Sentiment Analysis 2

Benjamin Moscona

4/20/2022

Assignment

You will use the tweet data from class today for each part of the following assignment.

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.

```
tweets$text <- str_remove(tweets$text, "@.*")
```

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

```
#tokenize tweets to individual 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")

words %>%
  group_by(word) %>%
  summarize(n = n()) %>%
  arrange(desc(n))

## # A tibble: 5,261 × 2
##    word           n
##    <chr>      <int>
##  1 ipcc        1577
##  2 climate     1378
##  3 report      1084
##  4 change       621
##  5 world        332
##  6 emissions    326
##  7 scientists   269
##  8 fossil       255
##  9 warming      233
```

```
## 10 global        199
## # … with 5,251 more rows
```

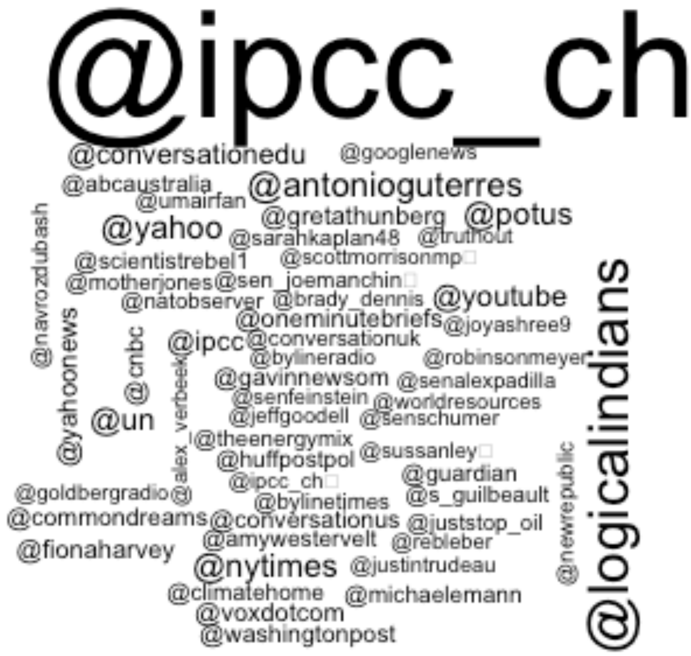
All 10 words are related directly to the IPCC report or climate change. “ipcc” is mentioned most, followed by “climate” and “report”. None of this is particularly surprising to me.

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

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

## Joining, by = "word"

## Joining, by = c("word", "sentiment").
```

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.

```r
hash_tweets <- tokens(corpus, remove_punct = TRUE) %>%
               tokens_keep(pattern = "@*")

dfm_hash<- dfm(hash_tweets)

tstat_freq <- textstat_frequency(dfm_hash, n = 100)
head(tstat_freq, 10)

##             feature frequency rank docfreq group
## 1          @ipcc_ch       131    1     131   all
## 2   @logicalindians        38    2      38   all
## 3  @antonioguterres        16    3      16   all
## 4          @nytimes        14    4      14   all
## 5            @yahoo        14    4      14   all
## 6            @potus        13    6      13   all
## 7               @un        12    7      12   all
## 8          @youtube        11    8      11   all
## 9  @conversationedu        10    9      10   all
## 10            @ipcc         9   10       9   all

#tidytext gives us tools to convert to tidy from non-tidy formats
hash_tib<- tidy(dfm_hash)

hash_tib %>%
   count(term) %>%
   with(wordcloud(term, n, max.words = 100))
```



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).

```
dat<- raw_tweets[,c(4,6, 10)] # Extract Date and Title fields

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

#let's clean up the URLs from the tweets
tweets$text <- gsub("http[^[:space:]]*", "",tweets$text)
tweets$text <- iconv(tweets$text, "latin1", "ASCII", sub="")
tweets$text <- str_to_lower(tweets$text)

#load sentiment lexicons
bing_sent <- get_sentiments('bing')
nrc_sent <- get_sentiments('nrc')

#tokenize tweets to individual words
words <- tweets %>%
```

```
  select(id, date, text, sentiment_brandwatch) %>%
  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")

compare_sents <- words %>%
  mutate(sent_score = replace_na(sent_score, 0)) %>%
  group_by(id) %>%
  summarize(sentiment = mean(sent_score),
            sentiment_brandwatch = first(sentiment_brandwatch)) %>%
  mutate(sentiment = case_when(sentiment >= 0.2 ~ "positive",
    sentiment < 0.2 & sentiment > -0.2 ~ "neutral",
    sentiment <= -0.2 ~ "negative"))

table(compare_sents$sentiment, compare_sents$sentiment_brandwatch)
```

```
##
##             negative neutral positive
##   negative       70     112        0
##   neutral       180    2009       18
##   positive        0      15        1
```

My method of taking the mean after assigning neutral words a zero and then splitting at -0.2 to 0.2 for neutral returns fairly similar results to the Brandwatch sentiment score. While both models are fairly similar for neutral and negative tweets, they disagree on which ones should be labeled positive vs. neutral. There aren't many particularly positive tweets in this set which is why there may be less agreement there.