Data 607 Sentiment Analysis With Tidy Data, Part 2

Enid Roman

2022-11-04

PART 2 OF 2 SENTIMENT ANALYSIS WITH TIDY DATASET

My corpus is from the Harry Potter series title "Half-Blood Prince" by J. K. Rowling.

```
# Install Libraries I felt I Needed

#install.packages("tidyverse")
#install.packages("textdata")
#install.packages("gutenbergr")
#install.packages("DT")
#install.packages("flextable")
#install.packages("wordcloud")
#devtools::install_github("ropensci/gutenbergr")
if (packageVersion("devtools") < 1.6) {
  install.packages("devtools")
}
devtools::install_github("bradleyboehmke/harrypotter")
```

Package was created by Bradley Boehmke and I retrieved it from https://afit-r.github.io/sentiment__analysis.

```
## Skipping install of 'harrypotter' from a github remote, the SHA1 (51f71461) has not changed since las
##   Use `force = TRUE` to force installation
```

THE SENTIMENTS DATASETS

```
# Load Libraries

library(tidyverse)      # data manipulation & plotting
```

```
## Warning: package 'tidyverse' was built under R version 4.2.2
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
```

```
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(stringr)        # text cleaning and regular expressions
library(tidytext)       # provides additional text mining functions
```

```
## Warning: package 'tidytext' was built under R version 4.2.2
```

```
library(textdata)       # Provides a framework to download, parse, and store text datasets on the disk
```

```
## Warning: package 'textdata' was built under R version 4.2.2
```

```
library(dplyr)          # aims to provide a function for each basic verb of data manipulation.
library(harrypotter)    # provides the first seven novels of the Harry Potter series
library(ggplot2)        # is an open-source data visualization package for the statistical programming
```

HARRY POTTER NOVELS IN THIS LIBRARY

philosophers_stone: Harry Potter and the Philosophers Stone (1997)

chamber_of_secrets: Harry Potter and the Chamber of Secrets (1998)

prisoner_of_azkaban: Harry Potter and the Prisoner of Azkaban (1999)

goblet_of_fire: Harry Potter and the Goblet of Fire (2000)

order_of_the_phoenix: Harry Potter and the Order of the Phoenix (2003)

half_blood_prince: Harry Potter and the Half-Blood Prince (2005)

deathly_hallows: Harry Potter and the Deathly Hallows (2007)

```
titles <- c("Philosopher's Stone", "Chamber of Secrets", "Prisoner of Azkaban",
            "Goblet of Fire", "Order of the Phoenix", "Half-Blood Prince",
            "Deathly Hallows")

books <- list(philosophers_stone, chamber_of_secrets, prisoner_of_azkaban,
           goblet_of_fire, order_of_the_phoenix, half_blood_prince,
           deathly_hallows)

series <- tibble()
```

```r
for(i in seq_along(titles)) {

        clean <- tibble(chapter = seq_along(books[[i]]),
                        text = books[[i]]) %>%
             unnest_tokens(word, text) %>%
             mutate(book = titles[i]) %>%
             select(book, everything())

        series <- rbind(series, clean)
}

# set factor to keep books in order of publication
series$book <- factor(series$book, levels = rev(titles))

series
```

To perform sentiment analysis we need to have our data in a tidy format. The following converts all seven Harry Potter novels into a tibble that has each word by chapter by book.

```
## # A tibble: 1,089,386 x 3
##    book                chapter word
##    <fct>                 <int> <chr>
##  1 Philosopher's Stone       1 the
##  2 Philosopher's Stone       1 boy
##  3 Philosopher's Stone       1 who
##  4 Philosopher's Stone       1 lived
##  5 Philosopher's Stone       1 mr
##  6 Philosopher's Stone       1 and
##  7 Philosopher's Stone       1 mrs
##  8 Philosopher's Stone       1 dursley
##  9 Philosopher's Stone       1 of
## 10 Philosopher's Stone       1 number
## # ... with 1,089,376 more rows
```

SENTIMENT ANALYSIS WITH INNER JOIN

```r
series %>%
        right_join(get_sentiments("nrc")) %>%
        filter(!is.na(sentiment)) %>%
        count(sentiment, sort = TRUE)
```

We use the nrc sentiment data set to assess the different sentiments that are represented across the Harry Potter series. There is a stronger negative presence than positive.

```
## Joining, by = "word"
```

```
## # A tibble: 10 x 2
##    sentiment        n
##    <chr>        <int>
##  1 negative     55091
```

```
##   2 positive     37758
##   3 sadness      34878
##   4 anger        32742
##   5 trust        23154
##   6 fear         21536
##   7 anticipation 20625
##   8 joy          13800
##   9 disgust      12861
##  10 surprise     12817
```

To visualize this analysis, we plot these sentiment scores across the plot trajectory of each novel. We are plotting against the index on the x-axis that keeps track of narrative time in sections of text.

We perform the following:

1. Create an index that breaks up each book by 500 words; this is the approximate number of words on every two pages so this will allow us to assess changes in sentiment even within chapters.

2. Join the bing lexicon with inner_join to assess the positive vs. negative sentiment of each word.

3. Count up how many positive and negative words there are for every two pages".

4. Spread our data and...

5. Calculate a net sentiment (positive - negative).

6. Plot our data.

```
series %>%
        group_by(book) %>%
        mutate(word_count = 1:n(),
               index = word_count %/% 500 + 1) %>%
        inner_join(get_sentiments("bing")) %>%
        count(book, index = index , sentiment) %>%
        ungroup() %>%
        spread(sentiment, n, fill = 0) %>%
        mutate(sentiment = positive - negative,
               book = factor(book, levels = titles)) %>%
        ggplot(aes(index, sentiment, fill = book)) +
          geom_bar(alpha = 0.5, stat = "identity", show.legend = FALSE) +
          facet_wrap(~ book, ncol = 2, scales = "free_x")
```

We can see how the plot of each novel changes toward more positive or negative sentiment over the trajectory of the story.

```
## Joining, by = "word"
```

COMPARING THE THREE SENTIMENTS

```
afinn <- series %>%
        group_by(book) %>%
        mutate(word_count = 1:n(),
               index = word_count %/% 500 + 1) %>%
        inner_join(get_sentiments("afinn")) %>%
        group_by(book, index) %>%
        summarise(sentiment = sum(value)) %>%
        mutate(method = "AFINN")
```

We compare each sentiments to get more information on which one is appropriate for your purposes. Lets use all three sentiment lexicons and examine how they differ for each novel.

```
## Joining, by = "word"
## `summarise()` has grouped output by 'book'. You can override using the
## `.groups` argument.
```

```
bing_and_nrc <- bind_rows(series %>%
                  group_by(book) %>%
                  mutate(word_count = 1:n(),
```

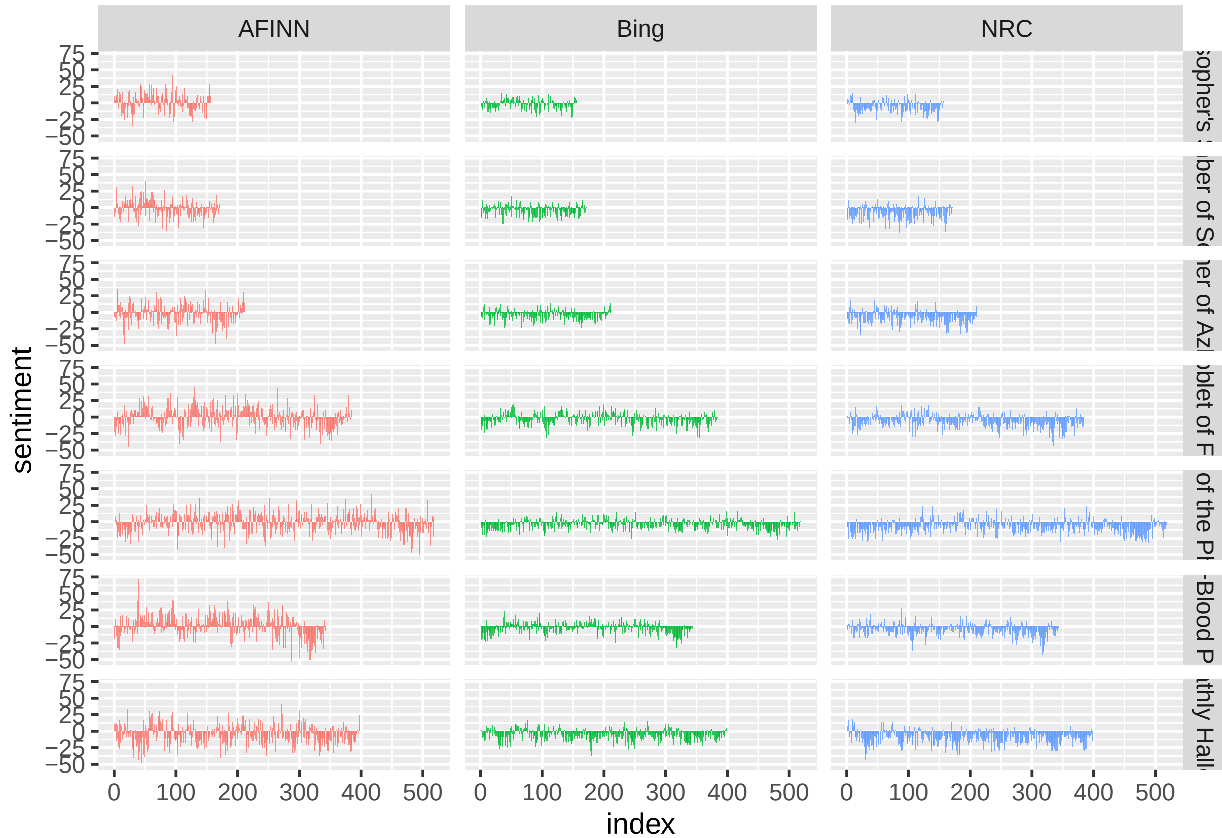
```
                        index = word_count %/% 500 + 1) %>%
                 inner_join(get_sentiments("bing")) %>%
                 mutate(method = "Bing"),
         series %>%
                 group_by(book) %>%
                 mutate(word_count = 1:n(),
                        index = word_count %/% 500 + 1) %>%
                 inner_join(get_sentiments("nrc") %>%
                                    filter(sentiment %in% c("positive", "negative"))) %>%
                 mutate(method = "NRC")) %>%
        count(book, method, index = index , sentiment) %>%
        ungroup() %>%
        spread(sentiment, n, fill = 0) %>%
        mutate(sentiment = positive - negative) %>%
        select(book, index, method, sentiment)
```

```
## Joining, by = "word"
## Joining, by = "word"
```

We now have an estimate of the net sentiment (positive - negative) in each chunk of the novel text for each sentiment lexicon. We bind them together and plot them.

```
bind_rows(afinn,
          bing_and_nrc) %>%
        ungroup() %>%
        mutate(book = factor(book, levels = titles)) %>%
  ggplot(aes(index, sentiment, fill = method)) +
  geom_bar(alpha = 0.8, stat = "identity", show.legend = FALSE) +
  facet_grid(book ~ method)
```

The three different lexicons for calculating sentiment give results that are different in an absolute sense but have fairly similar relative trajectories through the novels. We see similar dips and peaks in sentiment at about the same places in the novel, but the absolute values are significantly different. In some instances, it apears the AFINN lexicon finds more positive sentiments than the Bing and NRC lexicon. This output also allows us to compare across novels. First, you get a good sense of differences in book lengths - Order of the Pheonix is much longer than Philosopher's Stone. Second, you can compare how books differ in their sentiment (both direction and magnitude) across a series.



COMMON POSITIVE AND NEGATIVE SENTIMENT WORDS

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

We can analyze word counts that contribute to each sentiment.

```
## Joining, by = "word"
```

```
bing_word_counts
```

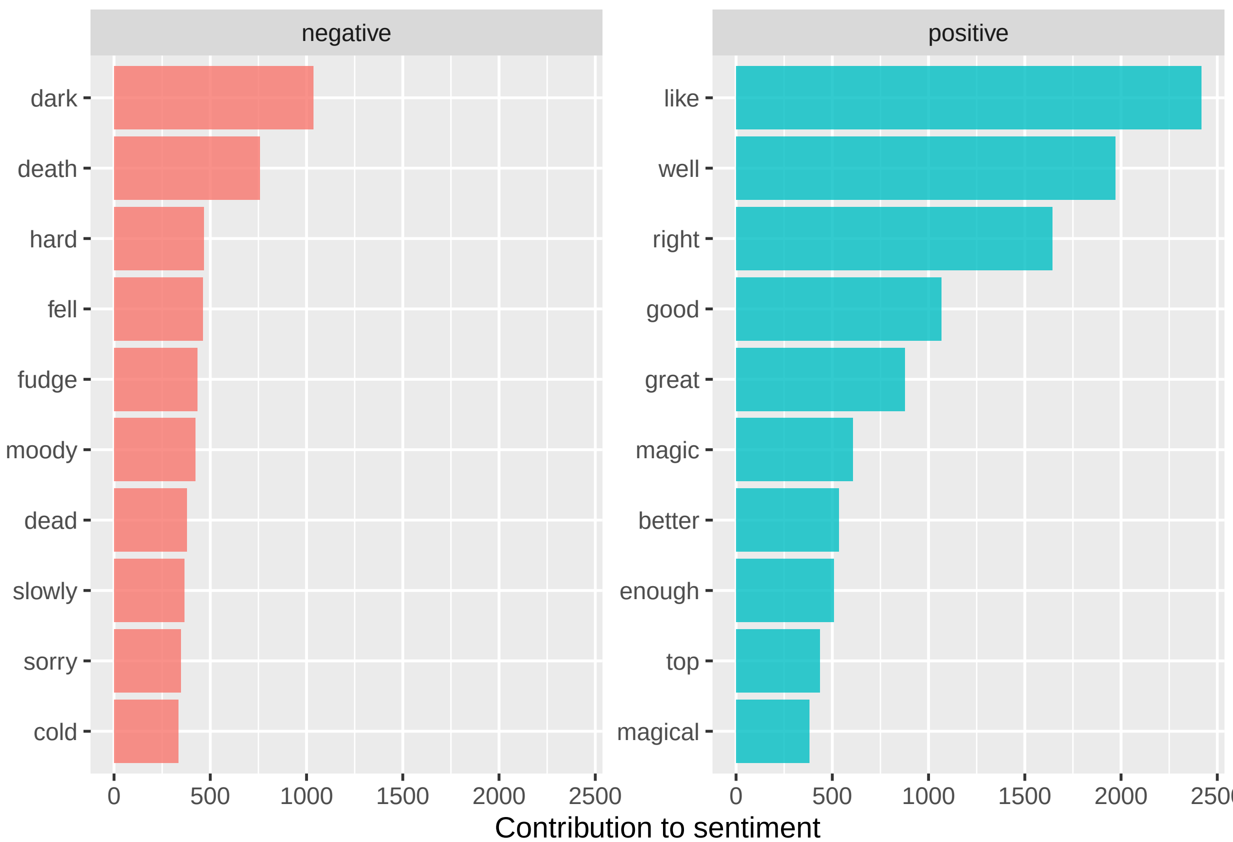
```
## # A tibble: 3,313 x 3
##    word   sentiment     n
##    <chr>  <chr>     <int>
##  1 like   positive   2416
##  2 well   positive   1969
##  3 right  positive   1643
##  4 good   positive   1065
##  5 dark   negative   1034
##  6 great  positive    877
##  7 death  negative    757
```

```
##   8 magic   positive    606
##   9 better positive    533
## 10 enough positive    509
## # ... with 3,303 more rows
```

```
bing_word_counts %>%
        group_by(sentiment) %>%
        top_n(10) %>%
        ggplot(aes(reorder(word, n), n, fill = sentiment)) +
          geom_bar(alpha = 0.8, stat = "identity", show.legend = FALSE) +
          facet_wrap(~sentiment, scales = "free_y") +
          labs(y = "Contribution to sentiment", x = NULL) +
          coord_flip()
```

By doing a geomplot we can view this visually to assess the top n words for each sentiment.

```
## Selecting by n
```



We see an anomaly in the sentiment analysis; the word "fudge" is coded as negative but it is used as a type of food in Harry Potter series. If it were appropriate for our purposes, we could easily add "fudge" to a custom stop-words list using bind_rows(). We could implement that with a strategy such as this.

```r
custom_stop_words <- bind_rows(tibble(word = c("fudge"),
                                      lexicon = c("custom")),
                               stop_words)

custom_stop_words
```

```
## # A tibble: 1,150 x 2
##    word        lexicon
##    <chr>       <chr>
##  1 fudge       custom
##  2 a           SMART
##  3 a's         SMART
##  4 able        SMART
##  5 about       SMART
##  6 above       SMART
##  7 according   SMART
##  8 accordingly SMART
##  9 across      SMART
## 10 actually    SMART
## # ... with 1,140 more rows
```

Using the wordcloud package, which uses base R graphics. Let's look at the most common words in the Harry Potter Series.

```r
library(wordcloud)
```

The size of a word's text below is in proportion to its frequency within its sentiment. We can use this visualization to see the most important positive and negative words, but the sizes of the words are not comparable across sentiments.

```
## Warning: package 'wordcloud' was built under R version 4.2.2

## Loading required package: RColorBrewer
```

```r
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
series %>%
  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, by = "word"
```

FURTER ANALYSIS

I chose to analyze one of my favorite Harry Pottr novels, Half-Blood Prince.

```
half_blood_prince[1:1]
```

Here I demonstrate a sample of the Half-Blood Prince. The following illustrates the raw text of the first chapter of the half_blood_prince. Each text is in a character vector with each element representing a single chapter.

```
## [1] " It was nearing midnight and the Prime Minister was sitting alone in his office, reading a long
```

THE SENTIMENTS DATASETS

Import the novel half_blood_prince.

```
titles <- c("Half-Blood Prince")
books <- list(half_blood_prince)
series <- tibble()

for(i in seq_along(titles)) {

  temp <- tibble(chapter = seq_along(books[[i]]),
                  text = books[[i]]) %>%
    unnest_tokens(word, text) %>%
    ##Here we tokenize each chapter into words
    mutate(book = titles[i]) %>%
    select(book, everything())

  series <- rbind(series, temp)
}
# set factor to keep books in order of publication
series$book <- factor(series$book, levels = rev(titles))

# This is what the tokenizing looks like

series
```

Code based on https://afit-r.github.io/sentiment_analysis.

```
## # A tibble: 171,284 x 3
##    book               chapter word
##    <fct>                <int> <chr>
##  1 Half-Blood Prince        1 it
##  2 Half-Blood Prince        1 was
##  3 Half-Blood Prince        1 nearing
##  4 Half-Blood Prince        1 midnight
##  5 Half-Blood Prince        1 and
##  6 Half-Blood Prince        1 the
##  7 Half-Blood Prince        1 prime
##  8 Half-Blood Prince        1 minister
##  9 Half-Blood Prince        1 was
## 10 Half-Blood Prince        1 sitting
## # ... with 171,274 more rows
```

```
str(half_blood_prince)
```

Here we see the number of chapters Half-Blood Prince contain and a glimpse of the first sentence in the first chapter.

```
##  chr [1:30] " It was nearing midnight and the Prime Minister was sitting alone in his office, reading
```

SENTIMENT ANALYSIS WITH INNER JOIN

```r
# Using the AFINN lexicon for sentiment analysis on Harry Potter

afinn <- series %>%
        group_by(book) %>%
        mutate(word_count = 1:n(),
               index = word_count %/% 500 + 1) %>%
        inner_join(get_sentiments("afinn")) %>%
        group_by(book, index) %>%
        summarise(sentiment = sum(value)) %>%
        mutate(method = "AFINN")
```

We compare each sentiments to get more information on which one is appropriate for your purposes. Lets use all three sentiment lexicons and examine how they differ in this particular novel, Half-Blood Prince.

```
## Joining, by = "word"
## `summarise()` has grouped output by 'book'. You can override using the
## `.groups` argument.
```

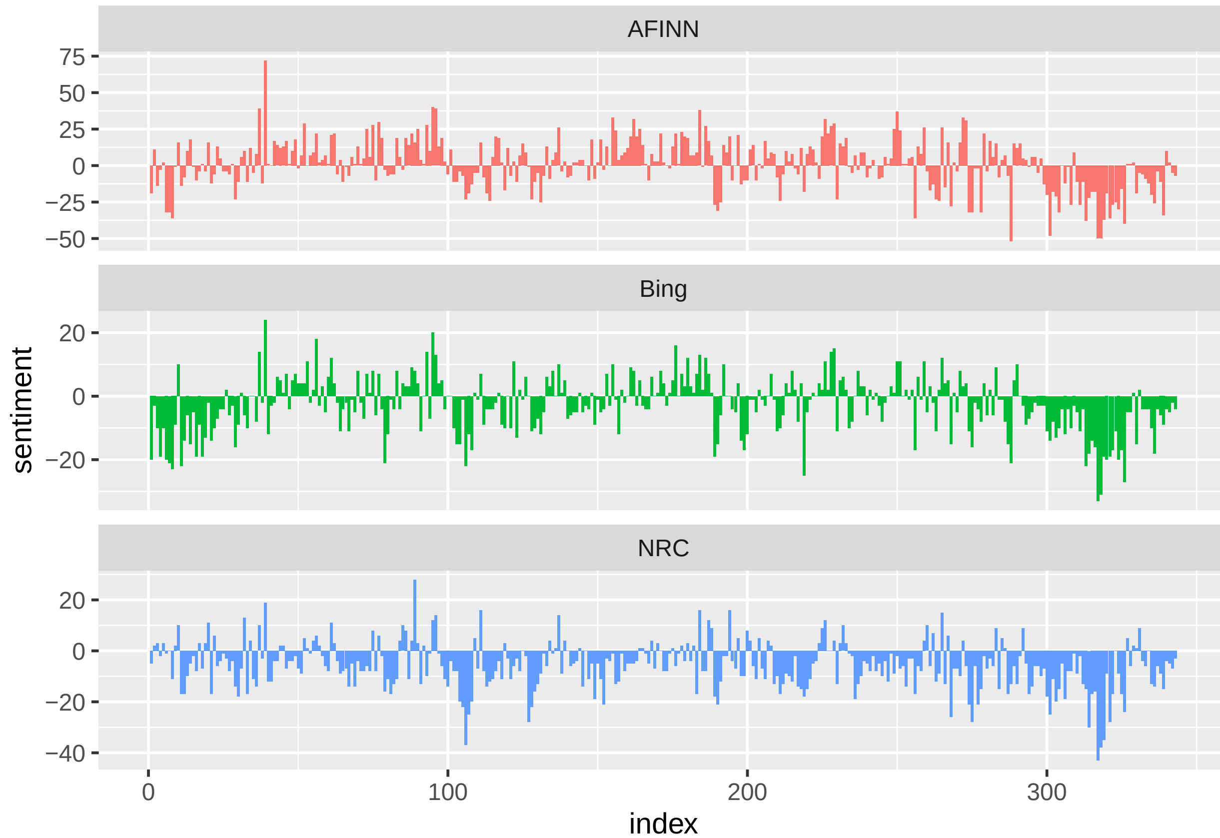
```r
bing_and_nrc <- bind_rows(series %>%
                  group_by(book) %>%
                  mutate(word_count = 1:n(),
                         index = word_count %/% 500 + 1) %>%
                  inner_join(get_sentiments("bing")) %>%
                  mutate(method = "Bing"),
          series %>%
                  group_by(book) %>%
                  mutate(word_count = 1:n(),
                         index = word_count %/% 500 + 1) %>%
                  inner_join(get_sentiments("nrc") %>%
                                   filter(sentiment %in% c("positive", "negative"))) %>%
                  mutate(method = "NRC")) %>%
        count(book, method, index = index , sentiment) %>%
        ungroup() %>%
        spread(sentiment, n, fill = 0) %>%
        mutate(sentiment = positive - negative) %>%
        select(book, index, method, sentiment)
```

```
## Joining, by = "word"
## Joining, by = "word"
```

We now have an estimate of the net sentiment (positive - negative) in each chunk of the novel text for each sentiment lexicon. We bind them together and plot them.

```r
bind_rows(afinn,
          bing_and_nrc) %>%
  ggplot(aes(index, sentiment, fill = method)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~method, ncol = 1, scales = "free_y")
```

The three different lexicons for calculating sentiment give results that are different in an absolute sense but have fairly similar relative trajectories through the novel. We see similar dips and peaks in sentiment at about the same places in the novel, but the absolute values are significantly different. In some instances, it apears the AFINN lexicon finds more positive sentiments than the Bing and NRC lexicon. NRC seems more negative sentiments then the other two sentiments. This output also allows us to compare across the chapters of the novel. You can compare how the chapters differ in their sentiment (both direction and magnitude) across the series.



Why the result for the NRC lexicon biased so high in sentiment compared to the Bing et al. result? Here we see how many positive and negative words are in these lexicons.

Both lexicons have more negative than positive words, but the ratio of negative to positive words is higher in the Bing lexicon than the NRC lexicon. This will contribute to the effect we see in the plot above, as will any systematic difference in word matches. Whatever the source of these differences, we see similar relative trajectories across the narrative arc, with similar changes in slope, but marked differences in absolute sentiment from lexicon to lexicon. This is all important context to keep in mind when choosing a sentiment lexicon for analysis.

```
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
```

```r
get_sentiments("bing") %>%
  count(sentiment)
```

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

MOST COMMON POSITIVE AND NEGATIVE WORDS

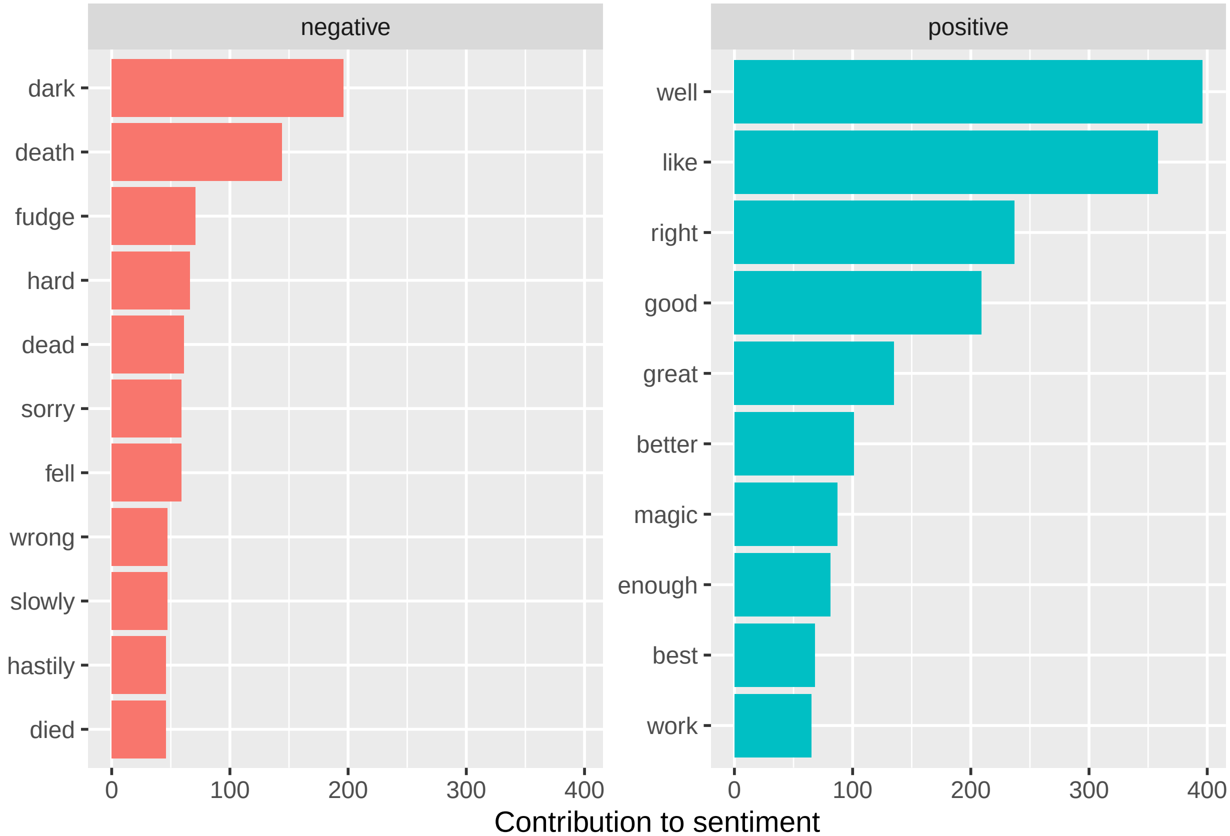
```r
bing_word_counts <- series %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

We can analyze word counts that contribute to each sentiment. By implementing count() here with arguments of both word and sentiment, we find out how much each word contributed to each sentiment.

```
## Joining, by = "word"
```

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

By doing a geomplot we can view this visually to assess the top n words for each senti-



ment.

```
custom_stop_words <- bind_rows(tibble(word = c("fudge"),
                                      lexicon = c("custom")),
                               stop_words)

custom_stop_words
```

We see an anomaly in the sentiment analysis; the word "fudge" is coded as negative but it is used as a type of food in Harry Potter series. If it were appropriate for our purposes, we could easily add "fudge" to a custom stop-words list using bind_rows(). We could implement that with a strategy such as this.

```
## # A tibble: 1,150 x 2
##    word        lexicon
##    <chr>       <chr>
##  1 fudge       custom
##  2 a           SMART
##  3 a's         SMART
##  4 able        SMART
##  5 about       SMART
##  6 above       SMART
##  7 according   SMART
##  8 accordingly SMART
```

```
##  9 across      SMART
## 10 actually    SMART
## # ... with 1,140 more rows
```

Using the wordcloud package, which uses base R graphics. Let's look at the most common words in the Harry Potter Series.

```
library(wordcloud)
library(reshape2)

series %>%
  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)
```

The size of a word's text below is in proportion to its frequency within its sentiment. We can use this visualization to see the most important positive and negative words, but the sizes of the words are not comparable across sentiments.

```
## Joining, by = "word"
```

THE LOUGHRAN LEXICON FOR SENTIMENT ANALYSIS

Using the Loughran lexicon for sentiment analysis on Harry Potter.

```
# Using the Loughran lexicon for sentiment analysis on Harry Potter
loughran <- series %>%
  right_join(get_sentiments("loughran")) %>%
  filter(!is.na(sentiment)) %>%
  count(sentiment, sort = TRUE)
```

The loughran lexicon divided words into constraining, litigious, negative, positive, superfluous and uncertainty.

```
## Joining, by = "word"
```

```
#A view of the Loughran analysis

loughran
```

We can see that in loughran negative words are more common than positive words while in bind this proportion is a bit over 4.

```
## # A tibble: 6 x 2
##   sentiment        n
##   <chr>        <int>
## 1 negative      4289
## 2 uncertainty   1709
## 3 positive      1481
## 4 litigious     1034
## 5 constraining   272
## 6 superfluous     56
```

```
#Prepares loughran for plotting
afinn <- series %>%
        group_by(book) %>%
        mutate(word_count = 1:n(),
               index = word_count %/% 500 + 1) %>%
        inner_join(get_sentiments("afinn")) %>%
        group_by(book, index) %>%
        summarise(sentiment = sum(value)) %>%
        mutate(method = "AFINN")
```

We compare each sentiments to get more information on which one is appropriate for your purposes. Lets use all four sentiment lexicons and examine how they differ in this particular novel, Half-Blood Prince.

```
## Joining, by = "word"
## `summarise()` has grouped output by 'book'. You can override using the
## `.groups` argument.
```

```
bing_and_nrc <- bind_rows(series %>%
                  group_by(book) %>%
                  mutate(word_count = 1:n(),
                         index = word_count %/% 500 + 1) %>%
                  inner_join(get_sentiments("bing")) %>%
                  mutate(method = "Bing"),
          series %>%
                  group_by(book) %>%
                  mutate(word_count = 1:n(),
                         index = word_count %/% 500 + 1) %>%
                  inner_join(get_sentiments("nrc") %>%
                                   filter(sentiment %in% c("positive", "negative"))) %>%
                  mutate(method = "NRC"))
```

```
## Joining, by = "word"
## Joining, by = "word"
```

```
loughran <- bind_rows(series %>%
                  group_by(book) %>%
                  mutate(word_count = 1:n(),
                         index = word_count %/% 500 + 1) %>%
                  inner_join(get_sentiments("loughran") %>%
                                   filter(sentiment %in% c("positive", "negative"))) %>%
                  mutate(method = "Loughran")) %>%
        count(book, method, index = index , sentiment) %>%
        ungroup() %>%
        spread(sentiment, n, fill = 0) %>%
        mutate(sentiment = positive - negative) %>%
        select(book, index, method, sentiment)
```

```
## Joining, by = "word"
```

```
# In order for the plot to work I need to bring this down from row 270.

afinn <- series %>%
        group_by(book) %>%
        mutate(word_count = 1:n(),
               index = word_count %/% 500 + 1) %>%
        inner_join(get_sentiments("afinn")) %>%
        group_by(book, index) %>%
        summarise(sentiment = sum(value)) %>%
        mutate(method = "AFINN")
```

```
## Joining, by = "word"
## `summarise()` has grouped output by 'book'. You can override using the
## `.groups` argument.
```

```r
bing_and_nrc <- bind_rows(series %>%
                  group_by(book) %>%
                  mutate(word_count = 1:n(),
                         index = word_count %/% 500 + 1) %>%
                  inner_join(get_sentiments("bing")) %>%
                  mutate(method = "Bing"),
          series %>%
                  group_by(book) %>%
                  mutate(word_count = 1:n(),
                         index = word_count %/% 500 + 1) %>%
                  inner_join(get_sentiments("nrc") %>%
                                   filter(sentiment %in% c("positive", "negative"))) %>%
                  mutate(method = "NRC")) %>%
        count(book, method, index = index , sentiment) %>%
        ungroup() %>%
        spread(sentiment, n, fill = 0) %>%
        mutate(sentiment = positive - negative) %>%
        select(book, index, method, sentiment)
```

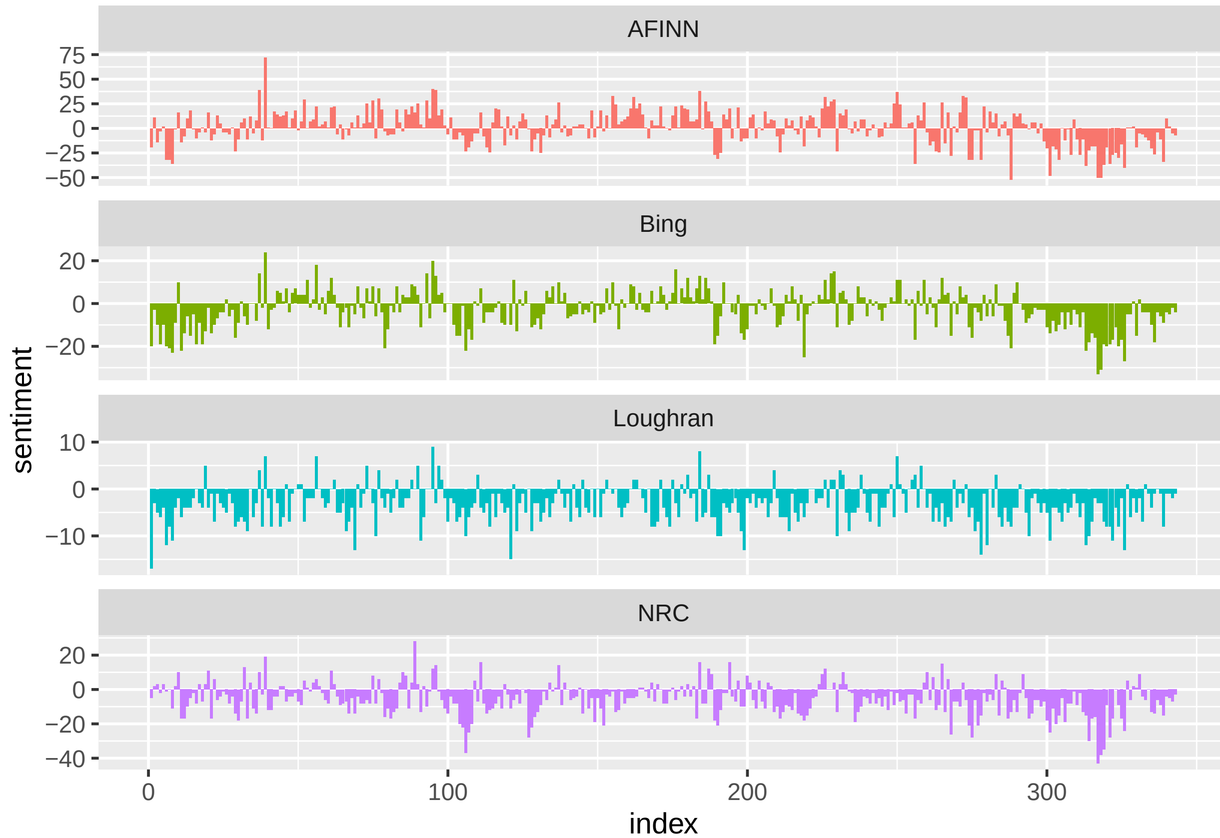
```
## Joining, by = "word"
## Joining, by = "word"
```

We now have an estimate of the net sentiment (positive - negative) in each chunk of the novel text for each sentiment lexicon. We bind them together and plot them.

```r
# Please note need to rerun line 270 alone in order for this plot to run.

bind_rows(afinn,
          bing_and_nrc, loughran) %>%
  ggplot(aes(index, sentiment, fill = method)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~method, ncol = 1, scales = "free_y")
```

The four different lexicons for calculating sentiment give results that are different in an absolute sense but have fairly similar relative trajectories through the novel. We see similar dips and peaks in sentiment at about the same places in the novel, but the absolute values are significantly different. In some instances, it apears the AFINN lexicon finds more positive sentiments than the Bing, NRC and Loughran lexicon. NRC and Loughran seems more negative sentiments then the other two sentiments. This output also allows us to compare across the chapters of the novel. You can compare how the chapters differ in their sentiment (both direction and magnitude) across the series.



The Stanford CoreNLP tools and the sentimentr R package (currently available on Github but not CRAN) are examples of such sentiment analysis algorithms. For these, we may want to tokenize text into sentences.

```
tibble(text = half_blood_prince) %>%
  unnest_tokens(sentence, text, token = "sentences")
```

```
## # A tibble: 12,295 x 1
##    sentence
##    <chr>
##  1 it was nearing midnight and the prime minister was sitting alone in his offi~
##  2 he was waiting for a call from the president of a far distant country, and b~
##  3 the more he attempted to focus on the print on the page before him, the more~
##  4 this particular opponent had appeared on the news that very day, not only to~
##  5 the prime minister's pulse quickened at the very thought of these accusation~
##  6 how on earth was his government supposed to have stopped that bridge collaps~
##  7 it was outrageous for anybody to suggest that they were not spending enough ~
##  8 the bridge was fewer than ten years old, and the best experts were at a loss~
##  9 and how dare anyone suggest that it was lack of policemen that had resulted ~
## 10 or that the government should have somehow foreseen the freak hurricane in t~
## # ... with 12,285 more rows
```

CONCLUSION

Sentiment analysis provides a way to understand the attitudes and opinions expressed in texts. We explored how to approach sentiment analysis using tidy data principles; when text data is

in a tidy data structure, sentiment analysis can be implemented as an inner join. We can use sentiment analysis to understand how a narrative arc changes throughout its course or what words with emotional and opinion content are important for a particular text.

We can clearly see that the lexicon that is chosen can have a big impact on the analysis and we need to be careful to take this into consideration on any analysis.