Assignment7-Week10-SentimentAnalysis

Ross Boehme

2023-04-02

Assignment 7

- In Text Mining with R, Chapter 2 looks at Sentiment Analysis. In this assignment, you should start by getting the primary example code from chapter 2 working in an R Markdown document. You're then asked to extend the code in two ways:

1. Work with a different corpus of your choosing
2. Incorporate at least one additional sentiment lexicon (possibly from another R package that you've found through research).

Initial Imports and Citations

```
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 4.2.3
```

```
library(janeaustenr)
```

```
## Warning: package 'janeaustenr' was built under R version 4.2.3
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0     v purrr   1.0.1
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.4     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(textdata)
```

```
## Warning: package 'textdata' was built under R version 4.2.3
```

Please note the base code for this project was taken from "Text Mining with R: A Tidy Approach" by Julia Silge and David Robinson. Sebastopol, CA: O'Reilly Media, 2017. ISBN 978-1-491-98165-8. Code below in named chunks "silge-robinson" and "silge-robinson2".

This first R chunk code shows how to "mine" words from Jane Austen's bibliography, with each observation in the dataframe a unique Book, Chapter, Line Number, and Word combination.

```
tidy_books <- austen_books() %>%
  group_by(book) %>%
  mutate(
    linenumber = row_number(),
    chapter = cumsum(str_detect(text,
                                regex("^chapter [\\divxlc]",
                                      ignore_case = TRUE)))) %>%
  ungroup() %>%
  unnest_tokens(word, text)
```

This second R chunk shows how to load three sentiment lexicons: afinn, bing, and NRC which I'd also like to cite (linked below). - afinn - bing - nrc

```
afinn <- get_sentiments('afinn')
bing <- get_sentiments('bing')
nrc <- get_sentiments('nrc')
```

I will use this base as inspiration for my own analysis.

Extending Base Code With Same Corpus

I will look at a the full available Jane Austen corpus to extend this code, evaluating which of the author's books have the highest percentage of neutral, positive, and negative words.

Showing word count by book. Sense & Sensibility is the longest book. Persuasion is the shortest.

```
austen_word_counts <- as.data.frame(table(tidy_books$book))
colnames(austen_word_counts) <- c("book","total_word_count")
austen_word_counts
```

```
##                  book total_word_count
## 1 Sense & Sensibility           119957
## 2   Pride & Prejudice           122204
## 3      Mansfield Park           160460
## 4                Emma           160996
## 5    Northanger Abbey            77780
## 6          Persuasion            83658
```

Taking positive and negative word lexicon from nrc which I'll use to evaluate Jane Austen's corpus. Merging with tidy_books df created previously, then showing positive and negative word count by book.

```
nrc_pos_neg <- nrc %>%
  filter(sentiment == 'negative'| sentiment=='positive')

austen_sentiment <- tidy_books %>%
```

```r
  inner_join(nrc_pos_neg) %>%
  group_by(book,sentiment) %>%
  summarise(total_count=n(),.groups = 'drop') %>%
  as.data.frame() %>%
  pivot_wider(names_from = sentiment,
              values_from = total_count)
```

```
## Joining with `by = join_by(word)`
```

```
## Warning in inner_join(., nrc_pos_neg): Each row in `x` is expected to match at most 1 row in `y`.
## i Row 251 of `x` matches multiple rows.
## i If multiple matches are expected, set `multiple = "all"` to silence this
##   warning.
```

```r
colnames(austen_sentiment) <- c("book","negative_words","positive_words")

austen_sentiment
```

```
## # A tibble: 6 x 3
##   book                negative_words positive_words
##   <fct>                        <int>          <int>
## 1 Sense & Sensibility           3994           7314
## 2 Pride & Prejudice             3630           7273
## 3 Mansfield Park                4694           9269
## 4 Emma                          4438           9262
## 5 Northanger Abbey              2456           4520
## 6 Persuasion                    2226           5248
```

Merging total word count df with positive and negative word count df to assess % of all words which were positive or negative by book.

```r
austen_combined <- austen_word_counts %>%
  inner_join(austen_sentiment) %>%
  transform(perc_neg = scales::percent(negative_words / total_word_count),
            perc_pos = scales::percent(positive_words / total_word_count),
            neutral_words = total_word_count - negative_words - positive_words)
```

```
## Joining with `by = join_by(book)`
```

```r
austen_combined <- austen_combined %>%
  transform(perc_neutr = scales::percent(neutral_words / total_word_count))

austen_final <- austen_combined[,c('book','total_word_count','positive_words','negative_words','neutral_

austen_final
```

```
##                  book total_word_count positive_words negative_words
## 1 Sense & Sensibility           119957           7314           3994
## 2   Pride & Prejudice           122204           7273           3630
## 3      Mansfield Park           160460           9269           4694
```

```
## 4                Emma           160996          9262           4438
## 5    Northanger Abbey            77780          4520           2456
## 6          Persuasion            83658          5248           2226
##   neutral_words perc_pos perc_neg perc_neutr
## 1        108649   6.097%   3.330%    90.573%
## 2        111301   5.952%   2.970%    91.078%
## 3        146497   5.777%   2.925%    91.298%
## 4        147296   5.753%   2.757%    91.490%
## 5         70804   5.811%   3.158%    91.031%
## 6         76184   6.273%   2.661%    91.066%
```

The book with the highest percentage of positive words was: Persuasion. It also had the lowest percentage of negative words.

```
austen_final %>% arrange(desc(perc_pos))
```

```
##                  book total_word_count positive_words negative_words
## 1          Persuasion            83658          5248           2226
## 2 Sense & Sensibility           119957          7314           3994
## 3   Pride & Prejudice           122204          7273           3630
## 4    Northanger Abbey            77780          4520           2456
## 5      Mansfield Park           160460          9269           4694
## 6                Emma           160996          9262           4438
##   neutral_words perc_pos perc_neg perc_neutr
## 1         76184   6.273%   2.661%    91.066%
## 2        108649   6.097%   3.330%    90.573%
## 3        111301   5.952%   2.970%    91.078%
## 4         70804   5.811%   3.158%    91.031%
## 5        146497   5.777%   2.925%    91.298%
## 6        147296   5.753%   2.757%    91.490%
```

The book with the highest percentage of negative words was "Sense & Sensibility" which is interesting because it also had the second highest percentage of positive words. It therefore had fewer "neutral" words, making it a more emotionally charged book than the others.

```
austen_final %>% arrange(desc(perc_neg))
```

```
##                  book total_word_count positive_words negative_words
## 1 Sense & Sensibility           119957          7314           3994
## 2    Northanger Abbey            77780          4520           2456
## 3   Pride & Prejudice           122204          7273           3630
## 4      Mansfield Park           160460          9269           4694
## 5                Emma           160996          9262           4438
## 6          Persuasion            83658          5248           2226
##   neutral_words perc_pos perc_neg perc_neutr
## 1        108649   6.097%   3.330%    90.573%
## 2         70804   5.811%   3.158%    91.031%
## 3        111301   5.952%   2.970%    91.078%
## 4        146497   5.777%   2.925%    91.298%
## 5        147296   5.753%   2.757%    91.490%
## 6         76184   6.273%   2.661%    91.066%
```

Additional Sentiment Analysis

Per the assignment description, I will now perform an additional sentiment analysis using a different corpus and different lexicon than previously mentioned.

For the lexicon I will use SentimentAnalysis.R package description linked here.

For the corpus I will use the "friends" package from R where each observation is a piece of speech said by a character in the TV show Friends.

```
library(SentimentAnalysis)
```

```
## Warning: package 'SentimentAnalysis' was built under R version 4.2.3
```

```
##
## Attaching package: 'SentimentAnalysis'
```

```
## The following object is masked from 'package:base':
##
##     write
```

```
library(friends)
```

```
## Warning: package 'friends' was built under R version 4.2.3
```

```
friends_lines <- friends
```

Data Cleaning and Exploratory Data Analysis As the value count generated below shows, there are 699 characters who have speaking lines over the course of Friends. To ensure an adequate sample size, I'll only look at the six main characters who account for the vast majority of lines: Monica Geller, Rachel Green, Ross Geller, Chandler Bing, Joey Tribbiani, and Phoebe Buffay.

Per this dataframe, the characters with the most lines are: Rachel Green (9312), Ross Geller (9157), and Chandler Bing (8465).

```
length(table(friends_lines$speaker))
```

```
## [1] 699
```

```
talkers <- as.data.frame(table(friends_lines$speaker))
top_10_talkers <- talkers %>%
  slice_max(order_by = Freq, n = 10)
names(top_10_talkers) <- c('character','speaking_lines')
top_10_talkers
```

```
##           character speaking_lines
## 1      Rachel Green           9312
## 2       Ross Geller           9157
## 3     Chandler Bing           8465
## 4     Monica Geller           8441
## 5    Joey Tribbiani           8171
```

```
## 6     Phoebe Buffay           7501
## 7  Scene Directions           6063
## 8             #ALL#            347
## 9     Mike Hannigan            330
## 10    Richard Burke            281
```

Only top 6 characters in terms of lines spoken.

```
friends_lines <- friends_lines %>%
  filter(speaker %in% c("Monica Geller", "Rachel Green", "Ross Geller", "Chandler Bing", "Joey Tribbiani
```

Splitting each row into multiple rows, where each word is its own row (to prepare for sentiment analysis). Delimeter between words is space " ".

```
friends_words <- friends_lines %>%
  separate_rows(text, sep = " ")
```

Showing which friends characters spoke the most, in terms of lines, words, and words per line. Monica has the fewest words per line (9.8) while Phoebe has the most (10.9). Rachel and Ross have the most lines overall (9312 and 9157) which drives their top total word count (97,633 and 95,561) among all characters.

```
friends_word_count <- friends_words %>%
  group_by(speaker) %>%
  summarise(total_word_count=n(),
            .groups = 'drop')

top_talkers_lines <- top_10_talkers %>%
  filter(character %in% c("Monica Geller", "Rachel Green", "Ross Geller", "Chandler Bing", "Joey Tribbia

friends_summary <- left_join(friends_word_count, top_talkers_lines, by=c('speaker'='character'))

friends_summary <- friends_summary %>%
  transform(words_per_line = round((total_word_count / speaking_lines),2))

friends_summary
```

```
##          speaker total_word_count speaking_lines words_per_line
## 1  Chandler Bing            86547           8465          10.22
## 2 Joey Tribbiani            86426           8171          10.58
## 3  Monica Geller            82988           8441           9.83
## 4  Phoebe Buffay            81506           7501          10.87
## 5   Rachel Green            97633           9312          10.48
## 6    Ross Geller            95561           9157          10.44
```

Sentiment Analysis I first attempted to perform sentiment analysis using the analyzeSentiment function in the SentimentAnalysis package. However that function is too slow to work well on individual words. EG friends_words.sentiment <- analyzeSentiment(friends_words.text). It's designed for smaller samples (e.g. a few paragraphs).

Therefore I'll change the SentimentAnalysis' word dictionary ("DictionaryGI") to a dataframe, and perform an analysis by joining the word dataframe with the sentiment dataframe, as we did with the Jane Austen data.

Preparing sentiment df.

```r
data(DictionaryGI)
str(DictionaryGI)
```

```
## List of 2
##  $ negative: chr [1:2005] "abandon" "abandonment" "abate" "abdicate" ...
##  $ positive: chr [1:1637] "abide" "ability" "able" "abound" ...
```

```r
length(DictionaryGI$positive) <- length(DictionaryGI$negative)

sa_df <- as.data.frame(DictionaryGI)

neg_words <- sa_df$negative
neg_words <- as.data.frame(neg_words)
neg_words <- neg_words %>%
  mutate(sentiment = "negative")
names(neg_words) <- c('text','sentiment')

pos_words <- sa_df$positive
pos_words <- as.data.frame(pos_words)
pos_words <- pos_words %>%
  mutate(sentiment="positive")
names(pos_words) <- c('text','sentiment')

sa_dict <- bind_rows(pos_words, neg_words)

friends_sentiment <- friends_words %>%
  inner_join(sa_dict) %>%
  group_by(speaker,sentiment) %>%
  summarise(total_count=n(),.groups = 'drop') %>%
  as.data.frame() %>%
  pivot_wider(names_from = sentiment,
              values_from = total_count)
```

```
## Joining with `by = join_by(text)`
```

```
## Warning in inner_join(., sa_dict): Each row in `x` is expected to match at most 1 row in `y`.
## i Row 69 of `x` matches multiple rows.
## i If multiple matches are expected, set `multiple = "all"` to silence this
##   warning.
```

```r
colnames(friends_sentiment) <- c("speaker","negative_words","positive_words")
```

Showing count of positive and negative words used by Friends characters.

```r
friends_sentiment
```

```
## # A tibble: 6 x 3
##   speaker        negative_words positive_words
##   <chr>                   <int>          <int>
## 1 Chandler Bing            2299           3512
## 2 Joey Tribbiani           2288           3564
```

```
## 3 Monica Geller           2303           3335
## 4 Phoebe Buffay           2030           3564
## 5 Rachel Green            2477           4321
## 6 Ross Geller             2329           3822
```

Merging total Friends word count df with positive and negative word count df to assess % of all words which were positive or negative by character.

```
friends_combined <- friends_summary %>%
  inner_join(friends_sentiment) %>%
  transform(perc_neg = scales::percent(negative_words / total_word_count),
            perc_pos = scales::percent(positive_words / total_word_count),
            neutral_words = total_word_count - negative_words - positive_words)
```

```
## Joining with `by = join_by(speaker)`
```

```
friends_combined <- friends_combined %>%
  transform(perc_neutr = scales::percent(neutral_words / total_word_count))

friends_final <- friends_combined[,c('speaker','total_word_count','speaking_lines','words_per_line','po
```

In general, this dataframe shows the characters with a relatively narrow band of sentiment: from 4.00% positive (Ross) to 4.43% positive (Phoebe) and from 2.43% negative (Ross) to 2.78% negative (Monica). Neutrality ranged from 93.04% (Rachel) to 93.56% (Ross).

These data show Rachel as the most sentimental character, Ross as the least sentimental, Phoebe as the most positive, and Monica as the most negative. All of this aligns with my domain knowledge, therefore the sentiment analysis appears successful.

```
friends_final
```

```
##           speaker total_word_count speaking_lines words_per_line positive_words
## 1  Chandler Bing            86547           8465          10.22           3512
## 2 Joey Tribbiani            86426           8171          10.58           3564
## 3  Monica Geller            82988           8441           9.83           3335
## 4  Phoebe Buffay            81506           7501          10.87           3564
## 5   Rachel Green            97633           9312          10.48           4321
## 6    Ross Geller            95561           9157          10.44           3822
##   negative_words neutral_words perc_pos perc_neg perc_neutr
## 1           2299         80736   4.058%  2.6564%    93.286%
## 2           2288         80574   4.124%  2.6474%    93.229%
## 3           2303         77350   4.019%  2.7751%    93.206%
## 4           2030         75912   4.373%  2.4906%    93.137%
## 5           2477         90835   4.426%  2.5371%    93.037%
## 6           2329         89410   4.000%  2.4372%    93.563%
```