Product)) %>%

summarise((number\_rows = n()))

# Average star ratings productwise

roomba\_reviews %>% group\_by((Product)) %>%

summarise((star\_mean = mean(Stars)))

# Tokenise

library(tidytext)

roomba\_tidy <- roomba\_reviews %>% unnest\_tokens(word, Review)

View(roomba\_tidy)

# Count

roomba\_tidy %>% count(word) %>% arrange(desc(n))

# Remove stop words -

# http://snowball.tartarus.org/algorithms/english/stop.txt

# http://www.lextek.com/manuals/onix/stopwords1.html

roomba\_tidy2 <- roomba\_tidy %>% anti\_join(stop\_words)

# Count

roomba\_tidy2 %>% count(word) %>% arrange(desc(n))

# Remove stop words (local)

local\_dict <- tribble(~word, ~lexicon, "roomba", "my\_dict", "2", "my\_dict", "ê", "my\_dict")

#Combining local dict with global dict

dict <- stop\_words %>% bind\_rows(local\_dict)

roomba\_tidy2 <- roomba\_reviews %>% unnest\_tokens(word, Review) %>% anti\_join(dict)

# Count

word\_count <- roomba\_tidy2 %>% count(word) %>% arrange(desc(n))

# Visualization

library(ggplot2)

ggplot(word\_count, aes(x = word, y = n)) + geom\_col()

# Count

word\_count2 <- roomba\_tidy2 %>% count(word) %>% filter(n>500) %>% arrange(desc(n))

ggplot(word\_count2, aes(x = word, y = n)) + geom\_col() + ggtitle("Word Counts")

# SENTIMENT Analysis

# Using BING(Binary), AFINN(Ordinal), LOUGHRAN(Other meanings), NRC(tidytext) dictionaries

# BING

get\_sentiments("bing")

#install.packages("textdata")

library(textdata)

get\_sentiments("afinn")

get\_sentiments("loughran")

get\_sentiments("nrc")

sentiment\_review <- roomba\_tidy2 %>% inner\_join(get\_sentiments("bing"))

sentiment\_review %>% count(sentiment)

sentiment\_review <- roomba\_tidy2 %>% inner\_join(get\_sentiments("loughran"))

sentiment\_review %>% count(sentiment)

sentiment\_review <- roomba\_tidy2 %>% inner\_join(get\_sentiments("nrc"))

sentiment\_review %>% count(sentiment)

sentiment\_review %>% count(sentiment) %>% arrange(desc(n))

pos\_neg <- sentiment\_review %>% count(sentiment) %>% filter(sentiment %in% c("positive", "negative"))

pos\_neg

ggplot(sentiment\_review, aes(x = word, y = sentiment)) + geom\_col()

# Regular expression

text <- c("Ania's favorite two colors are blue and red.", "Ania's favorite number is 1111.", "Ania lives at Dwarka, 42 Sub Way, Delhi", "She is 5 feet tall" , "Ania has visited 30 countries" , "Ania has ten fingers." , "Ania has worked at eleven different jobs" , " She can speak 3 languages" , "Ania's favorite food is pizza" , "Ania can name 10 facts about herself.")

str(text)

grep(pattern = "\\d\\s", x = text)

clean\_text <- gsub(pattern = "Ania", replacement = "She", text)

clean\_text

**Project Scrape**

library(dplyr)

library(rvest)

library(stringr)

library(tidyverse)

library(dplyr)

# Defining scraper function

scrape <- function(url){

flk <- read\_html(url)

flk %>% html\_nodes("[class='\_2xg6Ul']") %>% html\_text() %>% gsub("[^\x01-\x7F]", "", .) -> ttl

flk %>% html\_nodes("[class='qwjRop']") %>% html\_text() %>% gsub("READ MORE", "", .) %>% gsub("[^\x01-\x7F]", "", .)-> rv

rvw <- data.frame("Title" = ttl, "Review" = rv)

return(rvw)

}

# Defining blank data frame for capturing data

rvw\_master <- data.frame(Title=character(), Review=character())

#-------------------------------------------------------------------------------

url <- c()

for (i in seq(1,5)){

ur <- paste("https://www.flipkart.com/mobiles/mi~brand/pr?sid=tyy%2C4io&otracker=nmenu\_sub\_Electronics\_0\_Mi&page=",i) %>% gsub(" ", "", .)

url <- c(url,ur)

}

url\_list <- c()

for (i in url)

{

flk <- read\_html(i)

ff <- flk %>% html\_nodes("[class='\_31qSD5']") %>% html\_attr("href")

j <- 1

nn <- c()

for (i in ff)

{

nn[j] <- paste("https://www.flipkart.com",i) %>% gsub(" ", "", .)

j <- j + 1

}

url\_list <- c(url\_list, nn)

}

#-------------------------------------------------------------------------------

# Running scraper

for (i in url\_list){

rvw\_master <- rbind(rvw\_master,scrape(i))}

write.csv(rvw\_master, "MIReviews.csv", row.names = FALSE)

**Project Senti**

setwd("D:/Analytics/R/TA/Group Project")

textdata = read.csv("MIReviews.csv", header = TRUE)

library(SnowballC)

library(textstem)

library(tm)

library(NLP)

library(reshape2)

library(ggplot2)

library(tidyr)

library(igraph)

library(ggraph)

library(tidytext)

library(wordcloud)

library(dplyr)

colnames(textdata)

View(textdata)

str(textdata)

# Load the data as a corpus

docs <- VCorpus(VectorSource(textdata$Review))

View(docs)

#Text transformation

toSpace <- content\_transformer(function (x , pattern ) gsub(pattern, " ", x))

docs <- tm\_map(docs, toSpace, "/")

docs <- tm\_map(docs, toSpace, "@")

docs <- tm\_map(docs, toSpace, "\\|")

#Cleaning the text

# Convert the text to lower case

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

# Remove numbers

docs <- tm\_map(docs, removeNumbers)

# Remove english common stopwords

docs <- tm\_map(docs, removeWords, stopwords("english"))

# Remove punctuations

docs <- tm\_map(docs, removePunctuation)

# Eliminate extra white spaces

docs <- tm\_map(docs, stripWhitespace)

for(i in 1:100){

writeLines(as.character(docs[[i]]))

}

# Text stemming

# docs <- tm\_map(docs, stemDocument)

#Build a term-document matrix

dtm <- TermDocumentMatrix(docs)

m <- as.matrix(dtm)

v <- sort(rowSums(m),decreasing=TRUE)

d <- data.frame(word = names(v),freq=v)

head(d, 10)

#Generate the Word cloud

set.seed(1234)

wordcloud(words = d$word, freq = d$freq, min.freq = 1,

max.words=200, random.order=FALSE, rot.per=0.35,

colors=brewer.pal(8, "Dark2"))

names(textdata$Review) = "text"

textdata<-mutate(textdata,text= as.character(Review))

data\_bigrams <- textdata %>%

unnest\_tokens(bigram, text, token = "ngrams", n = 2)%>%

count(bigram,sort=TRUE)

data\_bigrams

### deleting stop words

bigrams\_separated <- data\_bigrams %>%

separate(bigram, c("word1", "word2"), sep = " ")

bigrams\_filtered <- bigrams\_separated %>%

filter(!word1 %in% stop\_words$word) %>%

filter(!word2 %in% stop\_words$word)

# new bigram counts:

bigram\_counts <- bigrams\_filtered %>%

count(word1, word2, sort = TRUE)

bigram\_counts

## combine words

bigrams\_united <- bigrams\_filtered %>%

unite(bigram, word1, word2, sep = " ")

#SENTIMENT ANALYSIS

# Tokenise

library(tidytext)

textdata$Review<-as.character(textdata$Review)

review\_tidy <- textdata %>% unnest\_tokens(word, Review)

View(textdata)

View(review\_tidy)

# Count

review\_tidy %>% count(word) %>% arrange(desc(n))

# Remove stop words -

# http://snowball.tartarus.org/algorithms/english/stop.txt

# http://www.lextek.com/manuals/onix/stopwords1.html

review\_tidy2 <- review\_tidy %>% anti\_join(stop\_words)

# Count

word\_count<-review\_tidy2 %>% count(word) %>% arrange(desc(n))

# Visualization

library(ggplot2)

dev.off()

ggplot(word\_count, aes(x = word, y = n)) + geom\_col()

# Count

word\_count2 <- review\_tidy2 %>% count(word) %>% filter(n>200) %>% arrange(desc(n))

ggplot(word\_count2, aes(x = word, y = n)) + geom\_col() + ggtitle("Word Counts")

# SENTIMENT Analysis

# Using BING(Binary), AFINN(Ordinal), LOUGHRAN(Other meanings), NRC(tidytext) dictionaries

# BING

get\_sentiments("bing")

install.packages("textdata")

library(textdata)

get\_sentiments("afinn")

get\_sentiments("loughran")

get\_sentiments("nrc")

sentiment\_review1 <- review\_tidy2 %>% inner\_join(get\_sentiments("bing"))

sentiment\_review1<-sentiment\_review1 %>% count(sentiment)%>% filter(n>200)%>% arrange(desc(n))

sentiment\_review2 <- review\_tidy2 %>% inner\_join(get\_sentiments("loughran"))

sentiment\_review2<-sentiment\_review2 %>% count(sentiment)%>% filter(n>200)%>% arrange(desc(n))

sentiment\_review3 <- review\_tidy2 %>% inner\_join(get\_sentiments("nrc"))

sentiment\_review3<-sentiment\_review3 %>% count(sentiment)%>% filter(n>200)%>% arrange(desc(n))

sentiment\_review3 %>% count(sentiment) %>% arrange(desc(n))

pos\_neg <- sentiment\_review3 %>% count(sentiment) %>% filter(sentiment %in% c("positive", "negative"))

pos\_neg

library(dplyr)

ggplot(sentiment\_review1, aes(x = n, y = sentiment)) + geom\_col() + ggtitle("BING library results") + xlab("Counts")

ggplot(sentiment\_review2, aes(x = n, y = sentiment)) + geom\_col() + ggtitle("Loughran library results") + xlab("Counts")

ggplot(sentiment\_review3, aes(x = n, y = sentiment)) + geom\_col() + ggtitle("NRC library results") + xlab("Counts")

**Assignment**

# Assignment 1 --- Date - 18/07/2020 --- ROOM -1

library(dplyr)

library(rvest)

library(stringr)

func <- function(htm, xp){

article\_html <- read\_html(htm)

article\_body <- html\_nodes(article\_html,xpath=xp)

article\_body\_text <- html\_text(article\_body)

textdata = data.frame(article\_body\_text)

names(textdata) = "text"

textdata <- data.frame(textdata)

return(textdata)

}

shrutika\_htm <- "https://www.dw.com/en/coronavirus-triggers-mental-health-crisis-in-india/a-54011738"

shrutika\_xp <- '//p[(((count(preceding-sibling::\*) + 1) = 28) and parent::\*)] | //p[(((count(preceding-sibling::\*) + 1) = 32) and parent::\*)] | //p[(((count(preceding-sibling::\*) + 1) = 33) and parent::\*)] | //p[(((count(preceding-sibling::\*) + 1) = 31) and parent::\*)] | //p[(((count(preceding-sibling::\*) + 1) = 30) and parent::\*)] | //p[(((count(preceding-sibling::\*) + 1) = 27) and parent::\*)] | //p[(((count(preceding-sibling::\*) + 1) = 25) and parent::\*)] | //p[(((count(preceding-sibling::\*) + 1) = 24) and parent::\*)] | //\*[(@id = "bodyContent")]//p[(((count(preceding-sibling::\*) + 1) = 1) and parent::\*)]'

souvik\_htm <- "https://www.newindianexpress.com/lifestyle/health/2020/may/17/61-indians-suffering-from-mental-health-issues-during-lockdown-survey-2144506.html"

souvik\_xp <- '//\*[(@id = "storyContent")] | //\*[(@id = "content\_head")]'

amandeep\_htm <- "https://indianexpress.com/article/lifestyle/life-style/matters-of-the-mind-hold-the-line-to-collectively-beat-the-crisis-6508614/"

amandeep\_xp <- '//\*[@id="section"]/div/div[3]/div[1]/div/div/div/div/p[1] | //\*[@id="section"]/div/div[3]/div[1]/div/div/div/div/p[2] | //\*[@id="section"]/div/div[3]/div[1]/div/div/div/div/p[3] | //\*[@id="section"]/div/div[3]/div[1]/div/div/div/div/p[4] | //\*[@id="section"]/div/div[3]/div[1]/div/div/div/div/p[6] | //\*[@id="section"]/div/div[3]/div[1]/div/div/div/div/p[7] | //\*[@id="section"]/div/div[3]/div[1]/div/div/div/div/p[9] | //\*[@id="section"]/div/div[3]/div[1]/div/div/div/div/p[11]'

shatakshi\_htm <- "https://www.edexlive.com/opinion/2020/jun/25/mental-health-matters-why-young-india-is-depressed-and-anxious-about-life-in-a-post-covid-world-12875.html"

shatakshi\_xp <- '//\*[(@id = "storyContent")]//p[(((count(preceding-sibling::\*) + 1) = 1) and parent::\*)]'

shriya\_htm <- "https://www.downtoearth.org.in/blog/health/covid-19-lockdown-why-india-must-brace-for-a-mental-health-crisis-71889"

shriya\_xp <- '//\*[@id="box\_0"]/div[2]/p[27]'

shubham\_htm <- "https://fit.thequint.com/mind-it/why-is-mental-healthcare-is-hard-to-get-in-india"

shubham\_xp <- '//\*[@id="8b899606-c515-4b6e-98c4-275339b9eff2"]/div[3]/div/h2 | //\*[@id="8b899606-c515-4b6e-98c4-275339b9eff2"]/div[3]/div/p[1] | //\*[@id="8b899606-c515-4b6e-98c4-275339b9eff2"]/div[3]/div/p[2] | //\*[@id="8b899606-c515-4b6e-98c4-275339b9eff2"]/div[3]/div/p[3]'

apurva\_htm <- "https://theprint.in/india/a-huge-mental-health-crisis-awaits-india-post-covid-but-only-the-power-of-community-will-help/427146/"

apurva\_xp <- '//\*[contains(concat( " ", @class, " " ), concat( " ", "st\_\_content-block--text", " " ))]//\*[contains(concat( " ", @class, " " ), concat( " ", "st\_\_content-block--text", " " ))]//\*[contains(concat( " ", @class, " " ), concat( " ", "st\_\_content-block--text", " " )) and (((count(preceding-sibling::\*) + 1) = 4) and parent::\*)]//p'

df <- data.frame()

df <- rbind(df, func(souvik\_htm,souvik\_xp))

df <- rbind(df, func(shrutika\_htm,shrutika\_xp))

df <- rbind(df, func(amandeep\_htm,amandeep\_xp))

df <- rbind(df, func(shriya\_htm,shriya\_xp))

df <- rbind(df, func(shatakshi\_htm,shatakshi\_xp))

df <- rbind(df, func(shubham\_htm,shubham\_xp))

df <- rbind(df, func(apurva\_htm, apurva\_xp))

# Eleminating 11th row as it is irrelevant

df2 <- data.frame(df[-11,])

names(df2) <- "text"

# Create corpus

library(tm)

myCorpus <- VCorpus(VectorSource(df2$text))

# Viewing the corpus

for(i in 1:33){

writeLines(as.character(myCorpus[[i]]))

}

## data pre-processing

# convert to lower case

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

# remove punctuation

myCorpus <- tm\_map(myCorpus, removePunctuation)

# remove numbers

myCorpus <- tm\_map(myCorpus, removeNumbers)

# Removing stop-words

myStopwords <- c(stopwords("english"), "ians", "â", 'cent', "per")

myCorpus <- tm\_map(myCorpus, removeWords, myStopwords)

# Removing white spaces (extra spaces)

myCorpus <- tm\_map(myCorpus,stripWhitespace)

# Term document matrix

dtm <- DocumentTermMatrix(myCorpus)

dtm.matrix <- as.matrix(dtm)

wordcount <- colSums(dtm.matrix)

topten <- head(sort(wordcount, decreasing=TRUE), 10)

topten # Top ten words with counts

# Plotting the data -

# For top 10 -

dtm <- DocumentTermMatrix(myCorpus)

dtm.matrix <- as.matrix(dtm)

wordcount <- colSums(dtm.matrix)

topten <- head(sort(wordcount, decreasing=TRUE), 10)

library(reshape2)

library(ggplot2)

dfplot <- as.data.frame(melt(topten))

dfplot$word <- dimnames(dfplot)[[1]]

dfplot$word <- factor(dfplot$word,

levels=dfplot$word[order(dfplot$value,

decreasing=TRUE)])

fig <- ggplot(dfplot, aes(x=word, y=value)) + geom\_bar(stat="identity")

fig <- fig + xlab("Word in Corpus")

fig <- fig + ylab("Count")

print(fig)

# N-gram analysis -------------

textdata <- data.frame(text = sapply(myCorpus, as.character), stringsAsFactors = FALSE)

library(tidytext)

library(dplyr)

uni\_gram <- as.data.frame(textdata%>%

unnest\_tokens(word,text) %>%

count(word,sort=TRUE))

uni\_gram

str(uni\_gram)

data\_bigrams <- textdata %>%

unnest\_tokens(bigram, text, token = "ngrams", n = 2)%>%

count(bigram,sort=TRUE)

data\_bigrams

str(data\_bigrams)

### deleting stop words

bigrams\_separated <- data\_bigrams %>%

separate(bigram, c("word1", "word2"), sep = " ")

bigrams\_filtered <- bigrams\_separated %>%

filter(!word1 %in% stop\_words$word) %>%

filter(!word2 %in% stop\_words$word)

# new bigram counts:

bigram\_counts <- bigrams\_filtered %>%

count(word1, word2, sort = TRUE)

bigram\_counts

# Tri-grams

tri\_grams<-textdata %>%

unnest\_tokens(trigram, text, token = "ngrams", n = 3) %>%

separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%

filter(!word1 %in% stop\_words$word,

!word2 %in% stop\_words$word,

!word3 %in% stop\_words$word) %>%

count(word1, word2, word3, sort = TRUE)

head(tri\_grams,10)

## Visualization

library(igraph)

# original counts

bigram\_counts

# filter for only relatively common combinations

bigram\_graph <- bigram\_counts %>%

filter(n > 0) %>%

graph\_from\_data\_frame()

bigram\_graph

library(ggraph)

set.seed(2017)

ggraph(bigram\_graph, layout = "fr") +

geom\_edge\_link() +

geom\_node\_point() +

geom\_node\_text(aes(label = name), vjust = 1, hjust = 1)

## combine words

bigrams\_united <- bigrams\_filtered %>%

unite(bigram, word1, word2, sep = " ")

bigrams\_united

##bar chart

ggplot(head(bigrams\_united,20), aes(reorder(bigram,n), n)) +

geom\_bar(stat = "identity") + coord\_flip() +

xlab("Bigrams") + ylab("Frequency") +

ggtitle("Most frequent bigrams")

# Visualising Tri-grams

# original counts

tri\_grams

# filter for only relatively common combinations

trigram\_graph <- tri\_grams %>%

filter(n > 0) %>%

graph\_from\_data\_frame()

trigram\_graph

ggraph(trigram\_graph, layout = "fr") +

geom\_edge\_link() +

geom\_node\_point() +

geom\_node\_text(aes(label = name), vjust = 1, hjust = 1)