Group Assignment – Shiny App

Are you normal?

TA and NLP – DAT-5317

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

```
########################################################################
#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)
}
```