Assignment #10 DATA 607

Anthony Conrardy

2024-03-28

Assignment #10 Details

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 should provide a citation to this base code. You're then asked to extend the code in two ways:

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

As usual, please submit links to both an .Rmd file posted in your GitHub repository and to your code on rpubs.com. You make work on a small team on this assignment.

Book Citation

For the purposes of this exercise, we shall be using the code and information contained in the following book:

Silge, J., & Robinson, D. (2017). Text mining with R: A tidy approach. " O'Reilly Media, Inc."

Reviewing Lexicon Availability

In this section we are accessing the sentiment lexicons from AFINN, NRC, and Bing.

```
# get_sentiments() can select a specific lexicon from AFINN, NRC, or Bing.
get_sentiments("afinn")
```

```
## # A tibble: 2,477 x 2
##    word       value
##    <chr>      <dbl>
##  1 abandon       -2
##  2 abandoned     -2
##  3 abandons      -2
##  4 abducted      -2
##  5 abduction     -2
##  6 abductions    -2
##  7 abhor         -3
##  8 abhorred      -3
##  9 abhorrent     -3
## 10 abhors        -3
## # i 2,467 more rows
```

```
get_sentiments("nrc")
```

```
## # A tibble: 13,872 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 abacus      trust
##  2 abandon     fear
##  3 abandon     negative
##  4 abandon     sadness
##  5 abandoned   anger
##  6 abandoned   fear
##  7 abandoned   negative
##  8 abandoned   sadness
##  9 abandonment anger
## 10 abandonment fear
## # i 13,862 more rows
```

```
get_sentiments("bing")
```

```
## # A tibble: 6,786 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 2-faces     negative
##  2 abnormal    negative
##  3 abolish     negative
##  4 abominable  negative
##  5 abominably  negative
##  6 abominate   negative
##  7 abomination negative
##  8 abort       negative
##  9 aborted     negative
## 10 aborts      negative
## # i 6,776 more rows
```

Sentiment Analysis with Inner Join

In this section we are taking the text from the books of Jane Austen and assigning each word to it's own row, based on chapter and line number. We are also removing the stop_words, identifying the words associated with the sentiment of Joy, and then applying it to the book "Emma" and getting word counts.

```
#Un-nesting the data and assigning one word per row.
tidy_books1 <- 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)

# Remove the Stop Words
```

```
tidy_books <- tidy_books1 |>
  anti_join(stop_words)
```

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

```
# Using the NRC lexicon, filter the words associated with joy.  Identifies 687 words.
nrcjoy <- get_sentiments('nrc') |> filter(sentiment =="joy")

# Using the un-nested data frame, filter the book "Emma" and join the columns with the "joy" sentiment

tidy_books |> filter(book == "Emma") |>
  inner_join(nrcjoy) |>
  count(word, sort = TRUE)
```

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

## # A tibble: 297 x 2
##    word          n
##    <chr>     <int>
##  1 friend      166
##  2 hope        143
##  3 happy       125
##  4 love        117
##  5 deal         92
##  6 found        92
##  7 happiness    76
##  8 pretty       68
##  9 true         66
## 10 comfort      65
## # i 287 more rows
```

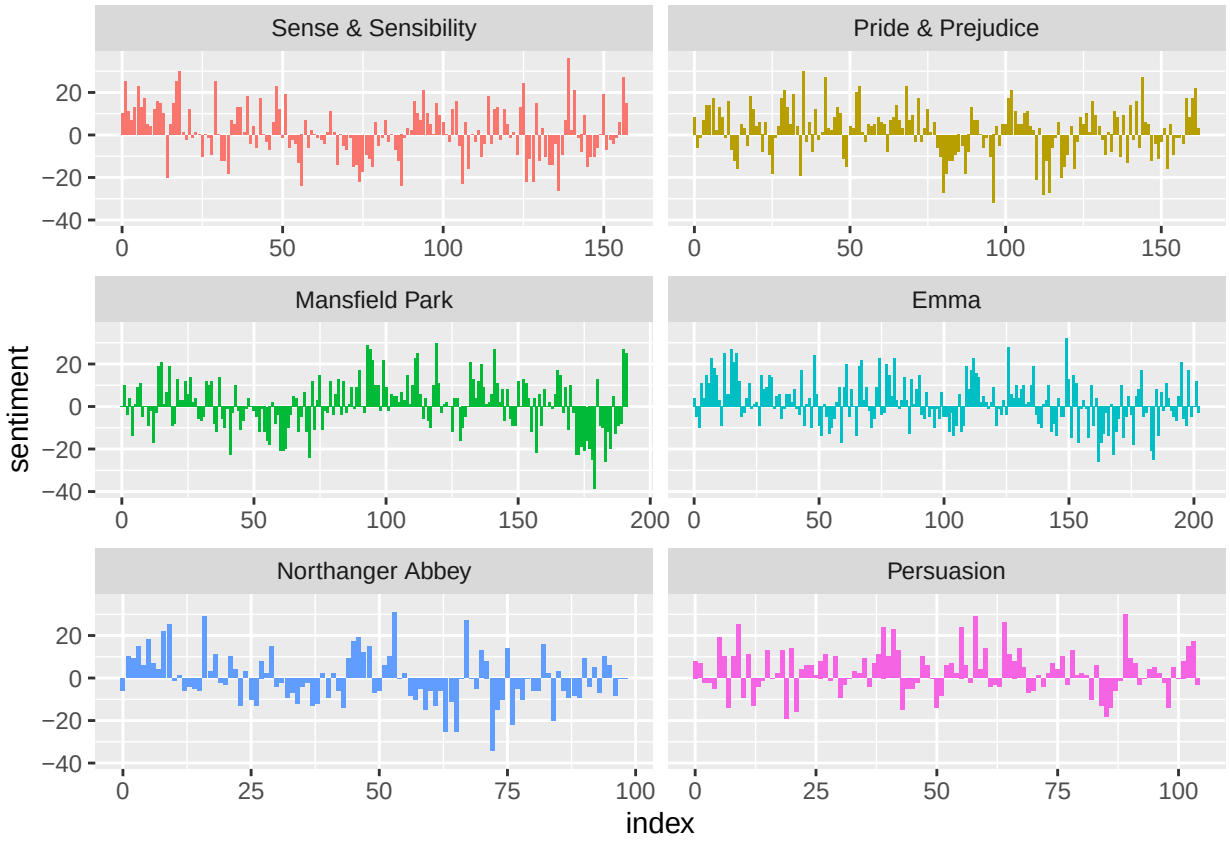
Net Sentiment Calculation

In this section, we assign the Bing sentiment lexicon to the entire Jane Austen collection and then group by book and summarize positive and negative word counts by 80 row intervals. We also create a new variable to assign a sentiment value to each interval and then plot those overall negative or positive sentiment values per book.

```
# Create data frame names janeaustensentiment from tidy_books.  Join the Bing sentiments lexicon with t
janeaustensentiment <- tidy_books |>
  inner_join(get_sentiments("bing")) |>
  count(book, index = linenumber %/% 80, sentiment) |>
  spread(sentiment, n, fill=0) |>
  mutate(sentiment = positive - negative)
```

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

```
# Create a sentiment plot comparing the sentiment values for each 80 row interval for each book.
ggplot(janeaustensentiment, aes(index, sentiment, fill=book)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~book, ncol = 2, scales = "free_x")
```




```
#ggsave("sentiment_plot.png", gg, width = 11, height = 8, dpi = 600)
```

Comparing Three Sentiment Libraries

In this section we compare the sentiment assessments of Jane Austen using all three lexicons. For the most part each lexicon seems to indicate qualitatively the same result; however, the NRC lexicon seems to be the most forgiving as it relates to overall sentiment of Jane Austen works.

```
# Limit to one book by author
pride_prejudice <- tidy_books |> filter(book == "Pride & Prejudice")

# Create data frame of afinn lexicon sentiment for "pride & prejudice. Inner join based upon "word", an
afinn <- pride_prejudice |>
  inner_join(get_sentiments("afinn")) |>
  group_by(index=linenumber %/% 80) |>
  summarize(sentiment = sum(value)) |>
  mutate(method = "AFINN")
```

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

```
# Combining Bing and NRC rows, and identifying the method being used for tidy dataset.
# Positive and Negative Sentiment for both Bing and NRC
bing1 <- pride_prejudice |>
  inner_join(get_sentiments("bing")) |>
```

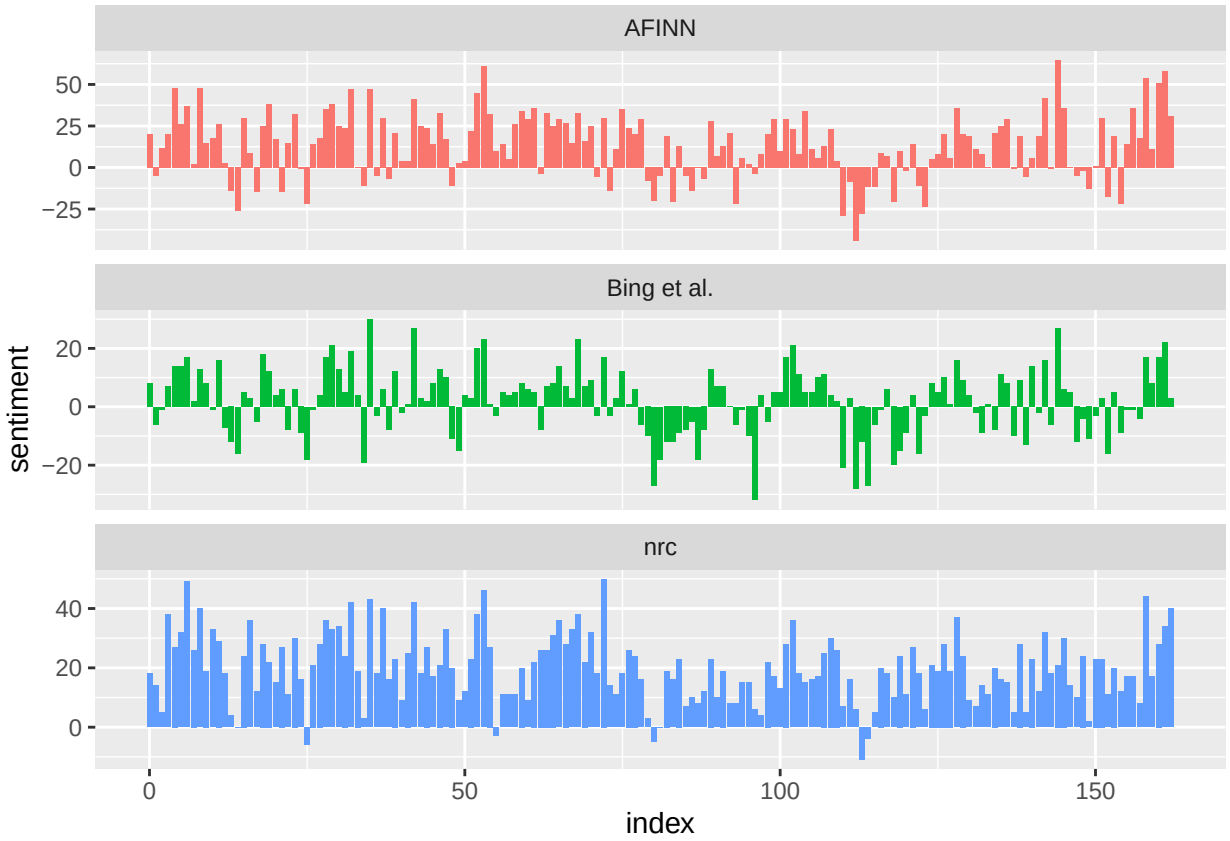
```
  mutate(method ="Bing et al.") |>
  count(method, index=linenumber %/% 80, sentiment) |>
  spread(sentiment, n, fill = 0) |>
  mutate(sentiment = positive - negative)
```

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

```
nrc1 <- pride_prejudice |>
    inner_join(get_sentiments("nrc")) |>
    filter((sentiment %in% c("positive", "negative"))) |>
    mutate(method="nrc") |>
    count(method, index=linenumber %/% 80, sentiment) |>
    spread(sentiment, n, fill = 0) |>
    mutate(sentiment = positive - negative)
```

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

```
# Bind the rows of the two data frames.
bing_and_nrc <- bind_rows(bing1, nrc1)

# Bind all three together
lex_all <- bind_rows(afinn, bing_and_nrc)

# Plot all results for comparison

gg2 <- ggplot(lex_all, aes(index, sentiment, fill=method)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~method, ncol = 1, scales = "free_y")
gg2
```



Checking the Lexicon Bias

In this section we check the positive/negative bias of the lexicons. We see that the both lexicons have more negative sentiment words than positive; however, Bing lexicon seems to have the most negative sentiment words.

```
get_sentiments("nrc") |> filter(sentiment %in% c('positive', 'negative')) |> count(sentiment)
```

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative   3316
## 2 positive   2308
```

```
get_sentiments("bing") |> count(sentiment)
```

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative   4781
## 2 positive   2005
```

Most Common Postive and Negative Words

In this section we identify the top 10 positive and negative sentiment words from the works of Jane Austen. We see that the word "miss" appears almost 1000 times more often as the next negative sentiment word of

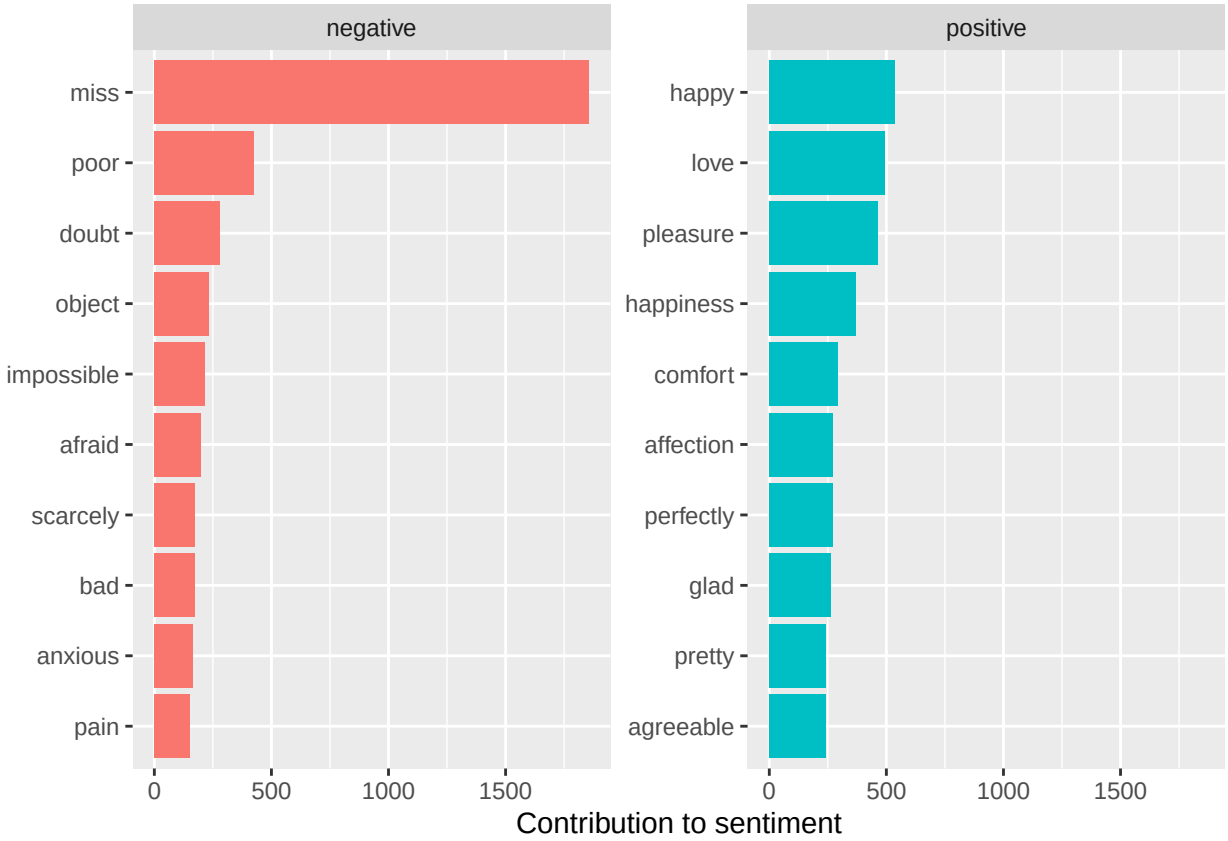
"poor". This is a concern of miss application in this context since "miss" is used as a form of address, and not that of loss.

```
bing_word_counts <- tidy_books |> inner_join(get_sentiments("bing")) |>
  count(word, sentiment, sort = TRUE) |>
  ungroup()
```

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

```
bing_word_counts |>
  group_by(sentiment) |>
  top_n(10) |>
  ungroup() |>
  mutate(word=reorder(word, n)) |>
  ggplot(aes(word, n, fill=sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Contribution to sentiment",
       x=NULL) +
  coord_flip()
```

```
## Selecting by n
```

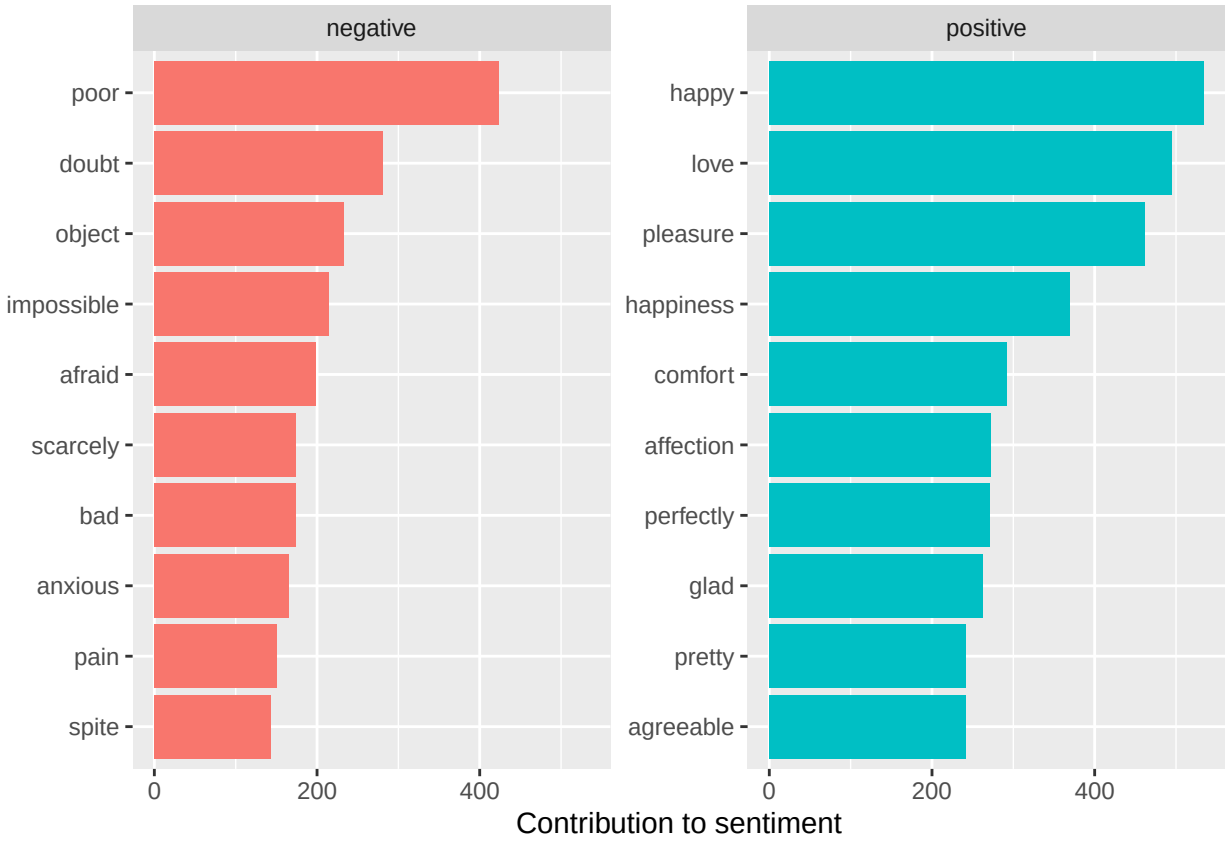


Creating Custom Stop Word Addition

In this section, we create a custom stop_word lexicon by adding the word "miss" to the lexicon and then re-apply it to the data frame. We see that the results are more in line with all the other sentiment words.

```
# Create a row to add to the custom stop words
custom <- tibble(word="miss", lexicon="custom")
#bind the rows
custom_stop_words <- bind_rows(custom, stop_words)
#run through the pipe
bing_word_counts |>
  anti_join(custom_stop_words) |>
  group_by(sentiment) |>
  top_n(10) |>
  ungroup() |>
  mutate(word=reorder(word, n)) |>
  ggplot(aes(word, n, fill=sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Contribution to sentiment",
       x=NULL) +
  coord_flip()
```

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



Word Cloud Generation

In this section we create a word cloud to display the top sentiment word counts and limit to a maximum of 50 words in the cloud.

```
tidy_books |> anti_join(custom_stop_words) |>
  count(word) |>
  with(wordcloud(word, n, max.words = 50))
```

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

```
## Warning in wordcloud(word, n, max.words = 50): fanny could not be fit on page.
## It will not be plotted.
```

Positive and Negative Word Cloud

In this section we create a positive/negative word cloud with the dark gray text indicating negative sentiment words and light gray positive sentiment words.

```
tidy_books |> anti_join(custom_stop_words) |>
inner_join(get_sentiments("bing")) |>
  count(word, sentiment, sort = TRUE) |>
  acast(word~sentiment, value.var = "n", fill = 0) |>
  comparison.cloud(colors = c("gray20", "gray80"), max.words = 100)
```

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

Import New Text for Analysis

In this section, I chose to extend the use of sentiment analysis that was used for the works of Jane Austen to the classic work of "The Adventures of Tom Sawyer" by Mark Twain. While Jane Austen was a Victorian-era, English author in the genre of romantic fiction, Mark Twain was a 19th century American author in the genre of adventure fiction, who is also know for being a humorist. While both these authors have very little in common, I thought it interesting to compare the overall sentiment and word use. I chose not to create a word cloud for this analysis, and simply evaluate top 10 word counts and overall book sentiment.

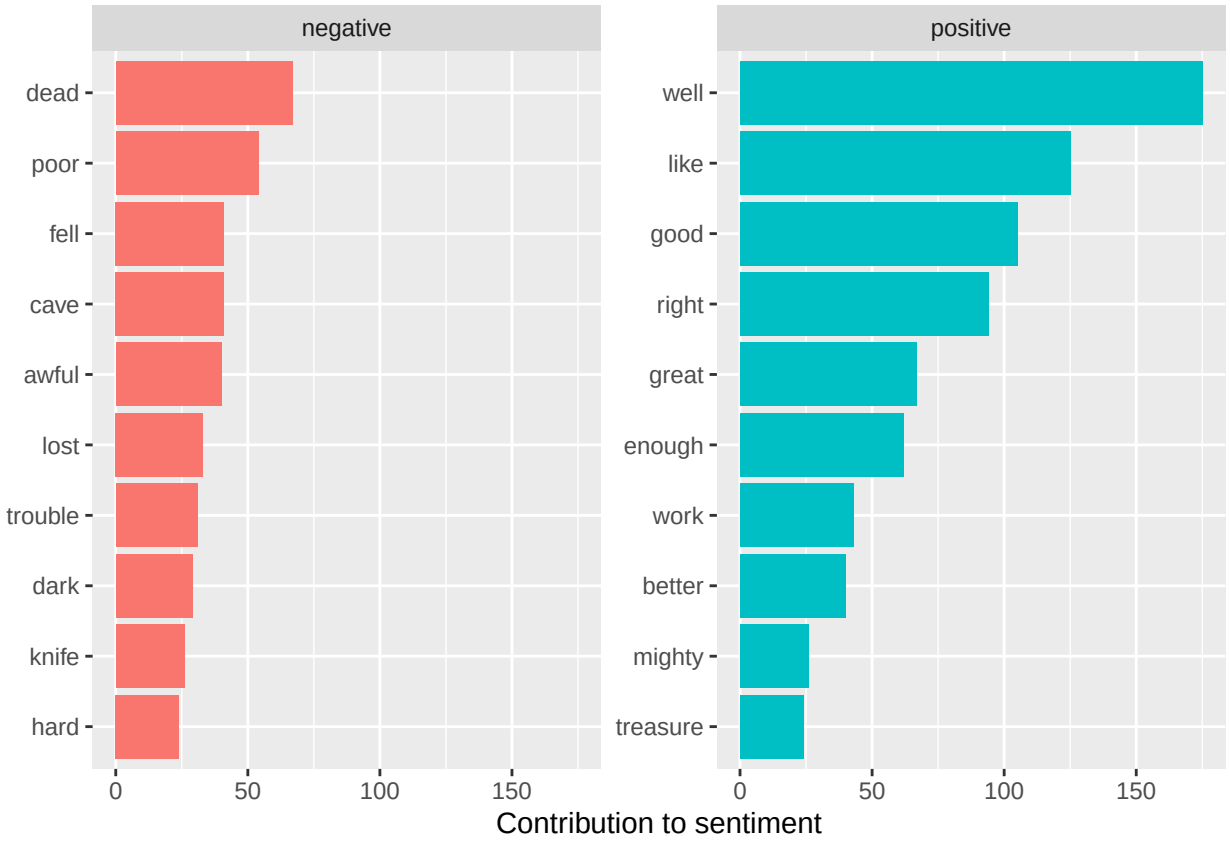
```
# Pull the text file of Tom Sawyer in.
location <- "~/tom_sawyer_1.txt"

# Read the lines of text
text_lines <- readLines(location)

# Create a data frame with one column containing the text lines
tom_sawyer <- data.frame(Text = text_lines, stringsAsFactors = FALSE)

# Remove the rows that have empty values or character strings in the column.  Remove the first 237 rows
tom_sawyer_rows <- tom_sawyer |> filter(Text !="") |> slice(-(1:237)) |> filter(!grepl("CHAPTER", Text))

# Remove the first 237 rows of text as they are irrelevant to the story of Tom Sawyer
```

```
tom_sawyer_rows_less <- tom_sawyer_rows[-(1:237), ]

tom_sawyer_parsed <- tom_sawyer_rows |>
  mutate(linenumber = row_number()) |>
  unnest_tokens(word, Text)

bing_word_counts_tom <- tom_sawyer_parsed |> inner_join((get_sentiments("bing"))) |>
  count(word, sentiment, sort = TRUE) |>
  ungroup()
```

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

```
bing_word_counts_tom |>
  group_by(sentiment) |>
  top_n(10) |>
  ungroup() |>
  mutate(word=reorder(word, n)) |>
  ggplot(aes(word, n, fill=sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Contribution to sentiment",
       x=NULL) +
  coord_flip()
```

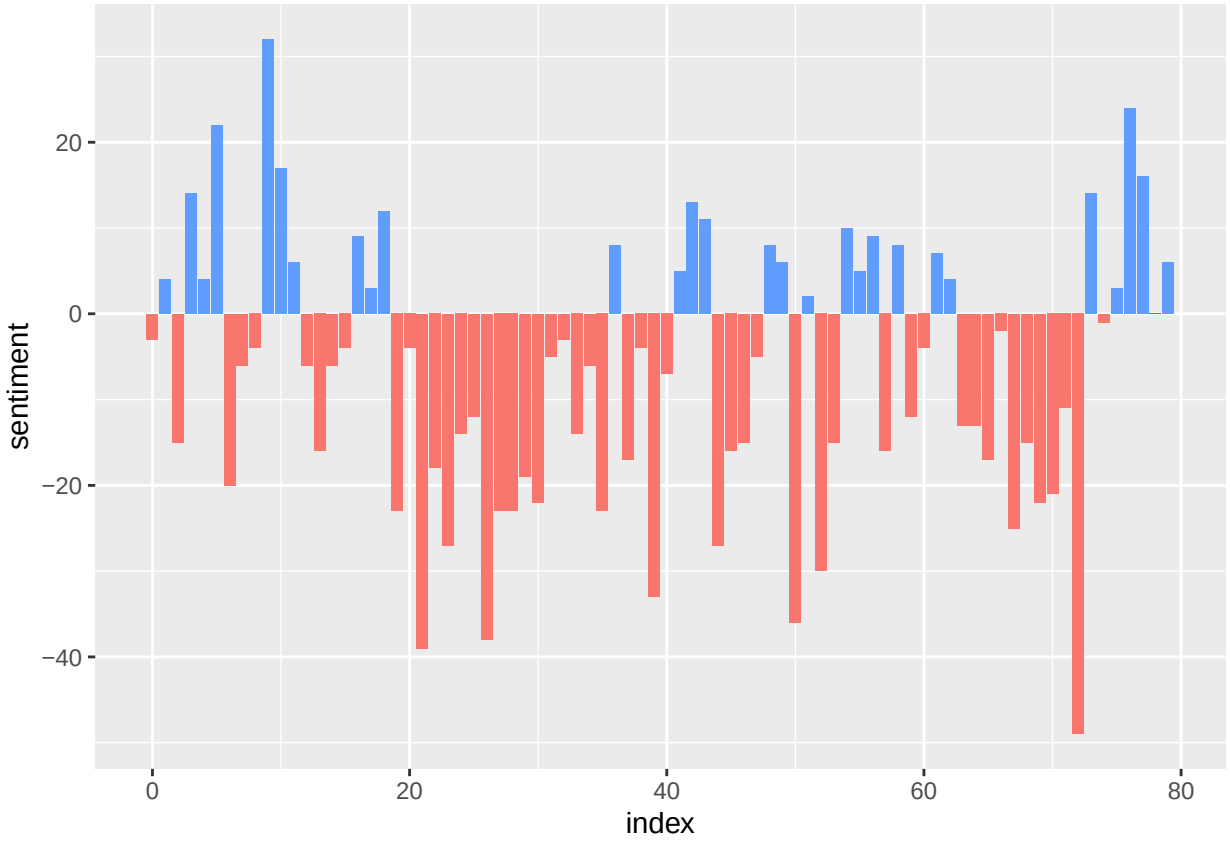
```
## Selecting by n
```



```
tom_sawyer_sentiment <- tom_sawyer_parsed |>
  inner_join(get_sentiments("bing"))|>
  count(, index = linenumber %/% 80, sentiment)|>
  spread(sentiment, n, fill=0) |>
  mutate(sentiment = positive - negative)
```

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

```
# Create a sentiment plot comparing the sentiment values for each 80 row interval for each book.

ggplot(tom_sawyer_sentiment, aes(index, sentiment, fill = factor(sign(sentiment)))) +
  geom_col(show.legend = FALSE)
```



Conclusion and Discussion

It is interesting to note that we see the same contextual issue with the use of the word "well" with Mark Twain that we saw with the word "miss" with Jane Austen. The use of the word "well" is being used as an introductory word to the continuation of a statement, or commonly known as a discourse marker (i.e. Well, I never!, Well, I reckon so.).

Also, we see that overall there seems to be a net negative sentiment for many of the intervals giving the impression that the book has an overall negative sentiment. This seems counter-intuitive for an author that is credited with being both a humorist and adventure fiction writer. However, when one views it from the perspective of an "adventure", once can easily conclude that adventure comes with danger, danger comes with darkness and foreboding, and that sets the stage for the progression of the book.