# Independent Study - Healthcare Analytics Case Study Development with R Professor: Divakaran Liginlal

## Phase 1:

For the purposes of this project, I tried to map the people’s experiences in the Healthcare sector in reference to meetings doctors in Qatar. Meddy is a great new resource In Qatar for creating public reviews for doctors and allowing the general public to be more aware of what kind of doctor they are visiting. This not only helps put the power of choosing the best specialist in the hands of the patient but also motivates doctors to make their experience more wholesome for the patients.

Although Meddy is a great resource to gauge which doctor to consult when looking for oneself, it doesn’t paint a holistic picture of what the general atmosphere of Qatar’s healthcare system is like. One way to do this is to go around hospitals and pass out surveys to all the patients that go to doctor in Qatar and gauge their experience. But this old-school method can be problematic. Apart from the time and effort required to tediously collect all this data, the cost of hiring people that collect this may be immense. Then there is also a huge problem in regards to stopping people right after they may have to get medicine or may have received bad news, which may affect their willingness to partake in this survey. The overall problem is also that of data redundancy. The fact that Meddy already collects this data makes the process of collecting it again redundant. This project aims to make more use of the data available on Meddy’s website and paint a clearer picture on Qatar’s healthcare system.

The statistical coding language R and its packages were used in Phase 1. Parts and achievements of Phase 1 are broken down into its components below:

#### How to scrape a website:

The R package ‘rvest’ is one of the leading web scrapping packages for R. It is being utilized for its easy operation and large community support. Using it is as simple as installing, loading and running the package on a supplied URL. This package will read data from the whole page that has been specified and save it as htm. From here the data can be manipulated with other packages from R.

#### How to identify the correct part of website to scrape:

Meddy’s website was easy to scrape, but in order to get the names of the doctors and their ratings, the correct area on the website needed to be identified. For the purposes of this project, the focus was on the names of the doctors, their ratings and their comments by patients. A web browser add-on called ‘SelectorGadget’ came in handy here. It is able to identify the exact html tag within the CSS of the website that contains the relevant data. This tag can be input into the ‘rvest’ package to only take data from these tags. For example the tag that stored comments was under ‘.comment-body’.

#### How to store the data that has been scraped:

To start off, only the data from one doctor was obtained. The comments were stored in a CSV file. An appropriate data table was created to store the comments, and then saved a CSV file. Meddy gives its users a choice of 3 emoticons when submitting a review. The challenge was to capture the image from these three choices and derive sentiments for a doctor through them. An if condition list when logging data from the doctors page helps identify what type of image was stored using its name helped identify the correct emoticon that was selected. The number of ‘’satisfied’, ‘unsatisfied’ and ‘neutral’ patients were combined to form a pie chart that displays the percentage of the comments that fall into each of these categories.

#### How to automate the process of multiple pages:

* Comments about doctors
  + Each doctor’s name is used to create the name of a CSV file.
  + Each comment and name of commenter is stored in the file using an appropriate table format.
* Ratings of doctors
  + The emoticon face of the doctor is captured using the name of the image associated with it. They are projected as a pie chart to display perchance of positive, negative and neutral comments.
* Multiple Pages automation
  + This is a multi part process that starts with an URL for a particular field of doctors. The same technique used to gather comments for a single doctor is used to gather all the doctors’ names, which is then put into a list.
  + The loop iterates through the list of doctors, changing the name to the appropriate format (dashes instead of spaces and meddy.com at the start) and adding it to a new URL list. This list is saved with the cleaned doctor’s names and their corresponding URL as a CSV file.
  + The code runs through the whole process of capturing doctors comments and overall review emoticons for each page in the URL list.

## Phase 2:

Analyze the data gathered from the Meddy website (e.g. sentiment analysis and more).

#### Word cloud

A world cloud is a quick and easy way to visualize the captured data and see patterns and trends in it. It helps capture the over all sentiments that patients may feel towards a certain doctor or even how patients on Meddy (representing Qatar’s healthcare users) feel about the state of healthcare in Qatar.

#### Sentiment analysis

The comments on the website are associated with an emoticon. But there are stronger tools available that utilize machine learning to gauge whether or not the comment made is in a positive or negative context. These can be used when the data is not as well organized as it is in Meddy and can be an excellent way to understand the context of lots of textual data in very little time.

#### Future Potential

Given that we are already moving towards a world where all information is electronically recorded, applications of such a program can lead to better understanding of all this stored data. Patients won’t even need to log on to a website like Meddy, find the right doctor and then write the review for them. Just by using one’s voice, the thoughts about the experience can be recorded. With advancements in natural language processing, these comments can be converted in R compatible text and then analyzed almost instantly. The patient may not even need to mention the name of the doctor as the name and location can be taken from the calendar appointment that the patient has saved.

#### Adaptability

R is a fairly simple statistical language to use and get started with. But it also has great depth that can be utilized to organize and manipulate data to derive meaning out of it. This project and can be adapted to any sort of website that has data structures that follow a particular CSS format for gathering. The automated multiple page data gathering can also be adapted with an appropriate pattern to get to the next page. It does not necessarily have to be related to healthcare as long is it has data to be mined and analyzed, the R script can adapt to it.

## R code Appendix

#### Getting Doctors Names from a given Field:

#Loading the rvest package

install.packages('rvest')

library('rvest')

setwd("~/Desktop")

#Specifying the url for desired website to be scrapped

url <- 'https://www.meddy.co/doctors/general+practitioner/'

#Reading the HTML code from the website

webpage <- read\_html(url)

doctorList <- html\_nodes(webpage,'.listings-card-header')

names <- html\_text(doctorList)

write.csv(names, file="docname.csv")

cleannames = gsub("\nDr. ", "", names)

cleannames = gsub("\n", "", cleannames)

cleannames2 = gsub(" ", "-", cleannames)

nameTable <- data.frame(matrix(nrow=0,ncol=2))

colnames(nameTable)[1]="Doctor Name"

colnames(nameTable)[2]="Profile Link"

for(j in 1:length(cleannames2)){

nameTable[j,1] <- cleannames[j]

nameTable[j,2] <- paste("https://www.meddy.co/",cleannames2[j],sep="")}

write.csv(nameTable, file="link.csv")

#### Single Doctors Sentiments (Pie Chart) and Comments:

#Loading the rvest package

library('rvest')

setwd("~/Desktop")

url <- 'https://www.meddy.co/mohammad-asha/'

# print(url)

webpage <- read\_html(url)

#Sentiment for each doctor

sentimentPos <- length(html\_nodes(webpage,'.meddy-satisfied'))

sentimentNeg <- length(html\_nodes(webpage,'.meddy-unsatisfied'))

sentimentNeu <- length(html\_nodes(webpage,'.meddy-neutral'))

sentimentTotal <- length(html\_nodes(webpage,'.status-icon'))

# sentimentNeg

# sentimentNeu

# sentimentPos

# sentimentTotal

# Pie Chart with Percentages

slices <- c(sentimentNeu, sentimentNeg, sentimentPos)

lbls <- c("Neutral", "Negative", "Positive")

pct <- round(slices/sentimentTotal\*100)

lbls <- paste(lbls, pct) # add percents to labels

lbls <- paste(lbls,"%",sep="") # ad % to labels

pie(slices,labels = lbls, col=rainbow(length(lbls)),

main="Pie Chart of Sentiments")

#Using CSS selectors to scrap the names section

names\_data\_html <- html\_nodes(webpage,'.comment-name')

#Converting the ranking data to text

name\_data <- html\_text(names\_data\_html)

#Using CSS selectors to scrap the comments section

comment\_data\_html <- html\_nodes(webpage,'.comment-body')

#Converting the ranking data to text

comment\_data <- html\_text(comment\_data\_html)

commentTable <- data.frame(matrix(nrow=0,ncol=2))

colnames(commentTable)[1]="Patient Name"

colnames(commentTable)[2]="Patient Comment"

for(k in 1:length(name\_data)){

commentTable[k,1] <- name\_data[k]

commentTable[k,2] <- comment\_data[k]

}

commentTable <- apply(commentTable,2,as.character)

write.csv(commentTable, file="AshaMuhammad.csv")

#### Multiple Doctors Comments and Sentiments:

#Loading the rvest package

library('rvest')

setwd("~/Desktop")

docUrl <- read.csv(file="link.csv", header=TRUE, stringsAsFactors=FALSE)

docUrl

for(j in 1:6){ #currently limited to 6 because of Meddy’s allowance

url <- docUrl[j,3]

print(url)

webpage <- read\_html(url)

#Sentiment for each doctor

sentimentPos <- length(html\_nodes(webpage,'.meddy-satisfied'))

sentimentNeg <- length(html\_nodes(webpage,'.meddy-unsatisfied'))

sentimentNeu <- length(html\_nodes(webpage,'.meddy-neutral'))

sentimentTotal <- length(html\_nodes(webpage,'.status-icon'))

# sentimentNeg

# sentimentNeu

# sentimentPos

# sentimentTotal

# Pie Chart with Percentages

slices <- c(sentimentNeu, sentimentNeg, sentimentPos)

lbls <- c("Neutral", "Negative", "Positive")

pct <- round(slices/sentimentTotal\*100)

lbls <- paste(lbls, pct) # add percents to labels

lbls <- paste(lbls,"%",sep="") # ad % to labels

pie(slices,labels = lbls, col=rainbow(length(lbls)),

main="Pie Chart of Sentiments")

#Using CSS selectors to scrap the names section

names\_data\_html <- html\_nodes(webpage,'.comment-name')

#Converting the ranking data to text

name\_data <- html\_text(names\_data\_html)

#Using CSS selectors to scrap the comments section

comment\_data\_html <- html\_nodes(webpage,'.comment-body')

#Converting the ranking data to text

comment\_data <- html\_text(comment\_data\_html)

commentTable <- data.frame(matrix(nrow=0,ncol=2))

colnames(commentTable)[1]="Patient Name"

colnames(commentTable)[2]="Patient Comment"

for(k in 1:length(name\_data)-1){

commentTable[k,1] <- name\_data[k]

commentTable[k,2] <- comment\_data[k]

}

commentTable <- apply(commentTable,2,as.character)

doctor <- docUrl[j,2]

doctorFile <- paste(doctor, ".csv")

write.csv(commentTable, file=doctorFile)

}

#### Sentiment Analysis Individual Doctor (Word cloud and Sentiment Graph):

#adapted from https://www.springboard.com/blog/text-mining-in-r/

# Install

install.packages("tm") # for text mining

install.packages("SnowballC") # for text stemming

install.packages("wordcloud") # word-cloud generator

install.packages("RColorBrewer") # color palettes

install.packages("syuzhet")

install.packages("ggplot2")

# Load

library("tm")

library("SnowballC")

library("wordcloud")

library("RColorBrewer")

library('syuzhet')

library('rvest')

library("ggplot2")

#Specifying the url for desired website to be scrapped

url <- 'https://www.meddy.co/mohammad-asha/'

#Reading the HTML code from the website

webpage <- read\_html(url)

#Using CSS selectors to scrap the names section

comments\_data\_html <- html\_nodes(webpage,'.comment-body')

#Converting the ranking data to text

textdata <- html\_text(comments\_data\_html)

# Load the data as a corpus

docs <- Corpus(VectorSource(textdata))

inspect(docs)

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

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

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

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

# 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)

# Text stemming (reduces words to their root form)

docs <- tm\_map(docs, stemDocument)

dtm <- TermDocumentMatrix(docs)

m <- as.matrix(dtm)

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

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

head(d, 10)

par(bg="grey30")

png(file="WordCloud.png",width=1000,height=700, bg="grey30")

wordcloud(d$word, d$freq, col=terrain.colors(length(d$word), alpha=0.9), random.order=FALSE, rot.per=0.3 )

title(main = "Meddy's Most Used Terms to Describe Doctors", font.main = 1, col.main = "cornsilk3", cex.main = 1.5)

d<-get\_nrc\_sentiment(textdata)

td<-data.frame(t(d))

#Transformation and cleaning

names(td)[1] <- "count"

td\_new <- cbind("sentiment" = rownames(td), td)

rownames(td\_new) <- NULL

td\_new2<-td\_new[1:8,]

dev.off()

qplot(sentiment, data=td\_new2, weight=count, geom="bar",fill=sentiment)+ggtitle("Patient sentiments")

#### Sentiment Analysis for multiple csv doctor files saved in directory (Word cloud and Sentiment Graph):

install.packages("tidyverse")

setwd("~/Desktop")

# Load

library("tidyverse")

tbl <-

list.files(pattern="\*.csv") %>%

map\_df(~read\_csv(., col\_types = cols(.default = "c")))

#tbl

# Install

install.packages("tm") # for text mining

install.packages("SnowballC") # for text stemming

install.packages("wordcloud") # word-cloud generator

install.packages("RColorBrewer") # color palettes

install.packages("syuzhet")

install.packages("ggplot2")

# Load

library("tm")

library("SnowballC")

library("wordcloud")

library("RColorBrewer")

library('syuzhet')

library('rvest')

library("ggplot2")

textdata <- subset(tbl, select=c('Patient Comment'))

# Load the data as a corpus

docs <- Corpus(VectorSource(textdata))

inspect(docs)

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

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

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

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

# 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)

# Text stemming (reduces words to their root form)

docs <- tm\_map(docs, stemDocument)

dtm <- TermDocumentMatrix(docs)

m <- as.matrix(dtm)

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

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

head(d, 10)

par(bg="grey30")

png(file="WordCloud.png",width=1000,height=700, bg="grey30")

wordcloud(d$word, d$freq, col=terrain.colors(length(d$word), alpha=0.9), random.order=FALSE, rot.per=0.3 )

title(main = "Meddy's Most Used Terms to Describe Doctors", font.main = 1, col.main = "cornsilk3", cex.main = 1.5)

d<-get\_nrc\_sentiment(textdata)

td<-data.frame(t(d))

#Transformation and cleaning

names(td)[1] <- "count"

td\_new <- cbind("sentiment" = rownames(td), td)

rownames(td\_new) <- NULL

td\_new2<-td\_new[1:8,]

dev.off()

qplot(sentiment, data=td\_new2, weight=count, geom="bar",fill=sentiment)+ggtitle("Patient sentiments")