Text analysis of 2024 US Presidential convention speeches

Posted on August 25, 2024 by **Jerry Tuttle** in **R bloggers** | 0 Comments

Text analysis of 2024 US Presidential convention speeches, by Jerry Tuttle

Can an actuary / mathematician / data analyst say anything objective and data-oriented about the 2024 US presidential campaign?

Yes, if I confine my remarks to a numerical text analysis of the candidates' words, rather than attempt to comment on the candidates' political and economic views. This analysis is based solely on data that I collected from the two convention speeches.

I performed a sentiment analysis of the candidates' emotional words. Here is a graphical summary, discussion to follow:

Sentiment analysis

I also calculated the candidates' most frequent words, excluding stop words like "a", "and", "the" and so on that don't carry much useful information:
Most frequent words

  I calculated statistics of things like average sentence length and average word length:
Summary statistics

This blog is my attempt at some objective and data-oriented analysis of the two presidential candidates from their convention sppeches. This is a text analysis of their speeches. Text analysis is a branch of data analysis that takes unstructured text documents, breaks them into sentences and words, and attempts to find some meaning. Text analysis is used by marketers to evaluate customer opinions, by police departments to monitor potential or actual criminal activites, by researchers to evaluate things like whether Shakespeare really wrote all the plays attributed to him and which Beatles songs were mostly written by John versus which by Paul, and by many others.

This is intended as an objective analysis. I am not trying to make either candidate look good, or bad. I have used these same text analysis techniques in other projects such as analyzing *Hamlet*, analying short stories, and analyzing the Twitter tweets of a radio talk show host.

For each of Trump and Harris, I started with a transcript of their convention sppeches. I believe the transcripts are their spoken transcripts, not their written transcripts, based on their very first few sentences. I used various computer packages in R such as tm and tidytext to tokenize the documents into individual sentences, words, and characters. I was guided by the works of Silge and Robinson in *Text Mining with R* and Williams in *Data Science Desktop Survival Guide*.

A summary and a comparison of the of the tokenization of the speeches is the following, repeated from before.

Recent Posts

Text analysis of 2024 US Presidential convention speeches
Top 25 R Packages (You Need To Learn In 2024)
biorecap: an R package for summarizing bioRxiv preprints with a local LLM
Adding Subtitles in ggplot2
Add Footnote to ggplot2
Correlation By Group in R
shortuuid: Generate and Translate Standard UUIDs
Polar-centred maps by @ellis2013nz
Out-of-sample Imputation with {missRanger}
SQL of the Rings: One Language to Query Them All (with R)
Unlocking the Power of the Linux Shell
Best Books on Generative AI
Comparing Many Models: Crude Oil Futures Price
Understanding the main() Function in C
armadillo 0.3.0 is available on CRAN

Obviously Trump spoke for much longer than Harris, so he had many more total words. Harris had a larger average sentence length (18.06 versus 10.41), but the two were close in average word length and average number of syllables per word. Their numerical Flesch scores were about equal – this is a measure of the grade level required to understand the text. Harris had a larger percentage of unique or different words in her speech than Trump (22% to 12%). Trump had a larger percentage of negative words than Harris (43% to 37%), which I will discuss below.

Sentiment analysis ia analyzing text to determine its emotional tone. Linguistics experts have built dictionaries that associate a large list of words with eight emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, and trust) and also negative and positive sentiments. For example, the word “ache” is associated with sadness. Some words have more than one emotion; for example “admirable” is associated with both joy and trust. Further, some words have either negative or positive associations.

There are some limitations in sentiment analysis. The sentiment dictionary does not recognize sarcasm, and I am limiting my analysis to single words so I am not picking up negation (such as “not expensive”) or other instances where the emotion requires several words. A conclusion from the sentiment distribution graph is that the candidates are surprisingly similar in most of these emotions. The biggest differences are that Trump has a greater portion of his words categorized as anger and negative than Harris has.

Most frequent positive and megative words

Trump's frequent positive words include "love", adjectives "beautiful" and "wonderful", and also "money". His negative words include "tax", "invasion", "inflation", "war", "illegal", and "crime".

Harris's frequent positive words include "freedom", "love", "opportunity", and "forward". Her negative words include "tax", "enforcement", "abuse", and "abortion". Interestingly "mother" is both a positive word and a negative word, as is "vote". (I rechecked the sentiment dictionary to confirm these.)

Trump has a larger percentage of negative words (negative divided by positive plus negative) than Harris (43% to 37%). These

positive and negative lists seem consistent with my memory of their speeches.

Most frequent words

Word frequency provides some clue on what is important to each candidate. Not surprisingly, both candidates frequently used words like “America”, “Americans”, “country”, “nation”, and “border”. Trump used adjectives like “beautiful” and “incredible”. Harris spent considerable time talking about Trump and also about her mother. Harris frequently used “middle”, “class”, “women”, and “law”.
Distribution of word sizes

Distribution of syllables per word

The distributions of size of words and of syllables per word are pretty similar for both candidtaes.

Final thoughts

It is hard to be indifferent about the 2024 US presidential election. You have your opinion, and I have mine. Much of what the candidates say is their opinion, or their plan if elected, and these things are not things we can verify.

Some things the candidates say as if they are facts, are stated in a way that is open to interpretation. A good example is that “You are better (or worse) off financially today than four years ago.” I can choose one measure, collect some data, and show I am better off; or I can choose a very different measure, collect some data, and show I am worse off.

Some things the candidates say as facts ARE verifiable. I am in no position to do such verifying, but a number of third parties do this. Here are a few links. I can not vouch for their reliability or bias.

- AP News
- FACT Check
- Reuters
- Snopes
- Verify This

Text analysis is a popular data analysis technique, and I hope I have added some insight into the candidates in an objective manner.

The following is my R code:

```
# Trump convention speech, July 19, 2024
# https://www.nytimes.com/2024/07/19/us/politics/trump-rnc-speech-transcript.html

# Harris convention speech, August 23, 2024
# https://singjupost.com/full-transcript-kamala-harriss-2024-dnc-speech/?singlepage=1

library(tidytext)
library(tm)
library(dplyr)
library(nsyllable)
library(SnowballC)
library(ggplot2)
library(forcats)
library(ggpubr)

speaker <- readline(prompt = "Enter Trump, or enter Harris: ")
if (speaker == “Trump” | speaker == “Harris”) print(speaker) else print(“Invalid input”)
trump_file <- "C:/Users/Jerry/Desktop/Harris_Trump/trump_convention_speech.txt"
harris_file <- "C:/Users/Jerry/Desktop/Harris_Trump/harris_convention_speech.txt"
```

```
textfile <- ifelse(speaker=="Trump", trump_file, harris_file)
textfile <- readLines(textfile)
text_df <- data.frame(line = 1:length(textfile), text=textfile)
names(text_df)[2] <- "text"

docs <- Corpus(VectorSource(text_df$text))
docs <- tm_map(docs, content_transformer(tolower))
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, removePunctuation, ucp=TRUE)
docs <- tm_map(docs, stripWhitespace)
inspect(docs[1:8])

custom_colors <- c("#1f77b4", "#ff7f0e", "#2ca02c", "#d62728",
"#9467bd",
    “#8c564b”, “#e377c2”, “#7f7f7f”, “#bcbd22”, “#17becf”,
    “#6a3d9a”, “#ff9e1b”, “#f6c6c7”, “#8dd3c7”, “#ffffb3”,
    “#bebada”, “#fb8072”, “#80b1d3”, “#fdb462”, “#b3e2cd”,
    “#ccebc5”)

common_theme <- theme(
    legend.position=”NULL”,
    plot.title = element_text(size=15, face=”bold”),
    plot.subtitle = element_text(size=12.5, face=”bold”),
    axis.title = element_text(size=15, face=”bold”),
    axis.text = element_text(size=15, face=”bold”),
    legend.title = element_text(size=15, face=”bold”),
    legend.text = element_text(size=15, face=”bold”))

docs_df <- data.frame(text = sapply(docs, as.character)) # Convert
Corpus to data.frame
wordfile <- unnest_tokens(docs_df, word, text, token = "words")
wordfile %>% count(word, sort=TRUE)
wordcountfile <- mutate(wordfile, numbchars = nchar(word)) #
characters per word
long1 <- wordcountfile[which(wordcountfile$numbchars ==
max(wordcountfile$numbchars)),1] # longest word long2 <-
wordcountfile[which(wordcountfile$numbchars ==
max(wordcountfile$numbchars)),2]
numberchars <- sum(wordcountfile$numbchars)
numberwords <- sum(count(wordcountfile, word, sort=TRUE)$n) #
no. words
avgcharperword <- round(numberchars / numberwords, digits=2)
sentencefile <- unnest_tokens(text_df, sentence, text, token =
"sentences")
sentencecount <- sum(count(sentencefile, sentence, sort=TRUE)$n)
avgwordpersent <- round(numberwords/sentencecount,2)
```

```
wordcountdist <- wordcountfile %>% count(numbchars)
wordcountdist$numbchars <- as.factor(wordcountdist$numbchars)
title <- paste(speaker, "- Distribution of Word Size")
subtitle <- paste("Longest word: ", long1, long2, "characters")
ggplot(wordcountdist, aes(numbchars, n, fill=numbchars)) +
    geom_bar(stat="identity", position = "dodge", width=0.5) +
    labs(title=title, subtitle=subtitle) +
    xlab("number of characters per word") + ylab("") +
    scale_fill_manual(values = custom_colors) +
    theme(legend.position = "none") +
    common_theme

syls <- nsyllable(wordfile$word, language = "en")
syls[which(wordfile$word == "jd")] <- 2 # used because nsyllable
generated error here
syls[which(wordfile$word == "nd")] <- 1 # used because nsyllable
generated error here
syls[which(wordfile$word == "st")] <- 3 # s/b 21st; used because
nsyllable generated error here
syls[which(wordfile$word == "gasolinepowered")] <- 5 # used
because nsyllable erred here
long2 <- min(syls[(syls == max(syls, na.rm=TRUE))], na.rm=TRUE)
w <- min(which(syls == long2))
long1 <- wordfile$word[w]
avgsylperword <- round(sum(syls)/numberwords, digits = 2)
avgsylperword
syls <- data.frame(syls) %>% count(syls)
syls$syls <- as.factor(syls$syls)
colnames(syls) <- c("syllables", "n")
title <- paste(speaker, "- Distribution of No. Syllables per Word")
subtitle <- paste("Most syllables: ", long1, long2, "syllables")
ggplot(syls, aes(syllables, n, fill = syllables)) +
    geom_bar(stat="identity", position = "dodge", width=0.5) +
    labs(title=title, subtitle=subtitle) +
    xlab("number of syllables per word") + ylab("") +
    scale_fill_manual(values = custom_colors) +
    theme(legend.position = "none") +
    common_theme

# Flesch-Kincaid reading ease formula
flesch <- round(206.835 - 1.015*(numberwords/sentencecount) -
84.6*(sum(syls$n)/numberwords),2) # Flesch reading ease
flesch
flesch_df <- data.frame(score = c(0,30,50,60,70,80,90,100),
    grade = c("College graduate","College","10th – 12th grade","8th –
```

```
9th grade”,
      “7th grade”,”6th grade”,”5th grade”,”below 5th grade”))

# Function to find the grade based on score; vlookup
find_grade <- function(score, flesch_df) {
  idx <- findInterval(score, flesch_df$score)
  if (idx == 0) {
   return(“below 5th grade”) # Handle case where score is below the
minimum
  } else {
   return(flesch_df$grade[idx])
  }
}

score_to_find <- flesch
flesch_grade <- find_grade(score_to_find, flesch_df)
flesch_grade

# delete stop words
docs_df <- data.frame(text = sapply(docs, as.character)) # Convert
Corpus to data.frame
wordfile <- unnest_tokens(docs_df, word, text, token = "words")
stop_words <- data.frame(tidytext::stop_words) # more words than
tm
my_stop_words <- data.frame(word = c("theyre", "hes", "dont",
   “didnt”,”youre”,”cant”, “im”,”whats”, “weve”, “theyve”, “youve”,
   “couldnt”, “wont”, “youd”))
wordfile <- anti_join(wordfile, stop_words)
wordfile <- anti_join(wordfile, my_stop_words)

wordfreq <- wordfile
wordfreq <- count(wordfreq, word, sort=TRUE) # word frequency
excl stop words
wordfreqdf <- data.frame(wordfreq)

unique_words <- nrow(wordfreq)
portion_unique_words <- round(unique_words / numberwords,
digits=2)
wordfreqdf20 <- wordfreqdf[1:21,] # Think about threshold
wordfreqdf20

graphtitle <- paste(speaker, "Word Frequency")
wordfreqdf20$word <- fct_reorder(wordfreqdf20$word,
wordfreqdf20$n, .desc = FALSE)
ggplot(data=wordfreqdf20, aes(x=word, y=n, fill=word)) +
     geom_bar(stat=”identity”, position = “dodge”, width=0.5) +
```

```
    coord_flip() +
    common_theme +
    xlab(“”) + ylab(“Frequency”) +
    ggtitle(graphtitle) +
    scale_fill_manual(values = custom_colors) +
    theme(legend.position = “none”)

# sentiments; note mother is both positive and negative!
df1 <- data.frame(wordfile)
colnames(df1) <- "word"
df2 <- get_sentiments("nrc")
df3 <- merge(x=df1, y=df2, by="word", all.x=TRUE,
stringsAsFactors=FALSE)
df3 <- subset(df3, !is.na(sentiment))
table(df3$sentiment)
w <- data.frame(table(df3$sentiment))
colnames(w) <- c("sentiment", "n")

sentiment_colors <- c(
  “Anger” = “red”,
  “Anticipation” = “green”,
  “Disgust” = “brown”,
  “Fear” = “purple”,
  “Joy” = “yellow”,
  “Negative” = “gray”,
  “Positive” = “lightblue”,
  “Sadness” = “blue”,
  “Surprise” = “pink”,
  “Trust” = “turquoise”)

title <- paste(speaker, "- Sentiment Plot")
ggplot(w, aes(sentiment, n)) +
    geom_bar(stat = “identity”, position = “dodge”, width = 0.5, fill =
sentiment_colors) +
    ggtitle(title) +
    ylab(“”) +
    common_theme +
    theme(axis.text.x = element_text(angle = 45, hjust=1))

df4 <- df3 %>% filter(sentiment == “positive” | sentiment ==
“negative”)
w <- with(df4, table(sentiment))
neg <- w[1]
pos <- w[2]
neg_ratio <- round(w[1] / (w[1] + w[2]), digits=2)
df5 <- df4 %>% group_by(sentiment) %>% count(word, sort=TRUE)
```

```
pos_freq <- df5 %>% filter(sentiment==”positive”) %>% top_n(10, wt =
n) %>% slice_head(n = 10)
neg_freq <- df5 %>% filter(sentiment==”negative”) %>% top_n(10, wt
= n) %>% slice_head(n = 10) # ties
pos_freq$word <- fct_reorder(pos_freq$word, pos_freq$n, .desc =
FALSE)
neg_freq$word <- fct_reorder(neg_freq$word, neg_freq$n, .desc =
FALSE)

title <- paste(speaker, "- Most Frequent Positive and Negative
Words")
p1 <- ggplot(pos_freq, aes(word, n)) +
    geom_bar(stat=”identity”, position = “dodge”, width=0.5,
fill=”darkgreen”) +
    ggtitle(“Positves”) +
    common_theme +
    xlab(“”) +
    coord_flip()
p2 <- ggplot(neg_freq, aes(word, n)) +
    geom_bar(stat=”identity”, position = “dodge”, width=0.5, fill=”red”)
+
    ggtitle(“Negatives”) +
    common_theme +
    xlab(“”) +
    coord_flip()
plot <- ggarrange(p1,p2, ncol=2, nrow=1, legend=NULL)
annotate_figure(plot, top = text_grob(title,
   color = “black”, face = “bold”, size = 14))

if (speaker == “Trump”){
  t <- data_frame(speaker, numberwords, avgwordpersent,
avgcharperword, avgsylperword, flesch, flesch_grade,
portion_unique_words, neg_ratio)
  print(t)
  } else {h <- data_frame(speaker, numberwords, avgwordpersent,
avgcharperword, avgsylperword, flesch, flesch_grade,
portion_unique_words, neg_ratio)
   conclusion <- data.frame(rbind(t,h))
   conclusion <- t(conclusion)
   colnames(conclusion) <- c("Trump", "Harris")
   conclusion <- conclusion[-1,]
   print(conclusion)
}
```

```
# the results of stemming and lemmatizing were not used in the
report
# stemming
wordfile <- wordfile %>%
   mutate(stem = wordStem(word)) %>%
   count(stem, sort = TRUE)

# lemmatize
df1 <- wordfile # df1 has col named stem
url <- "https://raw.githubusercontent.com/michmech/lemmatization-
lists/master/lemmatization-en.txt"
df2 <- read.table(url, header = FALSE, sep = "\t", quote = "",
stringsAsFactors = FALSE)
names(df2) <- c("stem", "word")
df3 <- merge(x = df1, y = df2, by = "stem", all.x = TRUE,
stringsAsFactors=FALSE)

# if word is not in dictionary, then leave word as is; otherwise, use
stemmed word.
df3$word <- ifelse(is.na(df3$word), df3$stem, df3$stem)

# End
```


← **Previous post**