Which Modern-Day Music Artist is William Shakespeare Most Similar to

Angel Sierra, Dominique McDonald, Mariana Gonzalez Castro,
Danny Ying, and Carina Kalaydjian

June 7th, 2022

Abstract

abstract text goes here here here here here here here here!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

1 Introduction

here we cite [4] our glorius leader [7] yeah yeah cool cool [10] and this too [9] this as well [14] woohooo!!! [13] and [6] and [8] and [1] sdljdj [3] , [12] and [11], [2], [5]

William Shakespeare is a world-renowned poet from 16th-century England. In his lifetime of 52 years, he wrote 37 plays, 154 sonnets, and many poems[1]. William Shakespeare has recognition for being one of the most revolutionary Literarry Artists to this date[1]. The Oxford Dictionary of Quotations states that William Shakespeare wrote close to one tenth of the most quoted lines ever written or spoken in English[1]. Shakespeare is also the second most quoted English writer after the writers of the bible[1]. Through his work, Shakespear introduced almost 3,000 words to the English language, these words can still be found in the Oxford English Dictionary today[1]! Shakespeare has an incredible ability to deeply express emotions through his work. Reders have been inspired and moved by his work for centuries. Shakespeare changed the world through his literary works. That is why William Shakespear is the best candidate to analyze and see how his Literary work compares to modern-day music artists song lyrics.

For this project we will be analzing Shakespeare's words and usage of emotion from his 154 sonnets from the Gutenberg Project[14]. We will then compare these words and emotions to another dataset of 45 different modern-day music artists and their song lyrics[13]. Our interest is seeing which music artists song lyrics are the most similar to Shakespeare's sonnets in the aspect of two categories, similarities words used and similarities in emotions expressed. Then we will see which modern-day music artist is the most similar to Willam Shakspare.

In order to answer this question, we are going to use various methods of Text Analysis, analysis of unorganized text data[9]. The different forms of Text Analysis that we will be using for this project are Sentiment Analysis and Keyword Extraction. Sentiment Analysis involves analyzing the text and being able to extract the different emotions the text is expressing[9]. Keyword Extraction analyzes the text and identifies which words appear most frequently in the text, these words are called Keywords[9]. The combination of these two methods will bring the result of which music artist is the most similar to shakespear in therms of types of words used and which music artist is the most similar to Shakespeare in terms of emotion expressed through words.

2 Methods

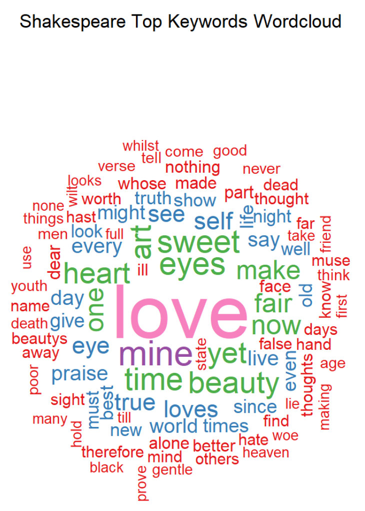
There is a growing demand for the application of Natural Language Processing to drive music knowledge discovery [11]. Within song lyrics there is a wealth of information that can be used to gain insights about the song and its listeners. The workflow used to conduct NLP is outlined in Oramas et. Al [12] as the following steps 1. Corpus creation (collection of separated documents), 2. Text mining (accessing desired info and eliminating the excess), 3. Information extraction (word frequency, collocation, word position, etc.), 4. Knowledge graph generation (a directed labeled graph in which we have associated domain specific meanings with nodes and edges [2]), 5. Sentiment Analysis (Identify feelings and emotions present in a text [9]).

This NLP pipeline can be applied to music to provide recommendations on what songs a listener might like based on what they already listen to. [11] This information is gathered by performing what's called a similarity search. Mahedero et al [11] conducts a similarity search for their project by first calculating a similarity measure. The similarity measure used is known as the Standard Cosine Distance (SCD). The calculations of the SCD are beyond the scope of this paper, but something to note is that the SCD relies on the Inverse Document Frequency as a way to measure the prevalence of words in a document and compare that to other documents.

As previously stated, we relied on Natural Language Processing methods to calculate the similarity between Shakespeare and current artists with the hope of identifying a single artist whose work has similar themes as Shakespeare's. Our pipeline was structured similarly to that proposed in [12] with the main difference being that we do not rely on knowledge graph generation to store any information or display findings. The methods that were most useful for this research were keyword extraction and sentiment analysis. Both methods of analysis provided us with crucial insights, so the details of each seem pertinent to share.

2.1 Keyword Extraction

Keyword extraction is largely aimed at identifying the most relevant words in different texts and utilizing those words to understand common theme or popular topics.[9] We used the tm library in R to perform keyword extraction[8] on the sonnets. This process involves first prepping the text by removing unwanted punctuation or numbers, eliminating stop words, changing everything to lower case. These transformations are necessary because when working with strings, you not only have to be precise, but you also must be exact. The cleaning process sometimes includes stemming the words, but we opted not to do that. Once the text is clean the idea is to create a table containing each word used in the text and the frequency with which it is used. To identify the top ten key words we sorted the words in the matrix by their frequency.



Once Shakespeare's keywords were identified, the next step was to identify the keywords for each of the 45 music artists. This process mirrored that of extracting keywords from the sonnets except we utilized an algorithm to automate the process for each artist. Once keywords were identified we converted the word frequencies for Shakespeare and all other artists into proportions, in order to standardize for comparison. To find the artists who were most like Shakespeare based on keywords, we checked for artist who had the highest number of matching keywords. Finally, then we ranked those by who used keywords in similar proportions to Shakespeare by calculating the Euclidian distance between the proportions of each of the matching keywords. This is similar to what Mahedero et. Al [11] does for their similarity search except we rely on word proportions and Euclidean distance, while their research utilizes Inverse Document Frequency and the Standard Cosine Distances.

2.2 Sentiment Analysis

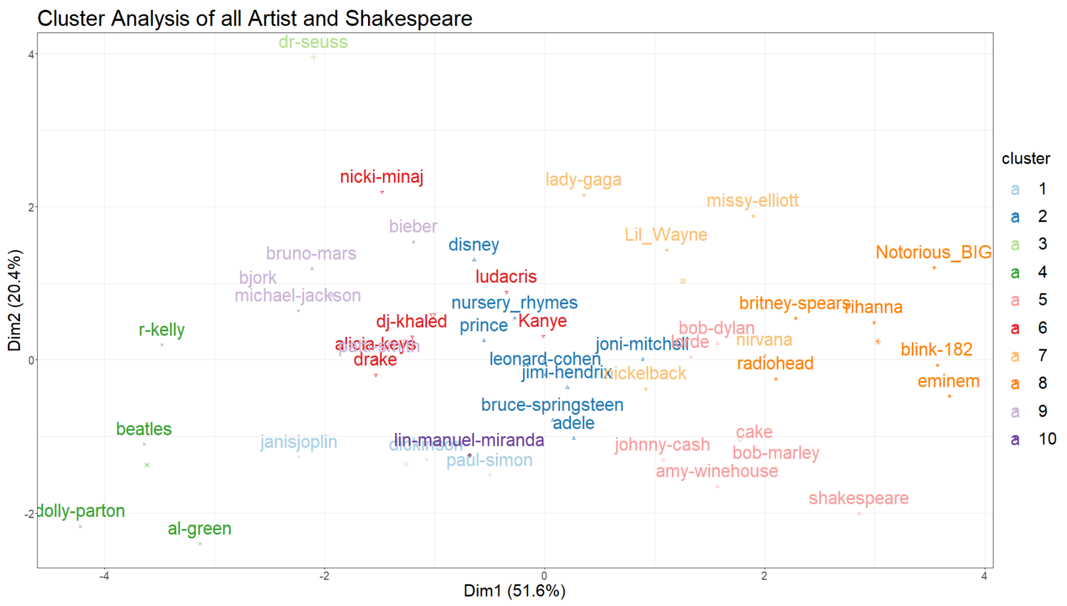
A similar sort of ranking was achieved from Sentiment Analysis, but the process has notable differences. The library syuzhet was used for the analysis. There is a wide variety of paths one could explore when performing sentiment analysis. Two that seemed viable for the purpose of this research were calculating an overall sentiment score and identifying the different emotions present in each text. [6] The latter option proved to be more fruitful, as the analysis is more detailed. This process of classification based on emotion is known as NRC sentiment analysis. NRC Sentiment Analysis uses the National Research Council (NRC) Word-Emotion Association Lexicon to classify words in a text into eight categories of emotions. [5] It is important to note that a word may be associated with more than one emotion. The eight emotions are anger, anticipation, disgust, fear, joy, sadness, surprise, trust (include simple equation). [5] The objective of NRC Sentiment Analysis is to calculate the frequency with which each emotion is conveyed. This is calculated by identifying the emotions associated with the unique words in a text and summing up all the instances of each emotion[6].

Once the NRC sentiments were calculated for both Shakespeare and the music artists, the frequencies were converted to proportions for accurate comparison. Using the proportions of each emotion, Euclidean distance was calculated for between each music artist and Shakespeare. The artists with the shortest distances from Shakespeare were considered the most similar to him, and therefore ranked higher in regard to comparison of the emotions conveyed in their works.

After comparing results from each analysis method, a final ranking was calculated. This overall ranking was calculated by summing up the rankings from both analysis methods. Because similarity to Shakespeare was assessed using Euclidean distance, lower rankings signify higher similarity to Shakespeare. This meant that the artists with the lowest overall ranking mirrored Shakespeare's work emotionally more than other artists.

2.3 K-Means Clustering

The NRC sentiments were utilized even further as predictors in K-Means Cluster Analysis. The objective of cluster analysis within the scope of this research is to utilize an unsupervised learning model [3] to assess commonalities between the work of each music artist and Shakespeare. K-Means clustering is an iterative method that categorizes each data point into one of k predefined groups, or clusters. The process is driven by two objectives. The first being maximizing the distance between clusters, so that they are distinct. The second is minimizing the data points within a cluster, so the clusters themselves are homogenous. [3] By employing K-means Clustering we were able to identify a group of artists that whose work most closely matches Shakespeare's.



2.4 Data Management

Before discussing the results of our various methods, it is important to address what made such an undertaking possible: data management. The size of this project necessitated multiple team members using multiple platforms. The data was stored in a relational database and then hosted on Amazon Web Services(AWS) service called Relational Database Services(RDS). Using RDS preserves the database from alterations, each memeber each member provided the read only user credentials to their database to enable the team to access data without making changes to the database itself. Once we had the data it the next steps were to process and analyze it.

Again, with so many contributors working to advance the project it was necessary to have an avenue for efficient and organized sharing of code. For this aspect of the project, GitHub was utilized and it allowed team members to work on the same files from different locations and share them as frequently as necessary. Along with our code we are also able to store and share important information that aided us in our research. The different information sharing structures employed allowed for efficient progression and ultimately valuable results.

3 Results

Artist	Sent. Euclidean Distance	Sentiment Rank
amy-winehouse	0.001227	1
eminem	0.001237	2
cake	0.001304	3
nirvana	0.001365	4
bob-dylan	0.001471	5
bob-marley	0.001492	6
johnny-cash	0.001589	7
nickelback	0.001827	8
britney-spears	0.002097	9
rihanna	0.002249	10

Table 1: Ranked Top 10 Most Similar Music Artist to Shakespeare Based on Sentiments

Artist	Word Rank	Sentiment Rank	Overall Rank
nickelback	2	8	10
amy-winehouse	6	4	10
cake	9	3	12
adele	1	21	22
joni-mitchell	11	11	22
leonard-cohen	8	17	25
paul-simon	10	22	32
blink-182	18	14	32
bob-marley	27	6	33
rihanna	25	10	35

Table 2: Ranked Top 10 Most Similar Music Artist to Shakespeare

4 Conclusions

Artist	Word Count	Frequency	Keyword	Word Rank
adele	3	626.00	love time heart	1
nickelback	3	11349.00	love time yet	2
bieber	2	648.00	love time	3
dolly-parton	2	685.00	love time	4
dj-khaled	2	928.00	mine time	5
amy-winehouse	2	3700.00	love time	6
bjork	2	5341.00	love yet	7
leonard-cohen	2	5972.00	love time	8
cake	2	7690.00	love time	9
paul-simon	2	8464.00	love time	10

Table 3: Ranked Top 10 Most Similar Music Artist to Shakespeare Based on Keywords

5 Appendix

```
options(width=90, xtable.comment = FALSE)

if(!dir.exists("_assets")) {
    dir.create("_assets")
}

###############################################
#list of libraries that we use
library(RMySQL)
library(DBI)

library(dplyr)
library(stringr)

library(tm)
library(syuzhet)
library(wordcloud)
library(RColorBrewer)

library(cluster)
library(factoextra)

library(ggplot2)
library(xtable)
########################################################
```

```
#access database to acquire datasets

drv <- dbDriver("MySQL")
xdbsock <- ""

#Marianas ROUSER Account
xdbuser <-'ROuser'
xpw     <- 'Nom59trpy03'
xdbname <- 'MyDB'
xdbhost <- 'database-1.cjv4ba2ytv1n.us-west-1.rds.amazonaws.com'
xdbport <- 3306

con <-
  dbConnect(
    drv,
    user=xdbuser,
    password=xpw,
    dbname=xdbname,
    host=xdbhost,
    port=xdbport,
    unix.sock=xdbsock
  )


dbListTables(con)

original_sonnets <- dbGetQuery(con, "SELECT * FROM OG")

artists_complete <- dbGetQuery(con, "SELECT * FROM Artists_noDuplicates")

dbDisconnect(con)

############################################
#               DATA CLEANUP
############################################
# Cleaning Shakespeare sonnets: OG SONNETS
############################################
#going to remove of the column of row numbers
og <- select(original_sonnets,-c(1))

og[1,1] #it appears that there are special characters

names(og)
```

```
og <- rename(og, "og_sonnets" = "rep.NA..154.")

og <-
  og %>%
  mutate_at("og_sonnets", str_replace_all, "", "\'")


#I manually removed the punctuation except for apostrophes
#creating an empty dataframe first
og_sonnets <- data.frame(matrix(ncol=1,nrow=154,
                                 dimnames=list(NULL, "Sonnets")))

n <- 154
for (i in 1:154) {
  og_sonnets[i,]<- gsub("[,.;:?!]", "", og[i,1])
  rbind(og_sonnets[i,])
}

og_sonnets[1,1] #checking to see if it worked

############################################
#   CLEAN UP OF THE ARTIST DATA
############################################
#removing the column of row numbers
artists_complete <- select(artists_complete,-c(1))

###this code will take all the non-lyrical strings contained in a set of
#parenthesis and compile it into a vector of lists
df  <- vector(mode = "list", length = 45)
names(df) <- artists_complete$Artist

for (i in 1:45) {

  df[[i]] <- gsub("[\\(\\)]", "", unlist(regmatches(artists_complete$Lyrics[i],
                                           gregexpr("\\(.*?\\)",
                                           artists_complete$Lyrics[i]))))
}

df[["adele"]] #look at the first element as an example


# had to remove "Verse 1, Chorus 2, Verse 3, and Chorus 3
#you'll find that those phrases are no longer there and what remains is just empty quotations
```

```
artists_complete <-
  artists_complete %>%
  mutate_at("Lyrics", str_remove_all, pattern = c("Verse 1|Chorus 2|Verse 3|Chorus 3|x2|vocal s


df2  <- vector(mode = "list", length = 45)
names(df2) <- artists_complete$Artist

for (i in 1:45) {

  df2[[i]] <- gsub("[\\(\\)]", "", unlist(regmatches(artists_complete$Lyrics[i],
                                            gregexpr("\\(.*?\\)",
                                            artists_complete$Lyrics[i]))))
}

df2[["adele"]]


#the last thing to do is to remove unwanted punctuation
#this removes everything inside brackets, brackets included
artists_complete$Lyrics <- gsub("\\[.+?\\]", "",
                                      artists_complete$Lyrics)

#now with parenthesis that includes lyrics being sung...
#this removes all the parenthesis BUT without removing the text inside
artists_complete$Lyrics <- gsub("\\s*\\([^\\)]+\\)","",
                                        artists_complete$Lyrics)

#This replaces everything that's not alphanumeric signs, space or apostrophe with an empty stri
artists_complete$Lyrics <- gsub("[^[:alnum:][:space:]']", "",
                                      artists_complete$Lyrics)
artists_complete
#clean up is all done!

##############################################################
#                     METHODOLOGY
##############################################################
#finalized cleaned datasets
og_sonnets
artists_complete

##############################################################
#                 Key Word Extraction
```

```
###############################################################
# Shakespeare Key Word Extraction
###########################################
# Further data cleaning before keyword extraction

#the pattern for the special character may vary from computer to computer
# here are other patterns incase is the one bellow wont work
# ????T, ???T,
og_sonnets <- og_sonnets %>%
  mutate_at("Sonnets", str_replace_all, "???T", "\'") #the pattern for the special character ma

# need to load vector of text objects as a corpus. VectorSource() interprets each element of a
x_text <- Corpus(VectorSource(og_sonnets$Sonnets))
x_text

#Replacing "/", "@" and "|" with space
toSpace <- content_transformer(function (y , pattern ) gsub(pattern, " ", y))
x_text <- tm_map(x_text, toSpace, "/")
x_text <- tm_map(x_text, toSpace, "@")
x_text <- tm_map(x_text, toSpace, "\\|")

# Remove numbers
x_text <- tm_map(x_text, removeNumbers)

# Remove English common stop words
x_text <- tm_map(x_text, removeWords, stopwords("english"))

# Removing custom Shakespearean Stop words
# specify your custom stop words as a character vector
x_text <- tm_map(x_text, removeWords, c("thi", "thee", "thou", "may", "still", "thus", "though
                                        "upon", "from", "dost", "shall", "thy"))
# Remove punctuation
x_text <- tm_map(x_text, removePunctuation)

# Eliminate extra white spaces
x_text <- tm_map(x_text, stripWhitespace)

x_text #NO DOCUMENTS DROPPED

##### Creating The document term matrix and extracting keywords
# The term matrix contains all the words in your "documents" and their frequencies
# Build a term-document matrix
x_text_dtm <- TermDocumentMatrix(x_text)
```

```
x_text_dtm

#number of total terms is the non sparse entries
mat_dtm <- as.matrix(x_text_dtm)

# Sort by decreasing value of frequency
dtm_v <- sort(rowSums(mat_dtm),decreasing=TRUE)
dtm_d <- data.frame(word = names(dtm_v),freq=dtm_v)
# Display the top 50 most frequent words
head(dtm_d, 50)

####### Generate word cloud of Shakespeare Keywords from Sonnets and Write it out as a PNG
png("_assets/Shakespeare_Keywords_WordCloud.png", width=1980, height=1080, pointsize=35)

layout(matrix(c(1, 2), nrow=2), heights=c(1, 4))
par(mar=rep(0, 4))
plot.new()

text(x=0.5, y=0.5, "Shakespeare Top Keywords Wordcloud")

set.seed(314)
wordcloud(words = dtm_d$word,
          freq = dtm_d$freq,
          min.freq = 5,
          max.words=100,
          random.order=FALSE,
          rot.per=0.20,
          colors=brewer.pal(8, "Set1"),
          main = "Title")

dev.off()

#### world coud no title
png("_assets/Shakespeare_Keywords_WordCloud_No_title.png", width=1980, height=1080, pointsize=

layout(matrix(c(1, 2), nrow=2), heights=c(1, 4))
par(mar=rep(0, 4))
plot.new()

set.seed(314)
wordcloud(words = dtm_d$word,
          freq = dtm_d$freq,
          min.freq = 5,
          max.words=100,
          random.order=FALSE,
          rot.per=0.20,
          colors=brewer.pal(8, "Set1"),
          main = "Title")

dev.off()

##### proportion calculation for Shakespeare
head(dtm_d)
num_terms <- length(x_text_dtm$i); num_terms #total number of terms - 7611
nTerms(x_text_dtm) # unique terms in sonnets - 3089
head(Terms(x_text_dtm)) #just a list of all the terms that show up

prop_terms<-  dtm_d$freq/num_terms; head(prop_terms)

dtm_prop <- cbind(dtm_d, prop_terms)

##########################################
# Music Artist Key Word Extraction
##########################################
# This loop performs key word extraction on each artist's work
# Result: - A list containing a df for each artists.
#         - Each artist's df will contain: top 10 keywords, frequencies, proportions

#creating list that will be populated by loop
artist_keyword <- list()
artist_keyword

artist_text <- Corpus(VectorSource(artists_complete[,2])); artist_text

for(i in 1:length(artists_complete[,2])){
  kw <- rep(NA, 10)
  frq <- rep(NA, 10)
  prp <- rep(NA, 10)
  temp_mat <- cbind(kw, frq, prp)
  artist_keyword[[i]] <-as.data.frame(temp_mat) # ea. element will be df with 10 rows and 3 co
  colnames(artist_keyword[[i]]) <- c("keyword","frequency", "proportion")

  art_doc <- artist_text[i]

  #Replacing "/", "@" and "|" with space
  toSpace <- content_transformer(function (y , pattern ) gsub(pattern, " ", y))
  art_doc <- tm_map(art_doc, toSpace, "/")
  art_doc <- tm_map(art_doc, toSpace, "@")
  art_doc <- tm_map(art_doc, toSpace, "\\|")
  #art_doc <- tm_map(art_doc, to_e, "????T")

  # Convert the text to lower case
  art_doc <- tm_map(art_doc, content_transformer(tolower))

  # Remove numbers
  art_doc <- tm_map(art_doc, removeNumbers)

  # Remove common English stop words
  art_doc <- tm_map(art_doc, removeWords, stopwords("english"))

  # Removing custom stop words, specify stop words as a character vector
  art_doc <- tm_map(art_doc, removeWords, c("aint", "ooh", "thou", "never", "yeah", "hey", "th
                                             "upon", "from", "nah", "aint", "now", "one", "two"
  # Remove punctuation
  art_doc <- tm_map(art_doc, removePunctuation)

  # Eliminate extra white spaces
  art_doc <- tm_map(art_doc, stripWhitespace)

  #art_doc #NO DOCUMENTS DROPPED

  #number of total terms is the non sparse entries
  art_doc_dtm <- TermDocumentMatrix(art_doc)
  artist_mat_dtm <- as.matrix(art_doc_dtm)

  # Sort by decreasing value of frequency
  artisit_dtm_v <- sort(rowSums(artist_mat_dtm),decreasing=TRUE)
  artist_dtm_d <- data.frame(word = names(artisit_dtm_v),freq=artisit_dtm_v)
  artist_dtm_d

  artist_keyword[[i]]$keyword[1:10] <- artist_dtm_d$word[1:10] #populating keywords
  artist_keyword[[i]]$frequency[1:10] <- artist_dtm_d$freq[1:10]; artist_keyword #populating f

  art_num_terms <- length(art_doc_dtm$i); art_num_terms #total number of terms
  artist_keyword[[i]]$proportion[1:10] <- artist_dtm_d$freq[1:10]/art_num_terms #populating pr

  }

names(artist_keyword) <- artists_complete$artist_names
```

```
artist_keyword[3] #look at the data frame one at a time otherwise your computer won't like you
artist_keyword[1:45] #run this from your console to see results more easily


######## Loop to generate top Music Artists key words World Clouds
top_artists_df <- artists_complete[artists_complete$Artist %in% c("amy-winehouse", "nickelback"
                                                                  "joni-mitchell", "bob-dylan"
                                                                  "leonard-cohen", "britney-spe
top_artists_df

for(i in 1:nrow(top_artists_df)){

  art_doc <- Corpus(VectorSource(top_artists_df[i,2]))

  #Replacing "/", "@" and "|" with space
  toSpace <- content_transformer(function (y , pattern ) gsub(pattern, " ", y))
  art_doc <- tm_map(art_doc, toSpace, "/")
  art_doc <- tm_map(art_doc, toSpace, "@")
  art_doc <- tm_map(art_doc, toSpace, "\\|")
  #art_doc <- tm_map(art_doc, to_e, "????T")

  # Convert the text to lower case
  art_doc <- tm_map(art_doc, content_transformer(tolower))

  # Remove numbers
  art_doc <- tm_map(art_doc, removeNumbers)

  # Remove common English stop words
  art_doc <- tm_map(art_doc, removeWords, stopwords("english"))

  # Removing custom stop words, specify stop words as a character vector
  art_doc <- tm_map(art_doc, removeWords, c("aint", "ooh", "thou", "never", "yeah", "hey", "th
                                          "upon", "from", "nah", "aint", "now", "one", "two"
  # Remove punctuation
  art_doc <- tm_map(art_doc, removePunctuation)

  # Eliminate extra white spaces
  art_doc <- tm_map(art_doc, stripWhitespace)

  #art_doc #NO DOCUMENTS DROPPED

  #number of total terms is the non sparse entries
  art_doc_dtm <- TermDocumentMatrix(art_doc)
  artist_mat_dtm <- as.matrix(art_doc_dtm)

  # Sort by decreasing value of frequency
  artisit_dtm_v <- sort(rowSums(artist_mat_dtm),decreasing=TRUE)
  artist_dtm_d <- data.frame(word = names(artisit_dtm_v),freq=artisit_dtm_v)
  artist_dtm_d

  #word cloud generation for artist
  artistName <- top_artists_df$Artist[i]
  wc_Filename <- paste0("_assets/", artistName, "_KeyWord_WordCloud4.png")

  png(wc_Filename, width=1920, height=1080, pointsize=35)

  plot.new()

  set.seed(314)
  wordcloud(words = artist_dtm_d$word, freq = artist_dtm_d$freq, min.freq = 5,
            max.words=100, random.order=FALSE, rot.per=0.20,
            colors=brewer.pal(8, "Dark2"))

  dev.off()
}

##############################################################
#                 Sentiment Analysis
##############################################################
# Sentiment Analysis for Shakespeare's Sonnets
#############################################

# exporting plots as png
all_sonnets <- paste(og_sonnets$Sonnets[1:154], collapse = " ") #the collapse argument allows y
all_sonnets

shake_8sent <- get_nrc_sentiment(all_sonnets) #getting shakespeare sentiments
shake_8sent

shake_8sent_emotions <-shake_8sent[1:8]
shake_8sent_sents <- shake_8sent[9:10]

shake_8sent_emotions <- as.matrix(shake_8sent_emotions)
shake_8sent_sents <- as.matrix(shake_8sent_sents)

png("_assets/Barplot_Sonnets_8_Emotions.png", width=1920, height=1080, pointsize=35)
```

```
plot.new()

barplot(shake_8sent_emotions,
        main = "Frequency of Each Emotion",
        las = 2,
        col = brewer.pal(8, "Set1"))

dev.off()

png("_assets/Barplot_Sonnets_2_Sentiments.png", width=1920, height=1080, pointsize=35)
plot.new()
barplot(shake_8sent_sents,
        main = "Frequency of Positive and Negative Sentiments",
        las = 2,
        col = brewer.pal(3, "Set1"))

dev.off()

##################################
#  Music Artist Sentient Analysis
####################################
sent_scores<- get_sentiment(artists_complete$Lyrics) #returns overall sentiment value
summary(sent_scores)

artists_8sent <- get_nrc_sentiment(artists_complete$Lyrics) #returns classification into each 8
artists_8sent


all_sonnets <- paste(og_sonnets$Sonnets[1:154], collapse = " ") #the collapse argument allows y
all_sonnets

shake_8sent <- get_nrc_sentiment(all_sonnets) #getting shakespeare sentiments

shake_8sent


all_8sent <- rbind(artists_8sent, shake_8sent) #appending shakespeare's sentiments to table wit

names_8sent <- c(artists_complete$Artist, "shakespeare") #getting artist names for table
names_8sent

all_8sent <- cbind(names_8sent, all_8sent) #adding names
all_8sent #quick check
```

```
####### Creating a table of proportions instead of frequencies
prop_8sent <- data.frame(matrix(ncol = 9, nrow = 46))

prop_8sent[,1] <- all_8sent$names_8sent

for(i in 1:46){ #this loop divides each element in a row bu the row sum

  prop_8sent[i,2:9] <- all_8sent[i,2:9]/sum(all_8sent[i,2:9])

}

prop_8sent #another quick check

#Now that we have the sentiments in a table kmeans clustering will be straightforward.
#I will use MG's lovely code for this part :*
#identifying optimal number of clusters
k <- 12
vpc <- NULL #vpc here means "variance per cluster"
for (i in 1:k) {
    kfit <- kmeans(prop_8sent[,2:9], i)
    vpc <- c(vpc, kfit$betweenss/kfit$totss)
}
vpc

#writing plot out as a png
png("_assets/Variance_Explained_VS_Number_Clusters.png", width=1920, height=1080, pointsize=35
plot.new()
plot(1:k, vpc, xlab = "# of clusters", ylab = "explained variance", main = "Explained Variance

dev.off()

kfit7 <- kmeans(prop_8sent[,2:9], 7)
kfit8 <- kmeans(prop_8sent[,2:9], 8)
kfit10 <- kmeans(prop_8sent[,2:9], 10)
kfit12 <- kmeans(prop_8sent[,2:9], 12)

#clustering means were given for each of the eight emotions
kfit7$centers
kfit8$centers
kfit10$centers
kfit12$centers
```

```
########### Making Cluster Plots
##changing the row names so they will show up as labels for each point
row.names(prop_8sent) <- all_8sent$names_8sent

{ #Run this to generate plot all at once

png("_assets/ClusterAnalysis_Fit7.png", width=1920, height=1080, pointsize=35)
plot.new()
fviz_cluster(kfit7, data = prop_8sent[,2:9],
             palette = brewer.pal(7, "Set1"),
             geom = c("point", "text"),
             ellipse = F,
             ellipse.type = "t",
             ggtheme = theme_bw()
             )
dev.off()

png("_assets/ClusterAnalysis_Fit8.png", width=1920, height=1080, pointsize=35)
plot.new()
fviz_cluster(kfit8, data = prop_8sent[,2:9],
             palette = brewer.pal(8, "Set1"),
              geom = c("point", "text"),
             ellipse = F,
             ellipse.type = "t",
             ggtheme = theme_bw()
             )
dev.off()

#this is the plot we decided to use for final article
png("_assets/ClusterAnalysis_Fit10.png", width=1920, height=1080, pointsize=35)
plot.new()
fviz_cluster(kfit10, data = prop_8sent[,2:9],
             palette = brewer.pal(10, "Paired"),
             geom = c("point", "text"),
             ellipse = F,
             ellipse.type = "t",
             labelsize = 35 ,
             ggtheme = theme_bw()) + ggtitle("Cluster Analysis of all Artist and Shakespeare")




dev.off()
```

```
png("_assets/ClusterAnalysis_Fit12.png", width=1920, height=1080, pointsize=35)
plot.new()
fviz_cluster(kfit12, data = prop_8sent[,2:9],
             palette = brewer.pal(12, "Paired"),
              geom = c("point", "text"),
             ellipse = F,
             ellipse.type = "t",
             ggtheme = theme_bw()
             )
dev.off()

}
test <- cbind (all_8sent, kfit$cluster) #Appended the clusters to all artists sentiments to ge
#need to check which cluster shakespeare is in each time, because different groups might be fo

# test[test$‘kfit$cluster‘ == 4,] #these are the artists in shakespeare’s cluster

####################################
# Sentiment Ranking of Shakespeare vs Artists
####################################
x_text <- Corpus(VectorSource(og_sonnets$Sonnets))

x_text

#Replacing "/", "@" and "|" with space
toSpace <- content_transformer(function (y , pattern ) gsub(pattern, " ", y))
x_text <- tm_map(x_text, toSpace, "/")
x_text <- tm_map(x_text, toSpace, "@")
x_text <- tm_map(x_text, toSpace, "\\|")
x_text <- tm_map(x_text, removeNumbers)
x_text <- tm_map(x_text, removeWords, stopwords("english"))
x_text <- tm_map(x_text, removeWords, c("thi", "thee", "thou", "may", "still", "thus", "though
x_text <- tm_map(x_text, removePunctuation)
x_text <- tm_map(x_text, stripWhitespace)

# Build a term-document matrix
x_text_dtm <- TermDocumentMatrix(x_text)
x_text_dtm

#number of total terms is the non sparse entries
mat_dtm <- as.matrix(x_text_dtm)
```

```
# mat_dtm

# Sort by decreasing value of frequency
dtm_v <- sort(rowSums(mat_dtm),decreasing=TRUE)
dtm_d <- data.frame(word = names(dtm_v),freq=dtm_v)

# Display the top 50 most frequent words
head(dtm_d, 50)

num_terms <- length(x_text_dtm$i); num_terms #total number of terms - 7611
nTerms(x_text_dtm) # unique terms in sonnets - 3089
head(Terms(x_text_dtm)) #just a list of all the terms that show up


prop_terms<-  dtm_d$freq/num_terms; head(prop_terms)
dtm_prop <- cbind(dtm_d, prop_terms)
sonnet_data <- head(dtm_prop, 10); sonnet_data

#data cleanup using dplyr
# #the pattern for the special character may vary from computer to computer
# here are other patterns incase is the one bellow wont work
# ????T, ???T,
sonnets <-
  og_sonnets %>%
  mutate_at("Sonnets", str_replace, "???T", "\'")

sonnets_df1 <- data.frame(matrix(ncol=1,nrow=154,
                                 dimnames=list(NULL, "Sonnets")))

#I manually removed the punctuation except for apostrophes
n <- 154
for (i in 1:n) {
  sonnets_df1[i,]<- gsub("[,.;:?!]", "", og_sonnets[i,1])
  rbind(sonnets_df1[i,])
}

#create an empty dataframe
emotions <- data.frame(matrix(ncol=8,nrow=0, dimnames=list(NULL,
      c("anger", "anticipation", "disgust", "fear", "joy", "sadness",
        "surprise", "trust"))))

#n <- 154
#ran a for loop to conduct sentiment analysis to each individual psuedosonnet
```

```
#beware: it might take a lil while to run
#for (i in 1:n) {
  #emotions[i,]<- get_nrc_sentiment(sonnets_df1[i,1])
  #rbind(emotions[i,])
#}
#s <- colSums(emotions)

##### Loop calcuates proportions and frequencies of the different emotions used by artist and S
#the collapse argument allows you to create a single string by concatenating ea element of the
all_sonnets <- paste(sonnets_df1, collapse = " ")

#getting shakespeare sentiments
emotions <- get_nrc_sentiment(all_sonnets)

s_prop <- emotions[1,1:8] / sum(emotions[1,1:8])

artist_keyword <- data.frame(artist = artists_complete$Artist, sent = rep(NA, 45), word = rep(

artist_text <- Corpus(VectorSource(artists_complete[,2])); artist_text

for(i in 1:length(artists_complete[,2])){

  kw <- rep(NA, 10)
  frq <- rep(NA, 10)
  prp <- rep(NA, 10)
  temp_mat <- cbind(kw, frq, prp)

  #temp_mat is the information of the current artist
  temp_df <- as.data.frame(temp_mat)
  colnames(temp_df) <- c("word","frequency", "proportion" )

  art_doc <- artist_text[i]

  ## Data cleaning
  toSpace <- content_transformer(function (y , pattern ) gsub(pattern, " ", y))
  art_doc <- tm_map(art_doc, toSpace, "/")
  art_doc <- tm_map(art_doc, toSpace, "@")
  art_doc <- tm_map(art_doc, toSpace, "\\|")
  art_doc <- tm_map(art_doc, content_transformer(tolower))
  art_doc <- tm_map(art_doc, removeNumbers)
  art_doc <- tm_map(art_doc, removeWords, stopwords("english"))
  art_doc <- tm_map(art_doc, removeWords, c("aint", "ooh", "thou", "never", "yeah", "hey", "th
  art_doc <- tm_map(art_doc, removePunctuation)
  art_doc <- tm_map(art_doc, stripWhitespace)
  x_text <- tm_map(art_doc, stemDocument)

  art_doc_dtm <- TermDocumentMatrix(art_doc)
  #number of total terms is the non sparse entries
  #art_doc_dtm
  artist_mat_dtm <- as.matrix(art_doc_dtm)
  # mat_dtm
  # Sort by decreasing value of frequency
  artisit_dtm_v <- sort(rowSums(artist_mat_dtm),decreasing=TRUE)
  artist_dtm_d <- data.frame(word = names(artisit_dtm_v),freq=artisit_dtm_v)
  artist_dtm_d

  temp_df$word[1:10] <- artist_dtm_d$word[1:10]
  temp_df$frequency[1:10] <- artist_dtm_d$freq[1:10];
  art_num_terms <- length(art_doc_dtm$i); art_num_terms #total number of terms
  temp_df$proportion[1:10] <- artist_dtm_d$freq[1:10]/art_num_terms

  compare <- sonnet_data %>%
    left_join(temp_df, by = "word") %>%
    filter(!is.na(frequency))

  artist_keyword[i,]$word <- nrow(compare)
  artist_keyword[i,]$freq_df <- sum((compare$freq - compare$frequency)^2)
  artist_keyword[i,]$prop_df <- sum((compare$prop_terms - compare$proportion)^2)

  temp_word <- ""
  for (j in 1:length(compare$word)) {
    temp_word <- paste0(temp_word," ",compare$word[j])
  }

  artist_keyword[i,]$keyword <- temp_word

  ## Sentiment Analysis
  art_data <- gsub("[,.;:?!]", "", artists_complete[i,2])

  emotions <- data.frame(matrix(ncol=8,nrow=0, dimnames=list(NULL,
      c("anger", "anticipation", "disgust", "fear", "joy", "sadness",
        "surprise", "trust"))))

  emotions <- get_nrc_sentiment(art_data)
  emotions_prop <- emotions[1,1:8] / sum(emotions[1,1:8])
```

```
  artist_keyword[i,]$sent <- sum((s_prop - emotions_prop)^2)
}

table_word <- artist_keyword[order(artist_keyword$word,-rank(artist_keyword$freq_df), decreasi
  mutate(rank_word = 1:nrow(artist_keyword))
table_sent <- artist_keyword[order(artist_keyword$sent),]  %>%
  mutate(rank_sent = 1:nrow(artist_keyword))

rank_table <- table_word %>%
  left_join(table_sent, by = "artist") %>%
  select(artist, rank_word, rank_sent) %>%
  mutate(rank = rank_word + rank_sent)

rank_table
table_word
table_sent

## Writing resulting top 10 most similar artist to Shakespeare out as a write able .tex file
t <- rank_table[order(rank_table$rank),]

#selecting top ten
t1 <- t[1:10,]

####make bottom 5 least similar to shakespeare

t1 <- as.data.frame(t1)
#rename column name so table looks nice in latex
names(t1) <- c('Artist', 'Word Rank', 'Sentiment Rank', 'Overall Rank')
#making xtable to .tex file
xttbl <- xtable(t1, caption="Ranked Top 10 Most Similar Music Artist to Shakespeare", label="t
xxx <- print(xttbl, include.rownames=FALSE)
writeLines( xxx, file.path("_assets", "overall_rankTable_top10_Similar.tex") )

##writing bottom 5 just in case we want to use it

b_t1 <- t[40:45,]
names(b_t1) <- c('Artist', 'Word Rank', 'Sentiment Rank', 'Overall Rank')

xttbl <- xtable(b_t1, caption="Ranked Bottom 5 Least Similar Music Artist to Shakespeare", lab
xxx <- print(xttbl, include.rownames=FALSE)
writeLines( xxx, file.path("_assets", "bottom5_rankTable_top10_Similar.tex") )

##Writing top 10 Keyword Rankings
```

```
t2 <- table_word %>% select(artist, word, freq_df, keyword, rank_word)
t2 <- t2[1:10,]
names(t2) <- c('Artist', 'Word Count', 'Frequency', 'Keyword', 'Word Rank')

xttbl <- xtable(t2, caption="Ranked Top 10 Most Similar Music Artist to Shakespeare Based on Ke
xxx <- print(xttbl, include.rownames=FALSE)
writeLines( xxx, file.path("_assets", "word_rankTable_top10_Similar.tex") )

##Writing top 10 Sentiment Rankings

t3 <- table_sent %>% select(artist, sent, rank_sent)
t3 <- t3[1:10,]
names(t3) <- c('Artist', 'Sent. Euclidean Distance', 'Sentiment Rank')

xttbl <- xtable(t3, digits = 6, caption="Ranked Top 10 Most Similar Music Artist to Shakespeare
xxx <- print(xttbl, include.rownames=FALSE)
writeLines( xxx, file.path("_assets", "sent_rankTable_top10_Similar.tex") )
```

References

[1] 56 Fun Facts about William Shakespear. https://nosweatshakespeare.com/resources/shakespeare-facts/. Accessed May 31, 2022.

[2] An Introduction to Knowledge Graphs. http://ai.stanford.edu/blog/introduction-to-knowledge-graphs/. Accessed June 3, 2022.

[3] K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks. https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a. Accessed May 31, 2022.

[4] Natural Language Processing With R. https://www.udacity.com/blog/2020/10/natural-language-processing-with-r.html. Accessed May 20, 2022.

[5] NRC Word-Emotion Association Lexicon. https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm. Accessed June 3, 2022.

[6] Syuzhet Package Documentation in R. https://www.rdocumentation.org/packages/syuzhet/versions/1.0.6. Accessed May 20, 2022.

[7] Text Mining and Sentiment Analysis: Analysis with R. https://www.red-gate.com/simple-talk/databases/sql-server/bi-sql-server/text-mining-and-sentiment-analysis-with-r/. Accessed May 20, 2022.

[8] Tm: Text Mining Package Documentation in R. https://rdrr.io/rforge/tm/man/. Accessed May 20, 2022.

[9] What is Text Analysis? A Beginner's Guide. https://monkeylearn.com/text-analysis/. Accessed May 20, 2022.

[10] Kristin Briney. *Data Management for Researchers: Organize, maintain and share your data for research success.* Pelagic Publishing Ltd, 2015.

[11] Jose P. G. Mahedero, Álvaro MartÍnez, Pedro Cano, Markus Koppenberger, and Fabien Gouyon. Natural language processing of lyrics. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, MULTIMEDIA '05, page 475478, New York, NY, USA, 2005. Association for Computing Machinery.

[12] Sergio Oramas, Luis Espinosa-Anke, Francisco Gmez, and Xavier Serra. Natural language processing for music knowledge discovery. *Journal of New Music Research*, 47(4):365–382, 2018.

[13] Paul Mooney. Song Lyrics: Poetry and Lyric (TXT files). Accessed May 20, 2022.

[14] William Shakespear. Shakespear's Sonnets. Project Gutenberg, 1997 [Online]. EBook-No. 1041, Accessed May 20, 2022.