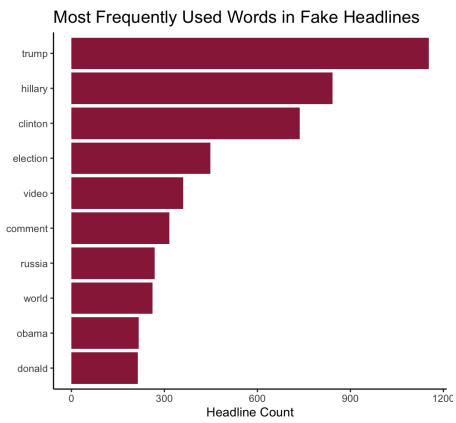
Dana DiVincenzo
4/18/2021

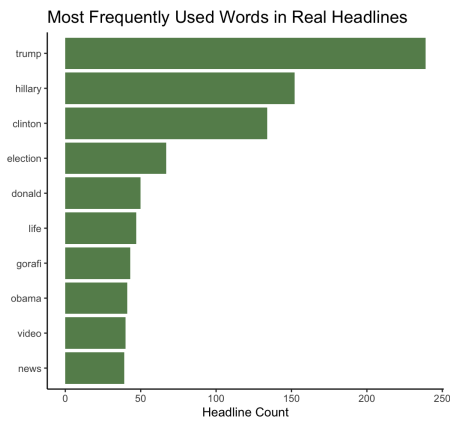
Assignment 2
MKTG.768

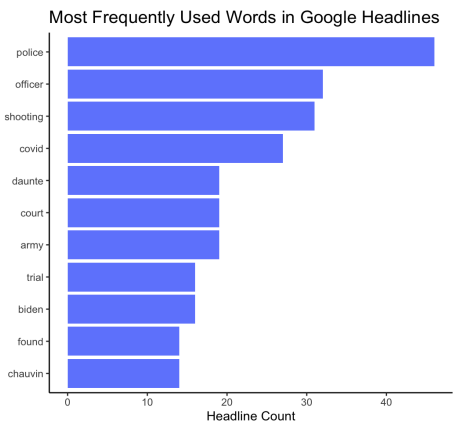
Real vs. Fake News Headlines

Fake news is when false information is presented as news. These fake news articles can be for satirical purposes, or can purposely mislead viewers with the intention of damaging a person's or organization's reputation. Some fake news websites operate in order to generate revenue through advertising. Besides being misleading, fake news can lead to general distrust in news networks and can diminish the impact of real news articles. The frequency of fake news has substantially increased with the rise of social media. Furthermore, studies have shown that fake news is especially dominant during election seasons, and at those times, are often created by foreign entities with the goal of swaying election results. In order to try and find possible differences between fake news headlines and real news headlines, I performed text mining techniques on fake news headlines, real news headlines, and Google News headlines which I collected myself. We will assume that articles from Google news are legitimate. The fake and real news headlines are from the same dataset, and although the dates of the headlines are unknown, I am guessing that they come from the year 2016 (based on topics in headlines). The Google News headlines were gathered from 4/9/2021 - 4/15/2021. The difference in dates will be important throughout this analysis, as the primary topics being discussed in the news are completely different in 2021 than they were in 2016.

To begin my analysis, I first looked at the top ten most frequently used words in the Fake data, the Real data, and the Google data. The graphs showing these results are below.

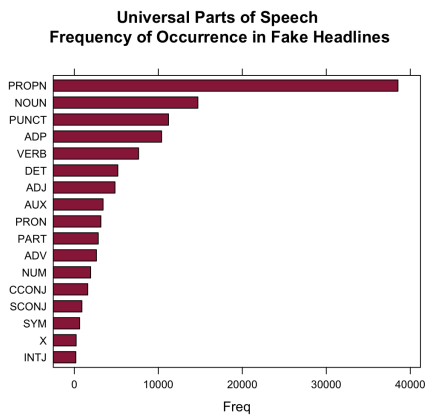


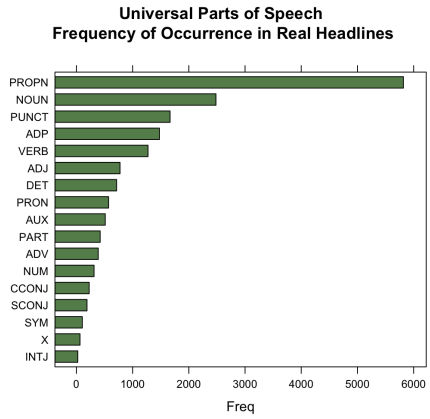


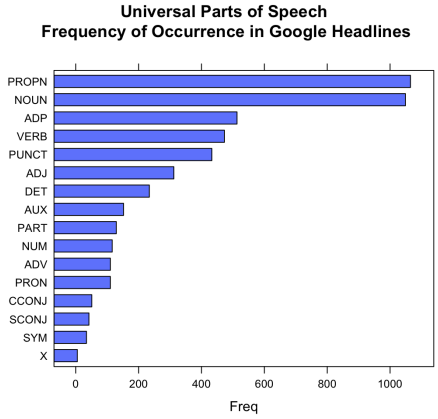


As we can see from the graphs, the Fake and Real headlines have very similar frequently used words. Seven out of the ten words on the Fake headlines graph are also on the Real headlines graph. We can also see that the words on both the Fake and Real headlines graphs seem to be about Donald Trump, Hillary Clinton, and the 2016 US presidential election. The 2016 election season was particularly heated and Trump and Clinton were the presidential candidates. Therefore, it makes sense that they would be main topics of discussion for real news articles, and that fake news articles would know to write articles about them as well. Furthermore, it is possible that these fake news articles about the election candidates came from foreign sources with the goal of making one of the candidates look bad, decreasing their election chances. When looking at the Google headlines, we can see that the topics discussed are completely different. Words like police, shooting, Covid, Daunte, and Biden all reflect what is being discussed in the current media cycle. It makes sense that these words are so different from the headlines in 2016, as Hillary Clinton and Donald Trump are not main topics of discussion in the news at this point. These graphs give us a general idea of what the headlines are discussing, but offer no clear way to differentiate real news from fake news.

Next, I created graphs which show the breakdown of the universal parts of speech (UPOS) for the Fake, Real, and Google headlines. These graphs are shown below.





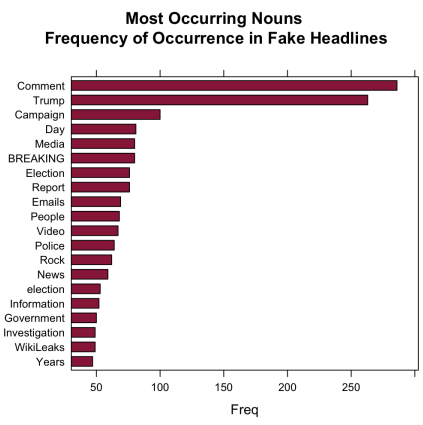


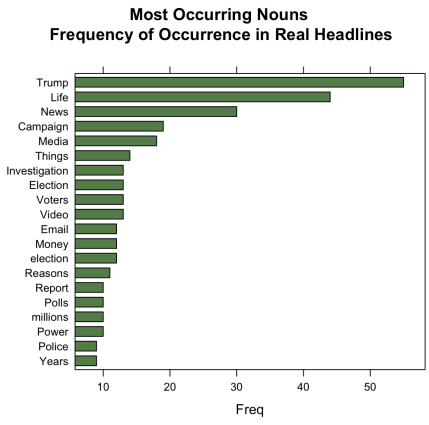
The graphs of the Fake and Real headlines look extremely similar. We can see that for both, proper nouns are the most commonly used part of speech, and proper nouns are used more than twice as much as the next most commonly used part of speech. Following proper nouns, the Fake and the Real graphs both

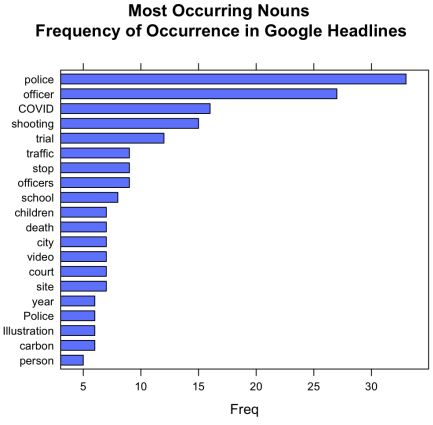
have nouns, punctuation, adpositions, and verbs as their next most frequently used parts of speech. Only when we get to the sixth most frequently used part of speech do the graphs finally become different. We can also see that the curve that the bars make on each graph look very similar for the Real vs Fake datasets. Comparing the Google headlines graph to the other two, it looks different. Although proper nouns are the most frequently used part of speech in the Google dataset, it is followed very closely by nouns. We can also see that the order of the parts of speech on the Google graph is more different from the Real graph when we also compare the Real graph to the Fake graph. Finally, we can see that the curve made by the bars on the Google graph looks different when compared to the Real and Fake graphs.

Some tentative conclusions can be made from these results. Because the Fake and Real graphs look more similar than the Real and Google graphs do, it could be that timing of article releases/ topics of article discussions are more important in determining UPOS than if the articles are real or if they are fake. It also seems that fake news articles are able to construct headlines which correctly mimic the UPOS breakdown of real articles. Furthermore, because the Google dataset was collected over such a small period of time, and the media was mainly covering a few very specific topics at that time, it is possible that the results from the Google dataset is atypical. We may be able to draw better conclusions from UPOS graphs if next time the Google News data is collected over a longer period of time, such as 6 months.

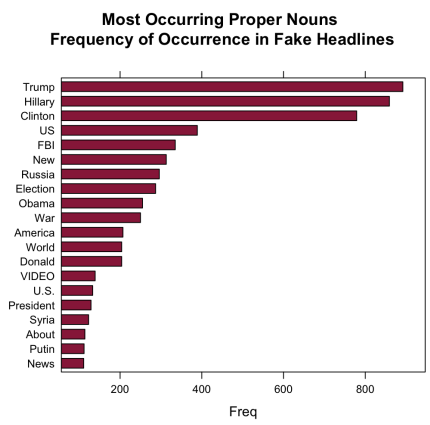
I then created graphs which show the top 20 most frequently occurring nouns, proper nouns, adjectives, and verbs in each dataset. These graphs are shown below.

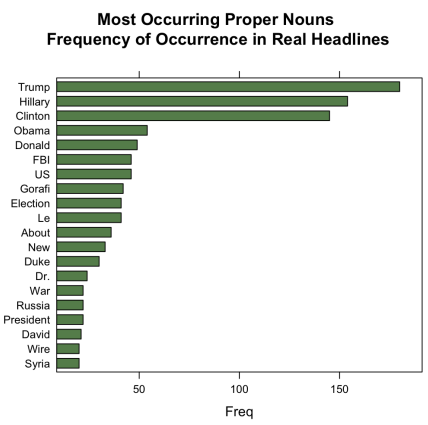


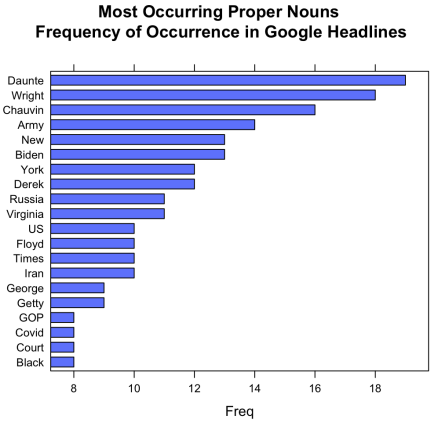


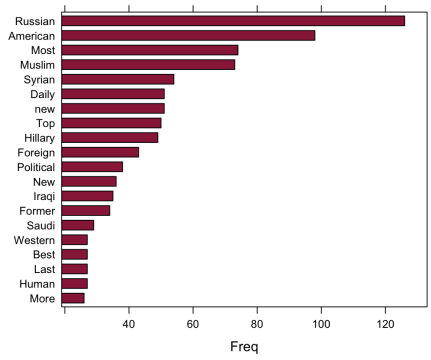


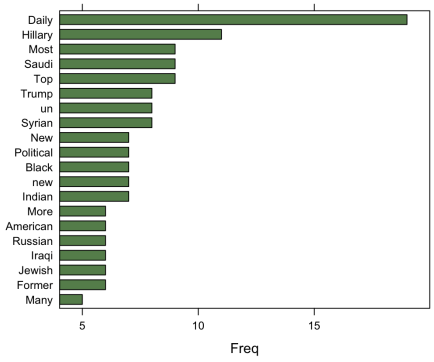
Dana DiVincenzo
4/18/2021

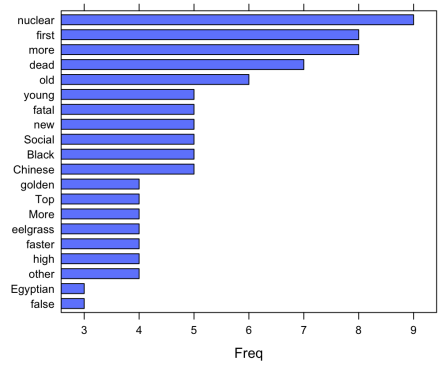


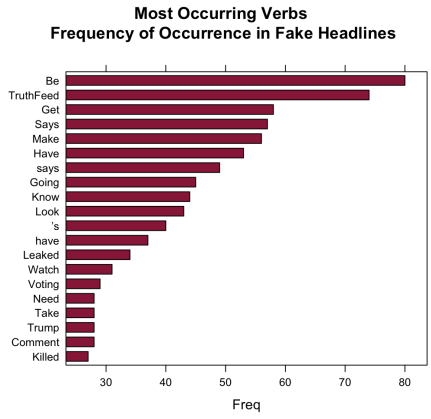


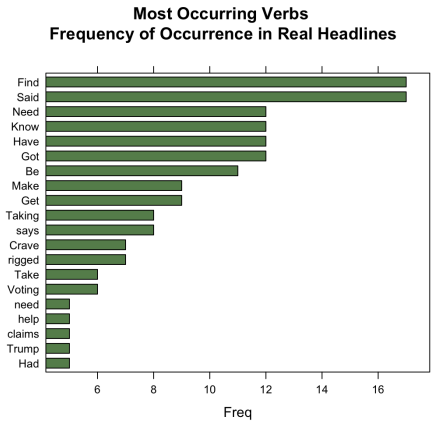


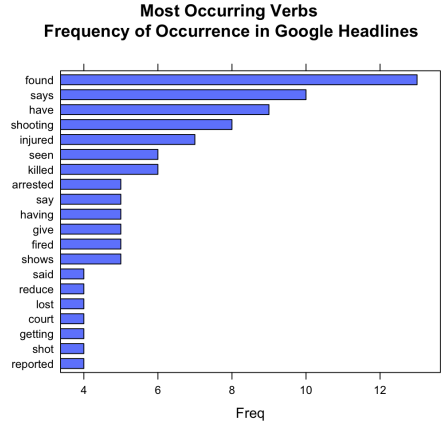










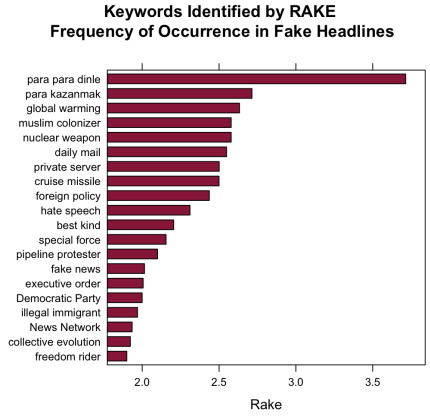


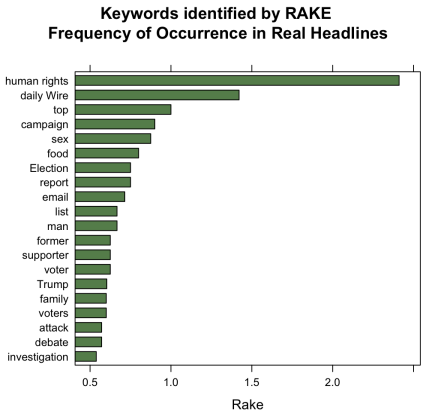
When looking at the noun and proper noun graphs, we can see that the Fake and Real news headlines contain similar people and topics. Trump, campaign, and Hiliary are all frequently used words. The nouns and proper nouns from the Google News headlines are completely different, and relate to current media events instead of the 2016 election, as expected. The curves made by the bars of the Fake and Real noun and proper noun graphs also look similar to each other when compared to the curves that the same Google

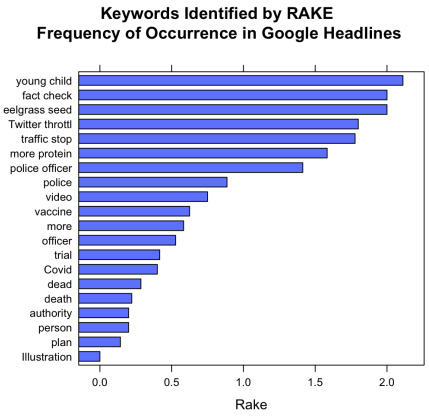
graph bars make. This may be due to the fact that the 2016 election and the candidates were discussed so intensely in the media at that time, causing the first few nouns and proper nouns on the graph to have a significantly higher frequency than other nouns and proper nouns in headlines. When looking at the adjective frequency graphs, we are able to see more differences between the Fake and Real graphs than we have seen previously. Many of the adjectives on both graphs are the same, but words like “Russian” and “American” which are at the top of the Fake list, are towards the bottom of the Real list. Furthermore, the Fake graph has words like “Muslim”, “Foreign”, and “Western”, which do not appear on the Real graph at all. These words have high potential to be clickbait words, which may be why they are used frequently within the Fake headlines. The adjectives on the Google graph are completely different from the other two, which again is likely due to the difference in media topics at the different times of data collection. Looking at the verb graphs for the Fake and Real data, we can see that most of the verbs used are basic, common verbs such as “get”, “says”, “be”, “talking”, etc. However, one verb stood out to me. The Fake graph’s second most frequently used verb is “TruthFeed”. TruthFeed is not a verb (sometimes words can be misclassified in these graphs), but instead is a website which is known for publishing fake and exaggerated stories with pro-Trump headlines. Therefore, if I did not know that the Fake dataset contained fake headlines, seeing the word TruthFeed being used so frequently could be a way to potentially alert me that these headlines may be false or exaggerated. If we look at the Google verb graph, we can see that the words here are less basic or common than the other two graphs, with verbs like “shooting”, “injured”, “killed”, and “arrested” being high on the list. Again, however, this may be because the data for this graph came from such a small window of time with specific media topics being covered. If data from Google News was collected over a longer period of time, we may see more common words used more frequently in the headlines. One interesting thing of note is that the nouns, proper nouns, adjectives, and verbs from the Fake and Real datasets almost all have capitalized first letters. The words shown in the same graphs from the Google dataset are typically lowercase, besides names of people and places. I would expect for fake news to capitalize every word in its headline in order to clickbait, and for real news articles to use more lowercase words. However, this may capitalization difference may be

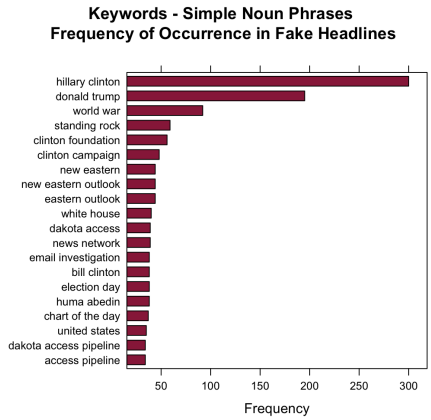
because of how the Fake and Real data was captured vs how the Google data was captured, and would therefore be of no interest to us.

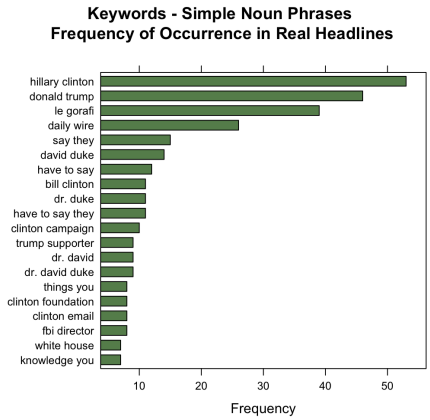
Next, I used two different methods to see which words frequently appear near each other within headlines. One method used RAKE, which extracts and ranks the keywords and phrases out of the headlines. Words are considered more important if they show up with specific words often, but are not generally scattered throughout all headlines. The other method found the top noun-verb phrases from the headlines and plotted them by frequency. This method allows us to see popular keywords and topics. The graphs showing this information are below.

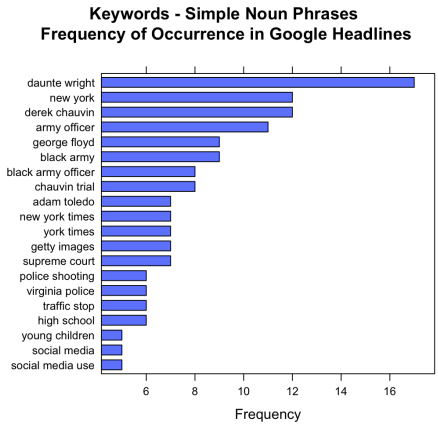








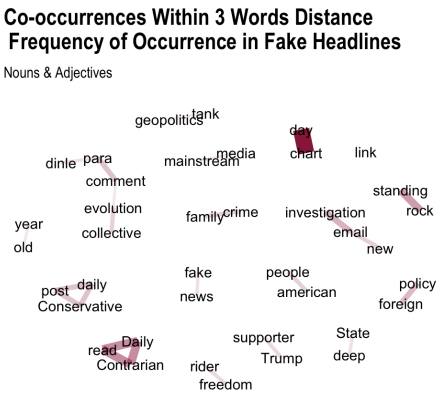


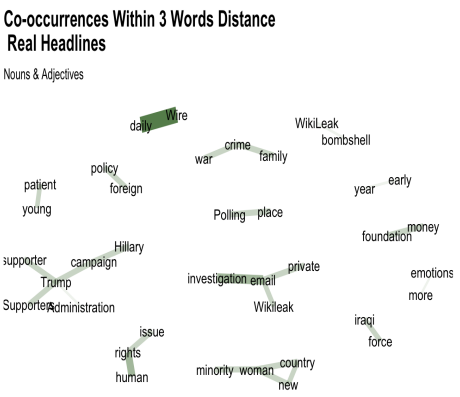


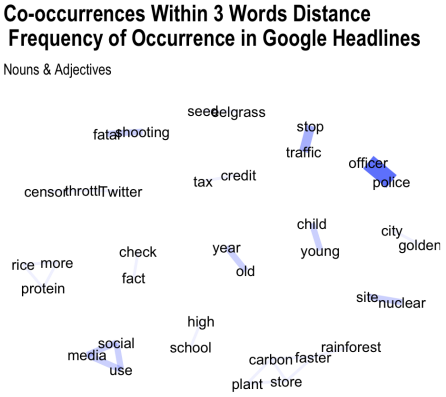
Looking at the RAKE keyword graphs, there are some interesting differences between the Fake and Real frequently found keywords. The first thing that jumps out is that the two most common keywords on the Fake graph appear to be in Turkish, not English. All of the headlines collected were marked as being written in English, so this is an interesting flag for headlines being fake. Another interesting difference

between the Fake and Real RAKE graphs is that every term in the Fake graph is made up of two words, while many of the terms on the Real graph are made of only one word. Looking at the Google News graph, we can see that many of the terms there are also only made of one word. Therefore, it could be that fake news headlines are more likely to have phrases identified by RAKE which are two words rather than one. Looking at the simple noun phrases graphs, it seems that the Fake and Real graphs have similar key phrases, and the Google graph has phrases more related to current events. This tells me that when analyzing future headlines, RAKE may be the better tool when trying to decipher real headline datasets from fake ones.

I was then able to create a word network graph for each of the datasets. A word network shows us the association and cohesion of specific words in the dataset. In this case, we mapped nouns and adjectives, so we are able to see which nouns and adjectives are related to each other for Fake, Real, and Google News headlines. These graphs are shown below.



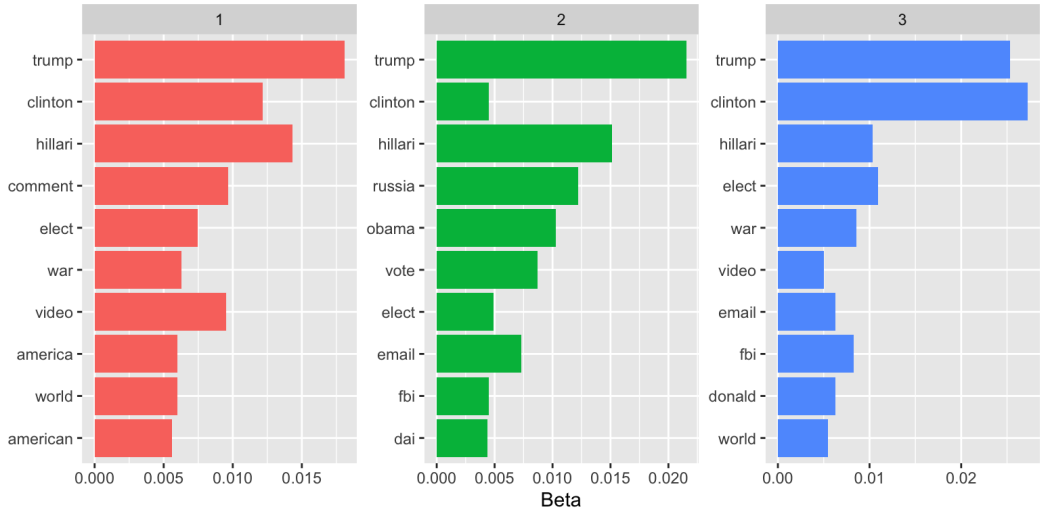




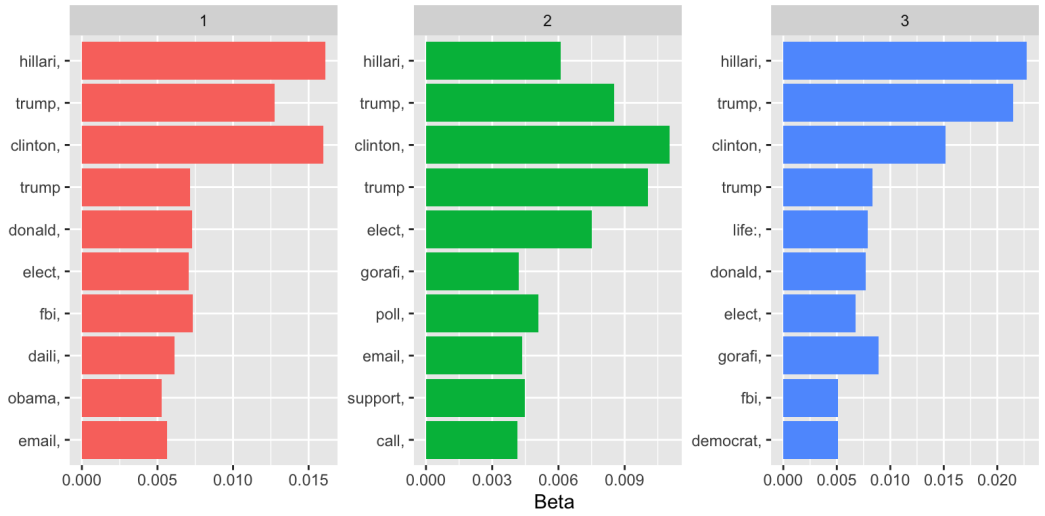
These word network graphs are where we are really able to see some differences between the Fake and Real headlines. By glancing at those two graphs, we are able to see that the topics discussed are not the same. If we look at the Fake graph specifically, some of the word connections make little sense. For example, the words “dinle”, “para”, “comment”, “evolution”, and “collective” are all linked together, but dinle and para are not English words while the other three are. It seems strange that they would be linked together. The words “day” and “chart” have the strongest connection on the Fake graph, but those words

are very common, making it suspicious that they would have such a strong connection. The words “read”, “daily”, and “contrarian” also have a strong connection, but I have no idea what those words would be referencing specifically, even after some internet research. When looking at the Real graph, the word clusters seem to make more natural sense, and also seem to be about more specific news topics. For example, the word Trump is connected to the words “supporter”, “administration”, “campaign”, and “Hillary”. The word email is connected to the words “wikileak”, “private”, and “investigation”. Seeing these clusters gives a viewer an immediate understanding into which news events are being discussed frequently - something I did not get with the Fake graph. The Google graph also seems to join together keywords that are more specific to a media story. For example, the words “police” and “officer”, and the words “traffic” and “stop”, make perfect sense to be connected, as the dominant news story at this time is about a police officer shooting an unarmed black person at a traffic stop. Other stories that were talked about are represented by specific words rather than vague ones, as we can see with the cluster which includes “carbon”, “plant”, and “rainforest”. Based on these three graphs, it seems that we could possibly identify fake news by seeing if a word network graph produces more vague or more specific topics. It also appears that understanding the dominant news stories at different time periods helps significantly in identifying which data contains fake headlines and which data contains real headlines.

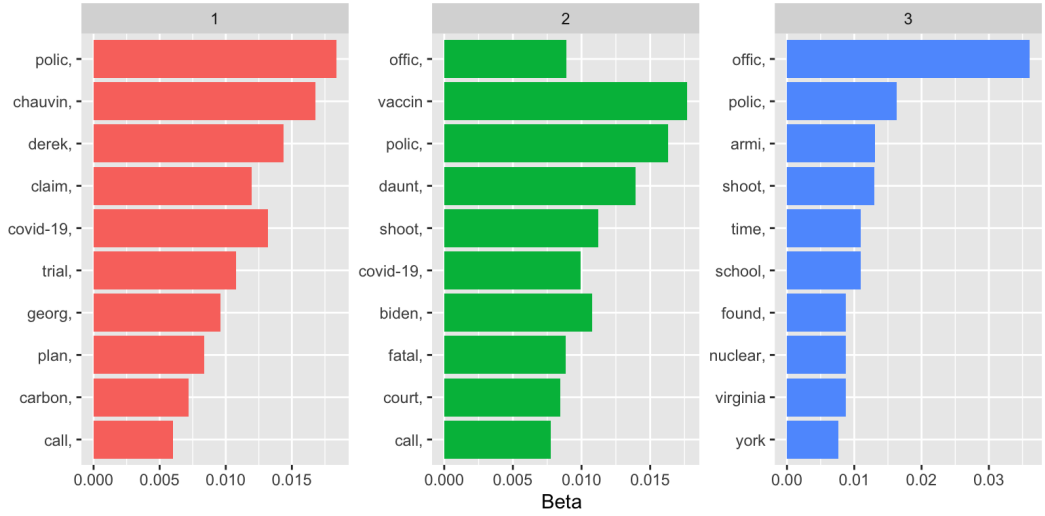
Moving forward, I created topic models using the LDA method. Topic modeling allows us to create clusters of headlines which contain similar words, phrases, or information. By looking at the different topics, we can quickly identify what each set of headlines are talking about most. The different topic models for the Fake, Real, and Google datasets are shown below.



Fake



Real



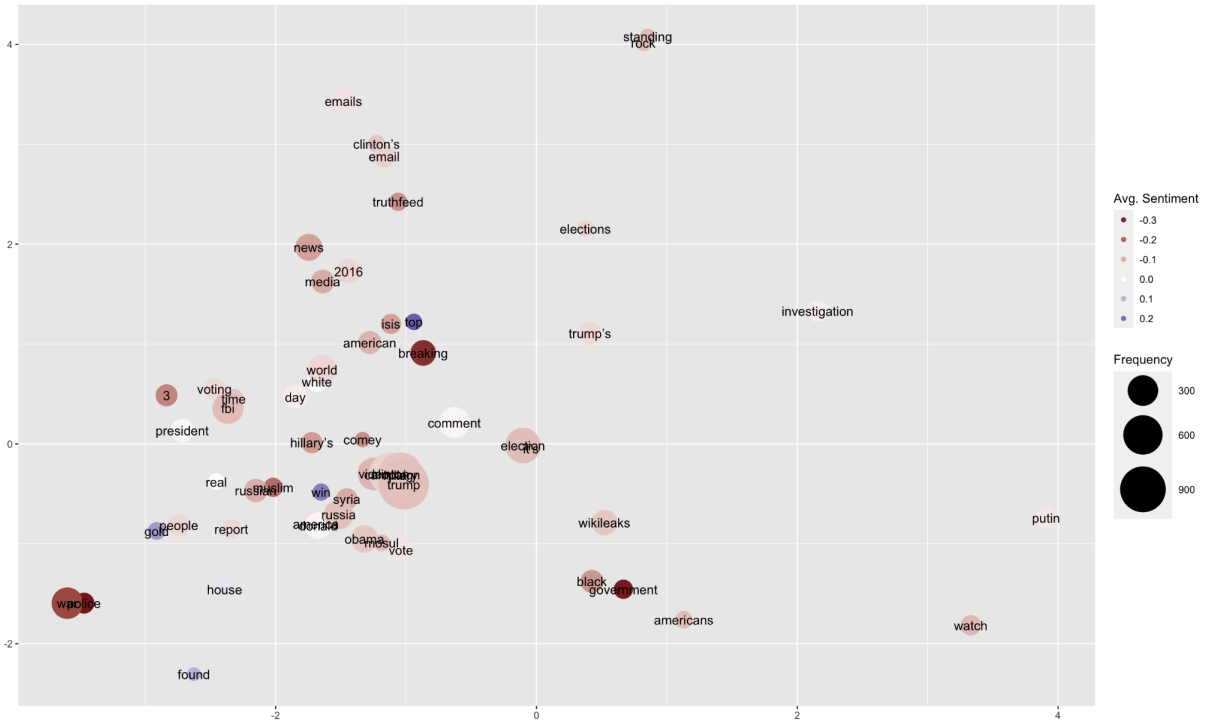
Google

Looking at the topic models for the Fake data, we can see that each topic mentioned Trump, Clinton, and the election. This tells me that those topics are being talked about in most of the headlines in the fake dataset. The first and third topic models mention the words “world” and “war”, which tell me that the fake headlines may have been discussing a possible World War Three. These words are not in the real topic models, and are fear-mongering and clickbait-like, so I can understand why they are in the fake dataset. Looking at the topic models for the real data, we can see that Trump, Clinton, and the election are again discussed in all models. Therefore, the fake headlines were right to discuss the election candidates as much as they did, as it makes them seem more like real news. Many other words in the real topic models are present in the fake ones, such as “emails”, “fbi”, and “Obama”. Overall, the real and fake data has similar topics being discussed. The real and fake models both also do not super distinct and unique topics, all the topics have similar keywords and blend together, which I think is representative of how focused the media was on the 2016 election at the time of headline collection. The topic models made from the Google News headlines have very different results from the other two datasets, which we have seen throughout this analysis. However, it is similar to the Fake and Real topic models in that some of the words are mentioned in every cluster, and many of the words are mentioned in multiple clusters. For example, the word “police” is in every topic, and the words “shoot”, “Covid-19”, “call”, and “officer” are mentioned in multiple. However, the third topic stands out as having unique words, with words like “Virginia”, “nuclear”, “found”, and “school” being in that topic and not in the others. Again, I believe the similarities in topics are due to the intense media coverage on a small number of events during the time of collection, and that the results may be different if the Google News headlines were collected during a longer period of time.

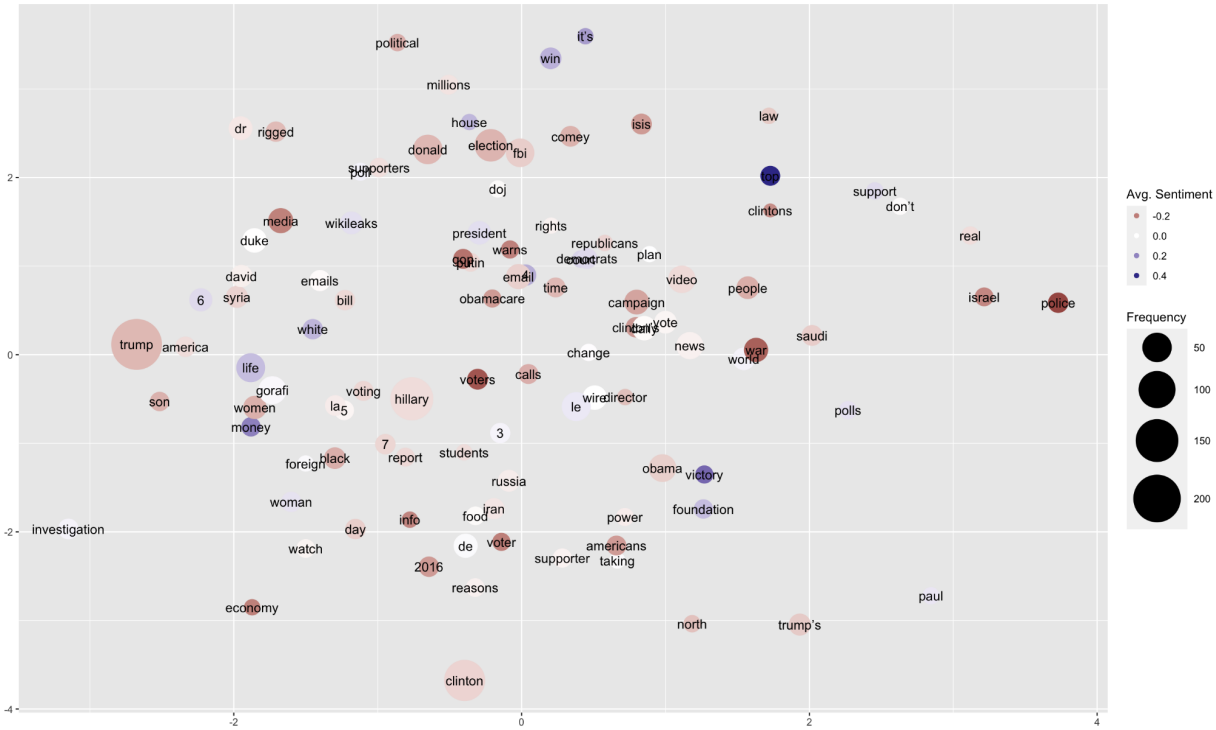
Finally, I investigated the sentiment of the different headline datasets. I took individual word level sentiment for each set of headlines, and plotted the frequently used words on a TSNE plot. This plot shows if words are used similarly or dissimilarly in the headlines, visualized by their proximity to each

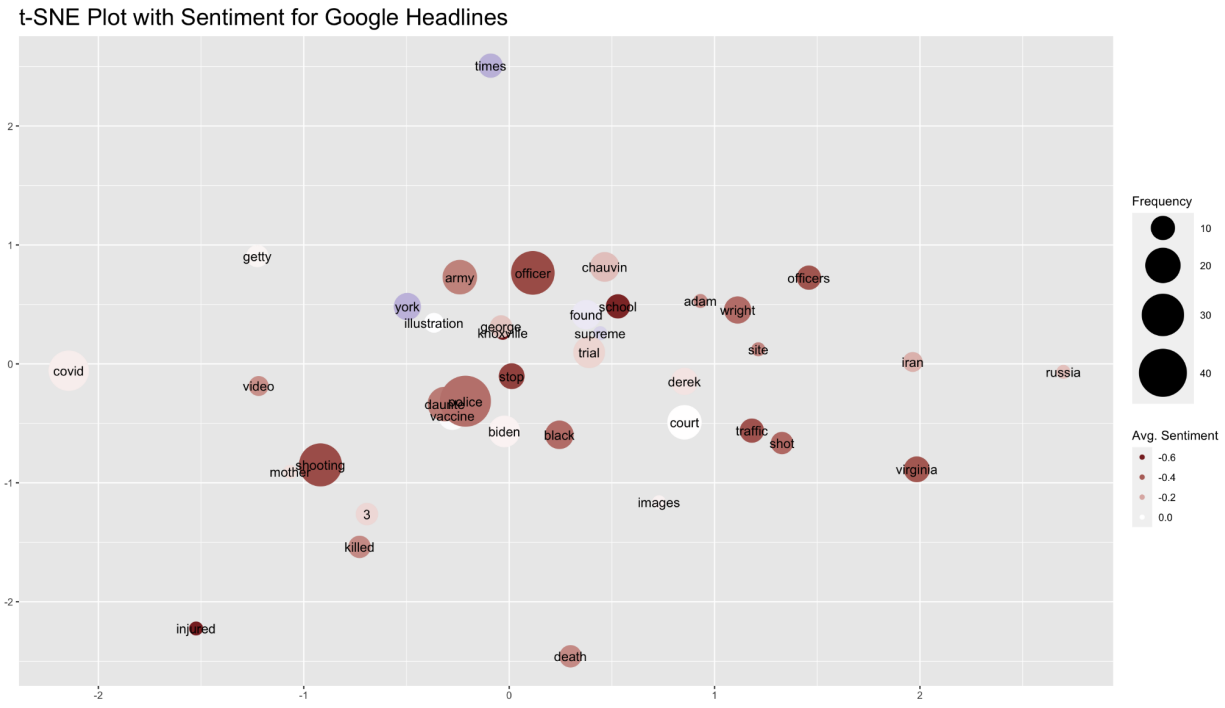
other. I then overlaid the sentiment of each word on top, so we could see which words are used together positively, and which are used together negatively. These graphs are shown below.

t-SNE Plot with Sentiment for Fake Headlines



t-SNE Plot with Sentiment for Real Headlines





This graph shows the average sentiment of each word, based on color, and the frequency of occurrence of each word, based on circle size. The first thing I noticed is that the average sentiment of the Fake headlines ranges from -.3 to .2, the average sentiment of the Real headlines ranges from -.2 to .4, and the average sentiment of the Google headlines ranges from -.6 to 0. This means that the Google headlines were the most negative, with the Real headlines were the most positive, and the Fake headlines were in the middle. However, this may be because the media focus for the week of Google News headline collection was on very negative topics. Better results may be found if the Google data is taken from a longer timeframe. Looking at the Fake TSNE plot specifically, we can see that the words “war”, “police”, “breaking”, and “government” are used the most negatively within the fake headlines. The words “top”, “win”, “gold”, and found are used the most positively. We can also see word clusters, such as “Clinton’s”, “email,” and “TruthFeed”, along with “black”, “government”, “wikileaks”, and “Americans”. Both of these clusters have negative sentiment, so we know news articles are speaking negatively about these topics. The real headline plot has negative words like “war”, “voters”, and “police”, and positive words such as “top”, “money”, and “victory”. The word top was one of the most positive words in both the real and the fake plots. The Google headlines plot showed that the words “york” and “times” were used most

positively, likely referring to the New York Times. We can also see that the words “officer”, “shooting”, “traffic”, and “stop” were some of the most negatively used words. Overall, there is no clear distinction between the Fake and Real plots that would allow me to easily identify the fake headlines as fake if I was unaware of their sources. Therefore, it seems that fake news has become so good at mimicking the sentiment of real news that it is difficult to distinguish without sophisticated models.

After doing some exploratory data analysis, text mining, and sentiment analysis, there are no clear conclusions on straightforward ways to detect fake news from real news. In most parts of this analysis, the Real news dataset looked more similar to the Fake news dataset than it did to the Google dataset - which tells me a few things. One, fake news outlets have become so good at creating believable headlines that they are almost impossible to tell apart from real news headlines. Two, the time at which the headlines are created/gathered is important, as comparing headlines from completely different times and media cycles is much more difficult than comparing headlines from the same time. Furthermore, it was very important that I knew the general news topics for each time period, otherwise some things from the fake dataset that stood out as strange to me may have appeared normal. This again shows how believable the fake headlines are.

If I were to continue analyzing headlines with the goal of being able to classify real from fake, I would repeat this same analysis but with multiple datasets from multiple different time periods. I would also extend the length of time that the Google data was taken, and gather it during a longer time period, to make sure the results are not skewed due to a randomly intense news weekend. I would then see if any results I found during one time period are the same as the results found during another time period. If so, I would feel that the results are more accurate and repeatable instead of possibly being due to specific media stories.

```
# extracting the whole google news website page
google <-read_html("https://news.google.com/")

# extracting headlines and cleaning using regex
headline_all <-google %>% html_nodes("article") %>% html_text("span") %>%
str_split("(?<=[a-z0-9!?\\.])(?=[A-Z])")

# getting only the headline title, which is the first element
headline_all <-sapply(headline_all, function(x) x[1])

# review the headlines and store them into a dataframe or file
# 4/15/21 5:00pm
google_headlines<-data.frame(headlines= headline_all, stringsAsFactors = F)
str(google_headlines)
write.csv(google_headlines, file="google_news_day6.csv")
# done scraping the data

# reading in the google headline data from the past few days
day1 <- read.csv("/Users/dana/google_news_day1.csv", stringsAsFactors = FALSE)
day2 <- read.csv("/Users/dana/google_news_day2.csv", stringsAsFactors = FALSE)
day3 <- read.csv("/Users/dana/google_news_day3.csv", stringsAsFactors = FALSE)
day4 <- read.csv("/Users/dana/google_news_day4.csv", stringsAsFactors = FALSE)
day5 <- read.csv("/Users/dana/google_news_day5.csv", stringsAsFactors = FALSE)
day6 <- read.csv("/Users/dana/google_news_day6.csv", stringsAsFactors = FALSE)

# joining the dataframes
google_headlines <- rbind(day1, day2)
google_headlines <- rbind(google_headlines, day3)
google_headlines <- rbind(google_headlines, day4)
google_headlines <- rbind(google_headlines, day5)
google_headlines <- rbind(google_headlines, day6)

nrow(google_headlines) #326
head(google_headlines)

# removing any duplicate headlines
google_headlines <- distinct(google_headlines)
nrow(google_headlines) # 321

# reading in the given dataset with mostly fake headlines and formatting
given_headlines <-read.csv("/Users/dana/Downloads/News Headlines Dataset Real Fake.csv",
stringsAsFactors = FALSE)
given_headlines <- given_headlines[, c("title", "language", "type")]
head(given_headlines)
str(given_headlines)
str(google_headlines)
nrow(given_headlines[given_headlines["type"] == "real",]) #1134 are real, 9355 are fake

#splitting the given dataset into 2, one with the fake headlines and one with the read headlines
```

```
fake_headlines <- given_headlines[given_headlines["type"] == "bs",]
real_headlines <- given_headlines[given_headlines["type"] == "real",]
nrow(fake_headlines)
nrow(real_headlines)
# time to do some text mining techniques we learned in class.

# let's first do it with the google dataset
# check out a small sample of stop words, at random
head(sample(stop_words$word, 15), 15)

#unnest and remove stop, undesirable and short words
# unnesting the headlines, removing all stopwords, undesirable words, and
# words smaller than 3 characters
undesirable_words <-c("ampvideo_youtube")

google_headline_words_filtered <-google_headlines %>%
  unnest_tokens(word, headlines) %>%
  anti_join(stop_words) %>%
  filter(!word %in% undesirable_words) %>%
  filter(nchar(word) > 3)

dim(google_headline_words_filtered)
head(google_headline_words_filtered,50)

# this shows me the top 10 most used words from the google headlines
# check this to see if there are any undesirable words
google_headline_words_filtered %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot() +
  theme(panel.grid.major = element_blank(), panel.background = element_blank(),
        axis.line = element_line(colour = "black"), plot.title = element_text(size=15)) +
  geom_col(aes(word, n), fill="#7189FF" ) +
  xlab("") +
  ylab("Headline Count") +
  ggtitle("Most Frequently Used Words in Google Headlines", ) +
  coord_flip()

# creating a wordcloud of the words in the google headlines
# tells you the frequency of that words presence in the text
google_word_counts <-google_headline_words_filtered %>% count(word, sort = TRUE)
wordcloud2(google_word_counts[1:300, ], size = .7)

# now let's do the same thing with the fake dataset

#unnest and remove stop, undesirable and short words
# unnesting the headlines, removing all stopwords, undesirable words, and
# words smaller than 3 characters for the fake headlines
```

```
fake_headline_words_filtered <- fake_headlines[, c("type", "title")] %>%
  unnest_tokens(word, title) %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 3)

head(fake_headline_words_filtered, 100)
dim(fake_headline_words_filtered)

# this shows me the top 10 most used words from the fake headlines
# all of these make sense except ampvideo_youtube, I will add to undesireable words
fake_headline_words_filtered %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot() +
  theme(panel.grid.major = element_blank(), panel.background = element_blank(),
        axis.line = element_line(colour = "black"), plot.title = element_text(size=15)) +
  geom_col(aes(word, n), fill = "#982649") +
  xlab("") +
  ylab("Headline Count") +
  ggtitle("Most Frequently Used Words in Fake Headlines") +
  coord_flip()

# creating a wordcloud of the words in the fake headlines
# tells you the frequency of that words presence in the text
fake_word_counts <-fake_headline_words_filtered %>% count(word, sort = TRUE)
wordcloud2(given_word_counts[1:300, ], size = 1)


# now let's do the same thing with the REAL dataset

#unnest and remove stop, undesirable and short words
# unnesting the headlines, removing all stopwords, undesirable words, and
# words smaller than 3 characters for the fake headlines

real_headline_words_filtered <- real_headlines[, c("type", "title")] %>%
  unnest_tokens(word, title) %>%
  anti_join(stop_words) %>%
  filter(nchar(word) > 3)

head(real_headline_words_filtered, 100)
dim(real_headline_words_filtered)

# this shows me the top 10 most used words from the real headlines
real_headline_words_filtered %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot() +
```

```
  theme(panel.grid.major = element_blank(), panel.background = element_blank(),
        axis.line = element_line(colour = "black"), plot.title = element_text(size=15)) +
  geom_col(aes(word, n), fill = "#678d58") +
  xlab("") +
  ylab("Headline Count") +
  ggtitle("Most Frequently Used Words in Real Headlines") +
  coord_flip()

# creating a wordcloud of the words in the real headlines
# tells you the frequency of that words presence in the text
real_word_counts <-real_headline_words_filtered %>% count(word, sort = TRUE)
wordcloud2(real_word_counts[1:300, ], size = 1)


###### Text Mining Parts of Speech and Keywords

# we will do google first

# udpipe needs a model file loaded.
# I am just downloading the most recent version here
x <- udpipe_download_model(language = "english")
udmodel_english <-
udpipe_load_model(file="/Users/dana/Downloads/english-ewt-ud-2.5-191206.udpipe")

# use udpipeto annotate the text in the google headlines and load into a frame
# this may take a while depending on how much data you are analyzing
s <-udpipe_annotate(udmodel_english, google_headlines$headlines)
# dumping the data into a dataframe
x <-data.frame(s)

# universal parts of speech for google_headlines
stats <-txt_freq(x$upos)
stats$key <-factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = stats, col = "#7189FF",
         main = "Universal Parts of Speech\n Frequency of Occurrence in Google Headlines",
         xlab = "Freq")

# most common nouns in google_headlines
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, 20), col = "#7189FF",
         main = "Most Occurring Nouns\n Frequency of Occurrence in Google Headlines", xlab = "Freq")

# most common proper nouns in google_headlines
stats <-subset(x, upos %in% c("PROPN"))
stats <-txt_freq(stats$token)
stats$key <-factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "#7189FF",
         main = "Most Occurring Proper Nouns\n Frequency of Occurrence in Google Headlines", xlab =
"Freq")
```

```
# most common adjectives in google_headlines
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, 20), col = "#7189FF",
       main = "Most Occurring Adjectives\n Frequency of Occurrence in Google Headlines", xlab =
"Freq")

# most common Verbs in google_headlines
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, 20), col = "#7189FF",
       main = "Most Occurring Verbs\n Frequency of Occurrence in Google Headlines", xlab = "Freq")

# if a word appears near another word at least 3 times, we see it on the graph
# tells me that these words appear most frequently and within a certain window of each other!
# these keywords tell us very useful things for quick analysis
# don't interpret the numbers on the x axis, only on the key words themselves
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), 20), col = "#7189FF",
       main = "Keywords Identified by RAKE\n Frequency of Occurrence in Google Headlines", xlab =
"Rake")

# here, we are looking at noun phrases (noun and verb in combination, within a certain distance of
eachother, a certain number of times)
# this tells us some things that people were talking about
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 = TRUE, detailed = FALSE)

stats <-subset(stats, ngram > 1 & freq > 3)
stats$key <-factor(stats$keyword, levels = rev(stats$keyword))
barchart(key ~ freq, data = head(stats, 20), col = "#7189FF",
       main = "Keywords - Simple Noun Phrases\n Frequency of Occurrence in Google Headlines", xlab =
"Frequency")


## Collocation identification –basically words following one another)
stats<-keywords_collocation(x = x, term = "token",
                     group = c("doc_id", "paragraph_id", "sentence_id"), ngram_max = 4)

## How frequently do words occur in the same sentence(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 frequent do words follow one another
stats <-cooccurrence(x = x$lemma, relevant = x$upos %in% c("NOUN", "ADJ"))
```

```
## Co-occurrences: How frequent do words follow one another even if we would
## skip 2 words in between. You can adjust this if you need to.
stats <-cooccurrence(x = x$lemma, relevant = x$upos %in% c("NOUN", "ADJ"), skipgram = 2)

head(stats)

# exploring stats visually instead of in a table
# tells us the connection between words that we could not see with the bar graphs
# can see which topics are most related to each other
# here, we only mapped nouns and adjectives, we could do nouns and verbs
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 = "#7189FF") +
  geom_node_text(aes(label = name), col = "black", size = 4) +
  theme_graph(base_family = "Arial Narrow") +
  theme(legend.position = "none") +
  labs(title = "Co-occurrences Within 3 Words Distance\n Frequency of Occurrence in Google Headlines",
subtitle = "Nouns & Adjectives")

# FAKE DATASET

d <-udpipe_annotate(udmodel_english, fake_headlines$title)
# dumping the data into a dataframe
z <-data.frame(d)

# universal parts of speech for fake dataset
stats <-txt_freq(z$upos)
stats$key <-factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = stats, col = "#982649",
        main = "Universal Parts of Speech\n Frequency of Occurrence in Fake Headlines",
        xlab = "Freq")

# nouns for fake dataset
stats <-subset(z, upos %in% c("NOUN"))
stats <-txt_freq(stats$token)
stats$key <-factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "#982649",
        main = "Most Occurring Nouns\n Frequency of Occurrence in Fake Headlines", xlab = "Freq")

# proper nouns for fake dataset
stats <-subset(z, upos %in% c("PROPN"))
stats <-txt_freq(stats$token)
stats$key <-factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "#982649",
        main = "Most Occurring Proper Nouns\n Frequency of Occurrence in Fake Headlines", xlab =
"Freq")

# adjectives in fake dataset
stats <-subset(z, upos %in% c("ADJ"))
```

```
stats <-txt_freq(stats$token)
stats$key <-factor(stats$key,levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "#982649",
       main = "Most Occurring Adjectives\n Frequency of Occurrence in Fake Headlines", xlab = "Freq")

# verbs for fake dataset
stats <-subset(z, upos %in% c("VERB"))
stats <-txt_freq(stats$token)
stats$key <-factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "#982649",
       main = "Most Occurring Verbs\n Frequency of Occurrence in Fake Headlines", xlab = "Freq")

# using RAKE on the fake dataset
stats <-keywords_rake(x = z, term = "lemma", group = "doc_id", relevant = z$upos %in% c("NOUN",
"ADJ"))
stats$key <-factor(stats$keyword, levels = rev(stats$keyword))
barchart(key ~ rake, data = head(subset(stats, freq > 3), 20), col = "#982649",
       main = "Keywords Identified by RAKE\n Frequency of Occurrence in Fake Headlines", xlab =
"Rake")

# here, we are looking at noun phrases (noun and verb in combination, within a certain distance of
eachother, a certain number of times)
# this is not a supervised model, so we have to be the one who supervises it
# this tells us some things that people were talking about
z$phrase_tag <-as_phrasemachine(z$upos, type = "upos")
stats <-keywords_phrases(x = z$phrase_tag, term = tolower(z$token),pattern =
                   "(A|N)*N(P+D*(A|N)*N)*", is_regex = TRUE, detailed = FALSE)

stats <-subset(stats, ngram > 1 & freq > 3)
stats$key <-factor(stats$keyword, levels = rev(stats$keyword))
barchart(key ~ freq, data = head(stats, 20), col = "#982649",
       main = "Keywords - Simple Noun Phrases\n Frequency of Occurrence in Fake Headlines", xlab =
"Frequency")

## Collocation identification –basically words following one another)
stats<-keywords_collocation(x = z, term = "token",
                     group = c("doc_id", "paragraph_id", "sentence_id"), ngram_max = 4)

## How frequently do words occur in the same sentence(nouns and adjectives)
stats <-cooccurrence(x = subset(z, upos %in% c("NOUN", "ADJ")),
              term = "lemma", group = c("doc_id", "paragraph_id", "sentence_id"))

## Co-occurrences: How frequent do words follow one another
stats <-cooccurrence(x = z$lemma, relevant = z$upos %in% c("NOUN", "ADJ"))

## Co-occurrences: How frequent do words follow one another even if we would
## skip 2 words in between. You can adjust this if you need to.
stats <-cooccurrence(x = z$lemma, relevant = z$upos %in% c("NOUN", "ADJ"), skipgram = 2)

head(stats)
```

```
# exploring stats visually instead of in a table
# tells us the connection between words that we could not see with the bar graphs
# can see which topics are most related to each other
# here, we only mapped nouns and adjectives, we could do nouns and verbs
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 = "#982649") +
  geom_node_text(aes(label = name), col = "black", size = 4) +
  theme_graph(base_family = "Arial Narrow") +
  theme(legend.position = "none") +
  labs(title = "Co-occurrences Within 3 Words Distance\n Frequency of Occurrence in Fake Headlines",
subtitle = "Nouns & Adjectives")


# REAL DATASET

h <-udpipe_annotate(udmodel_english, real_headlines$title)
# dumping the data into a dataframe
k <-data.frame(h)

# universal parts of speech for real dataset
stats <-txt_freq(k$upos)
stats$key <-factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = stats, col = "#678d58",
        main = "Universal Parts of Speech\n Frequency of Occurrence in Real Headlines",
        xlab = "Freq")

# nouns for real dataset
stats <-subset(k, upos %in% c("NOUN"))
stats <-txt_freq(stats$token)
stats$key <-factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "#678d58",
        main = "Most Occurring Nouns\n Frequency of Occurrence in Real Headlines", xlab = "Freq")

# proper nouns for real dataset
stats <-subset(k, upos %in% c("PROPN"))
stats <-txt_freq(stats$token)
stats$key <-factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "#678d58",
        main = "Most Occurring Proper Nouns\n Frequency of Occurrence in Real Headlines", xlab =
"Freq")

# adjectives in real dataset
stats <-subset(k, upos %in% c("ADJ"))
stats <-txt_freq(stats$token)
stats$key <-factor(stats$key,levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "#678d58",
        main = "Most Occurring Adjectives\n Frequency of Occurrence in Real Headlines", xlab = "Freq")

# verbs for real dataset
```

```
stats <-subset(k, upos %in% c("VERB"))
stats <-txt_freq(stats$token)
stats$key <-factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "#678d58",
       main = "Most Occurring Verbs\n Frequency of Occurrence in Real Headlines", xlab = "Freq")

# using RAKE on the real dataset
stats <-keywords_rake(x = k, term = "lemma", group = "doc_id", relevant = k$upos %in% c("NOUN",
"ADJ"))
stats$key <-factor(stats$keyword, levels = rev(stats$keyword))
barchart(key ~ rake, data = head(subset(stats, freq > 3), 20), col = "#678d58",
       main = "Keywords identified by RAKE\n Frequency of Occurrence in Real Headlines", xlab =
"Rake")

# here, we are looking at noun phrases (noun and verb in combination, within a certain distance of
eachother, a certain number of times)
# this is not a supervised model, so we have to be the one who supervises it
# this tells us some things that people were talking about
# real headlines
k$phrase_tag <-as_phrasemachine(k$upos, type = "upos")
stats <-keywords_phrases(x = k$phrase_tag, term = tolower(k$token),pattern =
                    "(A|N)*N(P+D*(A|N)*N)*", is_regex = TRUE, detailed = FALSE)

stats <-subset(stats, ngram > 1 & freq > 3)
stats$key <-factor(stats$keyword, levels = rev(stats$keyword))
barchart(key ~ freq, data = head(stats, 20), col = "#678d58",
       main = "Keywords - Simple Noun Phrases\n Frequency of Occurrence in Real Headlines", xlab =
"Frequency")

## Collocation identification –basically words following one another)
# real headlines
stats<-keywords_collocation(x = k, term = "token",
                     group = c("doc_id", "paragraph_id", "sentence_id"), ngram_max = 4)

## How frequently do words occur in the same sentence(nouns and adjectives)
stats <-cooccurrence(x = subset(k, upos %in% c("NOUN", "ADJ")),
                term = "lemma", group = c("doc_id", "paragraph_id", "sentence_id"))

## Co-occurrences: How frequent do words follow one another
stats <-cooccurrence(x = k$lemma, relevant = k$upos %in% c("NOUN", "ADJ"))

## Co-occurrences: How frequent do words follow one another even if we would
## skip 2 words in between. You can adjust this if you need to.
stats <-cooccurrence(x = k$lemma, relevant = k$upos %in% c("NOUN", "ADJ"), skipgram = 2)

head(stats)

# exploring stats visually instead of in a table
# tells us the connection between words that we could not see with the bar graphs
# can see which topics are most related to each other
# here, we only mapped nouns and adjectives, we could do nouns and verbs
```

```
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 = "#678d58") +
  geom_node_text(aes(label = name), col = "black", size = 4) +
  theme_graph(base_family = "Arial Narrow") +
  theme(legend.position = "none") +
  labs(title = "Co-occurrences Within 3 Words Distance\n Real Headlines", subtitle = "Nouns &
Adjectives")



####### Now I will use LDA for Topic Modeling

### GOOGLE DATA

# clean the google data, our headlines are in the 'headlines' column of the dataset
# check the column names for the dataset
google_headlines$headlines <-str_replace_all(google_headlines$headlines,"[^[:graph:]]", " ")

# write a function that allows us to extract topic models quickly and easily
top_terms_by_topic_LDA <- function(input_text, # should be a column from a data frame
                                   plot = T, # return a plot? TRUE by defult
                                   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
  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)
  }
}


# pay attention to column names
reviewsCorpus <- Corpus(VectorSource(google_headlines$headlines))
reviewsDTM <- DocumentTermMatrix(reviewsCorpus)

# convert the document term matrix to a tidytext corpus
reviewsDTM_tidy <- tidy(reviewsDTM)

# remove stopwords from the dataset of headlines
reviewsDTM_tidy_cleaned <- reviewsDTM_tidy %>% # take our tidy dtm and...
  anti_join(stop_words, by = c("term" = "word")) #%>% # remove Engligh stopwords and...
  #anti_join(custom_stop_words, by = c("term" = 'word')) # remove my custom stopwords

# recostruct cleaned documents (so that each word shows up the correct number of times)
cleaned_documents <- reviewsDTM_tidy_cleaned %>%
  group_by(document) %>%
  mutate(terms = toString(rep(term, count))) %>%
  select(document, terms) %>%
  unique()

# check out what the cleaned documents look like
head(cleaned_documents)

# start obtaining topic models
top_terms_by_topic_LDA(cleaned_documents, number_of_topics = 2)
top_terms_by_topic_LDA(cleaned_documents, number_of_topics = 3)
top_terms_by_topic_LDA(cleaned_documents, number_of_topics = 4) # 4 seems to be the best

reviewsDTM_tidy_cleaned <- reviewsDTM_tidy_cleaned %>%
  mutate(stem = wordStem(term))

# reconstruct our documents
cleaned_documents <- reviewsDTM_tidy_cleaned %>%
  group_by(document) %>%
  mutate(terms = toString(rep(stem, count))) %>%
  select(document, terms) %>%
  unique()

# revisit the topic models and created the stemmed word topic models
top_terms_by_topic_LDA(cleaned_documents$terms, number_of_topics = 3)
```

```
### REAL Headlines
# can just use the same function as above

real_headlines$title <-str_replace_all(real_headlines$title,"[^[:graph:]]", " ")

# pay attention to column names
real_headlines_Corpus <- Corpus(VectorSource(real_headlines$title))
realDTM <- DocumentTermMatrix(real_headlines_Corpus)

# convert the document term matrix to a tidytext corpus
realDTM_tidy <- tidy(realDTM)

# remove stopwords from the dataset of headlines
realDTM_tidy_cleaned <- realDTM_tidy %>% # take our tidy dtm and...
  anti_join(stop_words, by = c("term" = "word"))

# recostruct cleaned documents (so that each word shows up the correct number of times)
cleaned_real_headlines <- realDTM_tidy_cleaned %>%
  group_by(document) %>%
  mutate(terms = toString(rep(term, count))) %>%
  select(document, terms) %>%
  unique()

# check out what the cleaned documents look like
head(cleaned_real_headlines)

top_terms_by_topic_LDA(cleaned_real_headlines, number_of_topics = 2)
top_terms_by_topic_LDA(cleaned_real_headlines, number_of_topics = 4)

# stem the words (e.g. convert each word to its stem, where applicable)
realDTM_tidy_cleaned <- realDTM_tidy_cleaned %>%
  mutate(stem = wordStem(term))

# reconstruct our documents
cleaned_real_headlines <- realDTM_tidy_cleaned %>%
  group_by(document) %>%
  mutate(terms = toString(rep(stem, count))) %>%
  select(document, terms) %>%
  unique()

# revisit the topic models and created the stemmed word topic models
top_terms_by_topic_LDA(cleaned_real_headlines$terms, number_of_topics = 3)


### FAKE Headlines
# can just use the same function as above

fake_headlines$title <-str_replace_all(fake_headlines$title,"[^[:graph:]]", " ")
```

```
# pay attention to column names
fake_headlines_Corpus <- Corpus(VectorSource(fake_headlines$title))
fakeDTM <- DocumentTermMatrix(fake_headlines_Corpus)

# convert the document term matrix to a tidytext corpus
fakeDTM_tidy <- tidy(fakeDTM)

# remove stopwords from the dataset of reviews
fakeDTM_tidy_cleaned <- fakeDTM_tidy %>% # take our tidy dtm and...
  anti_join(stop_words, by = c("term" = "word"))

# recostruct cleaned documents (so that each word shows up the correct number of times)
cleaned_fake_headlines <- fakeDTM_tidy_cleaned %>%
  group_by(document) %>%
  mutate(terms = toString(rep(term, count))) %>%
  select(document, terms) %>%
  unique()

# check out what the cleaned documents look like (should just be a bynch of content words in alphabetical
order)
head(cleaned_fake_headlines)

top_terms_by_topic_LDA(cleaned_fake_headlines, number_of_topics = 2)
top_terms_by_topic_LDA(cleaned_fake_headlines, number_of_topics = 4)

# stem the words (e.g. convert each word to its stem, where applicable)
fakeDTM_tidy_cleaned <- fakeDTM_tidy_cleaned %>%
  mutate(stem = wordStem(term))

# reconstruct our documents
cleaned_fake_headlines <- fakeDTM_tidy_cleaned %>%
  group_by(document) %>%
  mutate(terms = toString(rep(stem, count))) %>%
  select(document, terms) %>%
  unique()

# removing punctuation as it was showing up in the LDA models
cleaned_fake_headlines$terms <- gsub("[^A-Za-z ]", "", cleaned_fake_headlines$terms)

# revisit the topic models and created the stemmed word topic models
top_terms_by_topic_LDA(cleaned_fake_headlines$terms, number_of_topics = 3)


#### now let's do some sentiment

#create a rowid for the googleheadlines
google_df <-google_headlines %>% mutate(id = row_number())
head(google_df)
# define the lexicon and any changes needed for our content
#get n rows–to see what we have in the lexicon
# Tyler Rinker is the author of sentimentr
```

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

#words to replace - didn't have any here
replace_in_lexicon <-tribble(~x, ~y,)

# create a new lexicon with modified sentiment
review_lexicon <-lexicon::hash_sentiment_jockers_rinker %>%
  filter(!x %in% replace_in_lexicon$x) %>%
  bind_rows(replace_in_lexicon) %>%
  setDT() %>%
  setkey("x")

# get sentence-level sentiment
google_sent_df <-google_df %>%
  get_sentences() %>%
  sentiment_by(by = c('headlines', 'id'), polarity_dt = review_lexicon)
head(google_sent_df)

# GloVe (Global Vectors for Word Representation). This step does two things.
# First, we calculate a contextualrepresentation of a word in a vector form
# Second we use an unsupervised learning on n-gram / dimensionality reduction

# create lists of reviews split into individual words (iterator over tokens)
tokens <-space_tokenizer(google_headlines$headlines %>%tolower() %>%removePunctuation())

# Create vocabulary. Terms will be unigrams (simple words).
it <-itoken(tokens, progressbar = FALSE)
vocab <-create_vocabulary(it)

# prune (remove) words that appear less than 3 times
vocab <-prune_vocabulary(vocab, term_count_min = 3L)

# Use our filtered vocabulary
vectorizer <-vocab_vectorizer(vocab)

# use skip gram window of 5 for context words
tcm <-create_tcm(it, vectorizer, skip_grams_window = 5L)

# fit the model
glove = GloVe$new(rank = 100, x_max = 5)
glove$fit_transform(tcm, n_iter = 20)

# get the processed word vector
word_vectors = glove$components

# Next, check which words have contextual similarity, expand to 20 if there are too many irrelevant words

# check for google
google<-word_vectors[, "google", drop = F]

# cosine similarity between word vectorstells us how similar they are
```

```
cos_sim = sim2(x = t(word_vectors), y = t(google), method = "cosine", norm = "l2")
head(sort(cos_sim[,1], decreasing = TRUE), 10)

# Next implement a quick t-SNE to visualize reviews by similarity of words

#create vector of words to keep, before applying tsne (remove stop words)
keep_words <-setdiff(colnames(word_vectors), stopwords())

# keep words in vector
word_vec <-word_vectors[, keep_words]

# prepare data frame to train
google_train_df <-data.frame(t(word_vec)) %>% rownames_to_column("word")

# train tsne for visualization
# perplexity is a parameter I can tune/adjust
# affects how tsne guesses how words lay in relation to previous words
tsne <-Rtsne(google_train_df[,-1], dims = 2, perplexity = 50, verbose=TRUE, max_iter = 500)

# Interpretation of a t-SNE plot is straightforward
# Similar objects(or words)appear nearby each other in the plot and
# dissimilar objects appear far away from each other.
# let's create the plot and examine it

# create plot
colors = rainbow(length(unique(google_train_df$word)))
names(colors) = unique(google_train_df$word)

plot_df <-data.frame(tsne$Y) %>% mutate(
  word = google_train_df$word,
  col = colors[google_train_df$word]
) %>% left_join(vocab, by = c("word" = "term")) %>%
  filter(doc_count >=7)

ggplot(plot_df, aes(X1, X2)) +
  geom_text(aes(X1, X2, label = word, color = col), size = 3) +
  xlab("") +
  ylab("") +
  theme(legend.position = "none")

# Step 8–calculate word level sentiment and overlay these on the t-SNE

head(google_df)
head(google_sent_df)
# calculate word-level sentiment
word_sent <-google_sent_df %>%
#  left_join(google_sent_df, by = "id") %>%
  select(id, headlines, ave_sentiment) %>%
  unnest_tokens(word, headlines) %>%
  group_by(word) %>%
  summarise(
```

```
    count = n(),
    avg_sentiment = mean(ave_sentiment),
    sum_sentiment = sum(ave_sentiment),
    sd_sentiment = sd(ave_sentiment)) %>%
  # remove stop words
  anti_join(stop_words, by = "word")

# filter to words that appear at least 3 times
pd_sent <-plot_df %>%
  left_join(word_sent, by = "word") %>%
  drop_na() %>%
  filter(count >= 3)

# plot the results
# size here represents the labels, want it to be easy to read
# x and y axis mean nothing
ggplot(pd_sent, aes(X1, X2)) +
  theme(plot.title = element_text(size = 20)) +
  geom_point(aes(X1, X2, size = count, alpha = .1, color = avg_sentiment)) +
  geom_text(aes(X1, X2, label = word), size = 4) +
  scale_colour_gradient2(low = muted("red"), mid = "white",high = muted("blue"), midpoint = 0) +
  scale_size(range = c(5, 20)) +
  xlab("") +
  ylab("") +
  ggtitle("t-SNE Plot with Sentiment for Google Headlines") +
  guides(color = guide_legend(title="Avg. Sentiment"), size = guide_legend(title = "Frequency"), alpha =
NULL) +
  scale_alpha(range = c(1, 1), guide = "none")



#### NOW lets do sentiment with the FAKE headlines
#create a rowid for the headlines
fake_df <-fake_headlines %>% mutate(id = row_number())


# define the lexicon and any changes needed for our content
#get n rows–to see what we have in the lexicon
# Tyler Rinker is the author of sentimentr
nrow(lexicon::hash_sentiment_jockers_rinker)#seems like 11,710 words

#words to replace - none here
replace_in_lexicon <-tribble(~x, ~y,)

# create a new lexicon with modified sentiment
review_lexicon <-lexicon::hash_sentiment_jockers_rinker %>%
  filter(!x %in% replace_in_lexicon$x) %>%
  bind_rows(replace_in_lexicon) %>%
  setDT() %>%
  setkey("x")
```

```
# get sentence-level sentiment
fake_sent_df <-fake_df %>%
  get_sentences() %>%
  sentiment_by(by = c('title', 'id'), polarity_dt = review_lexicon)

# GloVe (Global Vectors for Word Representation). This step does two things.
# First, we calculate a contextualrepresentation of a word in a vector form
# Second we use an unsupervised learning on n-gram / dimensionality reduction

# create lists of reviews split into individual words (iterator over tokens)
tokens <-space_tokenizer(fake_headlines$title %>%tolower() %>%removePunctuation())

# Create vocabulary. Terms will be unigrams (simple words).
it <-itoken(tokens, progressbar = FALSE)
vocab <-create_vocabulary(it)

# prune (remove) words that appear less than 3 times
vocab <-prune_vocabulary(vocab, term_count_min = 3L)

# Use our filtered vocabulary
vectorizer <-vocab_vectorizer(vocab)

# use skip gram window of 5 for context words
tcm <-create_tcm(it, vectorizer, skip_grams_window = 5L)

# 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

# cosine similarity between word vectorstells us how similar they are
cos_sim = sim2(x = t(word_vectors), method = "cosine", norm = "l2")
head(sort(cos_sim[,1], decreasing = TRUE), 10)

# Next implement a quick t-SNE to visualize reviews by similarity of words

#create vector of words to keep, before applying tsne (remove stop words)
keep_words <-setdiff(colnames(word_vectors), stopwords())

# keep words in vector
word_vec <-word_vectors[, keep_words]

# prepare data frame to train
fake_train_df <-data.frame(t(word_vec)) %>% rownames_to_column("word")

# train tsne for visualization
# perplexity is a parameter I can tune/adjust
# affects how tsne guesses how words lay in relation to previous words
tsne <-Rtsne(fake_train_df[,-1], dims = 2, perplexity = 50, verbose=TRUE, max_iter = 500)
```

```
# Interpretation of a t-SNE plot is straightforward
# Similar objects(or words)appear nearby each other in the plot and
# dissimilar objects appear far away from each other.
# let's create the plot and examine it

# create plot
colors = rainbow(length(unique(fake_train_df$word)))
names(colors) = unique(fake_train_df$word)

plot_df <-data.frame(tsne$Y) %>% mutate(
  word = fake_train_df$word,
  col = colors[fake_train_df$word]
) %>% left_join(vocab, by = c("word" = "term")) %>%
  filter(doc_count >= 70)

ggplot(plot_df, aes(X1, X2)) +
  geom_text(aes(X1, X2, label = word, color = col), size = 3) +
  xlab("") +
  ylab("") +
  theme(legend.position = "none")

# Step 8–calculate word level sentiment and overlay these on the t-SNE

head(fake_sent_df)

# calculate word-level sentiment
word_sent <-fake_sent_df %>%
  #left_join(given_sent_df, by = "id") %>%
  select(id, title, ave_sentiment) %>%
  unnest_tokens(word, title) %>%
  group_by(word) %>%
  summarise(
    count = n(),
    avg_sentiment = mean(ave_sentiment),
    sum_sentiment = sum(ave_sentiment),
    sd_sentiment = sd(ave_sentiment)) %>%
  # remove stop words
  anti_join(stop_words, by = "word")

# filter to words that appear at least 5 times
pd_sent <-plot_df %>%
  left_join(word_sent, by = "word") %>%
  drop_na() %>%
  filter(count >= 3)

# plot the results
# size here represents the labels, want it to be easy to read
# x and y axis mean nothing
ggplot(pd_sent, aes(X1, X2)) +
  theme(plot.title = element_text(size = 20)) +
```

```
  geom_point(aes(X1, X2, size = count, alpha = .1, color = avg_sentiment)) +
  geom_text(aes(X1, X2, label = word), size = 4) +
  scale_colour_gradient2(low = muted("red"), mid = "white",high = muted("blue"), midpoint = 0) +
  scale_size(range = c(5, 20)) +
  xlab("") +
  ylab("") +
  ggtitle("t-SNE Plot with Sentiment for Fake Headlines") +
  guides(color = guide_legend(title="Avg. Sentiment"), size = guide_legend(title = "Frequency"), alpha =
NULL) +
  scale_alpha(range = c(1, 1), guide = "none")




#### NOW lets do sentiment with the REAL headlines
#create a rowid for the headlines
real_df <-real_headlines %>% mutate(id = row_number())


# define the lexicon and any changes needed for our content
#get n rows–to see what we have in the lexicon
# Tyler Rinker is the author of sentimentr
nrow(lexicon::hash_sentiment_jockers_rinker)#seems like 11,710 words

#words to replace–none
replace_in_lexicon <-tribble(~x, ~y,)

# create a new lexicon with modified sentiment
review_lexicon <-lexicon::hash_sentiment_jockers_rinker %>%
  filter(!x %in% replace_in_lexicon$x) %>%
  bind_rows(replace_in_lexicon) %>%
  setDT() %>%
  setkey("x")

# get sentence-level sentiment
real_sent_df <-real_df %>%
  get_sentences() %>%
  sentiment_by(by = c('title', 'id'), polarity_dt = review_lexicon)

# GloVe (Global Vectors for Word Representation). This step does two things.
# First, we calculate a contextualrepresentation of a word in a vector form
# Second we use an unsupervised learning on n-gram / dimensionality reduction

# create lists of reviews split into individual words (iterator over tokens)
tokens <-space_tokenizer(real_headlines$title %>%tolower() %>%removePunctuation())

# Create vocabulary. Terms will be unigrams (simple words).
it <-itoken(tokens, progressbar = FALSE)
vocab <-create_vocabulary(it)

# prune (remove) words that appear less than 3 times
```

```
vocab <-prune_vocabulary(vocab, term_count_min = 3L)

# Use our filtered vocabulary
vectorizer <-vocab_vectorizer(vocab)

# use skip gram window of 5 for context words
tcm <-create_tcm(it, vectorizer, skip_grams_window = 5L)

# 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

# cosine similarity between word vectorstells us how similar they are
cos_sim = sim2(x = t(word_vectors), method = "cosine", norm = "l2")
head(sort(cos_sim[,1], decreasing = TRUE), 10)

# Next implement a quick t-SNE to visualize reviews by similarity of words

#create vector of words to keep, before applying tsne (remove stop words)
keep_words <-setdiff(colnames(word_vectors), stopwords())

# keep words in vector
word_vec <-word_vectors[, keep_words]

# prepare data frame to train
real_train_df <-data.frame(t(word_vec)) %>% rownames_to_column("word")

# train tsne for visualization
# perplexity is a parameter I can tune/adjust
# affects how tsne guesses how words lay in relation to previous words
tsne <-Rtsne(real_train_df[,-1], dims = 2, perplexity = 50, verbose=TRUE, max_iter = 500)

# Interpretation of a t-SNE plot is straightforward
# Similar objects(or words)appear nearby each other in the plot and
# dissimilar objects appear far away from each other.
# let's create the plot and examine it

# create plot
colors = rainbow(length(unique(real_train_df$word)))
names(colors) = unique(real_train_df$word)

plot_df <-data.frame(tsne$Y) %>% mutate(
  word = real_train_df$word,
  col = colors[real_train_df$word]
) %>% left_join(vocab, by = c("word" = "term")) %>%
  filter(doc_count >= 10)

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

```
  geom_text(aes(X1, X2, label = word, color = col), size = 3) +
  xlab("") +
  ylab("") +
  theme(legend.position = "none")

# Step 8–calculate word level sentiment and overlay these on the t-SNE

head(real_sent_df)

# calculate word-level sentiment
word_sent <-real_sent_df %>%
  #left_join(given_sent_df, by = "id") %>%
  select(id, title, ave_sentiment) %>%
  unnest_tokens(word, title) %>%
  group_by(word) %>%
  summarise(
    count = n(),
    avg_sentiment = mean(ave_sentiment),
    sum_sentiment = sum(ave_sentiment),
    sd_sentiment = sd(ave_sentiment)) %>%
  # remove stop words
  anti_join(stop_words, by = "word")

# filter to words that appear at least 5 times
pd_sent <-plot_df %>%
  left_join(word_sent, by = "word") %>%
  drop_na() %>%
  filter(count >= 5)

# plot the results
# size here represents the labels, want it to be easy to read
# x and y axis mean nothing
ggplot(pd_sent, aes(X1, X2)) +
  theme(plot.title = element_text(size = 20)) +
  geom_point(aes(X1, X2, size = count, alpha = .1, color = avg_sentiment)) +
  geom_text(aes(X1, X2, label = word), size = 4) +
  scale_colour_gradient2(low = muted("red"), mid = "white",high = muted("blue"), midpoint = 0) +
  scale_size(range = c(5, 20)) +
  xlab("") +
  ylab("") +
  ggtitle("t-SNE Plot with Sentiment for Real Headlines") +
  guides(color = guide_legend(title="Avg. Sentiment"), size = guide_legend(title = "Frequency"), alpha =
NULL) +
  scale_alpha(range = c(1, 1), guide = "none")
```