topic 3: sentiment analysis i

Mia Forsline

4/13/2022

Introduction

Sentiment analysis is a tool for assessing the mood of a piece of text. For example, we can use sentiment analysis to understand public perceptions of topics in environmental policy like energy, climate, and conservation.

This assignment is designed to teach the user more about sentiment analysis using tools such as the Nexis Uni database through the UCSB Library.

0. Using the "IPCC" Nexis Uni data set from the class presentation and the pseudo code we discussed, recreate Figure 1A from Froelich et al. (Date x # of 1) positive, 2) negative, 3) neutral headlines):

- read in the IPCC Nexis Uni data set
- extract 100 headlines
- clean the dataset
- extract headlines text
- determine sentiments of each headline

```r
my_files <- list.files(pattern = "Nexis_IPCC_Results.docx",
                       path = here("data"),
                       full.names = TRUE,
                       recursive = TRUE,
                       ignore.case = TRUE)

#use lnt_read to read in a LexisNexis file then convert to an object of class 'LNT output'
dat <- lnt_read(my_files)

meta_df <- dat@meta
articles_df <- dat@articles
paragraphs_df <- dat@paragraphs

dat2 <- data_frame(element_id = seq(1:length(meta_df$Headline)),
                   Date = meta_df$Date,
                   Headline = meta_df$Headline) %>%
        janitor::clean_names()

#remove duplicate headlines, if necessary (in this case, it's not)
```

```
dat2_clean <- dat2[!duplicated(dat2$headline), ]

#list of characters
mytext <- get_sentences(dat2$headline)

#sentiments from the text
sent <- sentiment(mytext)

#create dataframe of sentiments of each sentence
sentiment_df <- inner_join(dat2, sent, by = "element_id")

sentiment <- sentiment_by(sentiment_df$headline)

#display table of sentences ordered by sentiment
# sentiment_df %>%
#   arrange(sentiment)
```

- calculate polarity and assign values of -1, 0, or 1 to indicate negative, neutral, or positive

```
sentiment_df$polarity <- ifelse(sentiment_df$sentiment <0, -1, ifelse(sentiment_df$sentiment > 0, 1, 0)

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

- recreate a similar graph to figure 1A from Froelich et al.

```
ggplot(data = sentiment_df, aes(x = date, color = polarity)) +
  stat_count(geom = 'line', aes(y = ..count..)) +
  labs(y = "Developed Media Sentiment (no. headlines)",
       x = "Date",
       color = "Sentiment",
       title = "IPCC Nexis Uni Data Sentiment from April 4 - 11") +
  scale_color_manual(values = c("blue", "gray", "red"),
                     labels = c("Positive", "Neutral", "Negative")) +
  theme_classic()
```

1. Access the Nexis Uni database through the UCSB library

2. Choose a key search term or terms to define a set of articles.

I searched for the term “chytrid” from 2020 - 2022 and filtered my results to include only News.

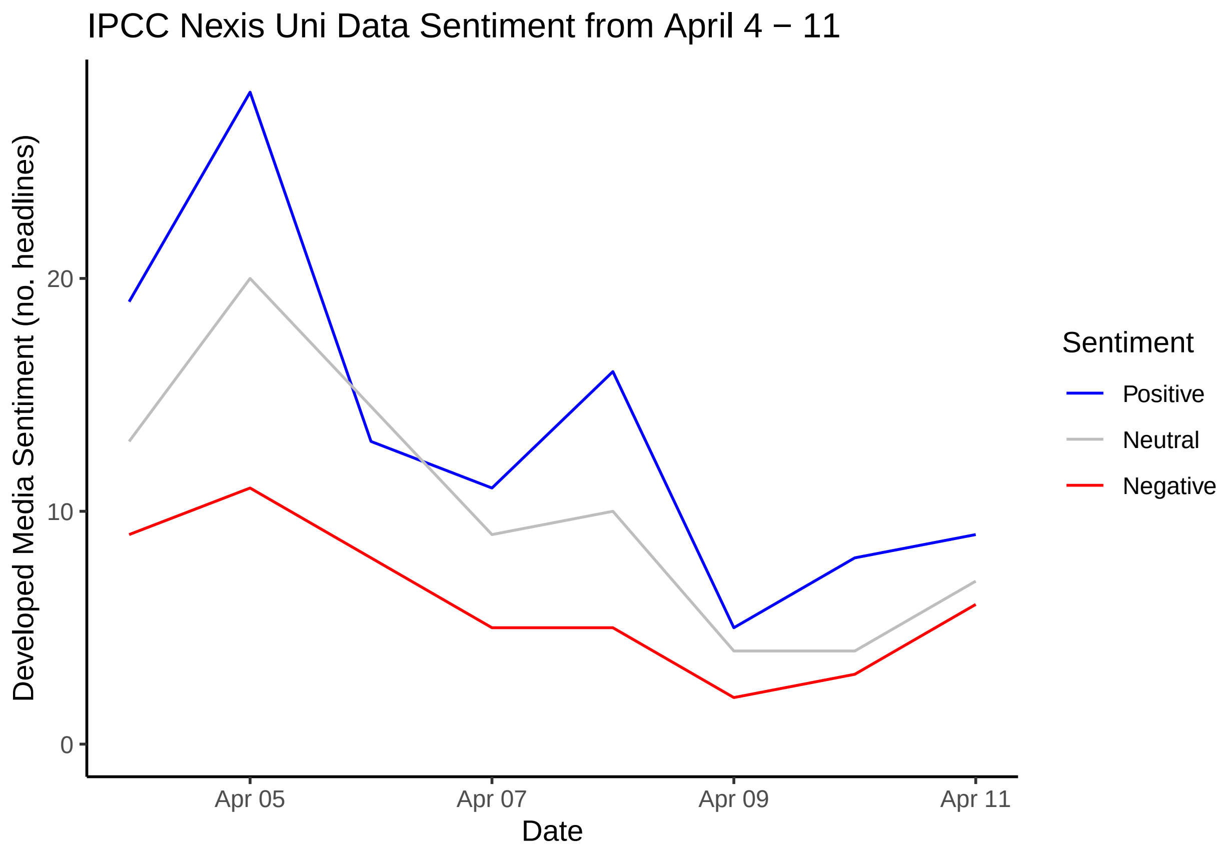


Figure 1: Sentiment over time based on headlines with negative (red), neutral (grey), and positive (blue) titles.

3. Use your search term along with appropriate filters to obtain and download a batch of at least 100 full text search results (.docx).

4. Read your Nexis article document into RStudio.

```r
my_files <- list.files(pattern = "chytrid.docx",
                       path = here("data"),
                       full.names = TRUE,
                       recursive = TRUE,
                       ignore.case = TRUE)

#use lnt_read to read in a LexisNexis file then convert to an object of class 'LNT output'
dat <- lnt_read(my_files)

meta_df <- dat@meta
articles_df <- dat@articles
paragraphs_df <- dat@paragraphs
```

5. This time use the full text of the articles for the analysis. First clean any artifacts of the data collection process (hint: this type of thing should be removed: "Apr 04, 2022(Biofuels Digest: http://www.biofuelsdigest.com/ Delivered by Newstex"))

- extract full text
- clean up text to remove duplicates, etc.

```r
# extract full text from the articles
paragraphs_dat <- data_frame(element_id = paragraphs_df$Art_ID,
                             Text = paragraphs_df$Paragraph)

dat3 <- inner_join(dat2,paragraphs_dat, by = "element_id") %>%
  janitor::clean_names()

#remove duplicate text rows
dat3 <- dat3[!duplicated(dat3$text), ]

#remove blank text rows
dat3 <- subset(dat3, text != " ")

#remove text rows that say "Graphic"
dat3 <- subset(dat3, text != "Graphic")

#remove rows that contain text that indicate an image
dat3 <- dat3[!grepl(" - ", dat3$text),]

#remove text rows that have tag lines
dat3 <- dat3[!grepl("Apr 0", dat3$text),]
```

```r
#save text dataframe as CSV
write_csv(dat3, here::here("data", "chytrid_dat3.csv"))
```

6. Explore your data a bit and try to replicate some of the analyses above presented in class if you'd like (not necessary).

- start by using the Bing sentiment analysis lexicon

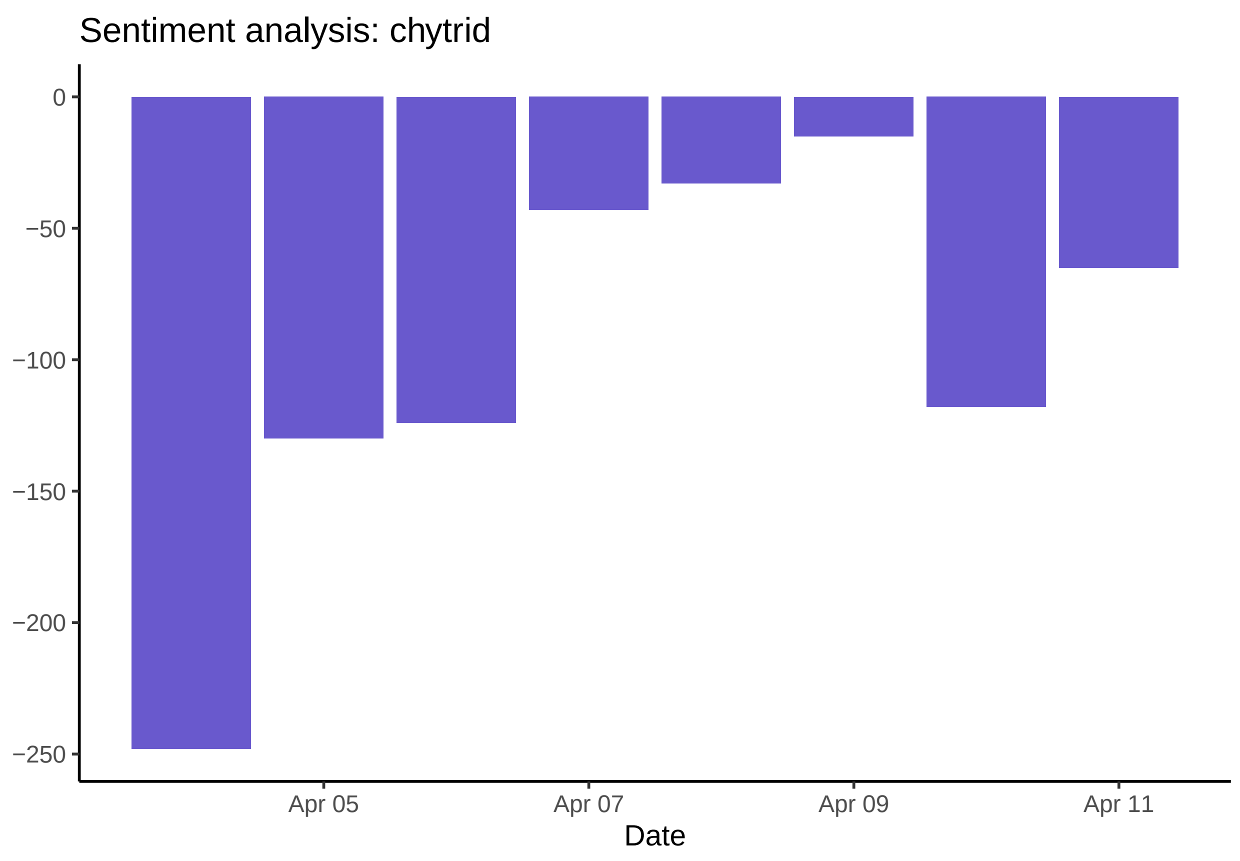
```r
bing_sent <- get_sentiments('bing') #grab the bing sentiment lexicon using the tidytext package

#head(bing_sent, n = 20)
```

- unnest the text to the word level so we can label the individual sentiment words
- remove stop words as standard text cleaning procedure

```r
chytrid_text <- read_csv(here::here("data", "chytrid_dat3.csv"))

#unnest to word-level tokens, remove stop words, and join sentiment words
 text_words <- chytrid_text  %>%
  unnest_tokens(output = word, input = text, token = 'words')

 sent_words <- text_words%>% #break text into individual words
  anti_join(stop_words, by = 'word') %>% #returns only the rows without stop words
  inner_join(bing_sent, by = 'word') #joins and retains only sentiment words

 #head(sent_words)
```

- create a sentiment score by counting the number of sentiment words occurring per day
- We can center the scores around an offset point equal to the average page sentiment score. This lets us measure the sentiment of a given page relative to the overall sentiment of the book.
- in short, is this page more or less pos vs neg ?

```r
sent_scores <- sent_words %>%
  count(sentiment, date) %>%
  spread(sentiment, n) %>%
  mutate(raw_score = positive - negative)

ggplot(sent_scores, aes(x = date)) +
  theme_classic() +
  geom_bar(aes(y = raw_score), stat = 'identity', fill = 'slateblue3') +
  #coord_flip() +
  theme(axis.title.y = element_blank()) +
  labs(title = "Sentiment analysis: chytrid",
       y = "Raw sentiment score",
       x = "Date")
```



7. Plot the amount of emotion words (the 8 from nrc) as a percentage of all the emotion words used each day (aggregate text from articles published on the same day). How does the distribution of emotion words change over time? Can you think of any reason this would be the case?

- use NRC Lexicon to look at the most common sentiment words in the data set

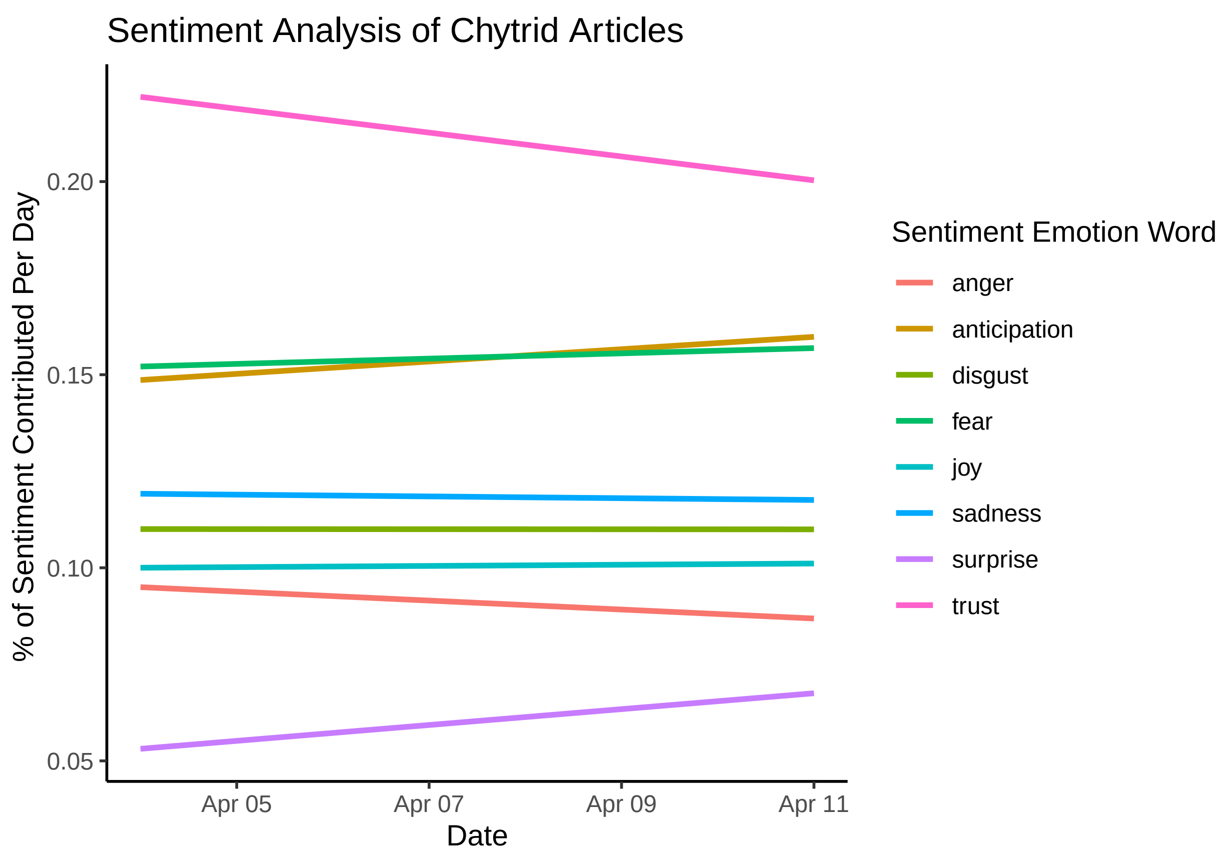
```
nrc_sent <- get_sentiments('nrc') #requires downloading a large dataset via prompt
#unique(nrc_sent$sentiment)

nrc_word_counts <- text_words %>%
  inner_join(get_sentiments("nrc")) %>%
  count(word, sentiment, date, sort = TRUE) %>%
  ungroup()

#remove "postive" and "negative" sentiments to leave only the 8 emotion words
nrc_word_counts <- subset(nrc_word_counts, sentiment != "positive" & sentiment != "negative")
```

- calculate the % of all emotion words used each day

```r
#count the number of words in each emotion category per day
nrc_word_counts_3 <- nrc_word_counts %>%
  group_by(date, sentiment) %>%
  summarise(total_n = sum(n)) %>%
  spread(sentiment, total_n) %>%
  ungroup()

#replace NA values with 0
nrc_word_counts_3[is.na(nrc_word_counts_3)] = 0

#calculate the total number of emotion words used each day
nrc_word_counts_3 <- nrc_word_counts_3 %>%
  mutate(total_n = anger + anticipation + disgust + fear + joy + sadness + surprise + trust)

#pivot longer to make a dataframe that can easily be used in ggplot()
nrc_word_counts_3 <- nrc_word_counts_3 %>%
  pivot_longer(cols = !c("date", "total_n"), names_to = "emotion", values_to = "n")

#calculate the %
nrc_word_counts_3 <- nrc_word_counts_3 %>%
  mutate(perc = n / total_n)
```

Plot the % of all emotion words used each day

```r
ggplot(data = nrc_word_counts_3, aes(x = date, y = perc, color = emotion)) +
  #geom_point(alpha = 0.25) +
  geom_smooth(method = lm, se = FALSE) +
  theme_classic() +
  labs(x = "Date",
       y = "% of Sentiment Contributed Per Day ",
       color = "Sentiment Emotion Word",
       title = "Sentiment Analysis of Chytrid Articles")
```


Sentiment Analysis of Chytrid Articles
% of Sentiment Contributed Per Day
0.20
0.15
0.10
0.05
Apr 05
Apr 07
Apr 09
Apr 11
Date
Sentiment Emotion Word
anger
anticipation
disgust
fear
joy
sadness
surprise
trust