EDS 231 Topic 4: Sentiment Analysis II Assignment

Mia Forsline

2022-04-26

Read in the Tweet data

```r
raw_tweets <- read.csv("https://raw.githubusercontent.com/MaRo406/EDS_231-text-sentiment/main/dat/IPCC_

dat <- raw_tweets[,c(4,6)]

tweets <- tibble(text = dat$Title,
                  id = seq(1:length(dat$Title)),
                 date = as.Date(dat$Date,'%m/%d/%y'))
```

1. Think about how to further clean a twitter data set. Let's assume that the mentions of twitter accounts is not useful to us. Remove them from the text field of the tweets tibble.

Clean up the Tweet data by removing mentions of Twitter accounts

```r
#let's clean up the URLs from the tweets
tweets$text <- gsub("http[^[:space:]]*", "",tweets$text) #pull out https and replace them with nothing

tweets$text <- gsub("@*", "",tweets$text) #pull out @ symbol to remove mentions

tweets$text <- str_to_lower(tweets$text) #convert text to lowercase
```

Load Bing and NRC sentiment lexicons

```r
bing_sent <- get_sentiments('bing')
nrc_sent <- get_sentiments('nrc')
```

Tokenize the Tweets into individual words

```r
#break it down to one word per row to pull out stop words and identify sentiment words
words <- tweets %>%
  select(id, date, text) %>%
  unnest_tokens(output = word, input = text, token = "words") %>%
  anti_join(stop_words, by = "word") %>%
  left_join(bing_sent, by = "word") %>%
  left_join(
    tribble(
      ~sentiment, ~sent_score,
      "positive", 1,
      "negative", -1),
    by = "sentiment")
```

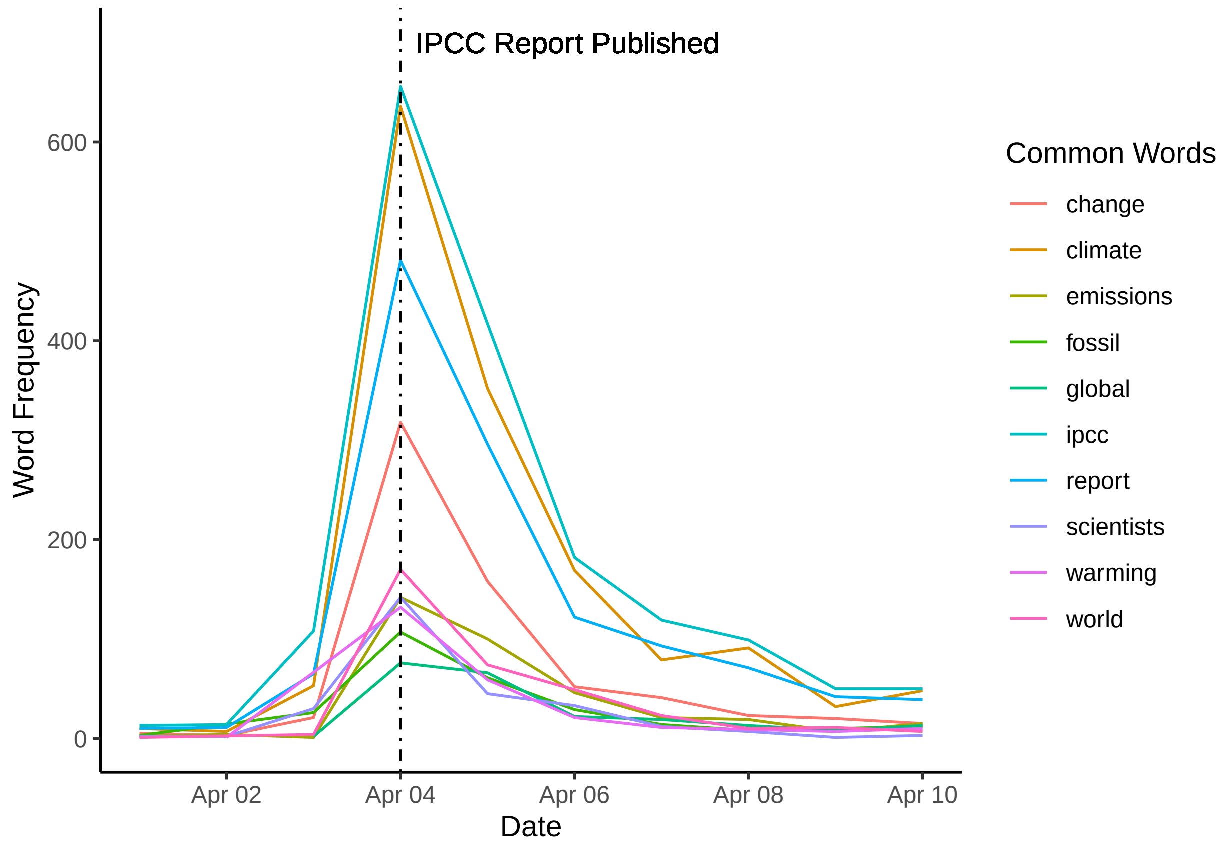
2. Compare the ten most common terms in the tweets per day. Do you notice anything interesting?

- identify the top 10 most common words in all tweets for the entire time period
- graph

```
#identify the top 10 most common words
common_words <- words %>%
  count(word) %>%
  arrange(desc(n)) %>%
  slice_head(n = 10)

#create a string of the top 10 most common words
common_words_string <- common_words$word

#subset the dataframe to include only the 10 most common words
common_words_over_time <- words %>%
  subset(word %in% c(common_words_string))

#count how many times each common word is used
common_words_count <- common_words_over_time %>%
  group_by(date) %>%
  count(word)
```

Plot the common word frequency

```
ggplot(data = common_words_count, aes(x = date, y = n, color = word)) +
  geom_line() +
  theme_classic() +
  labs(x = "Date",
       y = "Word Frequency",
       color = "Common Words") +
  geom_vline(xintercept = as.numeric(as.Date("2022-04-04")), linetype=4) +
  geom_text(aes(x = as.Date("2022-04-04"),
                label = "IPCC Report Published",
                y=700),
            colour="black",
            angle=0,
            hjust = -0.05,
            text=element_text(size=10))
```

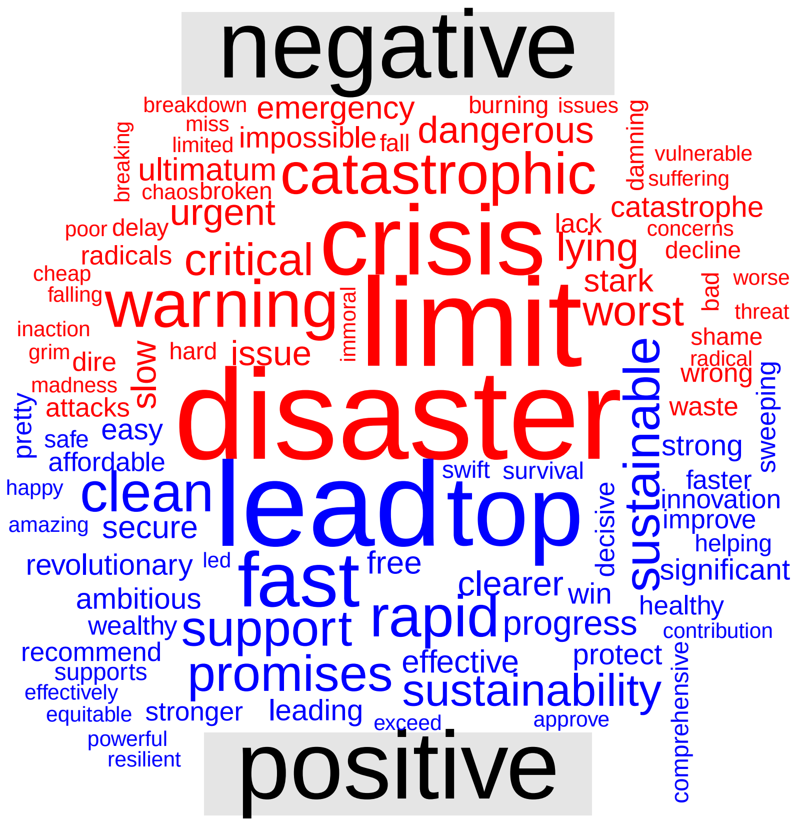


After graphing the daily frequency of the top ten most common terms in all the tweets, I noticed that the frequency of all ten words sharply increased on April 4, 2022, which is when the IPCC report was published. This makes sense because the day the report was published was when most people were Tweeting about it. Then the frequencies drop off after April 4 as people become less interested in discussing it on Twitter.

3. Adjust the wordcloud in the "wordcloud" chunk by coloring the positive and negative words so they are identifiable.

Comparison word cloud

```
words %>%
inner_join(get_sentiments("bing")) %>%
count(word, sentiment, sort = TRUE) %>%
acast(word ~ sentiment, value.var = "n", fill = 0) %>%
comparison.cloud(colors = c("red", "blue"),
                 max.words = 100)
```



4. Let's say we are interested in the most prominent entities in the Twitter discussion. Which are the top 10 most tagged accounts in the data set. Hint: the "explore_hashtags" chunk is a good starting point.

Use `quanteda` to tokenize and clean Tweet words

```
corpus <- corpus(dat$Title) #enter quanteda

tokens <- tokens(corpus) #tokenize the text so each doc (page, in this case) is a list of tokens (words

#clean it up
tokens <- tokens(tokens, remove_punct = TRUE, #remove punctuation
                 remove_numbers = TRUE) #remove numbers

#remove wtop words using the lexicon built into quanteda
tokens <- tokens_select(tokens, stopwords('english'), selection='remove')

#change all words to lower case
tokens <- tokens_tolower(tokens)
```

Mentions in tweets

Twitter Account	Frequency
@ipcc_ch	131
@logicalindians	38
@antonioguterres	16
@nytimes	14
@yahoo	14
@potus	13
@un	12
@youtube	11
@conversationedu	10
@ipcc	9

```r
#tokenize but keep only a particular pattern (in this case, the @ symbol)
mention_tweets <- tokens(corpus, remove_punct = TRUE) %>%
                  tokens_keep(pattern = "@*") #@ followed by any other string

dfm_mention <- dfm(mention_tweets) #shows location of each tweet in the corpus (document feature matrix

#identify the most frequently mentioned Twitter accounts
tstat_freq <- textstat_frequency(dfm_mention, n = 10)

#display the most frequently mentioned Twitter accounts in a table
tstat_freq %>% as.data.frame() %>%
  select(-rank, -group, -docfreq) %>%
  rename("Twitter Account" = feature,
         Frequency = frequency) %>%
  kableExtra::kbl() %>%
  kable_styling()
```

5. The Twitter data download comes with a variable called "Sentiment" that must be calculated by Brandwatch. Use your own method to assign each tweet a polarity score (Positive, Negative, Neutral) and compare your classification to Brandwatch's (hint: you'll need to revisit the "raw_tweets" data frame).

First, use the Brandwatch classifications to determine Tweet polarity scores

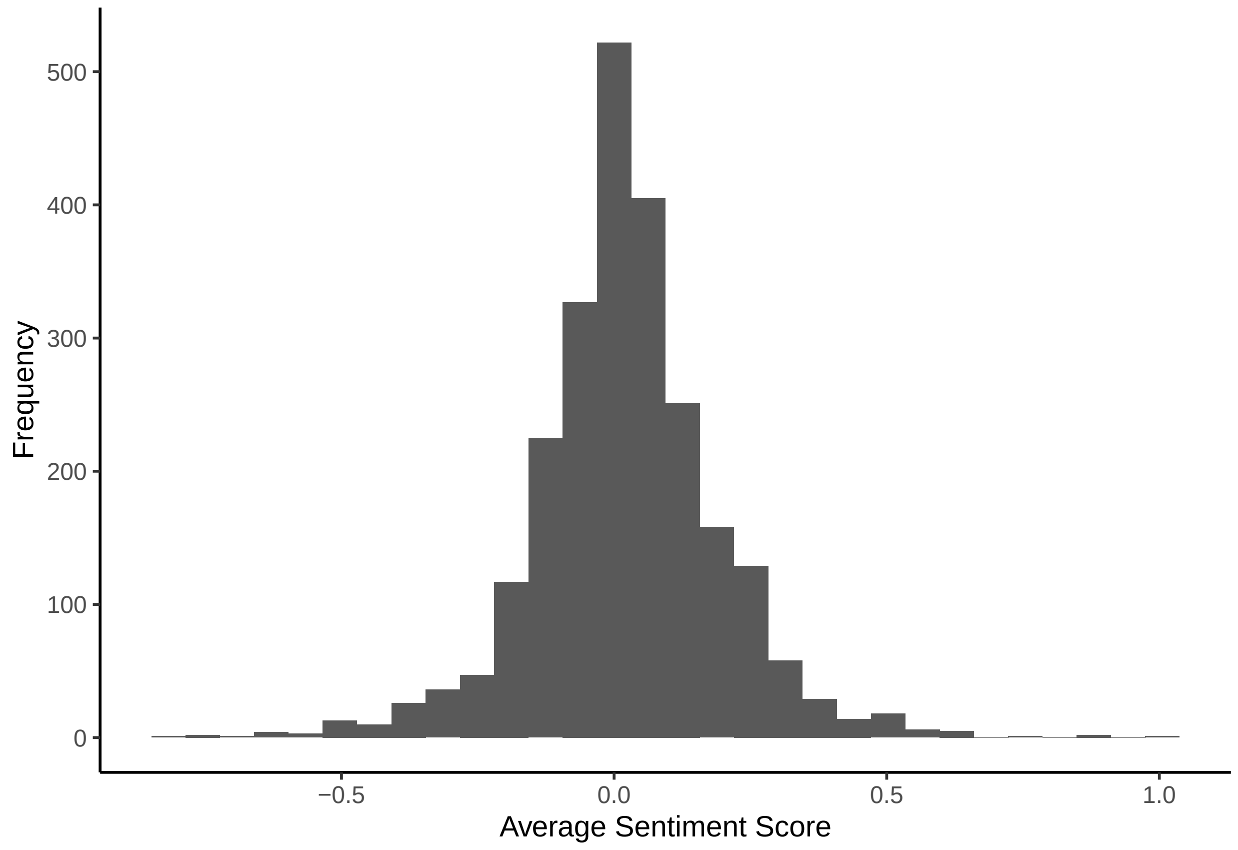
```r
raw_tweets <- read.csv("https://raw.githubusercontent.com/MaRo406/EDS_231-text-sentiment/main/dat/IPCC_

dat <- raw_tweets[,c(4,6, 10)]

tweets <- tibble(text = dat$Title,
                 id = seq(1:length(dat$Title)),
                 date = as.Date(dat$Date,'%m/%d/%y'),
                 sentiment = dat$Sentiment)

#assign 1,0,-1 to Tweets based on pos/neutral/neg Sentiment categories
tweets <- tweets %>%
  as.data.frame() %>%
  mutate(sentiment = as.factor(sentiment)) %>%
  mutate(
    polarity = ifelse(sentiment == "positive", 1,
                      ifelse(sentiment == "neutral", 0, -1)
                      )
        )

#count how many pos/neutral/neg tweets were calculated by Bandwatch
brandwatch_polarity_freq <- tweets %>%
  as.data.frame() %>%
  count(polarity) %>%
  arrange(desc(n)) %>%
  mutate(polarity = factor(polarity, levels = c("1", "0", "-1")))
```

Second, calculate average sentiment scores for each Tweet

```
raw_tweets <- read.csv("https://raw.githubusercontent.com/MaRo406/EDS_231-text-sentiment/main/dat/IPCC_

dat <- raw_tweets[,c(4,6, 10)]

tweets <- tibble(text = dat$Title,
                 id = seq(1:length(dat$Title)),
                 date = as.Date(dat$Date,'%m/%d/%y'),
                 sentiment = dat$Sentiment)

#list of characters
mytext <- get_sentences(tweets$text)

#sentiments from the text
mysent <- sentiment(mytext) %>%
  group_by(element_id) %>%
  summarise(avg_sentiment = mean(sentiment))
```

Plot to examine the distribution of average sentiment to best understand how to assign polarity scores

```
ggplot(data = mysent, aes(x = avg_sentiment)) +
  geom_histogram() +
  theme_classic() +
  labs(x = "Average Sentiment Score",
       y = "Frequency")
```

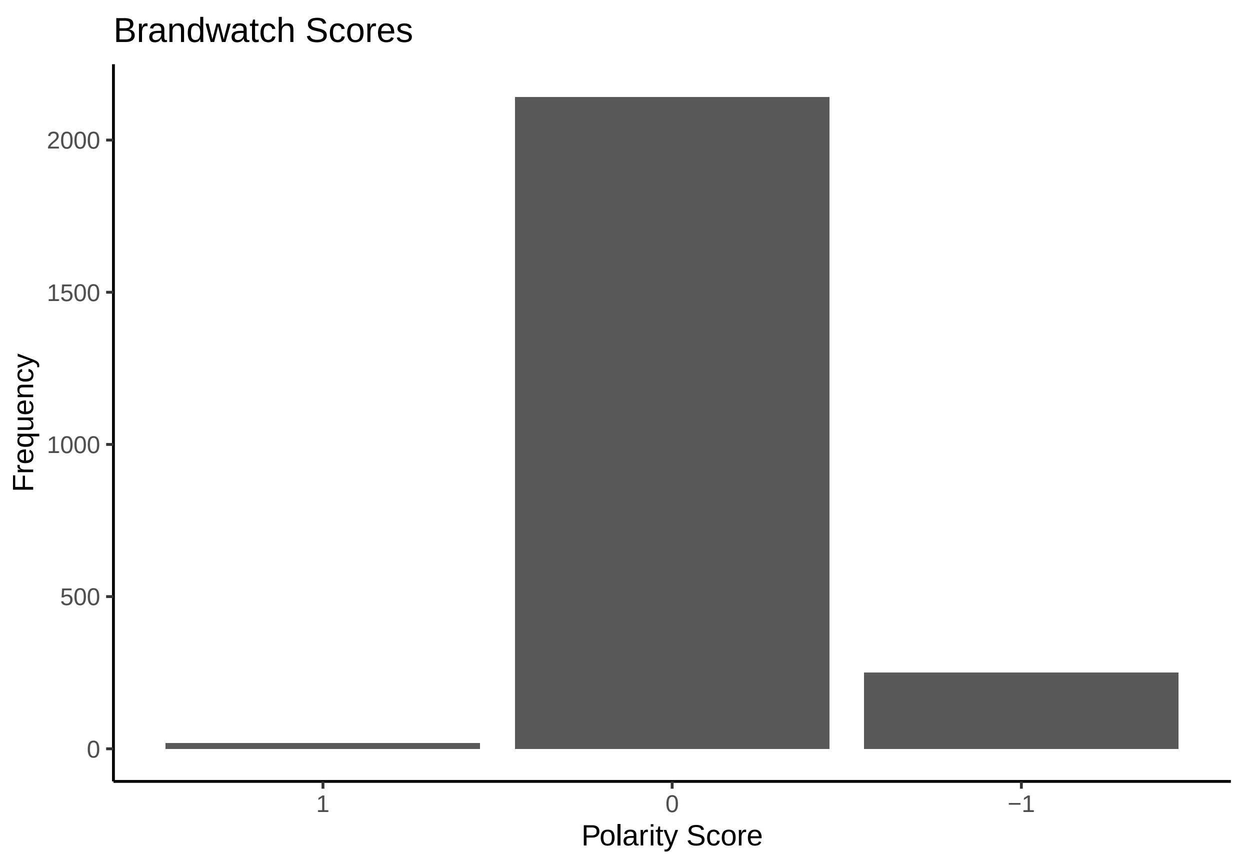


Assign polarity scores

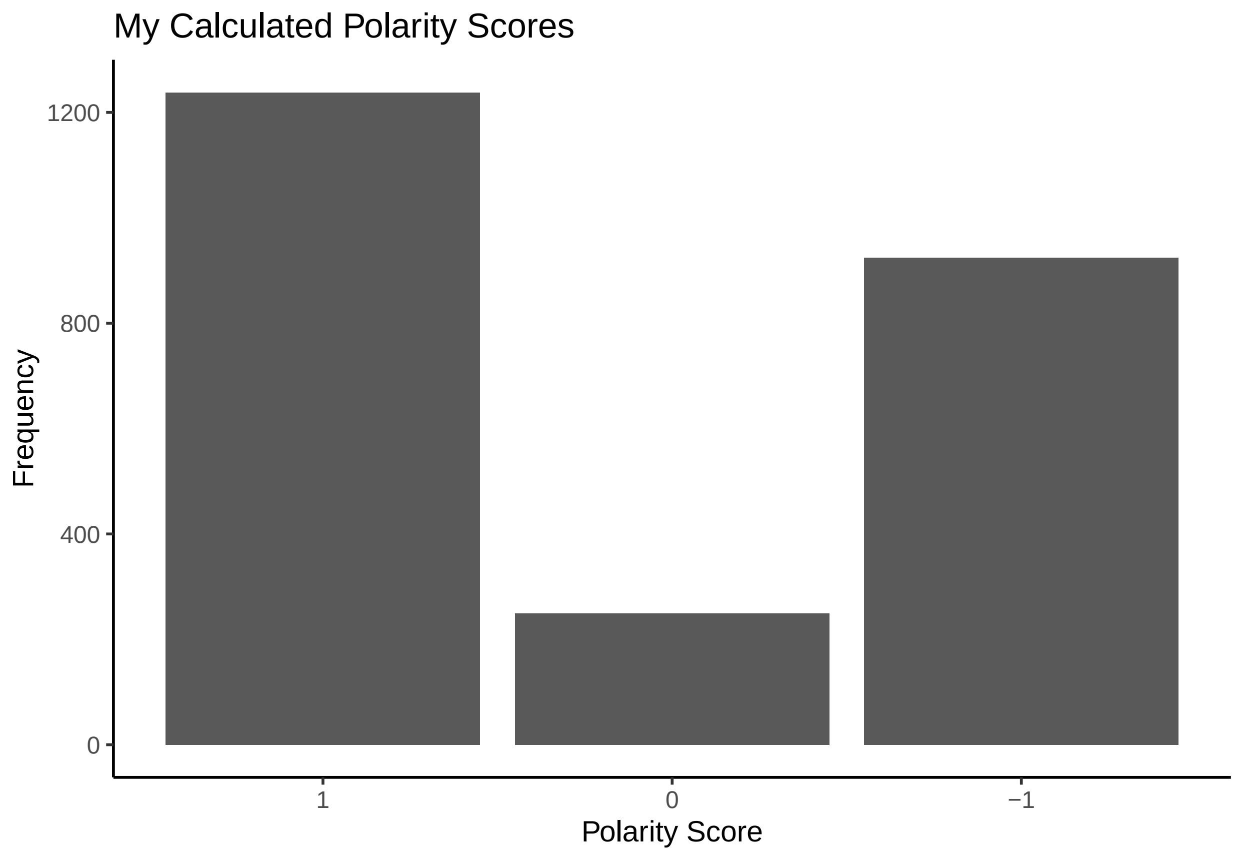
```
#positive: [0.33, 1)
#neutral: [-0.33, 0.33]
#negative: (-1, -0.33]
mysent$polarity <- ifelse(mysent$avg_sentiment < 0, -1,
                        ifelse(mysent$avg_sentiment > 0, 1, 0))

#change polarity to an ordered factor
mysent <- mysent %>%
  mutate(polarity = factor(polarity, levels = c("1", "0", "-1")))

#count how many pos/neutral/neg tweets there are
my_freq <- mysent %>%
  count(polarity) %>%
  arrange(desc(n))
```

Finally, compare the polarity scores I calcualted against Brandwatch's polarity scores

```
ggplot(data = brandwatch_polarity_freq, aes(x = polarity, y = n)) +
  geom_col() +
  theme_classic() +
  labs(x = "Polarity Score",
       y = "Frequency",
       title = "Brandwatch Scores")
```



```
ggplot(data = my_freq, aes(x = polarity, y = n)) +
  geom_col() +
  theme_classic() +
  labs(x = "Polarity Score",
       y = "Frequency",
       title = "My Calculated Polarity Scores")
```



The way I classified polarity scores resulted in much more positive and negative words and few neutral words. In contrast, Brandwatch's classification technique resulted in many neutral words compared to positive or negative words.