Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

We were given a dataset containing several thousand news headlines from around the world in U.S English covering U.S news. This dataset identifies the type of news, whether the news source is real vs. news sources that are fake or propaganda. Given this dataset we will use Text Mining tools to identify differences that could help someone understand key differences between real or fake news headlines. In parallel, we also web scrapped Google Headlines on 5 separate days between April 3, 2021 and April 18, 2021 that we assume is a valid news coverage source, to assist in our Text Mining analysis.

A great place to start with Text Mining analysis is to create word clouds the visualize the top most used words in each data set. Below in Figure 1, we see the common words among our Google Headlines. Figure 2 shows the general word cloud for the given dataset, while Figure 3 breaks out what we consider real in that data set, and Figure 4 breaks out what we consider fake.

Figure 1 – Google Headlines

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

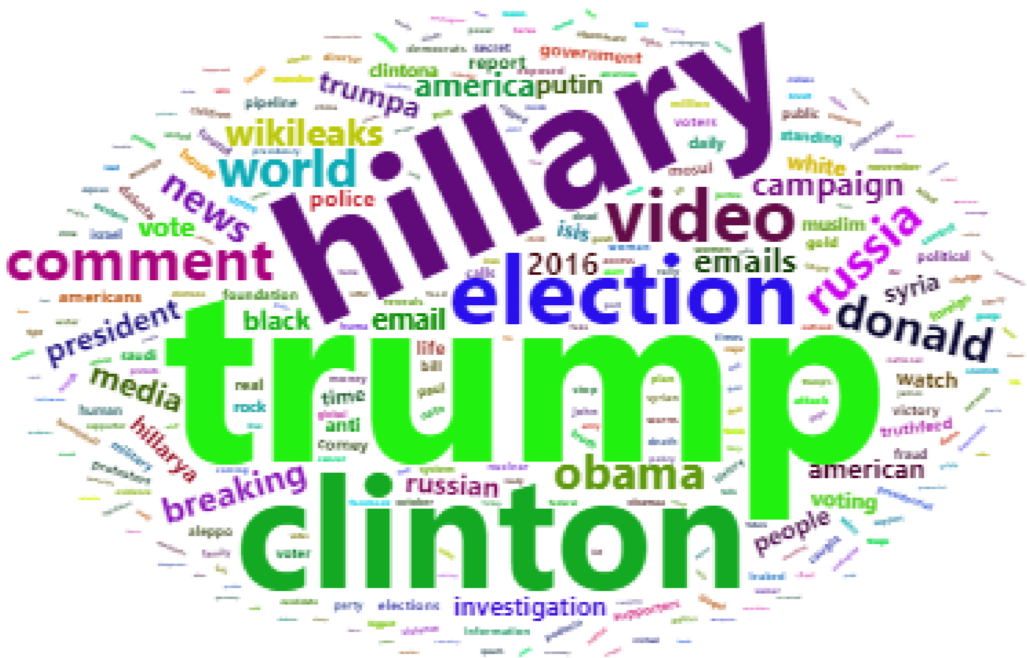

Figure 2 – Combined Given Dataset

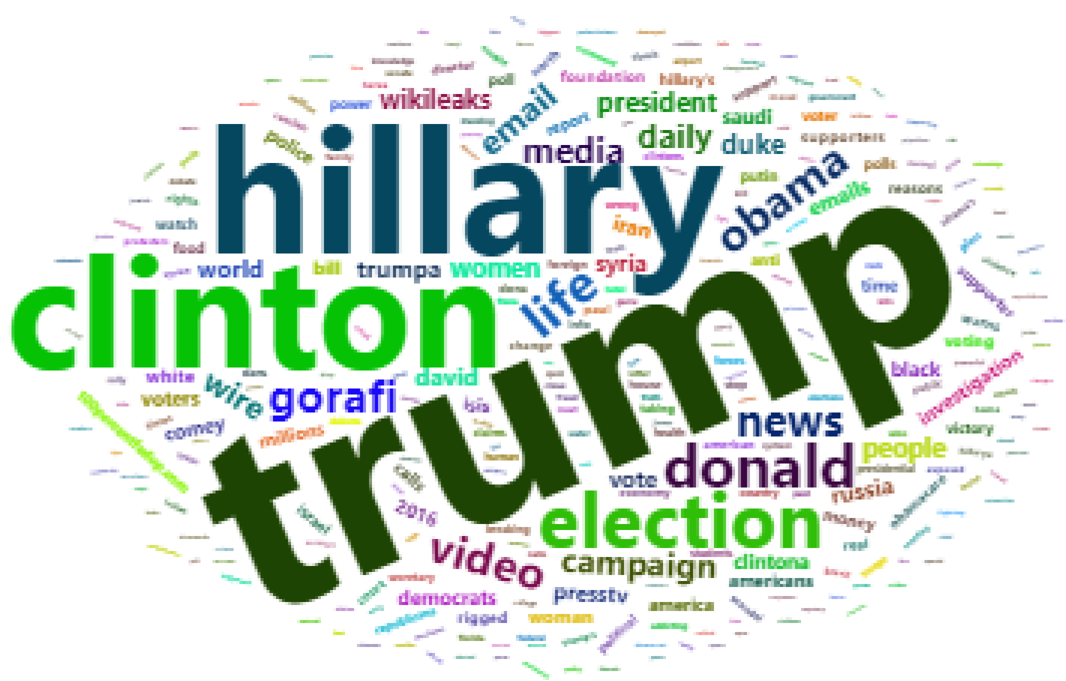

Figure 3 - Real Given Dataset

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

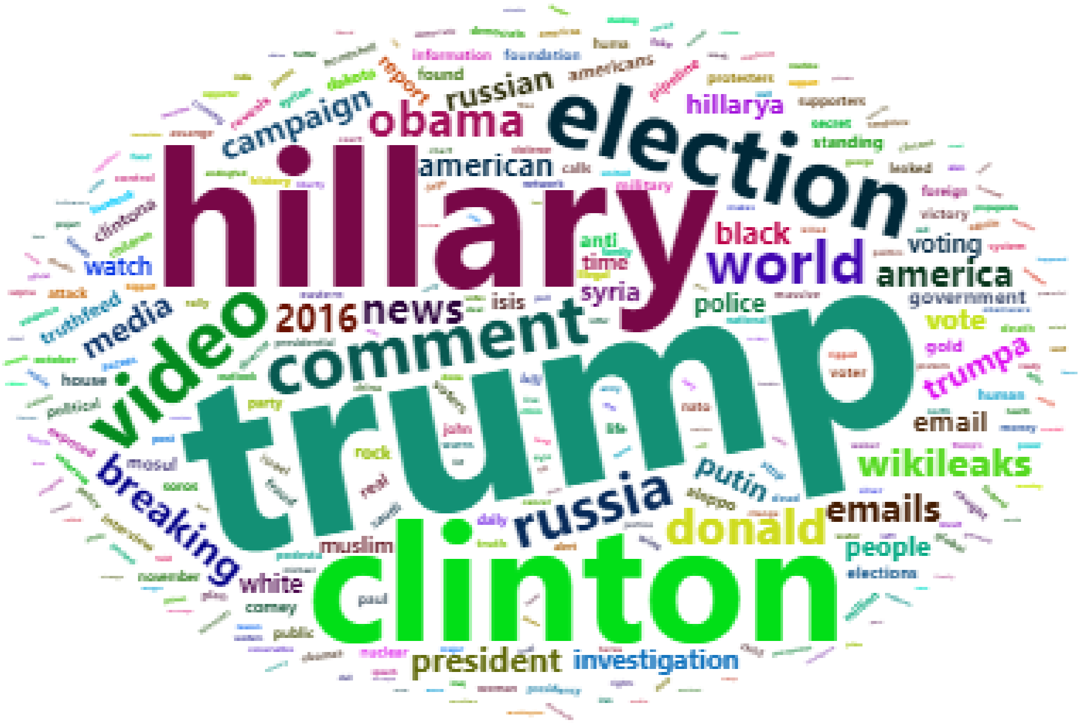

Figure 4 - Fake Given Dataset

Right away we can see that the Google Headlines we web scraped in April, 2021 are mostly about more recent events like the covid vaccine, civil rights trials, and mass-shootings. While, the dataset we were given must have been during the Hillary Clinton vs. Donald Trump U.S. presidential election. Another indicator is that the year 2016 appears in two of the three given dataset's word clouds.

We see similar trends in the three Figures below that visualize the Top 10 used words for each dataset. Figure 5 shows that 'police', 'covid', 'trial', and 'shooting' are most prevalent in the Google Headlines. Figure 6 and Figure 7, both display the words 'Trump', 'Hilary', 'Clinton', and 'election' are top frequent in the given dataset for both real and fake headlines. Some key differences are that the fake headlines focus on words like: 'Russia', 'comment', 'world', and 'breaking'. The real headlines use the words: 'life', 'media', 'email, and 'campaign'.

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

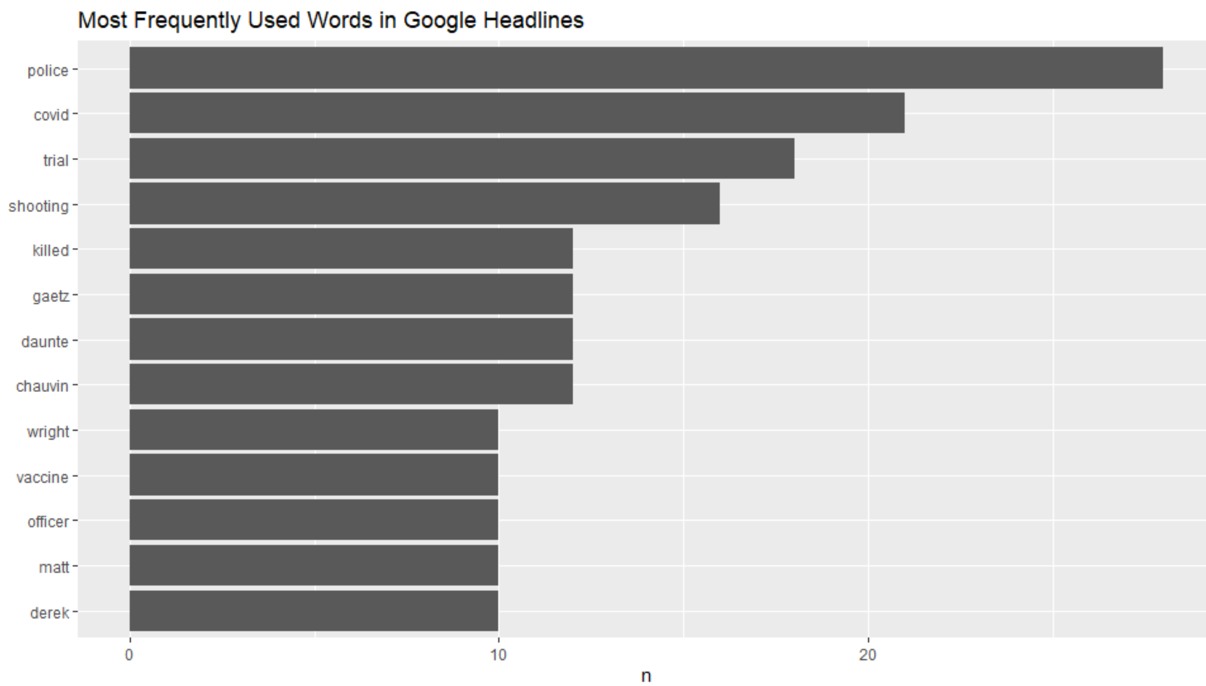

Figure 5 - Google Headlines

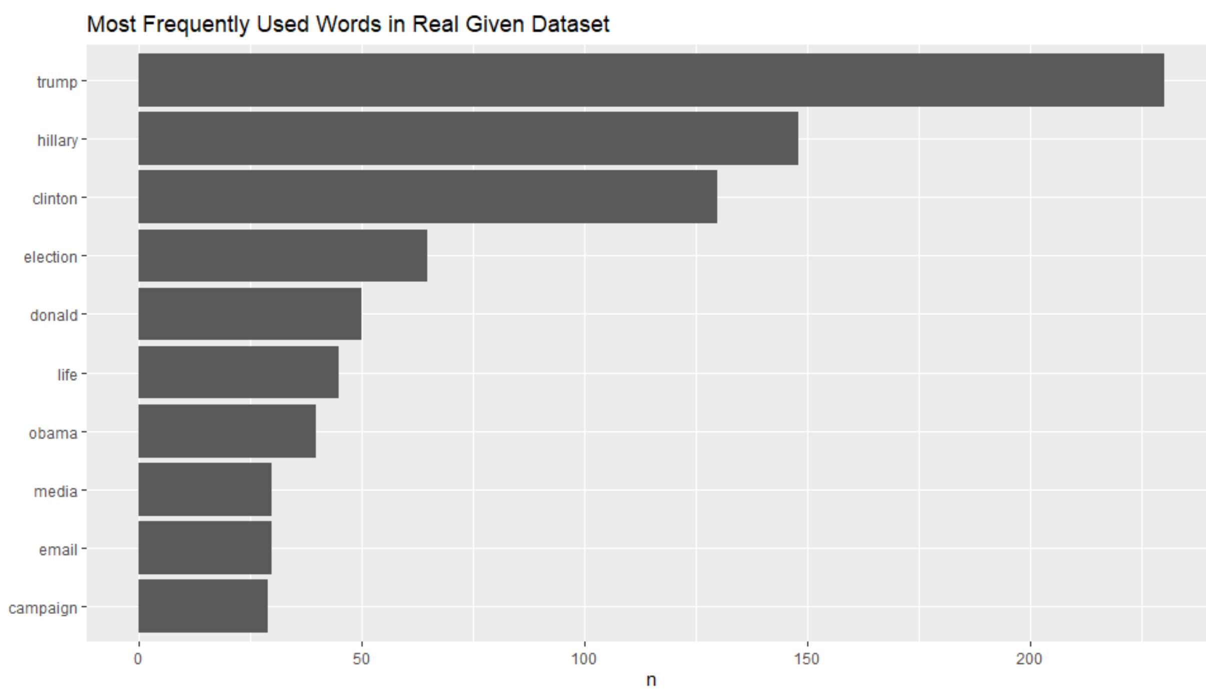

Figure 6 - Real Given Dataset

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

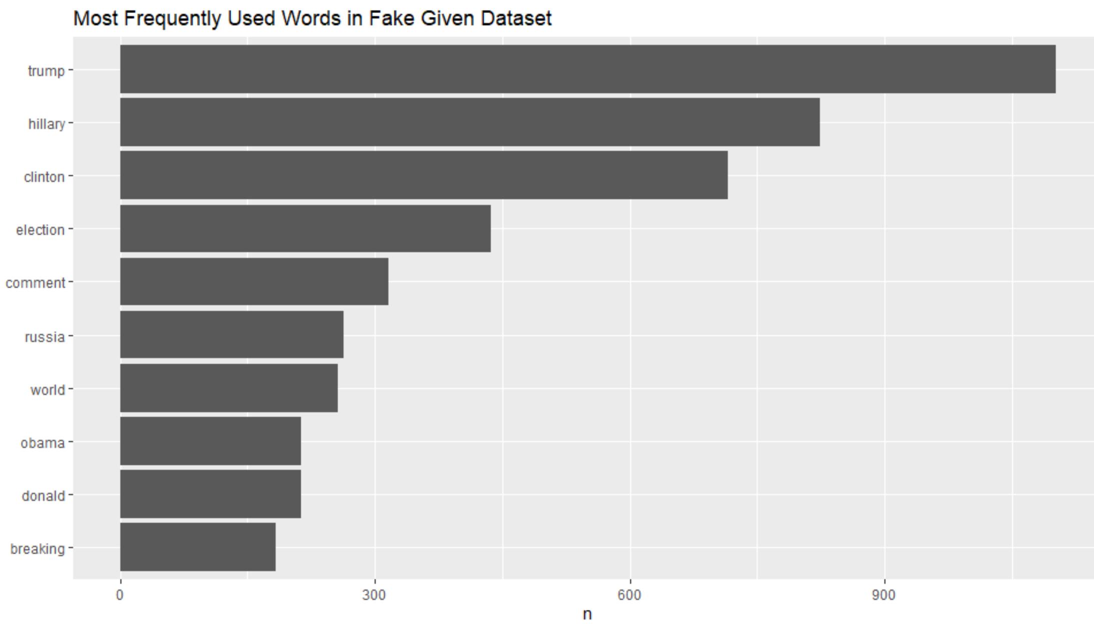

Figure 7 - Fake Given Dataset

Next, we visualize each dataset by analyzing universal parts of speech. All three datasets are very similar with the frequency of certain parts of speech used throughout the headlines. Below in Figure 8, 9 and 10, Google Headlines, the real given dataset, and the fake given dataset all used proper nouns, nouns, and adpositions the most. This makes sense because all of the headlines are written in English.

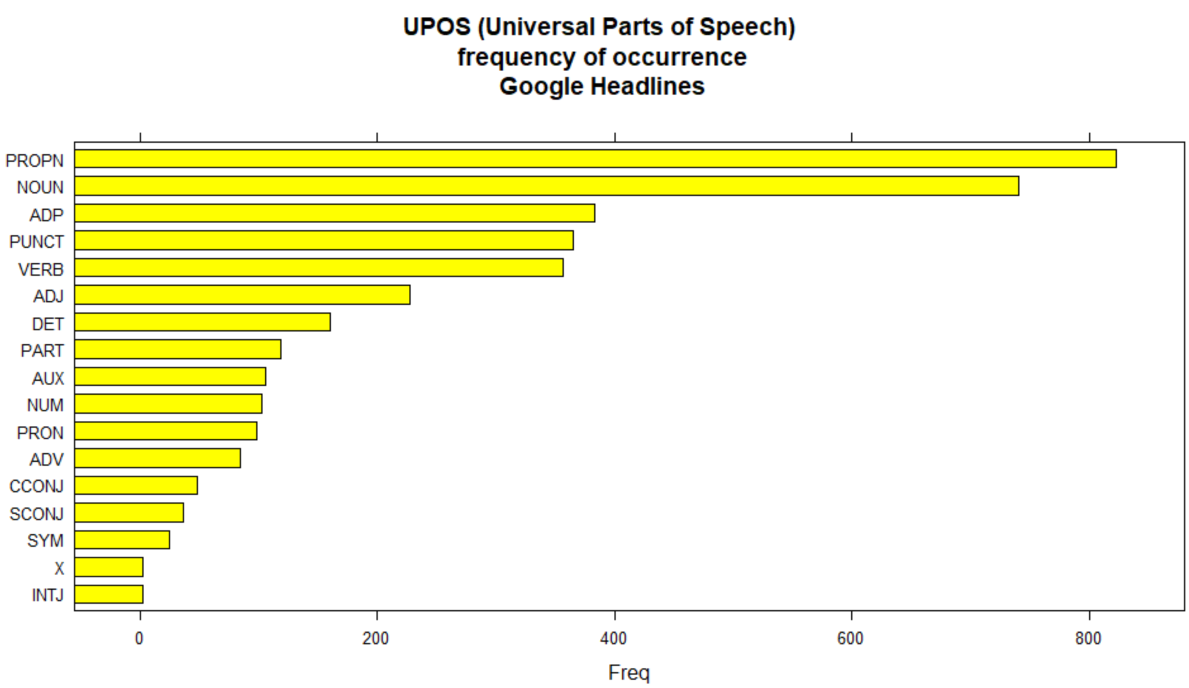

Figure 8 – Google Headlines

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

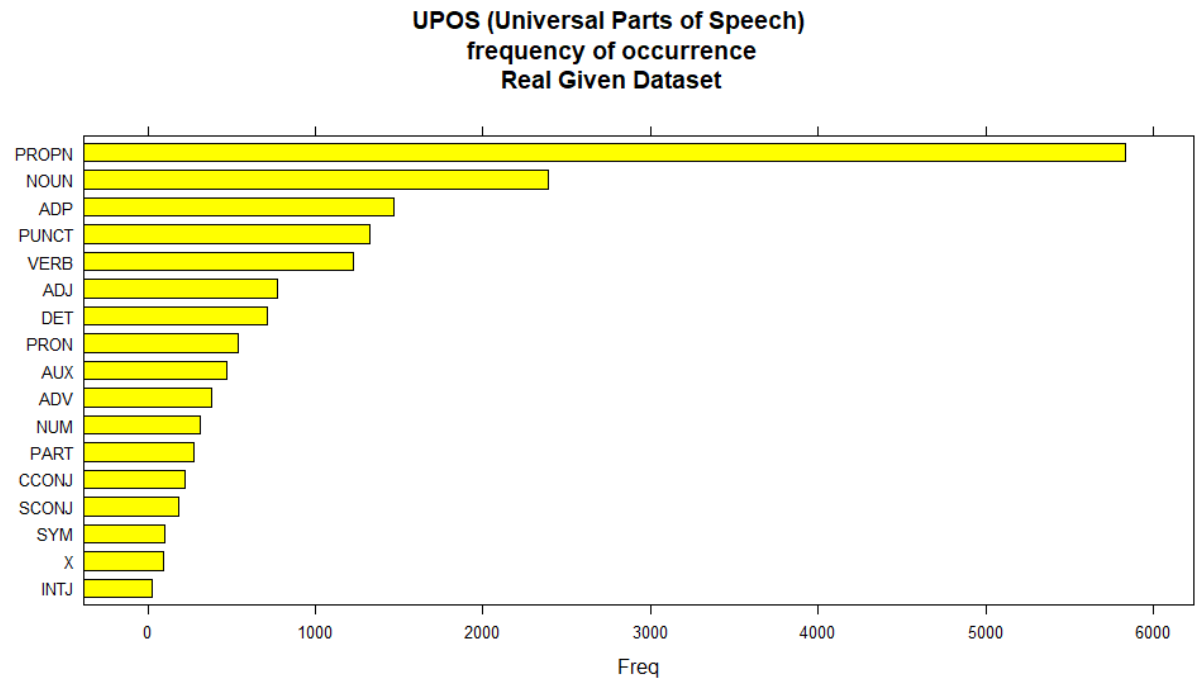

Figure 9 - Real Given Dataset

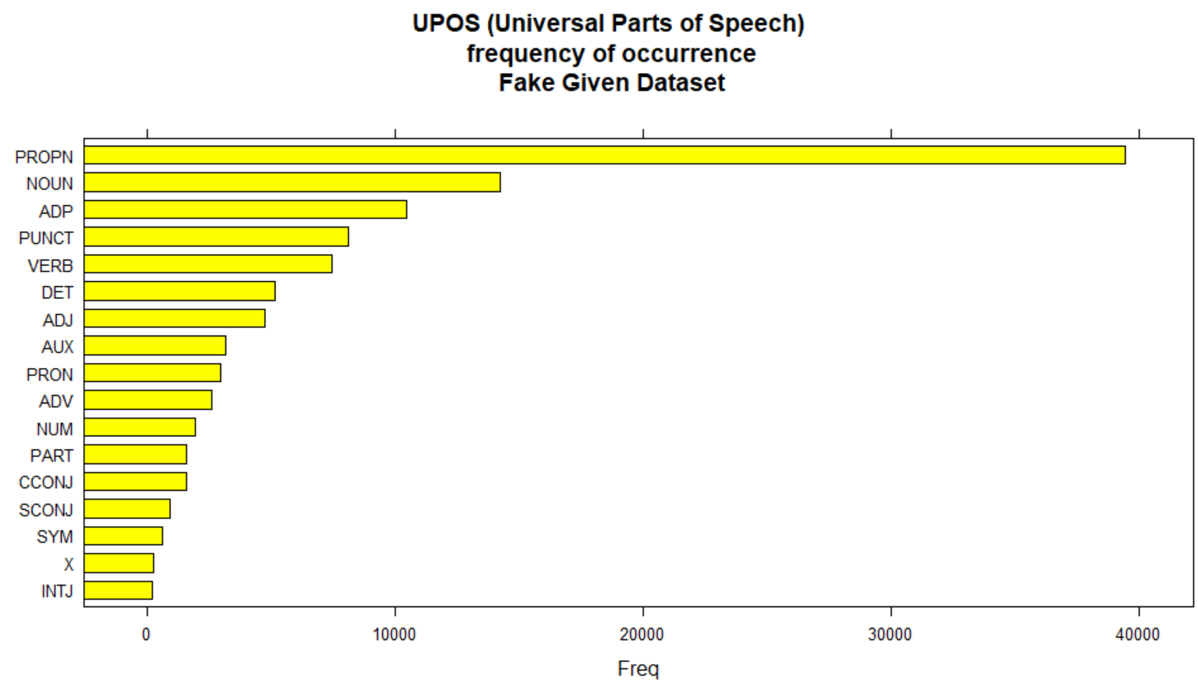

Figure 10 - Fake Given Dataset

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

If we dive into each of the most occurring nouns for each dataset, we can now begin to compare the overall context. The top four nouns found in the Google Headlines that we scraped are: 'police', 'COVID', 'officer', and 'trial' (Figure 11). In Figure 12, the top words used among the real given dataset are: 'Trump', 'Life', 'News, and 'Campaign'. In Figure 13, we see the most used words are: 'Comment', 'Trump', 'Campaign', and 'BREAKING'. The difference between the real vs. fake dataset are that the reputable headlines often use 'WATCH' as a noun, but the fake sources use the word 'BREAKING' as a noun.

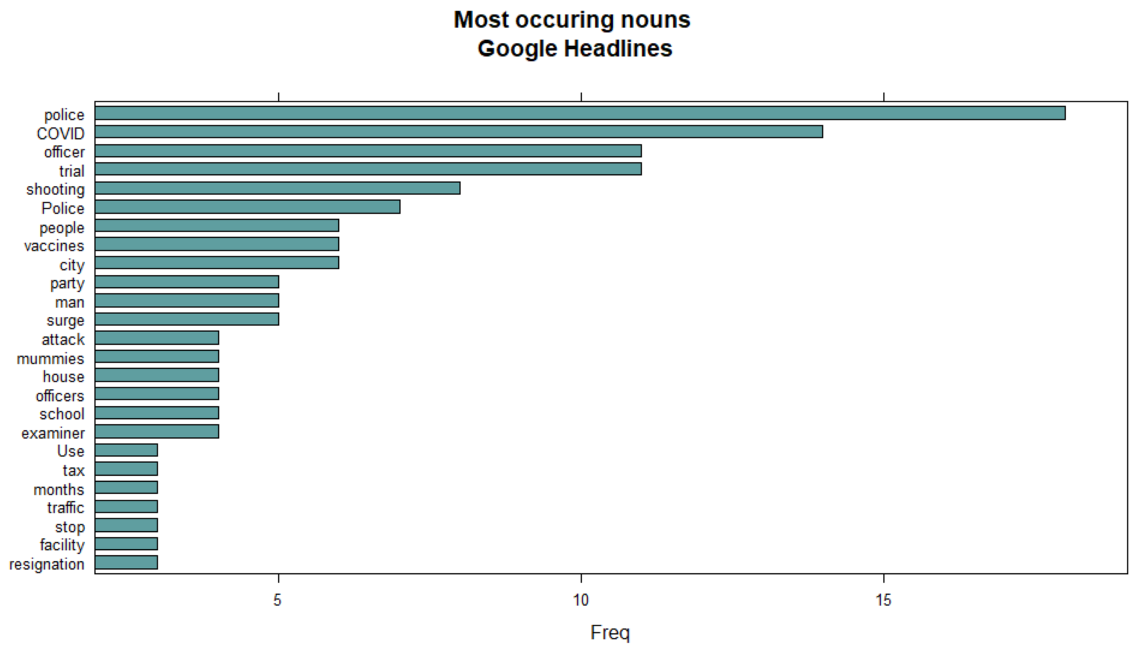

Figure 11 - Google Headlines

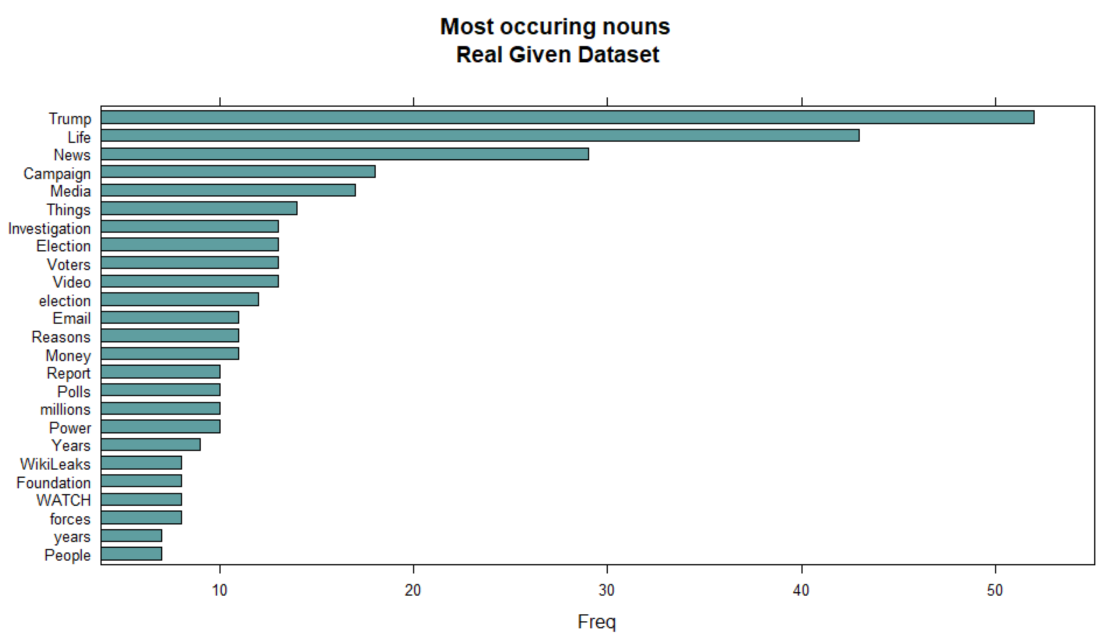

Figure 12 - Real Given Dataset

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

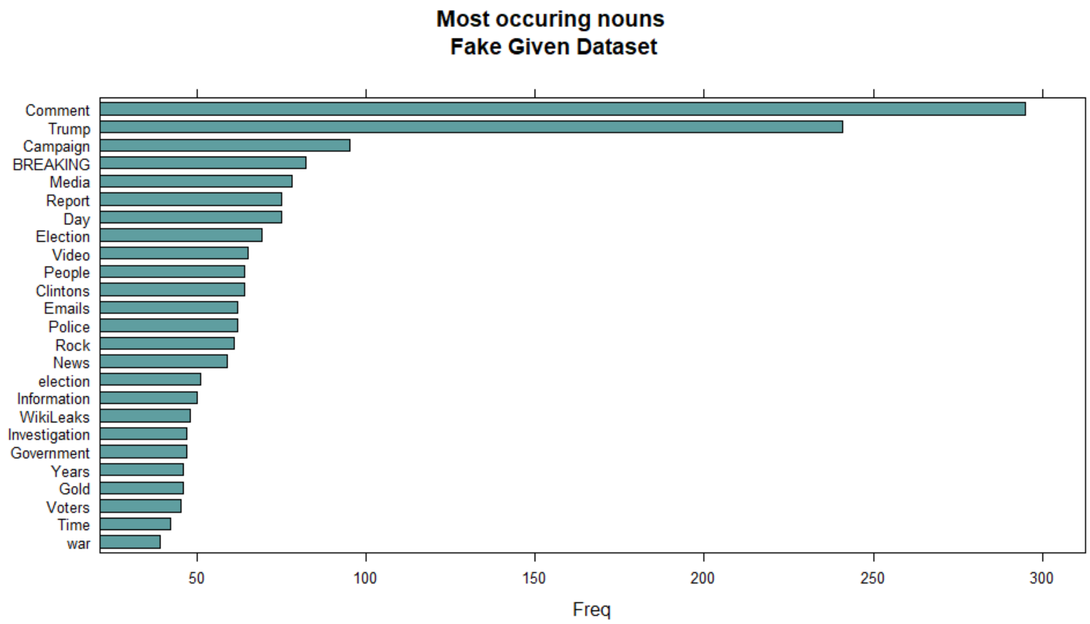

Figure 13 - Fake Given Dataset

In Figure 14-16 below, we see what happens if we pull out most occurring adjectives for each dataset. Google Headlines tend to use the adjectives: 'more', 'dead', 'ancient', and 'nuclear' frequently. The real given dataset uses adjectives: 'Daily', 'Hilary', 'Saudi', and 'Most' often. While, the fake given dataset uses: 'Russians', 'Americans', 'Most', and 'Muslim' as adjectives. It's definitely very interesting that the fake news focuses most on Russians in the headlines describing the U.S., before referring to actual Americans

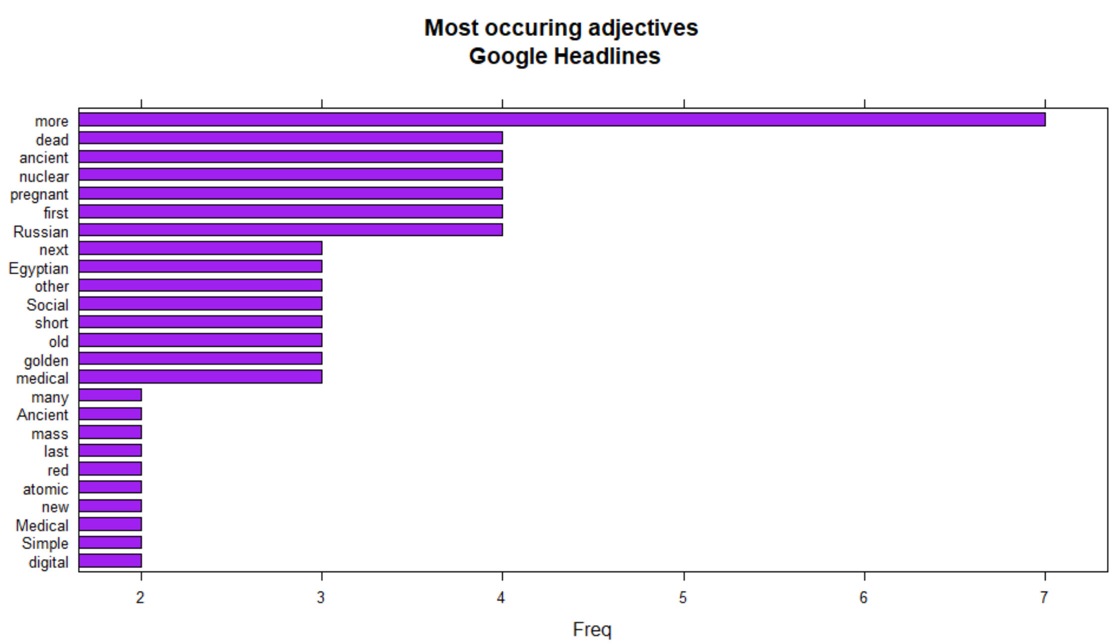

Figure 14 - Google Headlines

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

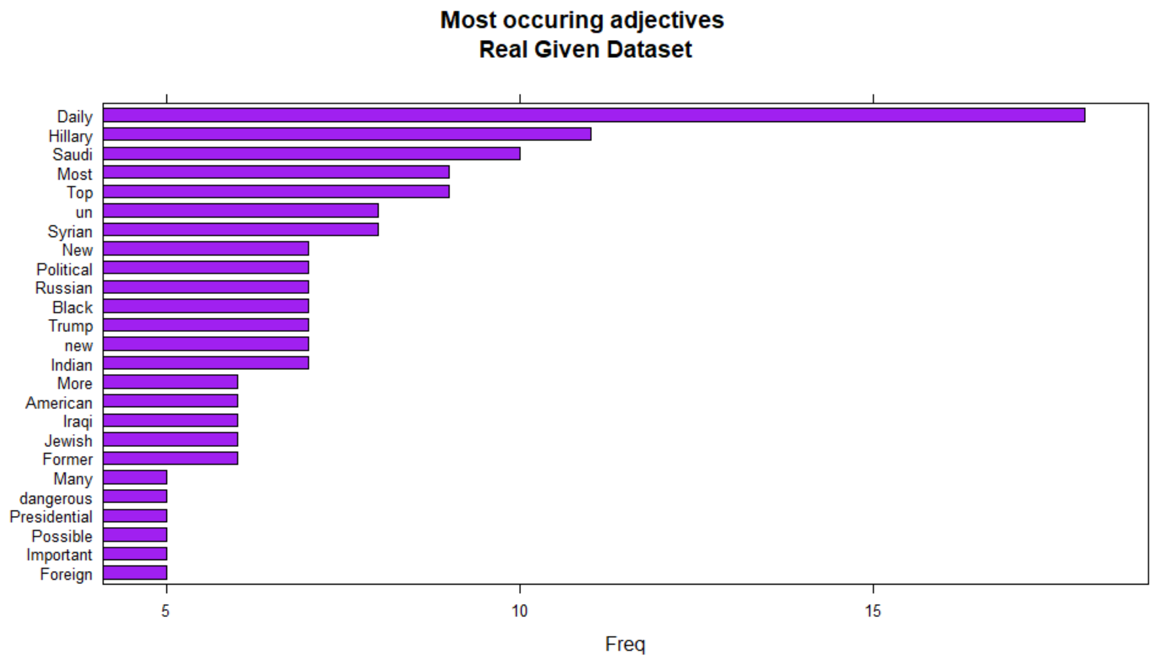

Figure 15 - Real Given Dataset

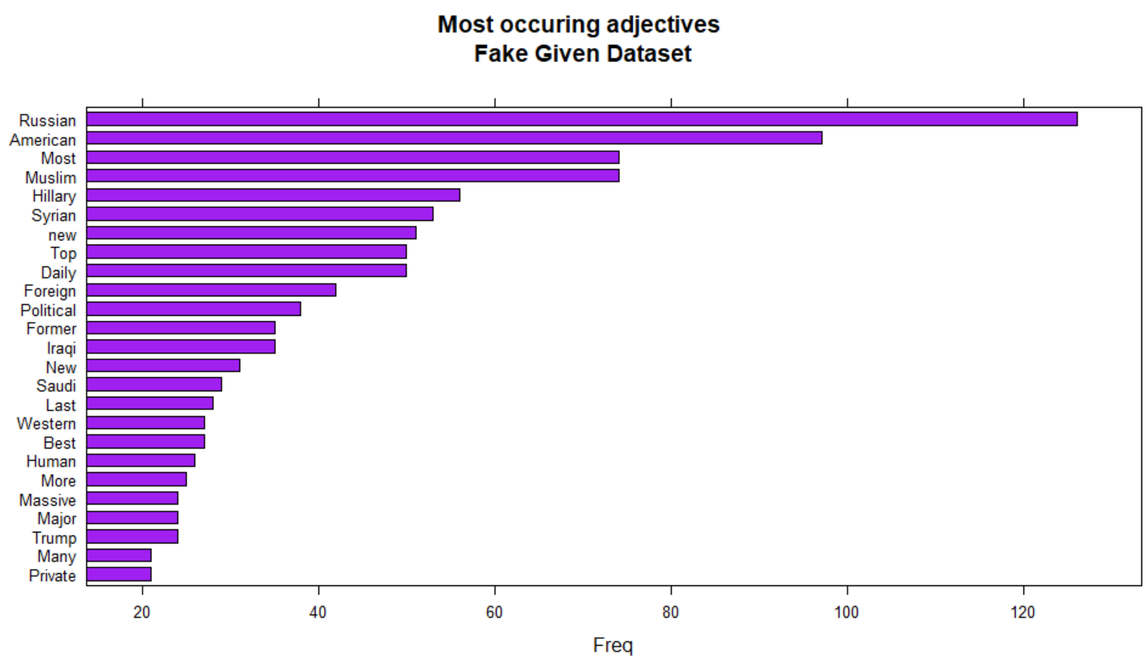

Figure 16 - Fake Given Dataset

Pulling out certain universal parts of speech is just a small step to fully understanding the context of the headlines in each of our datasets. We now apply a RAKE model, or Rapid Automatic Keyword Extraction Algorithm to the headlines. This will help us determine key phrases in body of texts by analyzing their frequency and co-occurrence with other words. Figure 17 below identifies the phrases ‘house party’ and, ‘police officer’. In Figure 18, the most frequent phrases used in the real given dataset is: ‘human rights’, and ‘daily wire’. The identified phrases in the fake give dataset are: ‘Muslim migrant’, ‘private server', 'daily mail', and 'Muslim colonizer'. The Google Headlines and the real given example datasets are relatively small and it's hard for the RAKE model to make complete sense of phrasing, we'll see a better example for those later. The fake phrases are quite alarming given that the word 'Muslim' is used in two of the top four phrases in the dataset.

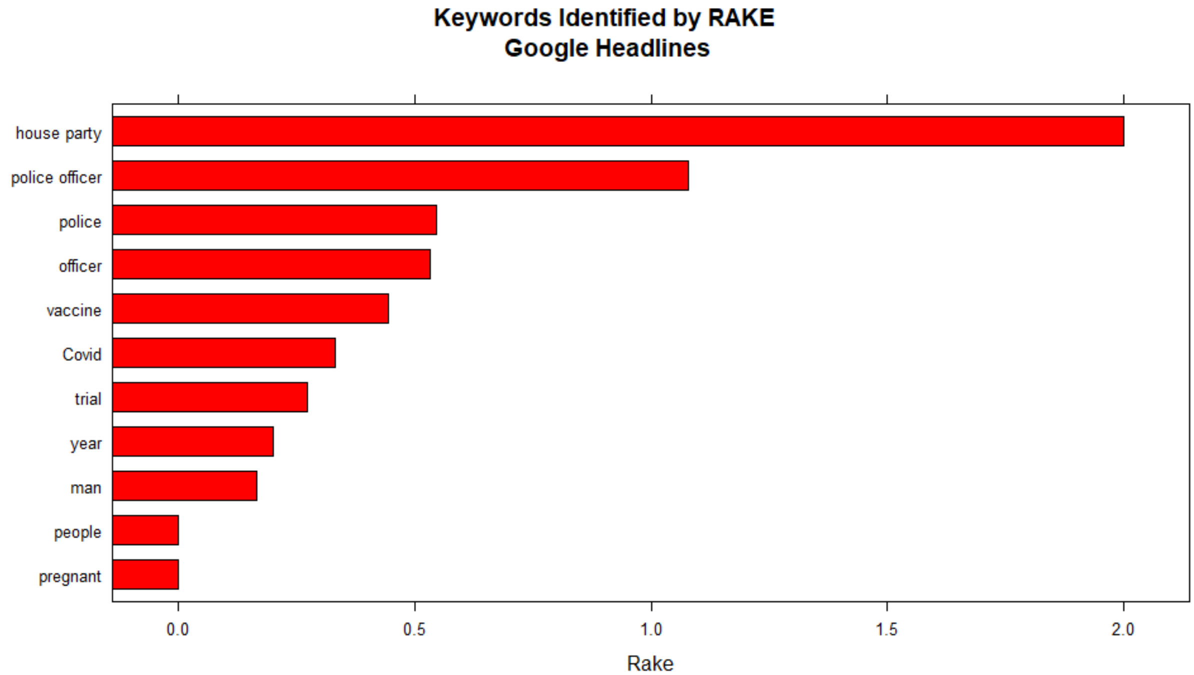

Figure 17 - Google Headlines

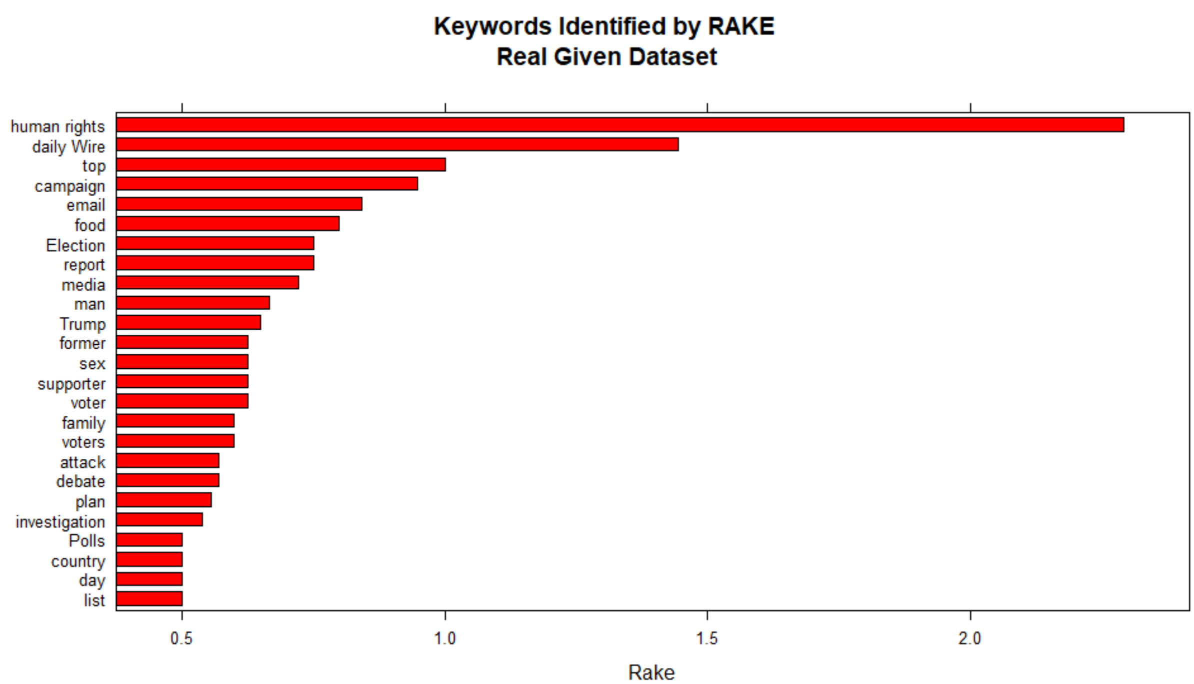

Figure 18 - Real Given Dataset

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

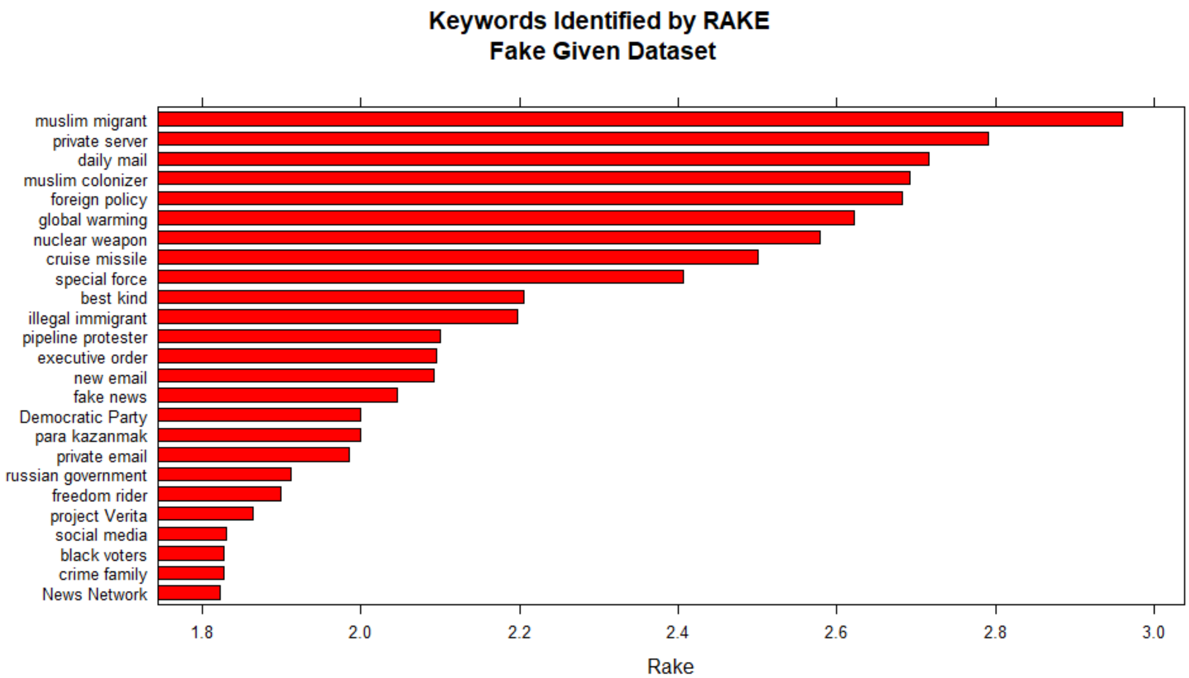

Figure 19 - Fake Given Dataset

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

Another similar approach to analyzing the context of each headline is by pulling out the frequent keywords, or simple noun phrases. This works well with the English language because most sentences have a noun or verb, plus a modifier that gives an idea of what the overall topic is. In Figure 20 below, the top simple noun phrases are: 'Daunte Wright', 'Matt Gaetz', 'Derek Chauvin', and 'Chauvin trial'. All of these people are key in the civil rights events that have been occurring during April, 2021. Figure 21 below, uses simple noun phrases like: 'Donald Trump', 'Hilary Clinton', 'Clinton campaign' among the real given dataset. The fake given dataset uses similar phrases to the real dataset, but also commonly uses: 'world war', and 'standing rock' (seen in Figure 23 below).

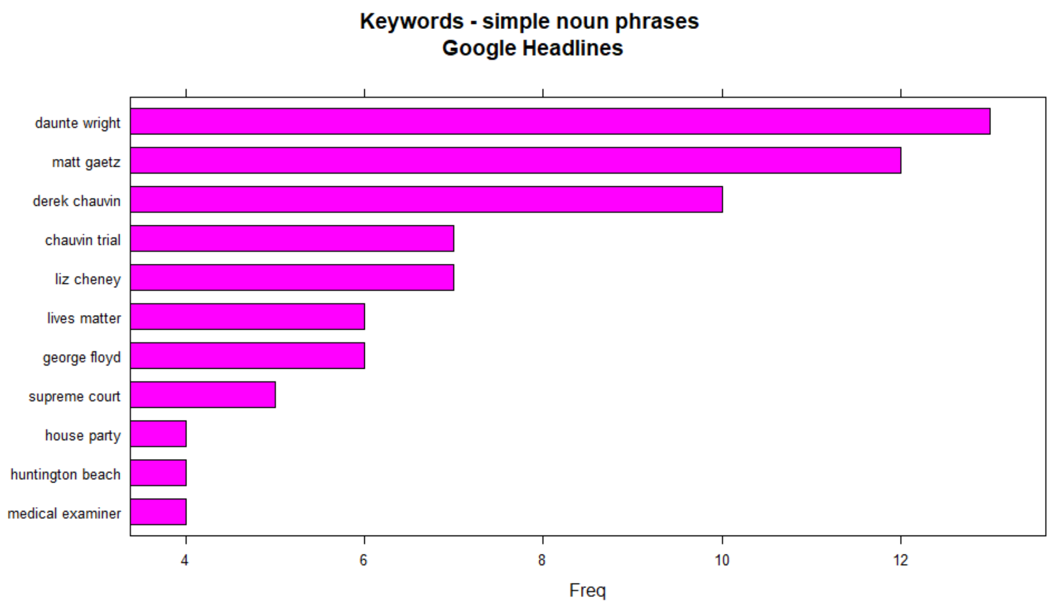

Figure 20 - Google Headlines

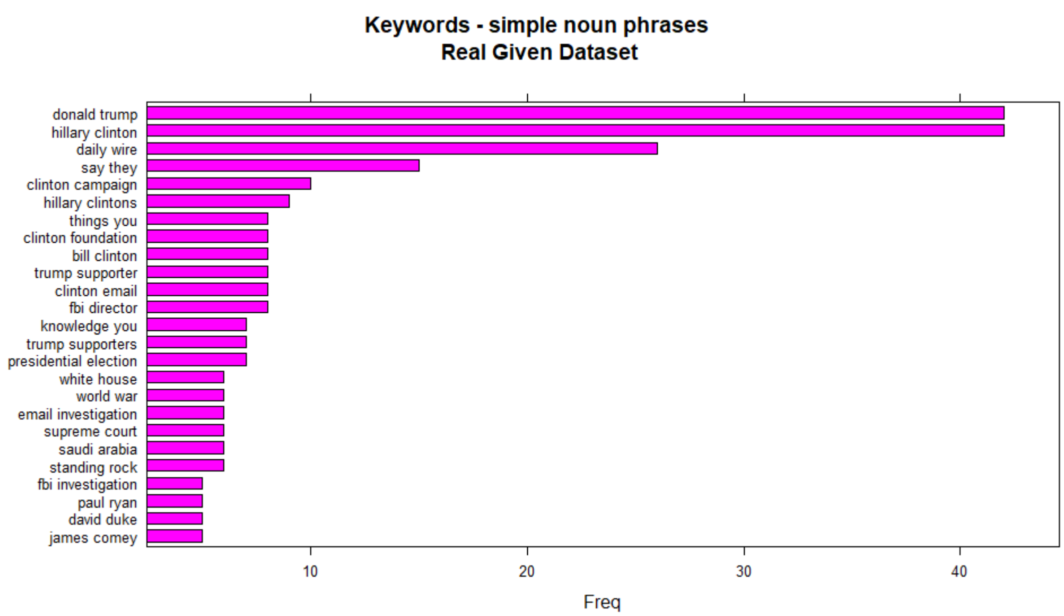

Figure 21 - Real Given Dataset

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

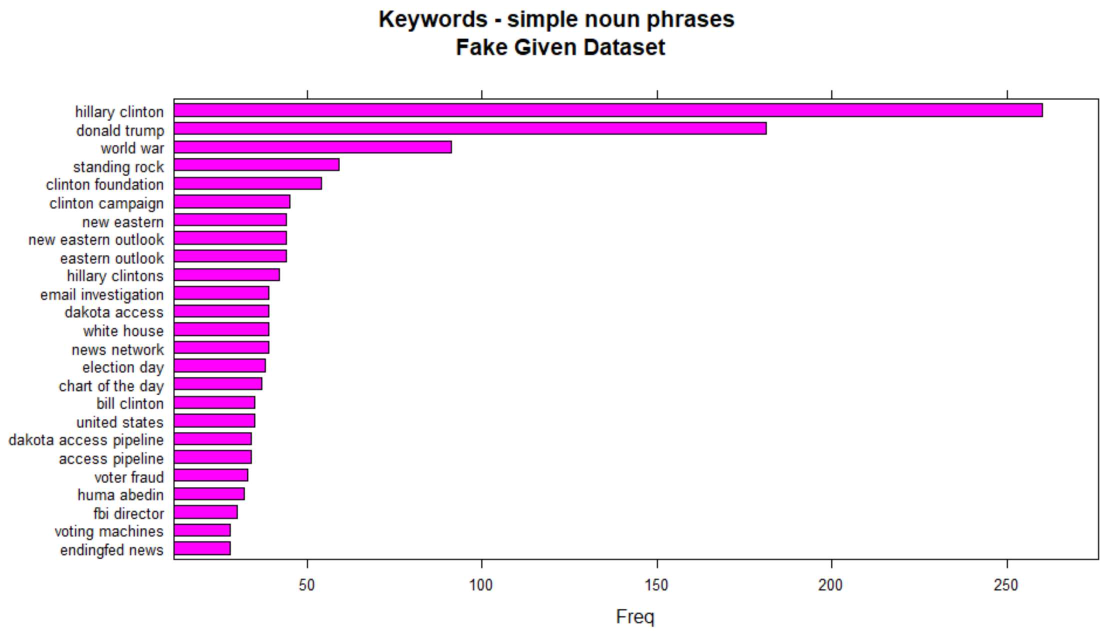

Figure 22 - Fake Given Dataset

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

To continue our co-occurrences of phrases across each dataset, we now create a word network of nouns and adjectives. This not only identifies phrases among headlines, it also visualizes how closely related each of these words are to one another. Figure 23 below displays words within 3 words distance in the Google Headlines, with the darker red connection lines indicting a stronger prevalence in the dataset. 'police officer', 'protein more vaccine covid surge', 'house party', and 'traffic stop' are frequent in the Google Headlines of April, 21. The real given dataset uses: 'human rights', 'private investigation email WikiLeaks', and 'Hilary campaign Trump administration supporter' (Figure 24). The fake given headlines in Figure 25, uses 'standing rock', 'freedom rider' and 'daily contrarian read' as words among each other.

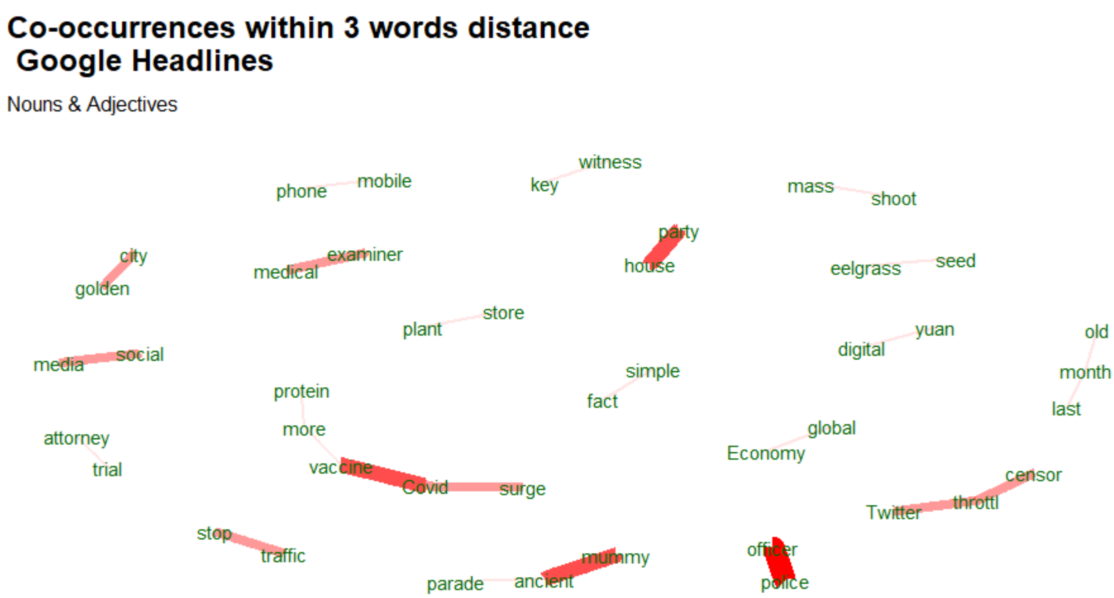

Figure 23 - Google Headlines

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

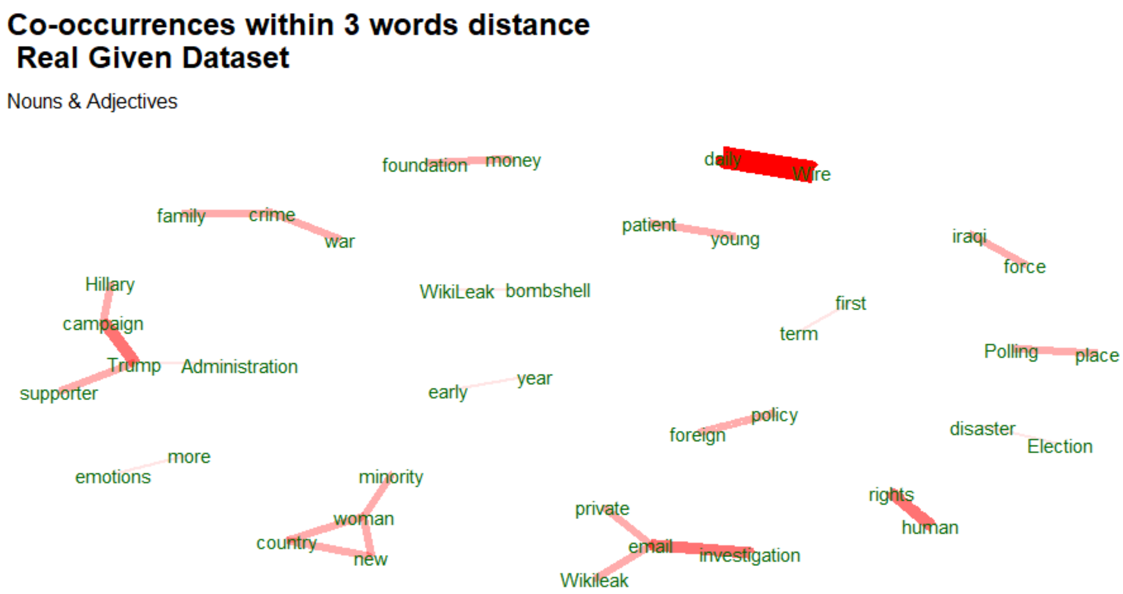

Figure 24 - Real Given Dataset

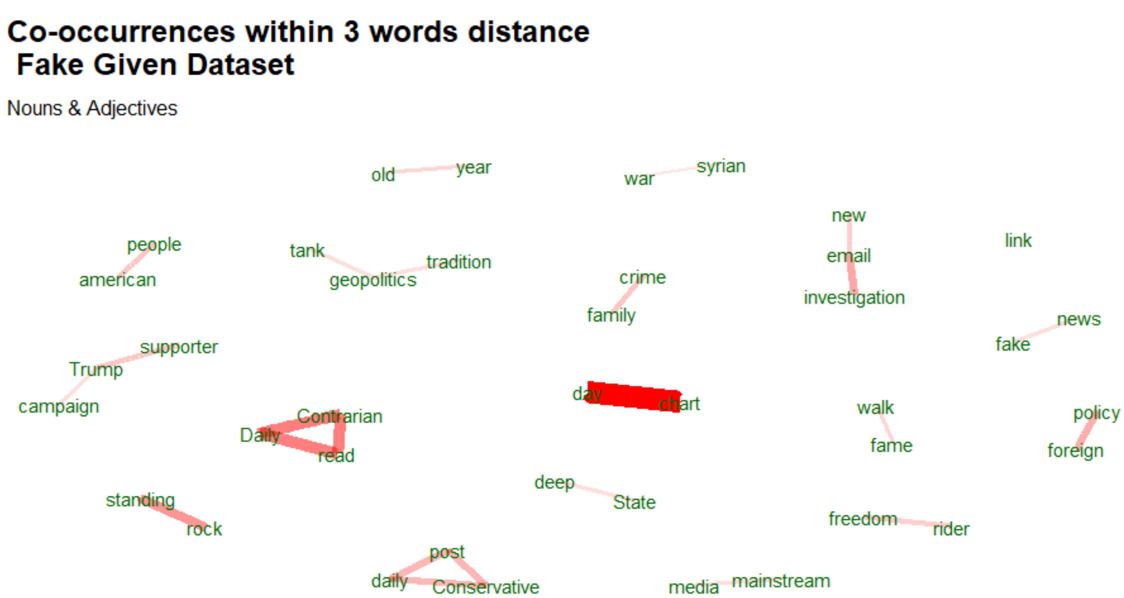

Figure 25 - Fake Given Dataset

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

The final step to our analysis is topic modeling, where we apply a Latent Dirichlet Allocation (LDA) model to the data. This is a natural language processing statistical model that observes sets of data and decides if their context is similar enough to be grouped into a particular topic. This type of modelling works really well if you have a lot of text data and you want to extract common themes. Figure 26 below, represents the Google Headlines parsed into 4 topic models. The groupings appear as four different topics regarding the Daunte Wright and Derek Chauvin civil rights trials, the Matt Gaetz allegations, the attack on the capital suspects, and the mass shootings occurring regularly 2021. Figure 27 below displays topic models for the real given dataset. We see three groupings the Clinton and Trump election, the Clinton and Trumping polling regarding emails, and the Clinton's having Obama's support against Trump. In Figure 28 below, we also identified three topic groupings: the Hillary and Trump election, the Clinton and Trump American emails, and the world war Black Muslim grouping.

Figure 26 - Google Headlines

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

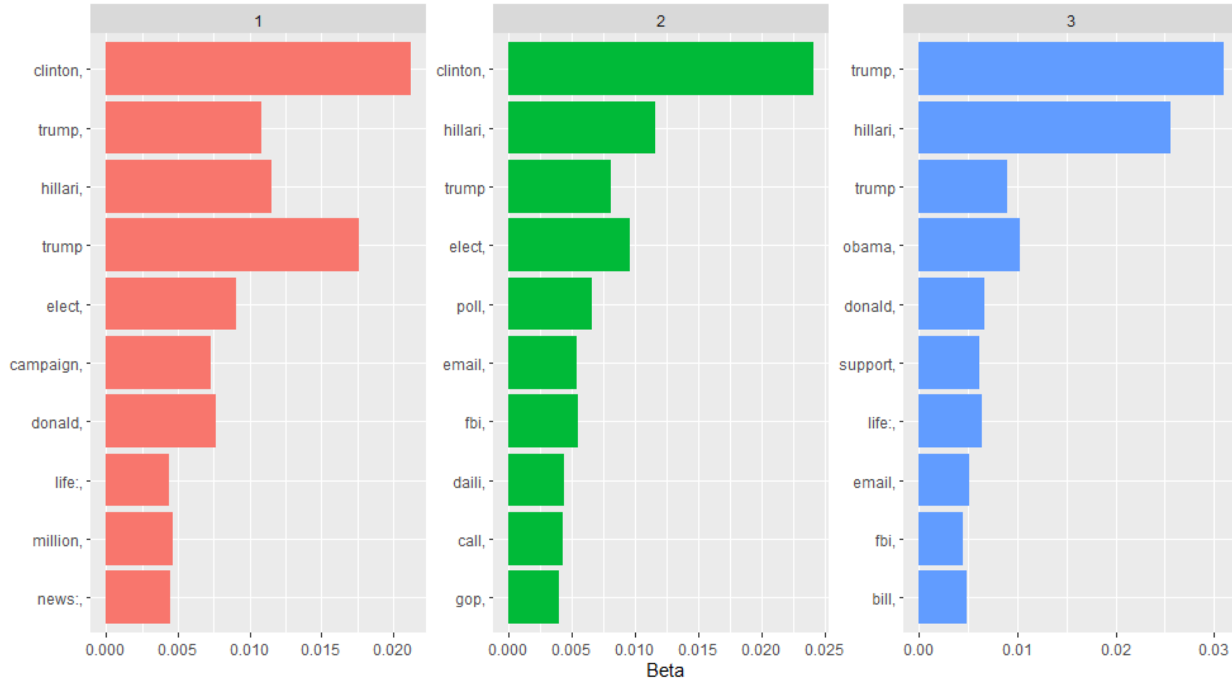

Figure 27 - Real Given Dataset

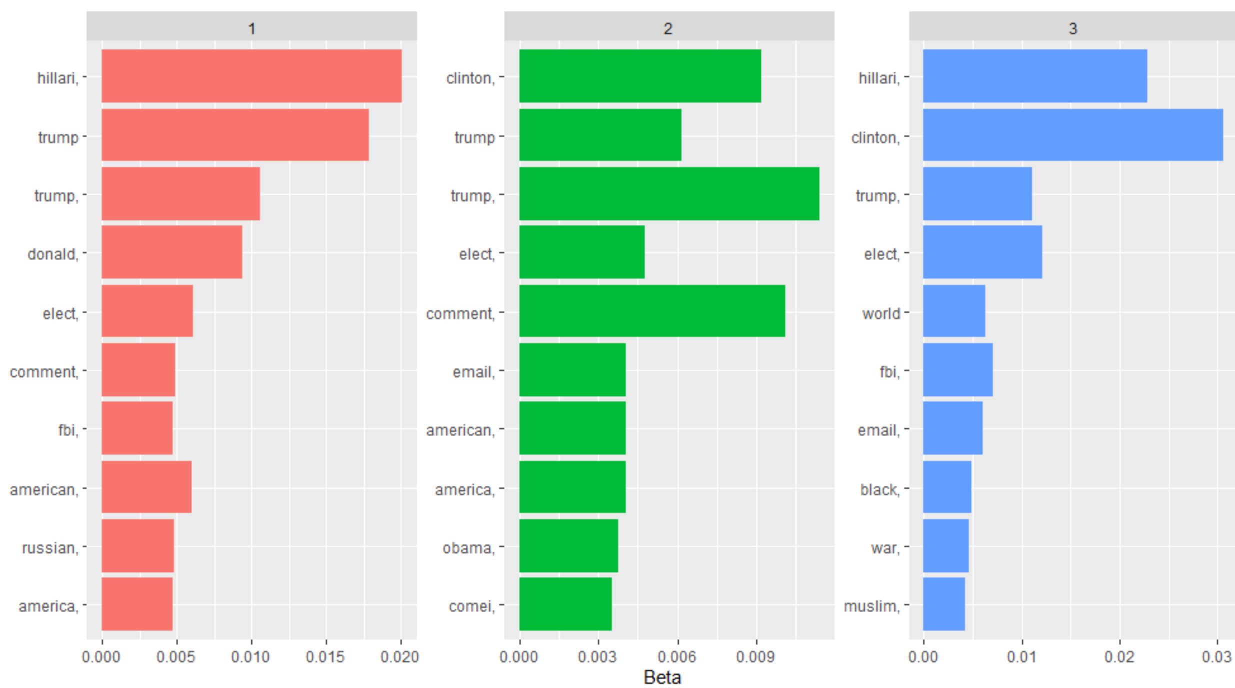

Figure 28 - Fake Given Dataset

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

When we compare just the real news verses the fake news in our given dataset, it is vey difficult to find key differences because they're both flooded with information about the Trump and Hilary election that year. The main difference is that the fake headlines tended to focus more on the standing rock and grouping world and war together. The real headlines tended to call out the word 'watch' as a noun, while the fake headlines called out 'breaking' as a noun. I think that has something to do with trying to capture someone's attention to get them to view their article.

When we compare just the real headlines given with the real Google Headlines we scraped, there aren't very many commonalities. The headlines in 2016 are much different than the headlines in 2021. The main topics discussed in headlines today are regarding Covid, vaccinations, civil rights trials and accusations, rather than headlines mostly about election polling and drama between the two 2016 candidates. The key to understanding the differences between real headlines is mostly attributed to understanding the timeline of current events. These two examples of real headlines are interesting because the headlines occurring during the 2016 election are very focused on one main topic, while headlines in April, 2021 are broad and have an expansive number of topics. Comparing two different time periods of news headlines isn't necessarily an apples-to-apples association.

To fully understand whether a news headline is real or fake is not exactly cut and dry. The best thing a person can do to decide if they should trust the news source is to first analyze the source of this information. If you're reading the comments section of an online article or browsing social media posts, realize that the presented argument is most likely going to be biased. These fake headlines or news reports use shocking words to get your attention and try to persuade you to spend more time on their page.

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

CODE USED FOR ANALYSIS

```
########WEB SCRAPE CODE Used 5 times between 4/3/21 & 4/18/21


#load packages

pacman::p_load(rvest, dplyr, stringr)


#extracts the whole news website page

google <- read_html("https://news.google.com/")


#extract the headlines & clean using regex

headline_all <- google %>%

  html_nodes("article") %>%

  html_text("span") %>%

  str_split("(?<=[a-z0-9!?\\.])(?=[A-Z])")


#get only headline title that is 1st element

headline_all <- sapply(headline_all, function(x)x[1])


#headline headlines and store into a dataframe

google_headlines <- data.frame(headlines = headline_all, stringsAsFactors=F)


str(google_headlines)


#set working directory to folder for dump

setwd("C:/Users/slaz9/Desktop/Business Analytics MS/MKTG.768.01/data/WebScraping")


write.csv(google_headlines, file="google_news_headlines.csv")
```

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

############ simple text analysis and word cloud

#install packages

pacman::p_load(dplyr, ggplot2, tidytext, wordcloud2, stringi)

#load data

headlines <- read.csv("C:/Users/slaz9/Desktop/Business Analytics MS/MKTG.768.01/data/WebScraping/google_news_headlines_210403_to_210418.csv", stringsAsFactors = F)

headlines <- read.csv("C:/Users/slaz9/Desktop/Business Analytics MS/MKTG.768.01/data/WebScraping/News Headlines Dataset Real Fake.csv", stringsAsFactors = F)

str(headlines)

#remove any unicode characters

headlines$headlines <- stringi::stri_trans_general(headlines$headlines, "latin-ascii")

#drop extra x.# columns

headlines <- headlines %>%

  select(X, headlines, type) %>%

  filter(type == 1)

#examine headlines

names(headlines)

glimpse(headlines)

str(headlines)

#delete all undesirable words

undesirable_words <- c("headlines", "headline", "video", "news", "gorafi")

#check small sample of stop-words at random

head(sample(stop_words$word, 15), 15)

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

```
#unnest headlines and remove stop-words & undesirable words <3 chars

headlines_words_filtered <- headlines %>%

  unnest_tokens(word, headlines) %>%

  anti_join(stop_words) %>%

  distinct() %>%

  filter(!word %in% undesirable_words) %>%

  filter(nchar(word) >3)


dim(headlines_words_filtered)


#get full word count from headlines

full_word_count <- headlines %>%

  unnest_tokens(word, headlines) %>%

  summarise(num_words=n()) %>%

  arrange(desc(num_words))


full_word_count


#plot commonly used words in headlines

headlines_words_filtered %>%

  count(word, sort=T) %>%

  top_n(10) %>%

  ungroup() %>%

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

  ggplot() +

  geom_col(aes(word, n)) +

  xlab("") +

  ggtitle("Most Frequently Used Words in Real Given Dataset") +
```

```
  coord_flip()


#create wordcloud

headlines_word_counts <- headlines_words_filtered %>%

  count(word, sort = T)


wordcloud2(headlines_word_counts[1:300, ], size=.5)




######################################## Other Topic Modeling using udpipe


#install packages

pacman::p_load(dplyr, ggplot2, stringr, udpipe, lattice)


#load data (change as needed)

headlines <- read.csv("C:/Users/slaz9/Desktop/Business Analytics MS/MKTG.768.01/data/WebScraping/google_news_headlines_210403_to_210418.csv", stringsAsFactors = F)

headlines <- read.csv("C:/Users/slaz9/Desktop/Business Analytics MS/MKTG.768.01/data/WebScraping/News Headlines Dataset Real Fake.csv", stringsAsFactors = F)


#remove any unicode characters

headlines$headlines <- iconv(headlines$headlines, "latin1", "ASCII", sub="")


#drop extra x.# columns and filter by type 0 or 1 (change as needed)

headlines <- headlines %>%

  select(X, headlines, type) %>%

  filter(type == 0)


#udpipe model load
```

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

```
udmodel_english <- udpipe_load_model(file="C:/Users/slaz9/Downloads/english-ewt-ud-2.5-191206.udpipe")



#use udpipe to annote the text in headlines

s <- udpipe_annotate(udmodel_english, headlines$headlines)

x <- data.frame(s)


#extract and display freq for universal parts of speech

stats <- txt_freq(x$upos)

stats$key <- factor(stats$key, levels=rev(stats$key))

barchart(key~freq, data=stats, col="yellow",

    main="UPOS (Universal Parts of Speech)\n frequency of occurrence\n Fake Given Dataset",

    xlab="Freq")


#extract & display most occurring nouns in headlines

#Nouns

stats <- subset(x, upos %in% c("NOUN"))

stats <- txt_freq(stats$token)

stats$key <- factor(stats$key, levels=rev(stats$key))

barchart(key~freq, data=head(stats, 25), col="cadetblue",

    main="Most occuring nouns\n Fake Given Dataset", xlab="Freq")


#Adjectives

stats <- subset(x, upos %in% c("ADJ"))

stats <- txt_freq(stats$token)

stats$key <- factor(stats$key, levels=rev(stats$key))

barchart(key~freq, data=head(stats, 25), col="purple",

    main="Most occuring adjectives\n Fake Given Dataset", xlab="Freq")
```

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

```
#Verbs

stats <- subset(x, upos %in% c("VERB"))

stats <- txt_freq(stats$token)

stats$key <- factor(stats$key, levels=rev(stats$key))

barchart(key~freq, data=head(stats, 25), col="gold",

        main="Most occuring verbs\n Fake Given Dataset", xlab="Freq")


#RAKE (rapid Automatic Keyword Extraction algorithm)

#determine key phrases in body of text by analyzing freq of word

#& co-occurance with other words in text

#RAKE

stats <- keywords_rake(x=x, term="lemma", group="doc_id",

                    relevant=x$upos %in% c("NOUN", "ADJ"))

stats$key <- factor(stats$keyword, levels=rev(stats$keyword))

barchart(key~rake, data=head(subset(stats, freq>3), 25), col="red",

        main="Keywords Identified by RAKE\n Fake Given Dataset",

        xlab="Rake")


#display plot seq of POS tags (noun/verb phrases)

x$phrase_tag <- as_phrasemachine(x$upos, type="upos")

stats <- keywords_phrases(x=x$phrase_tag, term=tolower(x$token),

                      pattern="(A|N)*N(P+D(A|N)*N)*",

                      is_regex=T, detailed=F)

stats <- subset(stats, ngram>1 & freq>3)

stats$key <- factor(stats$keyword, levels=rev(stats$keyword))

barchart(key~freq, data=head(stats,25), col="magenta",

        main="Keywords - simple noun phrases\n Fake Given Dataset", xlab="Freq")
```

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

```
#explore more words next to each other nouns/adjectives

#adjust ngram max levels as needed, 4 = co-occurrences within 3 words

stats <- keywords_collocation(x=x, term="token",

                        group=c("doc_id", "paragraph_id", "sentence_id"),

                        ngram_max=4)


#how freq words occur in same sentences (nouns and adjectives)

stats <- cooccurrence(x=subset(x, upos %in% c("NOUN", "ADJ")),

                term="lemma", group=c("doc_id", "paragraph_id",

                                "sentence_id"))


#co-occurrences: how freq do words follow eachother

stats <- cooccurrence(x=x$lemma, relevant=x$upos %in% c("NOUN", "ADJ"))


#co-occurrences: how freq do words follow & skip 2 words betwn

stats <- cooccurrence(x=x$lemma, relevant=x$upos %in% c("NOUN", "ADJ"),

                skipgram=2)

head(stats)



#display this visually

pacman::p_load(igraph, ggraph)


wordnetwork <- head(stats, 25)

wordnetwork <- graph_from_data_frame(wordnetwork)

ggraph(wordnetwork, layout="fr") +

  geom_edge_link(aes(width=cooc, edge_alpha=cooc),

              edge_colour="red") +

  geom_node_text(aes(label=name), col="darkgreen", size=4) +
```

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

```
  theme_graph(base_family="Arial Narrow") +

  theme(legend.position="none") +

  labs(title="Co-occurrences within 3 words distance\n Fake Given Dataset",

       subtitle="Nouns & Adjectives")



################################ Topic modeling



#load packages

pacman::p_load(tidyverse, tidytext, topicmodels, tm, SnowballC, stringr)


#load data (change as needed)

headlines <- read.csv("C:/Users/slaz9/Desktop/Business Analytics MS/MKTG.768.01/data/WebScraping/google_news_headlines_210403_to_210418.csv", stringsAsFactors = F)

headlines <- read.csv("C:/Users/slaz9/Desktop/Business Analytics MS/MKTG.768.01/data/WebScraping/News Headlines Dataset Real Fake.csv", stringsAsFactors = F)


#remove any unicode characters

headlines$headlines <- iconv(headlines$headlines, "latin1", "ASCII", sub="")


#drop extra x.# columns and filter by type (change as needed)

headlines <- headlines %>%

  select(X, headlines, type) %>%

  filter(type == 0)


#first create corpus from text

#second uses tidytext pkg

#topic model extracted based on # function invoked
```

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

```
#finally, top 10 terms that best represent each topic extracted


#fuction to get & plot the most informative terms by a specified number

# of topics, using LDA

top_terms_by_topic_LDA <- function(input_text, # should be a column from a data frame

                                   plot = T, # return a plot? TRUE by default

                                   number_of_topics = 4) # number of topics (4 by default)

{

  # create a corpus (type of object expected by tm) and document term matrix

  Corpus <- Corpus(VectorSource(input_text)) # make a corpus object

  DTM <- DocumentTermMatrix(Corpus) # get the count of words/document

  # remove any empty rows in our document term matrix (if there are any

  # we'll get an error when we try to run our LDA)

  unique_indexes <- unique(DTM$i) # get the index of each unique value

  DTM <- DTM[unique_indexes,] # get a subset of only those indexes

  # preform LDA & get the words/topic in a tidy text format

  lda <- LDA(DTM, k = number_of_topics, control = list(seed = 1234))

  topics <- tidy(lda, matrix = "beta")

  # get the top ten terms for each topic,

  # yes I made up the word informativeness

  top_terms <- topics  %>% # take the topics data frame and..

    group_by(topic) %>% # treat each topic as a different group

    top_n(10, beta) %>% # get the top 10 most informative words

    ungroup() %>% # ungroup

    arrange(topic, -beta) # arrange words in descending informativeness

  # if the user asks for a plot (TRUE by default)

  if(plot == T){

    # plot the top ten terms for each topic in order

    top_terms %>% # take the top terms
```

```
      mutate(term = reorder(term, beta)) %>%
      # sort terms by beta value
      ggplot(aes(term, beta, fill = factor(topic))) +
      # plot beta by theme
      geom_col(show.legend = FALSE) + # as a bar plot
      facet_wrap(~ topic, scales = "free") +
      # which each topic in a separate plot
      labs(x = NULL, y = "Beta") + # no x label, change y label
      coord_flip() # turn bars sideways
  }else{
    # if the user does not request a plot
    # return a list of sorted terms instead
    return(top_terms)
  }
}

#test function by starting w 2 topics
#identify irrelevant words 2 eliminate & add to stop word list
top_terms_by_topic_LDA(headlines$headlines, number_of_topics=2)

#begin topic modeling
#create tidytext corpus
#pay attention to column names
headlinesCorpus <- Corpus(VectorSource(headlines$headlines))
headlinesDTM <- DocumentTermMatrix(headlinesCorpus)

#convert doc term matrix to tidytext corpus
headlinesDTM_tidy <- tidy(headlinesDTM)
```

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

```
#add custom stop words

custom_stop_words <- tibble(word=c("headlines", "headline", "video", "news", "le gorafi","para para
dinle"))


#remove stop-words from dataset

headlinesDTM_tidy_cleaned <- headlinesDTM_tidy %>%

  anti_join(stop_words, by=c("term" = "word")) %>%

  anti_join(custom_stop_words, by=c("term" = "word"))


#reconstruct cleaned doc so each word shows correct num times

cleaned_documents <- headlinesDTM_tidy_cleaned %>%

  group_by(document) %>%

  mutate(terms=toString(rep(term, count))) %>%

  select(document,terms) %>%

  unique()


#verify cleaned docs in alphabetic order

head(cleaned_documents)


#obtain topic models

#try to & expand to 3 or 4 depending on data set size

top_terms_by_topic_LDA(cleaned_documents$terms, number_of_topics=2)

top_terms_by_topic_LDA(cleaned_documents$terms, number_of_topics=3)

top_terms_by_topic_LDA(cleaned_documents$terms, number_of_topics=4)


#stem topic models

headlinesDTM_tidy_cleaned <- headlinesDTM_tidy_cleaned %>%

  mutate(stem=wordStem(term))
```

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

```
#reconstruct our docs

cleaned_documents <- headlinesDTM_tidy_cleaned %>%

  group_by(document) %>%

  mutate(terms=toString(rep(stem,count))) %>%

  select(document, terms) %>%

  unique()


#revisit topic model & create stemmed word topic models

#try lower end of larger set

#then look at new most informative terms

top_terms_by_topic_LDA(cleaned_documents$terms, number_of_topics = 3)

top_terms_by_topic_LDA(cleaned_documents$terms, number_of_topics = 4)




#

# ###### attempt to get the sentiment & t-sne

#

# #load packages for this section

# pacman::p_load(tidyr, dplyr, stringr, data.table, sentimentr, ggplot2, text2vec, tm, ggrepel)

#

# #load data

# headlines <- read.csv("C:/Users/slaz9/Desktop/Business Analytics
MS/MKTG.768.01/data/WebScraping/google_news_headlines_210403_to_210418.csv",
stringsAsFactors = F)

# headlines <- read.csv("C:/Users/slaz9/Desktop/Business Analytics
MS/MKTG.768.01/data/WebScraping/News Headlines Dataset Real Fake.csv", stringsAsFactors = F)

#

# #remove any unicode characters

# headlines$headlines <- stringi::stri_trans_general(headlines$headlines, "latin-ascii")
```

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

```
#

# #drop extra x.# columns

# headlines <- headlines %>%

#   select(x, headlines, type) %>%

#   filter(type == 1)

#

# #create rowid for headlines

# headline_df <- as.data.frame(headlines) %>%

#   mutate(id=row_number())

#

# #define lexicon & add changes for our context

# #get n rows-to see what we have in the lexicon -

# # Tyler Rinker is the author of sentiment

# nrow(lexicon::hash_sentiment_jockers_rinker)#seems like 11,710 words

#

# #words to replace-in this example, there are switch, brand names etc.

# replace_in_lexicon <-tribble(

#   ~x, ~y,

#   "U.S", 0,       # not in dictionary

#   "America", 0,      # not in dictionary

#   "Biden", 0,         # not in dictionary

# )

#

# #create a new lexicon with modified sentiment

# headline_lexicon <-lexicon::hash_sentiment_jockers_rinker %>%

#   filter(!x %in% replace_in_lexicon$x) %>%

#   bind_rows(replace_in_lexicon) %>%

#   setDT() %>%

#   setkey("x")
```

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

```
#
# #get sentence level sentiment for testing
# sent_df <-headline_df %>%
#   get_sentences() %>%
#   sentiment_by(by = c('id', 'headlines'),
#               polarity_dt = headline_lexicon)
#
# str(sent_df)
#
# ##### continued
#
# #load packages for this section
# pacman::p_load(tidyr, dplyr, stringr, data.table, sentimentr, ggplot2, text2vec, tm, ggrepel)
#
#
# #GloVe (global vectors for word representation)
# #calc contextual representation of a word in vector form
# #use unsupervised learning on n-gram / dimensionality reduction
#
# #create lists of headlines split into indv words (iterator over tokens)
# tokens <- space_tokenizer(headlines$headlines %>%
#                          tolower() %>%
#                          removePunctuation())
#
# #create vocabulary. Terms will be unigrams (simple words)
# it <-itoken(tokens, progressbar = FALSE)
# vocab <-create_vocabulary(it)
#
# #remove words that appear less than 2 times
```

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

```
# vocab <-prune_vocabulary(vocab, term_count_min = 2L)

#

# #Use our filtered vocabulary

# vectorizer <-vocab_vectorizer(vocab)

#

# #use skip gram window of 3 for context words

# tcm <-create_tcm(it, vectorizer, skip_grams_window = 3L)

#

# #fit the model. It can take several minutes based on how much data you have

# glove = GloVe$new(rank = 100, x_max = 5)

# glove$fit_transform(tcm, n_iter = 20)

#

# #get the processed word vector

# word_vectors = glove$components

#

# #check which words have contextual similarity

# trump <- word_vectors[,"trump", drop=F]

#

# #cosine similarity btwn word vectors (how similar)

# cos_sim = sim2(x=t(word_vectors), y=t(trump),

#             method="cosine", norm="l2")

# head(sort(cos_sim[,1], decreasing=T),10)

#

# #implement quick t-SNE to visualize similarities

# pacman::p_load(tm, Rtsne, tibble, tidytext, scales)

#

# #create vector of words to keep, before applying tsne (remove stop words)

# keep_words <-setdiff(colnames(word_vectors), stopwords())

#
```

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

```
# # keep words in vector

# word_vec <-word_vectors[, keep_words]

#

# # prepare data frame to train

# train_df <-data.frame(t(word_vec)) %>%

#   rownames_to_column("word")

#

# # train tsne for visualization

# tsne <-Rtsne(train_df[,-1], dims = 2, perplexity = 50, verbose=TRUE,

#             max_iter = 500)

#

# #t-Distributed Stochastic Neighbor Embedding (t-SNE)

# #maps high dimensional data into lower dimensions

# #so that the distance btwn 2words roughly describes similarity

# #t-SNE creates naturally forming clusters

#

# #create plot

# colors = rainbow(length(unique(train_df$word)))

# names(colors) = unique(train_df$word)

#

# plot_df <-data.frame(tsne$Y) %>%  mutate(

#   word = train_df$word,

#   col = colors[train_df$word]

#   ) %>%left_join(vocab, by = c("word" = "term")) %>%

#   filter(doc_count >= 20)

#

# ggplot(plot_df, aes(X1, X2)) +

#   geom_text(aes(X1, X2, label = word, color = col), size = 3) +

#   xlab("") + ylab("") +
```

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

```
#   theme(legend.position = "none")

#

# str(headline_df)

# str(sent_df)

#

# #calculate word-level sentiment

# word_sent <-headline_df %>%

#   left_join(sent_df, by = "id") %>%

#   select(x, headlines, type, ave_sentiment) %>%

#   unnest_tokens(word, headlines) %>%

#   group_by(word) %>%

#   summarise(

#     count = n(),

#     avg_sentiment = mean(ave_sentiment),

#     sum_sentiment = sum(ave_sentiment)

#     ) %>%

#   # remove stop words

#   anti_join(stop_words, by = "word")

#

# # filter to words that appear at least 2 times

# pd_sent <-plot_df %>%

#   left_join(word_sent, by = "word") %>%

#   drop_na() %>%

#   filter(count >= 2)

#

# #plot results

# ggplot(pd_sent, aes(X1, X2)) +

#   geom_point(aes(X1, X2, size = count, alpha = .1, color = avg_sentiment)) +

#   geom_text(aes(X1, X2, label = word), size = 2) +
```

Problem 2: News Headlines Real vs. Fake
Sarah Lazio-Maimone

```
#   scale_colour_gradient2(low = muted("red"), mid = "white",
#                       high = muted("blue"), midpoint = 0) +
#   scale_size(range = c(5, 20)) +
#   xlab("") + ylab("") +
#   ggtitle("2-dimensional t-SNE Mapping of Word Vectors") +
#   guides(color = guide_legend(title="Avg. Sentiment"), size = guide_legend(title = "Frequency"), alpha =
NULL) +
#   scale_alpha(range = c(1, 1), guide = "none")
```