Group Assignment – Shiny App

**Are you normal?**

TA and NLP – DAT-5317

A picture containing drawing

Description automatically generated

**Professor**  
Thomas Kurnicki

**Submitted by**

Yağız Yaver Çiçek;

Alexander Gruber;

Karen Larios;

Dias Mussayev;

Elise Neumann

**Date**  
02/15/2021

Table of Contents

[## Loading all libraries and R data 2](#_Toc64307351)

[## Importing all files and consolidating into one dataframe 3](#_Toc64307352)

[## Calculations and analysis 4](#_Toc64307353)

[# Home 4](#_Toc64307354)

[# tf\_idf analysis 5](#_Toc64307355)

[# Pair-wide correlation 7](#_Toc64307356)

[# Sentiment 8](#_Toc64307357)

[# Naive Bayes 8](#_Toc64307358)

[## SHINY 10](#_Toc64307359)

[# UI 10](#_Toc64307360)

[# Server 14](#_Toc64307361)

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

# ## Loading all libraries and R data

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

#install.packages("shiny")

#install.packages("formattable")

#install.packages("DT")

# libraries

library(textreadr)

library(dplyr)

library(tidytext)

library(tidyverse)

library(stringr)

library(igraph)

library(ggplot2)

library(ggraph)

library(scales)

library(tm)

library(shinydashboard)

library(plotly)

library(widyr)

library(RTextTools)

library(e1071)

library(caret)

library(quanteda)

library(quanteda.textmodels)

library(RColorBrewer)

library(formattable)

library(DT)

# R data

data(stop\_words)

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

# ## Importing all files and consolidating into one dataframe

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

## for yes responses

# Importing all .txt files from one directory

setwd("/Users/anneliseneumann/Desktop/Text Analytics and NLP/Assignment 2/yes")

nm <- list.files(path="/Users/anneliseneumann/Desktop/Text Analytics and NLP/Assignment 2/yes")

# using read document to import the data:

my\_txt\_text <- do.call(rbind, lapply(nm, function(x) paste(read\_document(file=x))))

# restructuring data as a data frame

mydf\_yes1 <- data\_frame(response=c(seq(1,15)), question = 1, text = my\_txt\_text[,1], normal="yes")

mydf\_yes2 <- data\_frame(response=c(seq(1,15)), question = 2, text = my\_txt\_text[,2], normal="yes")

mydf\_yes3 <- data\_frame(response=c(seq(1,15)), question = 3, text = my\_txt\_text[,3], normal="yes")

mydf\_yes4 <- data\_frame(response=c(seq(1,15)), question = 4, text = my\_txt\_text[,4], normal="yes")

mydf\_yes5 <- data\_frame(response=c(seq(1,15)), question = 5, text = my\_txt\_text[,5], normal="yes")

mydf\_yes6 <- data\_frame(response=c(seq(1,15)), question = 6, text = my\_txt\_text[,6], normal="yes")

mydf\_yes <- rbind(mydf\_yes1, mydf\_yes2, mydf\_yes3, mydf\_yes4, mydf\_yes5, mydf\_yes6)

## for no responses

# Importing all .txt files from one directory

setwd("/Users/anneliseneumann/Desktop/Text Analytics and NLP/Assignment 2/no")

nm <- list.files(path="/Users/anneliseneumann/Desktop/Text Analytics and NLP/Assignment 2/no")

# using read document to import the data:

my\_txt\_text <- do.call(rbind, lapply(nm, function(x) paste(read\_document(file=x))))

# restructuring data as a data frame

mydf\_no1 <- data\_frame(response=c(seq(16,32)), question = 1, text = my\_txt\_text[,1], normal="no")

mydf\_no2 <- data\_frame(response=c(seq(16,32)), question = 2, text = my\_txt\_text[,2], normal="no")

mydf\_no3 <- data\_frame(response=c(seq(16,32)), question = 3, text = my\_txt\_text[,3], normal="no")

mydf\_no4 <- data\_frame(response=c(seq(16,32)), question = 4, text = my\_txt\_text[,4], normal="no")

mydf\_no5 <- data\_frame(response=c(seq(16,32)), question = 5, text = my\_txt\_text[,5], normal="no")

mydf\_no6 <- data\_frame(response=c(seq(16,32)), question = 6, text = my\_txt\_text[,6], normal="no")

mydf\_no <- rbind(mydf\_no1, mydf\_no2, mydf\_no3, mydf\_no4, mydf\_no5, mydf\_no6)

## combining the two together

mydf <- rbind(mydf\_yes, mydf\_no)

# creating a tidy version of the df

tidy\_df <- mydf %>%

unnest\_tokens(word,text) %>%

anti\_join(stop\_words) %>%

count(normal, word, sort=TRUE) %>%

mutate(word=reorder(word, n)) %>%

group\_by(normal)

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

# ## Calculations and analysis

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

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

## # Home

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

context <- c("In order to better understand our customers, we conducted interviews with 5 questions regarding different aspects of their life and personalities, and a final binary question regarding their ‘normality’. <br>

Due to covid circumstances, the interviews were conducted over Zoom, and otter.ai was used to transcribe the reponses. <br>

We conducted over 30 interviews, and this dashboard will go through some key analysis we have conducted on our data to generate insights.")

questions <- c("1. If you were arrested with no explanation, what would your friends and family assume you had done? <br>

2. What is your spirit animal and why? <br>

3. Who is your craziest family member and why? <br>

4. What is the dumbest way you hurt yourself? <br>

5. What is your most unusual talent? <br>

6. Would your friends describe you as normal? (Y/N) <br> ")

limitations <- c("<ul><li> Low number of responses collected limits the generalization of insights </li></ul>",

"<ul><li> Quality of transcription software distorted responses and may mask key words")

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

## # tf\_idf analysis

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

#Part 1: Normal-wise Analysis

tfidf\_total\_df <- mydf %>%

unnest\_tokens(word,text) %>%

anti\_join(stop\_words) %>%

count(normal, word, sort=TRUE) %>%

mutate(word=reorder(word, n))

tfidf\_total\_df <- tfidf\_total\_df %>% bind\_tf\_idf(word, normal, n) %>%

arrange(desc(tf\_idf))

tfidf\_total\_df %>%

filter(n>=2) %>%

filter(tf\_idf > 0.002) %>%

mutate(word=factor(word, levels = rev(unique(word)))) %>%

ungroup() %>%

ggplot(aes(word, tf\_idf, fill=normal))+

geom\_col(show.legend=FALSE)+

labs(x=NULL, y="tf-idf")+

facet\_wrap(~normal, ncol=2, scales="free")+

coord\_flip()

#Part 2: Question-wise Analysis

tfidf\_question\_df <- mydf %>%

unnest\_tokens(word,text) %>%

anti\_join(stop\_words) %>%

count(question, word, sort=TRUE) %>%

mutate(word=reorder(word, n))

tfidf\_question\_df <- tfidf\_question\_df %>% bind\_tf\_idf(word, question, n) %>%

arrange(desc(tf\_idf))

insight\_tfidf <- c("<ul><li>Top words include 'drinking', 'driving', 'drugs': Young population tendencies</li><li>

High frequency of word 'innocent': People tend to believe they're able to convince their innocence</li></ul>",

"<ul><li>Two types of words: animals and characteristics</li><li>

Attribute words like 'calm', 'sleep', 'chill' are possibly related to panda</li><li>

Other attributes like 'aggressive' or 'confident' are possibly related to 'tiger' or 'lion'</li></ul>",

"<ul><li>Male family members are ranked higher up the list</li><li>

Female family members are seen as more orderly, while males are seen as the jesters</li></ul>",

"<ul><li>Most words include usual and caricature ways to hurt oneself</li><li>

People tend to hurt particular limbs: hands, feet, ankles</li></ul>",

"<ul><li>Social skills are seen as unique talents: arguing, speaking, listening</li><li>

Further data is needed to fortify arguments</li></ul>")

#Part 3: Persona-wise Analysis

tfidf\_persona\_df\_yes <- mydf %>%

unnest\_tokens(word,text) %>%

anti\_join(stop\_words) %>%

filter(normal == "yes") %>%

count(response, word, sort=TRUE) %>%

mutate(word=reorder(word, n))

tfidf\_persona\_df\_yes <- tfidf\_persona\_df\_yes %>% bind\_tf\_idf(word, response, n) %>%

arrange(desc(tf\_idf))

tfidf\_persona\_df\_yes %>%

filter(tf\_idf > 0.08) %>%

mutate(word=factor(word, levels = rev(unique(word)))) %>%

mutate(word = reorder(word, tf\_idf)) %>%

ungroup() %>%

ggplot(aes(word, tf\_idf, fill=response))+

ggtitle("Normal Persona Unique Words") +

geom\_col(show.legend=FALSE)+

labs(x=NULL, y="tf-idf")+

facet\_wrap(~response, ncol=2, scales="free")+

coord\_flip()

tfidf\_persona\_df\_no <- mydf %>%

unnest\_tokens(word,text) %>%

anti\_join(stop\_words) %>%

filter(normal == "no") %>%

count(response, word, sort=TRUE) %>%

mutate(word=reorder(word, n))

tfidf\_persona\_df\_no <- tfidf\_persona\_df\_no %>% bind\_tf\_idf(word, response, n) %>%

arrange(desc(tf\_idf))

tfidf\_persona\_df\_no %>%

filter(tf\_idf > 0.08) %>%

mutate(word=factor(word, levels = rev(unique(word)))) %>%

#top\_n(50) %>%

ungroup() %>%

ggplot(aes(word, tf\_idf, fill=response))+

ggtitle("Anormal Persona Unique Words") +

geom\_col(show.legend=FALSE)+

labs(x=NULL, y="tf-idf")+

facet\_wrap(~response, ncol=2, scales="free")+

coord\_flip()

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

## # Pair-wide correlation

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

#Part 1: Pair-wise Correlation Analysis

# creating a new tidy version of the df without count

new\_tidy\_df <- mydf %>%

unnest\_tokens(word,text) %>%

anti\_join(stop\_words) %>%

group\_by(normal)

# insights pwcor

insight\_pwcor <- c("<ul><li>'tiger' is strongly correlated with 'argue' and 'wild', which relates to answers for people's spirit animal and their character.</li></ul>",

"<ul><li>'police' answers show high correlation with 'fun', 'mistake', and 'dying', so reasons for being arrested are often by mistake or for doing something fun.</li></ul>",

"<ul><li>'party' got used as part of an answer for several questions, as the mix of results shows.</li></ul>")

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

## # Sentiment

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

# insights sentiment

fleeb\_sent <- c("<ul><li>Abnormal responders use extreme words like 'super','amazing','craziest'.</li></ul>",

"<ul><li>'No' responders tend to have more negative words</li></ul>",

"<ul><li>Negative value for words “arrested” and “hurt” are higher among 'No' responders;

however, it is caused because there are more 'no' responses overall.</li></ul>",

"<ul><li>'Yes' responders tend to use more formal dialect.</li></ul>")

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

## # Naive Bayes

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

#creating a dfm

dfm\_df <- mydf %>%

filter(!question == 6) %>%

unnest\_tokens(word,text) %>%

anti\_join(stop\_words) %>%

count(response,normal, word, sort=TRUE) %>%

cast\_dfm(response,word,n)

#let's split the docs into training and testing data

d <- c(6,7,12,19,23,27)

dfm.train<-dfm\_df[-d,]

dfm.test<-dfm\_df[d,]

#building the Naive Bayes model:

isnaivebayesnormal <- textmodel\_nb(dfm.train, c(1,1,1,1,1,1,1,1,1,1,1,1,

0,0,0,0,0,0,0,0,0,0,0,0,0,0))

dfmat\_matched <- dfm\_match(dfm.test, features = featnames(dfm.train))

#dfmat\_matched: the ones that differ from the previous one are yeah, hurt, arrested, people, time

# predicting the testing data

offthewall\_predictions <- as.numeric(predict(isnaivebayesnormal, dfm.test))

offthewall\_predictions <- gsub('2',"Normal",offthewall\_predictions)

offthewall\_predictions <- gsub('1',"Abnormal",offthewall\_predictions)

# table with top predictive scores

words <- data.frame(isnaivebayesnormal$param)

t\_words <- t(words)

t\_words <- data.frame(t\_words)

colnames(t\_words) <- rownames(words)

rownames(t\_words) <- colnames(words)

t\_words$diff <- abs(t\_words$'1' - t\_words$'0')

k <- t\_words %>% arrange(desc(diff)) %>%

top\_n(10,diff) %>%

round(digits=3)

# insights naive bayes

insight\_nb <- c("<ul><li>According to our model we have 99.4% of sparsity which means we have 99% of unique words in our data set.</li></ul>",

"<ul><li>The model predicted that the use of words like yeah and guess in the text were the ones to contribute to a yes answer, we can say that these have no significant meaning into predicting if a person is considered normal or not.</li></ul>",

"<ul><li>The model predicted that the use of words like friends, craziest and animal in the text were the ones to contribute to a no answer, the same way these words have no significant meaning into predicting if is considered normal or not.</li></ul>",

"<ul><li>Our model is not making the correct predictions, but why is our model not good?

We have all sort of limitations, only 32 survey answers with a small number of tokens per question.</li></ul>",

"<ul><li>We have a high percentage of sparsity we have a lot of unique words and that could create a problem to the model, not having matches between the text could make the model mispredict.</li></ul>",

"<ul><li>Analyzing the answers individually, they do not provide a lot of insight to the question asked.</li></ul>")

#Format tools

improvement\_formatter <-

formatter("span",

style = x ~ style(

font.weight = "bold",

color = ifelse(x == "Correct", "green", "red")))

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

# ## SHINY

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

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

## # UI

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

library(shinydashboard)

ui <- dashboardPage(

# customizing the look of the app

skin = "purple",

# defining title

dashboardHeader(title = "Are you normal?"),

# list of tabs in sidebar menu

dashboardSidebar(

sidebarMenu(

# landing page

menuItem("Study Overview",

tabName = "home"),

# sentiment analysis

menuItem("Sentiment Analysis",

tabName = "sentiment"),

# tf\_idf

menuItem("Relative term frequency",

tabName = "tf\_idf"),

# pwcor

menuItem("Pair-wise correlation",

tabName = "pw\_cor\_tab"),

# ngrams

menuItem("N-Grams",

tabName = "ngrams"),

# naive Bayes

menuItem("Prediction",

tabName = "my\_prediction")

)

),

# filling body of the dashboard

dashboardBody(

tabItems(

# home

tabItem(tabName = "home",

fluidRow(

box(

width = 12,

title = "Context",

uiOutput("home\_context")

),

box(

width = 6,

title = "Questions",

uiOutput("home\_questions")

),

box(

width = 6,

title = "Limitations",

uiOutput("home\_limitations")

)

)

),

# sentiment analysis

tabItem(tabName = "sentiment",

fluidRow(

box(

width = 6,

sliderInput("plumbus","Select Minimum Word Frequency",

min = 0, max = 15, value = 3)

),

box(

status="success",

width = 6,

title = "INSIGHTS",

uiOutput("fleeb\_myList")

),

box(

width = 12,

title = "Contribution to Sentiment",

plotly::plotlyOutput("grumbo\_afinn")

)

)

),

# tf\_idf

tabItem(tabName = "tf\_idf",

fluidRow(

box(

width = 6,

selectInput("tf\_idf\_select","Select Question",selected = 1,choices = c(1,2,3,4,5)),

sliderInput("tf\_slider","Select tf\_idf Range",value= 0.005, min= 0.005, max = 0.08)

),

box(

status="success",

width = 6,

title = "INSIGHTS",

uiOutput("myList")

),

box(

width = 12,

title = "Proportionate Term Frequency for Survey Questions",

plotly::plotlyOutput("tf\_graph"))

)

),

# pw\_cor

tabItem(tabName = "pw\_cor\_tab",

fluidRow(

box(

width = 6,

selectInput("pw\_cor\_select","Select word for analysis",

selected = c("tiger","police"),

choices = c("tiger", "police", "party", "drinking", "super", "panda"),

multiple = T)

),

box(

status="success",

width = 6,

title = "INSIGHTS",

uiOutput("pwcor\_myList")

),

box(

width = 12,

title = "Pair-wise word correlation",

plotly::plotlyOutput("pwcor"))

)

),

# ngrams

tabItem(tabName = "ngrams",

fluidRow(

box(

width = 6,

sliderInput("ngram\_n", "How many combinations to see:",

min = 1, max = 10, value = 4),

selectInput("ngram\_normal", "Which business outcome to focus on:",

choices = c("yes", "no"), selected = c("yes", "no"), multiple = T)

),

box(

status="success",

width = 6,

title = "INSIGHTS",

"The only bigrams re-occuring at least twice are key phrases used to formulate the questions. As such, this analysis does not help identify one group of customers from another"

),

box(

width = 12,

title = "Most Frequent Bigrams",

tableOutput("ngram\_table")

)

)

),

#Naive Bayes

tabItem(tabName = "my\_prediction",

fluidRow(

box(

width = 6,

title = "Model Prediction",

tableOutput("model")

),

box(

width = 6,

title = "Top Chi Scores",

numericInput("k\_input", "Select number of rows to view", max=10, min=1, value=3),

tableOutput("top\_k")

),

box(

status="success",

width = 12,

title = "INSIGHTS",

uiOutput("nb\_myList")

)

)

)

)

)

)

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

## # Server

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

server <- function(input, output, session) {

# landing page

output$home\_context <- renderUI(HTML(paste(context)))

output$home\_questions <- renderUI(HTML(paste(questions)))

output$home\_limitations <- renderUI(HTML(paste(limitations)))

# ngrams

output$ngram\_table <- renderTable ({

mydf %>%

filter(normal == input$ngram\_normal) %>%

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

filter(!is.na(bigram)) %>%

separate(bigram, c("word1", "word2"), sep = " ") %>% # to split into col

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

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

mutate(bigram = paste(word1, word2, sep = " ")) %>%

count(normal, bigram, sort = TRUE) %>%

top\_n(input$ngram\_n, n)

})

#tf\_idf outputs

output$insight\_text <- renderText({paste(insight\_tfidf[as.numeric(input$tf\_idf\_select)])})

output$myList <- renderUI(HTML(paste(insight\_tfidf[as.numeric(input$tf\_idf\_select)])))

output$tf\_graph <- plotly::renderPlotly({

tfidf\_question\_df %>%

filter(question == input$tf\_idf\_select) %>%

filter(n>=2) %>%

filter(tf\_idf > input$tf\_slider) %>%

mutate(word=factor(word, levels = rev(unique(word)))) %>%

ungroup() %>%

ggplot(aes(word, tf\_idf))+

geom\_col(show.legend=FALSE, fill= "red3")+

theme\_minimal() +

#xlim(0,0.5) +

ylab("Frequency") +

xlab("")+

#facet\_wrap(~question, ncol=2, scales="free")+

coord\_flip()

})

# Pair-wise correlation

output$pwcor <- plotly::renderPlotly({

new\_tidy\_df %>%

group\_by(word) %>%

filter(n() >= 3) %>% #n() does the count

pairwise\_cor(word, response, sort=TRUE) %>%

filter(item1 %in% input$pw\_cor\_select) %>%

group\_by(item1) %>%

top\_n(6) %>%

ungroup() %>%

mutate(item2 = reorder(item2, correlation)) %>%

ggplot(aes(item2, correlation)) +

geom\_bar(stat = "identity", fill = "green4") +

ylab("Correlation") +

xlab(" ") +

facet\_wrap(~ item1, scales = "free\_y") +

coord\_flip()

})

# Pair-wise correlation Part 2

#output$pwcor\_insight\_text <- renderText({paste(insight\_pwcor)})

output$pwcor\_myList <- renderUI(HTML(paste(insight\_pwcor)))

output$pwcorplot <- renderPlot({

new\_tidy\_df %>%

group\_by(word) %>%

filter(n() >= 3) %>% #n() does the count

filter(correlation >.5) %>%

graph\_from\_data\_frame() %>%

ggraph(layout = "fr")+

geom\_edge\_link(aes(edge\_alpha = correlation), show.legend=F)+

geom\_node\_point(color = "lightblue", size=6)+

geom\_node\_text(aes(label=name), repel=T)+

theme\_void()

})

# Sentiment ggplot

#output$fleeb\_insight\_text <- renderText({paste(fleeb\_sent)})

output$fleeb\_myList <- renderUI(HTML(paste(fleeb\_sent)))

output$grumbo\_afinn <- plotly::renderPlotly({

tidy\_df %>%

inner\_join(get\_sentiments("afinn")) %>%

group\_by(normal) %>%

mutate(n = n\*value) %>%

filter(abs(n) > input$plumbus) %>%

mutate(word = fct\_reorder(word, n)) %>%

arrange(desc(n)) %>%

ggplot(aes(word, n, fill = ifelse(n<0, "green", "red"))) +

geom\_bar(stat = "identity") +

facet\_wrap(~normal, scales = "free\_y") +

ylab("Contribution to Sentiment") +

xlab(" ") +

coord\_flip() +

theme(legend.position = "none")

})

#Naive Bayes

#output$nb\_insight\_text <- renderText({paste(insight\_nb)})

output$nb\_myList <- renderUI(HTML(paste(insight\_nb)))

output$model <- renderTable({formattable(

data.frame(Response = c("6","7","12","19","23","27"),

Prediction = offthewall\_predictions,

Actual= c('Normal','Normal','Normal','Abnormal','Abnormal','Abnormal'),

Performance= c('Incorrect','Correct','Incorrect','Correct','Correct','Incorrect')),

align = c("l","c","c","r"), list('Performance' = improvement\_formatter))

})

output$top\_k <- renderTable({

k[1:input$k\_input,]

}, rownames = TRUE, digits=3)

}