Texts Analysis using Tweets from Congress

Amber

2025-02-03

Project Description

This project analyzes tweets from politicians, focused on pre-processing and analyzing textual contents. The project employs various natural language processing techniques - follows a systematic workflow, including:

1. Document sampling
2. Text preprocessing (tokenization, cleaning, and stemming)
3. Exploratory analysis using word clouds and TF-IDF to visualize word frequencies andhighlight discriminative terms
4. Sentiment classification through VADER dictionary to categorize tweets as positive, neutral, or negative
5. Similarity analysis through cosine similarity to measure thematic consistency

Load dataset

```
tweets <- read_csv("~/Desktop/Github_Projects/tweets_congress.csv")
```

Part 1

I **took a sample** of my documents and read them carefully. This sample is not random; instead, I chose documents that are particularly interesting to me (**tweets from only democrats**). After reviewing the sample, I reflect on my observations, key takeaways, and any notable patterns or insights that emerge.

Filter, subset and read my documents

```
# Filter all tweets from democrats
tweets_dem <- tweets %>%
  filter(Party == "D")
head(tweets_dem)

# Dataset too large ; create a subset.
set.seed(1) # Ensure we get the same output of a randomization
tweets_dem_sub <- tweets_dem %>%
  sample_n(50000) # Create a subset of 50000 observations

# Randomly read 10 rows
set.seed(2)
tweets_dem_sub %>%
  sample_n(10)
```

From the **data** side, these documents are long and messy, containing texts and author information. These texts include a lot of emojis, mentions , hash tags, and URLs which can make pre-processing complicated. These also include HTML escape characters like &amp which should be “&” and line breaks.

From the **content** side, after reading the tweets from some of the democrats, I found out that they brought up a lot frequently mentioned issues that democrats care about, including “LGBTQIA+”, “BorderWall”,“newest citizen”,“Covid-19”, etc. These relate to political topics like immigration policy, government spending, and civic engagement. We can also see some strong sentiments behind these tweets by looking at words like “useless” and “congrats”.

Part 2

I **tokenized** my documents and **pre-processed** them by removing any extraneous content identified during my close reading of a sample. I evaluated which content to remove based on its relevance and impact on the analysis.

Create a corpus

```
# Convert the dataset into a corpus
tweets_corpus <- corpus(tweets_dem_sub, text_field = "text")
summary(tweets_corpus, n =5) # Look at the first 5 documents
```

```
## Corpus consisting of 50000 documents, showing 5 documents:
##
##   Text Types Tokens Sentences        author                           date
##  text1    27     29         1  RepSusanWild Mon Apr 06 21:15:08 +0000 2020
##  text2    21     22         2        RepMGS Thu May 13 16:25:00 +0000 2021
##  text3    24     27         1  RepAndyKimNJ Sat Feb 27 16:48:02 +0000 2021
##  text4    23     24         2  repmarkpocan Wed Mar 31 18:23:01 +0000 2021
##  text5    21     25         1 SenGaryPeters Tue Apr 28 16:36:49 +0000 2020
##
bios
##
Proud to represent Pennsylvania's 7th Congressional District
##                                   Representing Pennsylvania's 5th Congressional Distri
ct. Vice Chair of @HouseAdm_Dems. Member of @HouseJudiciary @RulesDemocrats.
##  Husband to my college sweetheart. Father to two troublemaking baby boys. Congressman
for New Jersey's Third District. All tweets from Congressman Kim signed –AK
##  Honored to serve the people of Wisconsin's 2nd District. \nCo-Chair of @Labor_Caucus
& @LGBTEqCaucus 🏳️‍🌈. \nMember of @AppropsDems & @EdLaborCmte. \nHe/Him/His.
##                                                    U.S. Senator proudly representing th
e state of Michigan. For federal resources on the Coronavirus click here ⬇️⬇️⬇️
##  retweet_author              Name                              Link State Party
##            <NA>       Wild, Susan  https://twitter.com/RepSusanWild    PA     D
##            <NA> Scanlon, Mary Gay        https://twitter.com/RepMGS    PA     D
##            <NA>         Kim, Andy  https://twitter.com/RepAndyKimNJ    NJ     D
##    LGBTEqCaucus       Pocan, Mark  https://twitter.com/repmarkpocan    WI     D
##            <NA>   Peters, Gary C. https://twitter.com/SenGaryPeters    MI     D
##  congress
##     House
##     House
##     House
##     House
##    senate
```

Tokenization and Pre-processing

```
# Tokenize the corpus

text_clean <- gsub("'s|'m|'re|'d|'ve|'ll|n't|’s|’m|’re|’d|’ve|’ll|n’t", "", tweets_corpu
s)  # Remove common English contractions and avoid empty spaces being recognized by a to
ken

dem_tokens <- tokens(text_clean, remove_punct = TRUE, remove_numbers = TRUE) %>% # Remov
e punctuation and numbers
  tokens_tolower() %>% # Convert all tokens to lower case
  tokens_remove(c(stopwords("en"), "madam", "mr", "today","rt")) %>% # Remove common Eng
lish stopwords, retweet identifier and frequently appeared meaningless words
  tokens_remove(pattern = "#*|@*") %>%  # Remove hash tags and handles
  tokens_remove(pattern = "^https://.*", valuetype = "regex") %>% # Remove web links
  tokens_remove(c("amp","$")) %>% # Remove HTML escape characters
  tokens_wordstem() # Word stemming

# Look at the first few tokenized texts
head(dem_tokens)
```

```
## Tokens consisting of 6 documents and 9 docvars.
## text1 :
##  [1] "know"        "mani"        "laid"        "b"           "c"
##  [6] "find"        "new"         "sourc"       "incom"       "even"
## [11] "temporarili"
##
## text2 :
##  [1] "neither" "stigmat" "poor"    "leav"    "vulner"  "resent"  "special"
##  [8] "benefit" "say"     "instead"
##
## text3 :
##  [1] "rescu"     "kid"       "readi"     "kid"       "go"        "back"
##  [7] "school"    "in-person" "make"      "sure"      "school"
##
## text4 :
## [1] "check"   "along"   "show"    "support" "tran"    "peopl"
##
## text5 :
##  [1] "michigan" "hospit"   "health"   "care"     "provid"   "sever"
##  [7] "financi"  "strain"   "due"      "covid-19" "struggl"  "keep"
## [ ... and 1 more ]
##
## text6 :
## [1] "believ"     "whole"      "russia"     "investig"   "fraud"
## [6] "witch-hunt" "rod"        "rosenstein"
```

During my tokenization and preprocessing, I first cleaned the texts in my corpus by **removing common English contractions.** Then, I tokenized by **removing punctuation and numbers** and converting all tokens to **lowercase**. I removed **common English stop words** like “and” and meaningless words like “mr” that often appear in political contents. I also removed **hash tags (#), mentions (@) , re-tweets and website links** starting with "https". After I glimpsed the tokens and found out that there are a lot of words "amp" which can be resulted from **HTML escape characters** like &amp, I removed all "amp". Besides, I used **word stemming** to reduce duplicated words and reduce the dataset size for faster analysis later on.

I **didn't use n-grams** because I don't think it makes sense to make all individual tokens into, say, bi-grams as some words might make sense but most of them won't.

Part 3

I identified **location** as an important source of variation in my data. I then **subseted** the data along Northeastern and Western states and **created word clouds** for each category. Finally, I examined the differences between the word clouds to identify notable patterns or variations.

Subset my data along location, tokenize and preprocess

```
# Subset my original dataset by states that are in the West
tweets_west <- tweets_dem_sub %>%
  filter(State == "CA" | State == "WA" | State == "OR" | State == "AK" | State == "HI"
         | State == "MT" | State == "ID" | State == "WY" | State == "NV" | State == "UT"
         | State == "CO" | State == "AZ" | State == "NM")

# Subset my original dataset by states that are in the Northeast
tweets_east <- tweets_dem_sub %>%
  filter(State == "ME" | State == "NH" | State == "VT" | State == "MA" | State == "RI"
         | State == "CT" | State == "NY" | State == "NJ" | State == "PA")

# Create corpus for both
west_corpus <- corpus(tweets_west, text_field = "text")
east_corpus <- corpus(tweets_east, text_field = "text")

# Tokenize and pre-process both corpuses

## West

text_clean_west <- gsub("'s|'m|'re|'d|'ve|'ll|n't|’s|’m|’re|’d|’ve|’ll|n’t", "", west_co
rpus) # Remove common English contractions and avoid empty spaces being recognized by a
token

west_tokens <- tokens(text_clean_west, remove_punct = TRUE, remove_numbers = TRUE) %>% #
Remove punctuation and numbers
  tokens_tolower() %>% # Convert all tokens to lower case
  tokens_remove(c(stopwords("en"), "madam", "mr", "today","rt")) %>% # Remove common Eng
lish stopwords, retweets identifier and frequetly appeared meaningless words
  tokens_remove(pattern = "#*|@*") %>%  # Remove hashtags and handles
  tokens_remove(pattern = "^https://.*", valuetype = "regex") %>% # Remove web links
  tokens_remove(c("amp","$")) %>% # Remove HTML escape characters
  tokens_wordstem() # Word stemming

## Northeast

text_clean_east <- gsub("'s|'m|'re|'d|'ve|'ll|n't|’s|’m|’re|’d|’ve|’ll|n’t", "", east_co
rpus) # Remove common English contractions and avoid empty spaces being recognized by a
token

east_tokens <- tokens(text_clean_east, remove_punct = TRUE, remove_numbers = TRUE) %>% #
Remove punctuation and numbers
  tokens_tolower() %>% # Convert all tokens to lower case
  tokens_remove(c(stopwords("en"), "madam", "mr", "today","rt")) %>% # Remove common Eng
lish stopwords, retweets identifier and frequetly appeared meaningless words
  tokens_remove(pattern = "#*|@*") %>%  # Remove hashtags and handles
  tokens_remove(pattern = "^https://.*", valuetype = "regex") %>% # Remove web links
  tokens_remove(c("amp","$")) %>% # Remove HTML escape characters
  tokens_wordstem() # Word stemming

# Convert tokens to Document Term Matrix(DFM) and trim
west_dfm <- west_tokens %>%
  dfm() %>%
```

```
  dfm_trim(min_docfreq = 0.001, max_docfreq = 0.999, docfreq_type = "prop", verbose = TR
UE) # useing 0.001 as a document frequency threshold (a frequently used threshold) is to
o strict
```

```
## dfm_trim() changed from 13,982 features (16,046 documents) to 1,553 features (16,046
documents)
```

```
east_dfm <- east_tokens %>%
  dfm() %>%
  dfm_trim(min_docfreq = 0.001, max_docfreq = 0.999, docfreq_type = "prop", verbose = TR
UE)
```

```
## dfm_trim() changed from 12,340 features (13,619 documents) to 1,611 features (13,619
documents)
```

```
# Check top features of each DFM
topfeatures(west_dfm)
```

```
##     work american      act    trump     need     year     help    thank 
##     1014      971      847      837      833      791      760      758 
##   famili   health 
##      752      716
```

```
topfeatures(east_dfm)
```

```
## american     work      new    thank     need     year     join      act 
##      884      782      732      678      665      658      643      627 
##    trump   famili 
##      626      613
```

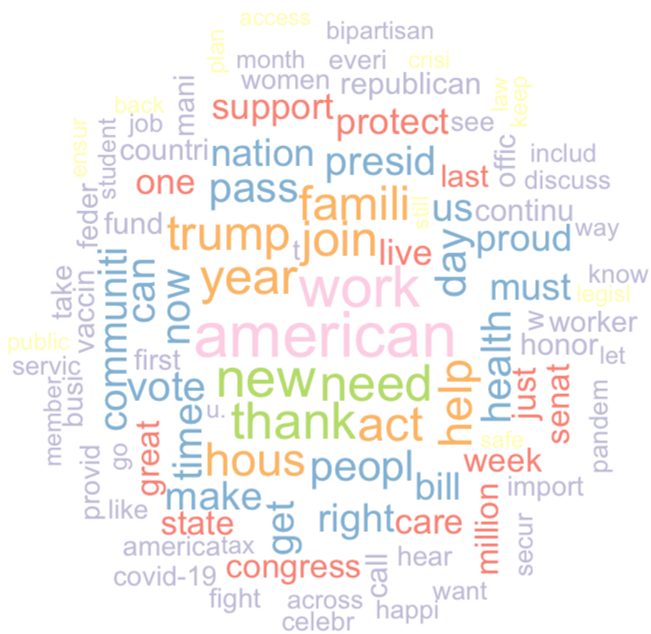
Create word clouds for comparison

```
# Load package necessary to create word clouds
pacman::p_load(quanteda.textplots)

# Create a word cloud for democrats' tweets from the West

wcWest <- textplot_wordcloud(west_dfm, random_order = FALSE,
                   rotation = .25, max_words = 100,
                   min_size = 0.5, max_size = 2.8,
                   color = RColorBrewer::brewer.pal(8, "Set3"))
```


work
american
trump
need
act
year
help
health
thank
famili
hous
presid
communiti
join
can
protect
support
pass
make
nation
peopl
congress
state
time
vote
million
care
one
bill
get
day
week
now
must
great
proud
countri
just
new
right
us
live
honor
continu
take
import
student
know
covid-19
everi
celebr
crisi
critic
discuss
america
senat
legisl
access
see
back
busi
colleagu
keep
provid
plan
ensur
administr
servic
veteran
bipartisan
across
member
mani
public
happi
job
last
hear
law
let
call
offic
women
vaccin
ca
invest
pandem
worker
feder
fight
includ
counti
fund
like
go
u.
wt
first
de



From these **word clouds**, there are generally **no major difference** between the contents between these two regions. They all mentioned “American” and “work” the most often, followed by words like “work”, “family” , “community” and “trump”.

Part 4

I subseted my data along the same variation and **identified words that differentiate between groups**. To achieve this, I applied **TF-IDF (Term Frequency-Inverse Document Frequency)**. This method **highlighted words that are more distinctive** in one group compared to others. By analyzing these key terms, I gained insights into **how language use varies across different categories** and what words are **most representative** of each group.

Compute TF-IDF for my data for both groups

```
pacman::p_load(quanteda.textstats)

# Proportional TF and log-scaled IDF for West
tfidf_west <- west_dfm %>%
  dfm_tfidf(scheme_tf = "prop", scheme_df = "inversemax")

# Proportional TF and log-scaled IDF for Northeast
tfidf_east <- east_dfm %>%
  dfm_tfidf(scheme_tf = "prop", scheme_df = "inversemax")

# Get top words for states in the West
top_west <- textstat_frequency(tfidf_west, n = 10, force = TRUE)  # Top 10 words
top_west <- top_west %>% select(feature, frequency) # Keeps only the word (feature) and
its TF-IDF frequency

# Get top words for states in the Northeast
top_east <- textstat_frequency(tfidf_east, n = 10, force = TRUE)
top_east <- top_east %>% select(feature, frequency)

# This gives us top words that are most distinctive for each group based on TF-IDF value
s.

print(top_west)
```

```
##      feature frequency
## 1         de  28.67775
## 2      happi  26.53185
## 3     celebr  24.11782
## 4    discuss  22.30793
## 5  congratul  22.05625
## 6      honor  21.74848
## 7       hear  21.74633
## 8      great  21.51188
## 9       read  21.30789
## 10    counti  21.12008
```

```
print(top_east)
```

```
##    feature frequency
## 1    happi  22.19833
## 2   celebr  19.88974
## 3    honor  19.67867
## 4    great  19.59006
## 5       go  19.44438
## 6       de  19.38012
## 7  discuss  19.33510
## 8     read  19.09488
## 9     hear  19.09453
## 10    call  18.88208
```

Several words appear in both lists with high frequency, suggesting shared themes across both datasets. Words like **“happi” (happy), “celebr” (celebrate), and “honor”** indicate a strong presence of **positive emotions**. Additionally, **“discuss”, “read”, and “hear”** suggest that **engagement with content** — whether through conversation, reading, or listening—is a common theme in both groups.

Part 5

I **utilized an existing dictionary called VADER** to measure sentiment, tone. I then **labeled** my documents based on the sentiment categories and **visualized** the prevalence of each class to analyze trends and patterns in the data.

```
# Here I used VADER as it is useful for sentiment analysis on social media contents like
tweets.
library(vader)

# Write a function to tidy the results
get_vader_tidy <- function(text_clean){
  get_vader(text_clean) %>%
    tibble(outcomes=names(.),
           values=.)
}

# Apply VADER to the first 100 comments and store results in a tidy format
vader_outputs <- map(tweets_dem_sub$text[1:100], get_vader_tidy)

# Bind all outputs into a single data frame and merge with original dataset
dem_vader <- tweets_dem_sub %>%
  slice(1:100) %>%   # Keep only the first 1000 rows (matching `vader_outputs`)
  mutate(vader_output = vader_outputs) %>%   # Attach sentiment results
  unnest(vader_output)

# Filter for compound sentiment scores and select relevant columns (text, outcomes and v
alues)
dem_vader_fil <- dem_vader %>%
  filter(outcomes == "compound") %>%
  select(text, outcomes, values)
head(dem_vader_fil)
```

```
## # A tibble: 6 × 3
##   text                                                          outcomes values
##   <chr>                                                         <chr>    <chr>
## 1 "I know that for so many who have been laid off b/c of #COVID… compound 0
## 2 "“It neither stigmatizes the poor nor leaves them vulnerable … compound 0.683
## 3 "RESCUE OUR KIDS - we’re all ready for our kids to go back to… compound 0.736
## 4 "RT @LGBTEqCaucus: Check it out! The @LGBTEqCaucus along with… compound 0.457
## 5 "Michigan hospitals &amp; health care providers are under sev… compound -0.34
## 6 "Me: Do you believe the whole Russia investigation was a frau… compound -0.718
```

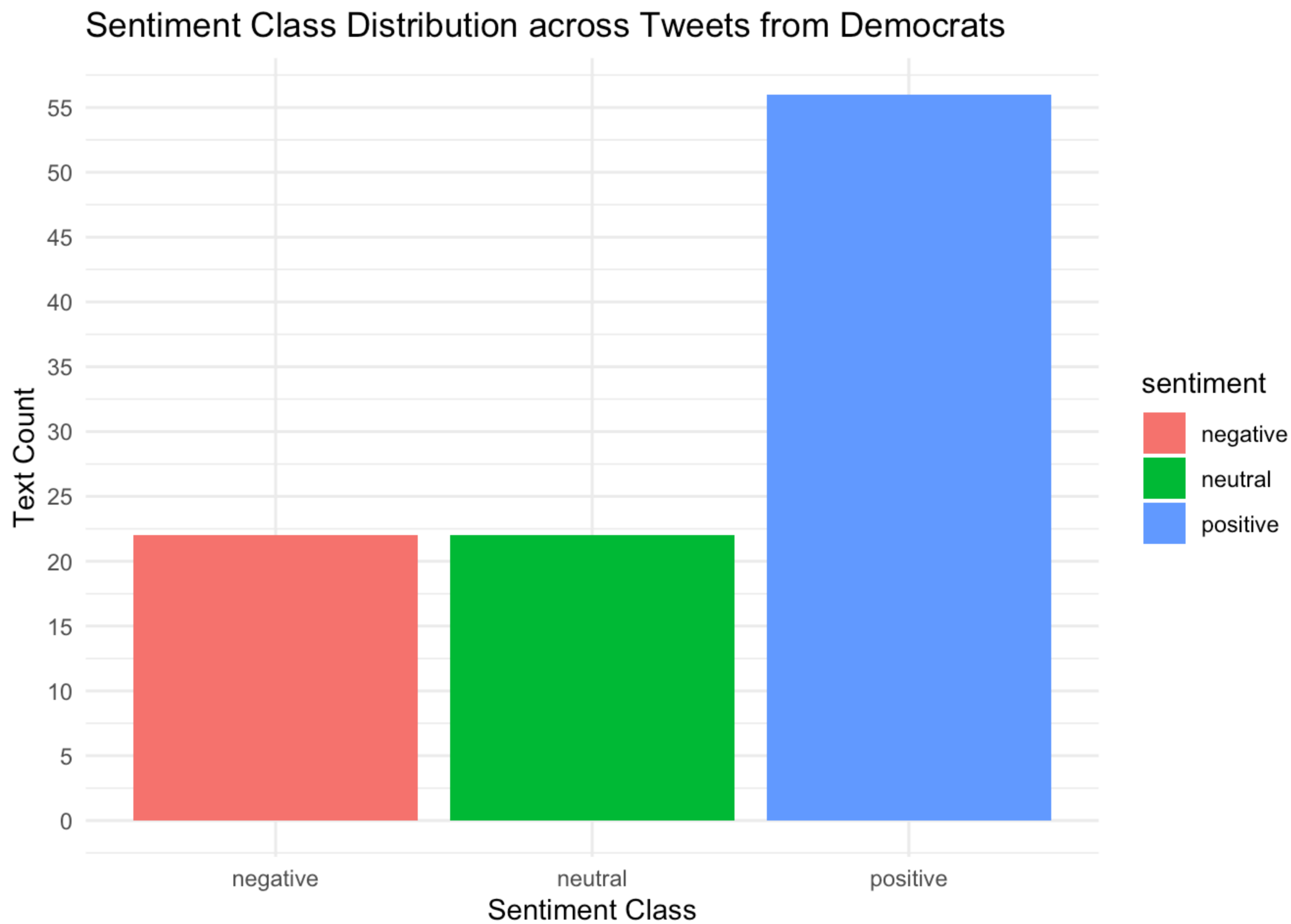
```r
# Sort from highest to lowest sentiment score
sorted_sentiment <- dem_vader_fil %>%
  arrange(desc(values))
head(sorted_sentiment)
```

```
## # A tibble: 6 × 3
##   text                                                                outcomes values
##   <chr>                                                               <chr>    <chr>
## 1 "“We do not serve one party or one interest, but we serve one… compound 0.874
## 2 "I was happy to help Tommy from @boyscouts Troop 1518 complet… compound 0.859
## 3 "Had great time @memgrizz victory in DC over the improved @Wa… compound 0.802
## 4 "RT @ProtectCareNH: Share an ACA health care story: Senator H… compound 0.796
## 5 "RT @RepCasten: The #IPCCReport made clear we’re running out … compound 0.785
## 6 "Washington state continues to make great progress towards ge… compound 0.785
```

```r
# Label my documents
sorted_sentiment <- sorted_sentiment %>%
  mutate(sentiment = case_when(
    values > 0  ~ "positive",
    values < 0  ~ "negative",
    values == 0 ~ "neutral"
  ))

# Count sentiment classes
sent_counts <- sorted_sentiment %>%
  count(sentiment)

# Create a bar plot to visualize the appearance of class
ggplot(sent_counts, aes(x = sentiment, y = n, fill = sentiment)) +
  geom_col() +
  labs(title = "Sentiment Class Distribution across Tweets from Democrats", x = "Sentime
nt Class", y = "Text Count") +
  scale_y_continuous(breaks = seq(0, 60, by = 5)) +
  theme_minimal()
```



The visualization reveals that **the majority of tweets exhibit a positive sentiment** (depicted in blue), significantly outnumbering both neutral (green) and negative (red) tweets. The counts for neutral and negative sentiments are relatively close, each accounting for roughly one-third of the positive sentiment count. This suggests that **Democratic tweets tend to convey more optimistic messages**.

Part 6

I seleced at least 10 documents that clearly, with high probability, represent each class in my sentiment classification dictionary.

Part 6.2

I used **cosine similarity** to identify the **10 documents** that are **most similar** to **a reference document with highest sentiment score** within each class. After reviewing these documents, I evaluated how well cosine similarity captures meaningful similarities between texts. Additionally, I experimented with **Euclidean distance measure** to compare the results and assess whether it provides different or improved document retrieval. This comparison helped determine **the effectiveness of different similarity measures** in identifying related content.

```
# Arrange with descending compound score
# sorted_sentiment <- sorted_sentiment %>%
#  arrange(desc(values))

# Convert to corpus
vader_corpus <- corpus(sorted_sentiment)

# Clean to corpus
vader_clean <- gsub("'s|'m|'re|'d|'ve|'ll|n't|’s|’m|’re|’d|’ve|’ll|n’t", "", vader_corpu
s)

# Convert to matrix
vader_dfm_matrix <- vader_clean %>%
  tokens(remove_punct = TRUE, remove_numbers = TRUE) %>%
    tokens_tolower() %>% #
    tokens_remove(c(stopwords("en"), "madam", "mr", "today","rt")) %>%
    tokens_remove(pattern = "#*|@*") %>%
    tokens_remove(pattern = "^https://.*", valuetype = "regex") %>%
    tokens_remove(c(":", "/")) %>%
    tokens_remove(c("amp","$")) %>%
    tokens_wordstem() %>%
  dfm() %>%
  as.matrix()

# Extract vectors for reference documents
mostpos <- vader_dfm_matrix["text1", ] # most positive
mostneg <- vader_dfm_matrix["text100", ] # most negative

# Function to calculate cosine similarity
calculate_cosine_similarity <- function(vec1, vec2) {
  dot_product <- sum(vec1 * vec2)
  magn1 <- sqrt(sum(vec1^2))
  magn2 <- sqrt(sum(vec2^2))
  return(dot_product / (magn1 * magn2))
}

# [Cosine similarity] Distance and similarity with reference to the most positive one
cosine_scores_pos <- apply(vader_dfm_matrix, 1, function(text) calculate_cosine_similari
ty(text, mostpos))
cosine_results_pos <- data.frame(text = rownames(vader_dfm_matrix), cosine_similarity =
cosine_scores_pos)

# Identify the 10 closet documents to the most positive

top10pos_co <- cosine_results_pos %>%
  arrange(desc(cosine_similarity)) %>%  # Sort in descending order
  slice(1:10)
print(top10pos_co)
```

```
##          text cosine_similarity
## text1   text1        1.00000000
## text97 text97        0.22613351
## text7   text7        0.19284730
## text56 text56        0.17739372
## text31 text31        0.07106691
## text15 text15        0.06741999
## text70 text70        0.06154575
## text98 text98        0.06154575
## text2   text2        0.00000000
## text3   text3        0.00000000
```

```
# [Cosine similarity] Distance and similarity with reference to the most negative one
cosine_scores_neg <- apply(vader_dfm_matrix, 1, function(text) calculate_cosine_similari
ty(text, mostneg))
cosine_results_neg <- data.frame(text = rownames(vader_dfm_matrix), cosine_similarity =
cosine_scores_neg)

# Identify the 10 closet documents to the most negative

top10neg_co <- cosine_results_neg %>%
  arrange(desc(cosine_similarity)) %>%  # Sort in descending order
  slice(1:10)
print(top10neg_co)
```

```
##            text cosine_similarity
## text100 text100        1.00000000
## text27   text27        0.19069252
## text29   text29        0.18257419
## text44   text44        0.17541160
## text77   text77        0.10540926
## text16   text16        0.10000000
## text73   text73        0.09128709
## text98   text98        0.09128709
## text68   text68        0.08451543
## text1     text1        0.00000000
```

Cosine Similarity focuses on the direction of text vectors rather than their magnitude, making it useful for comparing documents of different lengths.

```
# Function to calculate Euclidean distance between two vectors
calculate_euclidean_distance <- function(vec1, vec2) {
  ec_distance <- sqrt(sum((vec1 - vec2)^2)) # Euclidean formula
  return(ec_distance)
}

# [Euclidean similarity] Distance and similarity with reference to the most positive one
eu_scores_pos <- apply(vader_dfm_matrix, 1, function(text) calculate_euclidean_distance
(text, mostpos))
eu_results_pos <- data.frame(text = rownames(vader_dfm_matrix), euclidean_similarity = e
u_scores_pos)

# Identify the 10 closet documents to the most positive

top10pos_eu <- eu_results_pos %>%
  arrange(desc(euclidean_similarity)) %>%  # Sort in descending order
  slice(1:10)
print(top10pos_eu)
```

```
##          text euclidean_similarity
## text4   text4             6.480741
## text60 text60             6.244998
## text76 text76             6.164414
## text13 text13             6.082763
## text75 text75             6.082763
## text85 text85             6.082763
## text10 text10             6.000000
## text40 text40             6.000000
## text64 text64             6.000000
## text68 text68             6.000000
```

```
# [Euclidean similarity] Distance and similarity with reference to the most negative one
eu_scores_neg<- apply(vader_dfm_matrix, 1, function(text) calculate_euclidean_distance(t
ext, mostneg))
eu_results_neg <- data.frame(text = rownames(vader_dfm_matrix), euclidean_similarity = e
u_scores_neg)

# Identify the 10 closet documents to the most negative

top10neg_eu <- eu_results_neg %>%
  arrange(desc(euclidean_similarity)) %>%  # Sort in descending order
  slice(1:10)
print(top10neg_eu)
```

```
##          text euclidean_similarity
## text1   text1             5.656854
## text4   text4             5.477226
## text60 text60             5.196152
## text76 text76             5.099020
## text13 text13             5.000000
## text75 text75             5.000000
## text85 text85             5.000000
## text10 text10             4.898979
## text40 text40             4.898979
## text64 text64             4.898979
```

```
# Combine cosine and Euclidean similarity results for positive reference
top10pos_combined <- bind_rows(
  top10pos_co %>% mutate(similarity_type = "cosine"),
  top10pos_eu %>% mutate(similarity_type = "euclidean")
)
print(top10pos_combined)
```

```
##          text cosine_similarity similarity_type euclidean_similarity
## text1   text1        1.00000000          cosine                   NA
## text97 text97        0.22613351          cosine                   NA
## text7   text7        0.19284730          cosine                   NA
## text56 text56        0.17739372          cosine                   NA
## text31 text31        0.07106691          cosine                   NA
## text15 text15        0.06741999          cosine                   NA
## text70 text70        0.06154575          cosine                   NA
## text98 text98        0.06154575          cosine                   NA
## text2   text2        0.00000000          cosine                   NA
## text3   text3        0.00000000          cosine                   NA
## text4   text4                NA       euclidean             6.480741
## text60 text60                NA       euclidean             6.244998
## text76 text76                NA       euclidean             6.164414
## text13 text13                NA       euclidean             6.082763
## text75 text75                NA       euclidean             6.082763
## text85 text85                NA       euclidean             6.082763
## text10 text10                NA       euclidean             6.000000
## text40 text40                NA       euclidean             6.000000
## text64 text64                NA       euclidean             6.000000
## text68 text68                NA       euclidean             6.000000
```

```
# Combine cosine and Euclidean similarity results for negative reference
top10neg_combined <- bind_rows(
  top10neg_co %>% mutate(similarity_type = "cosine"),
  top10neg_eu %>% mutate(similarity_type = "euclidean")
)
print(top10neg_combined)
```

```
##                text cosine_similarity similarity_type euclidean_similarity
## text100    text100        1.00000000          cosine                   NA
## text27      text27        0.19069252          cosine                   NA
## text29      text29        0.18257419          cosine                   NA
## text44      text44        0.17541160          cosine                   NA
## text77      text77        0.10540926          cosine                   NA
## text16      text16        0.10000000          cosine                   NA
## text73      text73        0.09128709          cosine                   NA
## text98      text98        0.09128709          cosine                   NA
## text68      text68        0.08451543          cosine                   NA
## text1...10   text1        0.00000000          cosine                   NA
## text1...11   text1                NA       euclidean             5.656854
## text4        text4                NA       euclidean             5.477226
## text60      text60                NA       euclidean             5.196152
## text76      text76                NA       euclidean             5.099020
## text13      text13                NA       euclidean             5.000000
## text75      text75                NA       euclidean             5.000000
## text85      text85                NA       euclidean             5.000000
## text10      text10                NA       euclidean             4.898979
## text40      text40                NA       euclidean             4.898979
## text64      text64                NA       euclidean             4.898979
```

Using **Euclidean similarity** makes the results totally different, as it measures straight-line distance between two points in space. It focuses on the absolute size of text content in a document.

Q 6.3

I qualitatively analyzed the retrieved documents by examining their **top features** or **highest TF-IDF terms** to understand their most representative words. Additionally, I used **keyword-in-context (KWIC) analysis** to see how these terms appear within the text. Based on this qualitative assessment, I evaluated whether my sentiment dictionary accurately captures the intended themes or if adjustments are necessary.

```
# Select the top 10 texts based on the highest cosine similarity scores
top10_texts_pos <- cosine_results_pos %>%
  arrange(desc(cosine_similarity)) %>% # Sort the results in descending order of cosine
similarity
  slice(1:10) %>% # Select the top 10 entries
  pull(text)  # Extract text IDs

# Convert the merged_dem_vader dataframe into a corpus for text analysis
# dem_vader_corpus <- sorted_sentiment %>%
#  corpus()

# Subset the corpus to include only the documents that match the top 10 (most positive)
text IDs
subset_corpus_vadar <- corpus_subset(vader_clean, docnames(vader_clean) %in% top10_texts
_pos)

tfdif_vader_pos <- subset_corpus_vadar %>%
  tokens(remove_punct = TRUE, remove_numbers = TRUE) %>%
    tokens_tolower() %>% #
    tokens_remove(c(stopwords("en"), "madam", "mr", "today","rt")) %>%
    tokens_remove(pattern = "#*|@*") %>%
    tokens_remove(pattern = "^https://.*", valuetype = "regex") %>%
    tokens_remove(c(":", "/")) %>%
    tokens_remove(c("amp","$")) %>%
    tokens_wordstem() %>%
  dfm() %>%
  dfm_tfidf(scheme_tf = "prop", scheme_df = "inversemax")

# Create a Document-Feature Matrix (DFM) and apply TF-IDF weighting
# tfdif_vader_pos <- dfm(tokens(subset_corpus_vadar)) %>%
#  dfm_tfidf(scheme_tf = "prop", scheme_df = "inversemax")

# Convert the TF-IDF weighted DFM into a data frame
tfidf_df_pos <- convert(tfdif_vader_pos, to = "data.frame")

# Reshape and sort the top terms by TF-IDF score
top_tfidf_terms <- tfidf_df_pos  %>%
  pivot_longer(-doc_id, names_to = "term", values_to = "tfidf") %>%
  arrange(desc(tfidf)) %>%
  slice(1:10)  # Select the 10 highest TF-IDF terms
print(top_tfidf_terms)
```

```
## # A tibble: 10 × 3
##    doc_id term      tfidf
##    <chr>  <chr>     <dbl>
##  1 text3  time     0.120
##  2 text1  serv     0.0860
##  3 text97 heart    0.0753
##  4 text97 survivor 0.0753
##  5 text97 rememb   0.0753
##  6 text97 kill     0.0753
##  7 text97 night    0.0753
##  8 text2  happi    0.0669
##  9 text2  help     0.0669
## 10 text2  tommi    0.0669
```

```
# Define key words to search for in the text
patterns <- c("time","serve","heart","survivor","remember")

# Tokenize the first 10 texts from the merged_dem_vader dataframe
vader_tokens_toppos <- sorted_sentiment %>%
  slice(1:10) %>%
  corpus(text_field = "text") %>%
  tokens()

# Perform keyword-in-context (KWIC) search for the defined patterns within a window of 5
words
kwic_results_pos <- kwic(vader_tokens_toppos, pattern = patterns, window = 5)
print(kwic_results_pos)
```

```
## Keyword-in-context with 5 matches.
##   [text1, 5]                “ We do not | serve | one party or one interest
##  [text1, 14]       one interest, but we | serve | one nation. ” I’m
##   [text3, 3]                  Had great | time  | @memgrizz victory in DC over
##  [text3, 17]        . 4th quarter was Ja | time  | @JaMorant. Lots of…
##  [text5, 12] clear we’re running out of | time  | to address the#ClimateCrisis.
```

I think sentiment analysis here using Vader is a bit weird, as I looked at those key words and found out they are actually unrelated to sentiment. Those are more thematic words based on context, since, for example, serve shows strong thematic relevance like public service or civic duty. Therefore, if I would chnage my dictionary, I will change a topic dictionary.