Data 607 Sentiment Analysis With Tidy Data, Part 1

Enid Roman

2022-11-04

Part 1 of 2 I recreate the code from Chapter 2 of the book Text Mining with R by Juilia Silge and David Robinson. https://www.tidytextmining.com/sentiment.html

CHAPTER 2 SENTIMENT ANALYSIS WITH TIDY DATA

```
# Install Libraries I felt I Needed

#install.packages("tidyverse")
#install.packages("textdata")
#install.packages("gutenbergr")
#install.packages("DT")
#install.packages("flextable")
#install.packages("wordcloud")
```

THE SENTIMENTS DATASETS

```


# Load Libraries

library(tidyverse)
library(tidytext)
```

Afinn lexicon assigns words with a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment.

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

```
library(textdata)
```

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

```
# The function get_sentiments() allows us to get specific sentiment lexicons

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
## # ... with 2,467 more rows
```

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

The bing lexicon categorizes words in a binary fashion into positive and negative categories.

```
## # 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
## # ... with 6,776 more rows
```

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

The nrc lexicon categorizes words in a binary fashion ("yes"/"no") into categories of positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust.

```
## # 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
## # ... with 13,862 more rows
```

SENTIMENT ANALYSIS WITH INNER JOIN

```
library(janeaustenr)
```

Here we look at the words with a joy score from the NRC lexicon. First, we need to take the text of the novels and convert the text to the tidy format using unnest_tokens(). Set up some other columns to keep track of which line and chapter of the book each word comes from; we use group_by and mutate to construct those columns.

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

```
library(dplyr)
library(stringr)

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)
```

Now that the text is in a tidy format with one word per row, we are ready to do the sentiment analysis. First, let's use the NRC lexicon and filter() for the joy words. Next, let's filter() the data frame with the text from the books for the words from Emma and then use inner_join() to perform the sentiment analysis. What are the most common joy words in Emma? Let's use count() from dplyr.

```
nrc_joy <- get_sentiments("nrc") %>%
  filter(sentiment == "joy")

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

There are mostly positive, happy words about hope, friendship, and love here. We also see some words that may not be used joyfully by Austen ("found", "present")

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

```
## # A tibble: 301 x 2
##    word          n
##    <chr>     <int>
##  1 good        359
##  2 friend      166
##  3 hope        143
##  4 happy       125
##  5 love        117
##  6 deal         92
##  7 found        92
##  8 present      89
##  9 kind         82
## 10 happiness    76
## # ... with 291 more rows
```

Here we examine how sentiment changes throughout each novel. We can do this with just a handful of lines that are mostly dplyr functions. First, we find a sentiment score for each word using the Bing lexicon and inner_join().

Then, we count up how many positive and negative words there are in defined sections of each book. We define an index here to keep track of where we are in the narrative; this index (using integer division) counts up sections of 80 lines of text.

The %/% operator does integer division (x %/% y is equivalent to floor(x/y)) so the index keeps track of which 80-line section of text we are counting up negative and positive sentiment in.

```r
library(tidyr)

jane_austen_sentiment <- tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(book, index = linenumber %/% 80, sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(sentiment = positive - negative)
```

We use pivot_wider() so that we have negative and positive sentiment in separate columns, and lastly calculate a net sentiment (positive - negative).

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

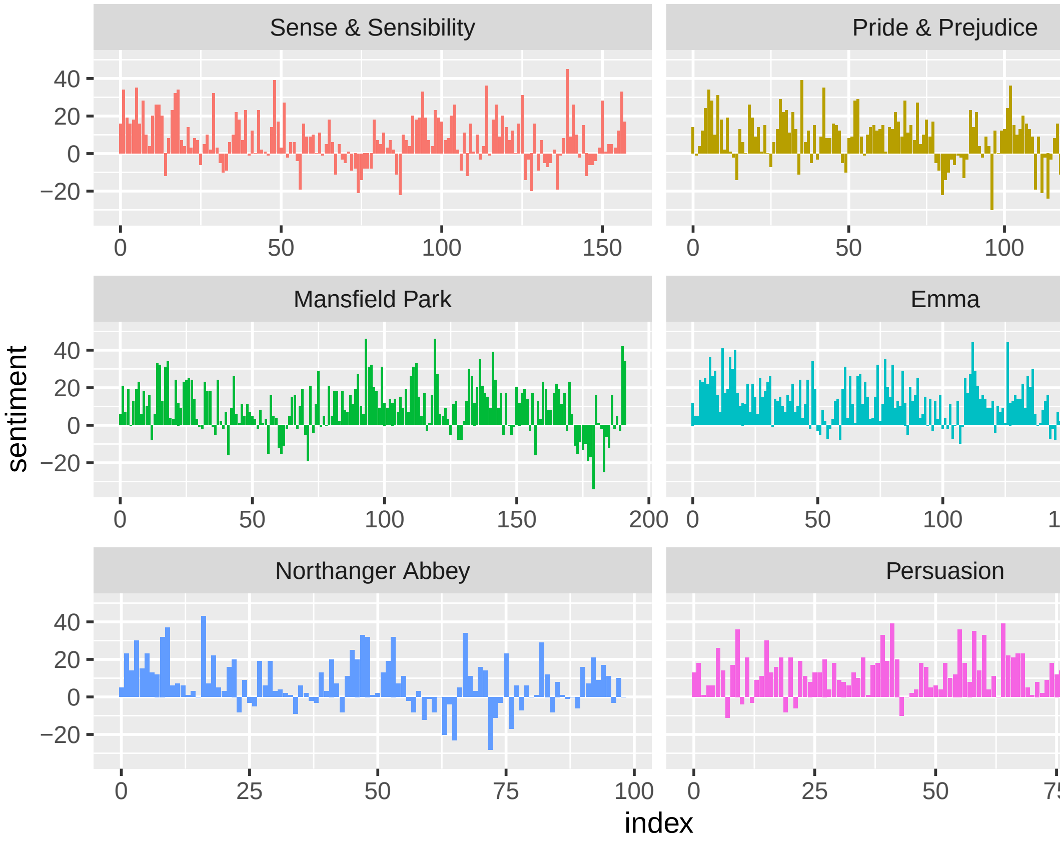
We sentiment through the narratives of Jane Austen's novels

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.

```
library(ggplot2)

ggplot(jane_austen_sentiment, aes(index, sentiment, fill = book)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~book, ncol = 2, scales = "free_x")
```

Here we see how the plot of each novel changes toward more positive or negative sentiment over



the trajectory of the story.
### COMPARING THE THREE SENTIMENT DICTIONARIES

```
pride_prejudice <- tidy_books %>%
  filter(book == "Pride & Prejudice")

pride_prejudice
```

We want some more information on which sentiment lexicons is appropriate for our purposes. We will use all three sentiment lexicons and examine how the sentiment changes across the narrative arc of Pride and Prejudice. First, we use filter() to choose only the words from the one novel we are interested in.

```
## # A tibble: 122,204 x 4
##    book              linenumber chapter word
##    <fct>                  <int>   <int> <chr>
##   1 Pride & Prejudice          1       0 pride
##   2 Pride & Prejudice          1       0 and
##   3 Pride & Prejudice          1       0 prejudice
##   4 Pride & Prejudice          3       0 by
##   5 Pride & Prejudice          3       0 jane
##   6 Pride & Prejudice          3       0 austen
##   7 Pride & Prejudice          7       1 chapter
##   8 Pride & Prejudice          7       1 1
##   9 Pride & Prejudice         10       1 it
## 10 Pride & Prejudice         10       1 is
## # ... with 122,194 more rows
```

We use inner_join() to calculate the sentiment in different ways.

To find a sentiment score in chunks of text throughout the novel, we will need to use a different pattern for the AFINN lexicon than for the other two.

Weuse integer division (%/%) to define larger sections of text that span multiple lines, and we can use the same pattern with count(), pivot_wider(), and mutate() to find the net sentiment in each of these sections of text.

```r
afinn <- pride_prejudice %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(index = linenumber %/% 80) %>%
  summarise(sentiment = sum(value)) %>%
  mutate(method = "AFINN")
```

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

```r
bing_and_nrc <- bind_rows(
  pride_prejudice %>%
    inner_join(get_sentiments("bing")) %>%
    mutate(method = "Bing et al."),
  pride_prejudice %>%
    inner_join(get_sentiments("nrc") %>%
                 filter(sentiment %in% c("positive",
                                         "negative"))
    ) %>%
    mutate(method = "NRC")) %>%
  count(method, index = linenumber %/% 80, sentiment) %>%
  pivot_wider(names_from = sentiment,
              values_from = n,
              values_fill = 0) %>%
  mutate(sentiment = positive - negative)
```

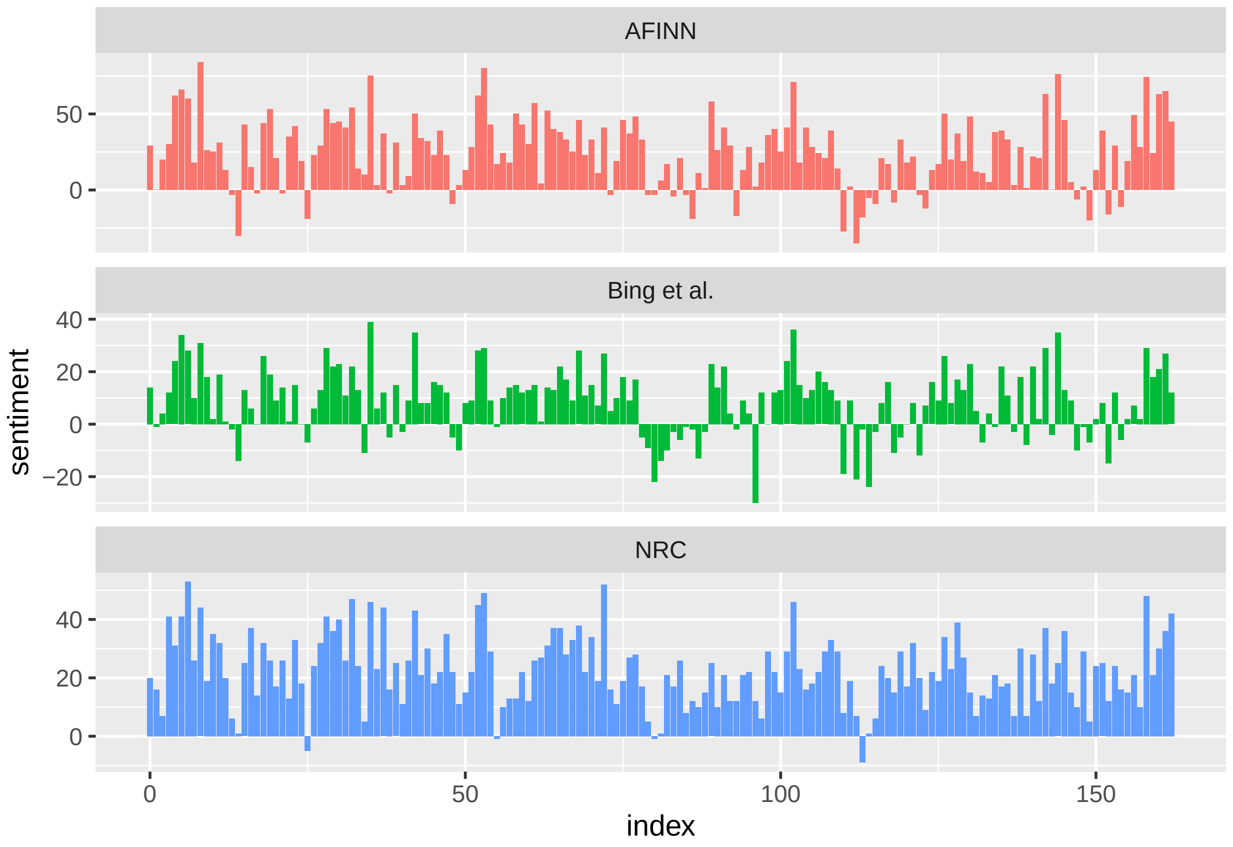
```
## Joining, by = "word"
## Joining, by = "word"
```

We compare three sentiment lexicons using Pride and Prejudice

We 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 visualize them.

The three different lexicons for calculating sentiment give results that are different in an absolute sense but have 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. The AFINN lexicon gives the largest absolute values, with high positive values. The lexicon from Bing et al. has lower absolute values and seems to label larger blocks of contiguous positive or negative text. The NRC results are shifted higher relative to the other two, labeling the text more positively, but detects similar relative changes in the text. We find similar differences between the methods when looking at other novels; the NRC sentiment is high, the AFINN sentiment has more variance, the Bing et al. sentiment appears to find longer stretches of similar text, but all three agree roughly on the overall trends in the sentiment through a narrative arc.

```
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")
```



### 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
```

```
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

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.

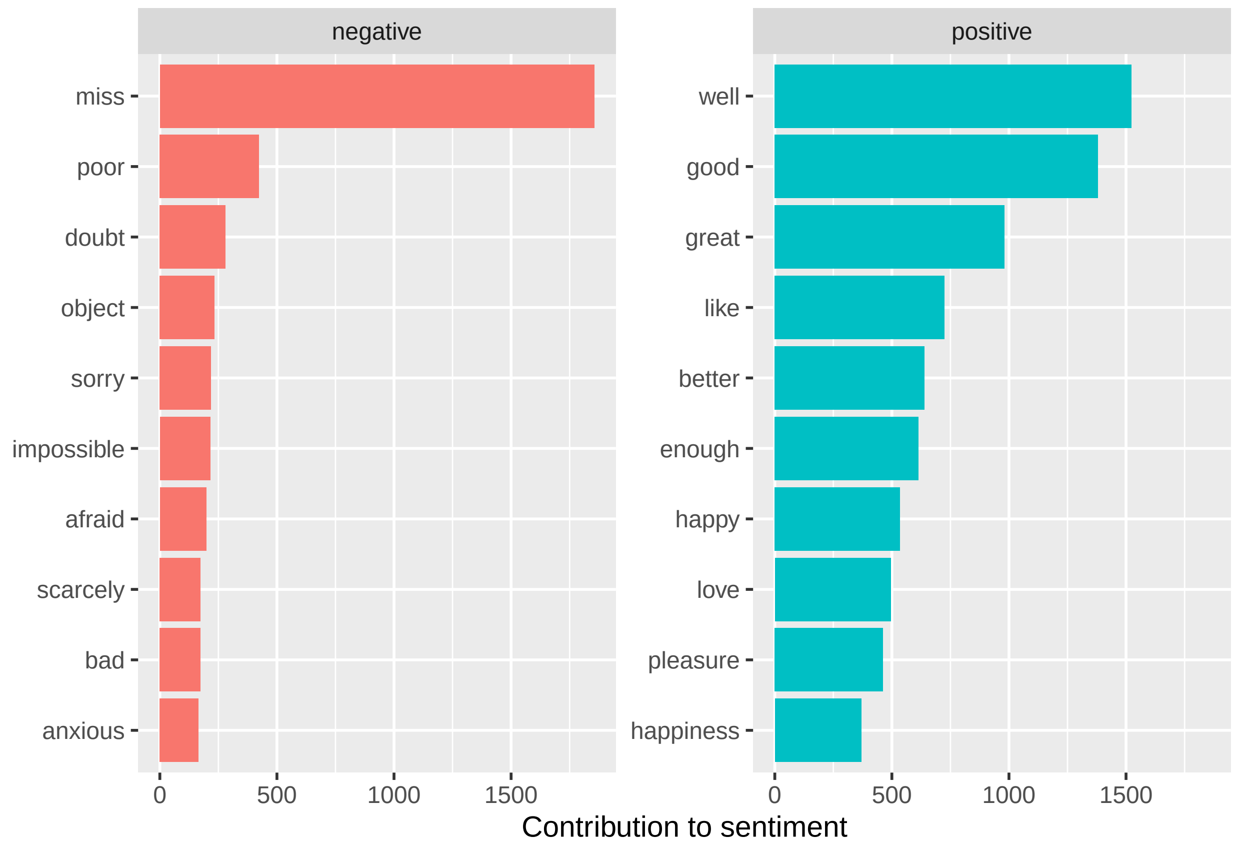
```
bing_word_counts <- tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

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

Words that contribute to positive and negative sentiment in Jane Austen's novels

Here we can pipe straight into ggplot2.

```
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)
```



### The word "miss" is coded as negative but it is used as a title for young, unmarried women in Jane Austen's works. If it were appropriate for our purposes, we could easily add "miss" to a custom stop-words list using bind_rows().

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

custom_stop_words
```

```
## # A tibble: 1,150 x 2
##    word        lexicon
##    <chr>       <chr>
##  1 miss        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
```

WORDCLOUDS

The most common words in Jane Austen's novels.

```
library(wordcloud)
```

The wordcloud package, which uses base R graphics. We see most common words in Jane Austen's works as a whole again, but this time as a wordcloud.

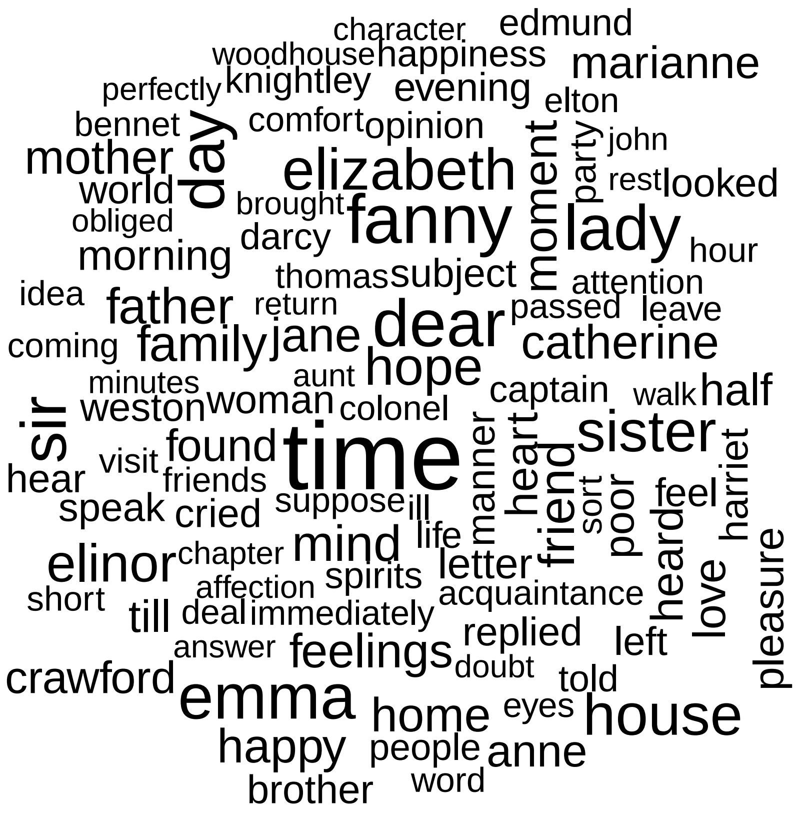
```
## Warning: package 'wordcloud' was built under R version 4.2.2
```

```
## Loading required package: RColorBrewer
```

```
tidy_books %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100))
```

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

```
## Warning in wordcloud(word, n, max.words = 100): miss could not be fit on page.
## It will not be plotted.
```



#### Most common positive and negative words in Jane Austen's novels

Using the function comparison.cloud(), you may need to turn the data frame into a matrix with reshape2's acast(). We do the sentiment analysis to tag positive and negative words using an inner join, then find the most common positive and negative words. Until the step where we need to send the data to comparison.cloud(), this can all be done with joins, piping, and dplyr because our data is in tidy format.

```
library(reshape2)
```

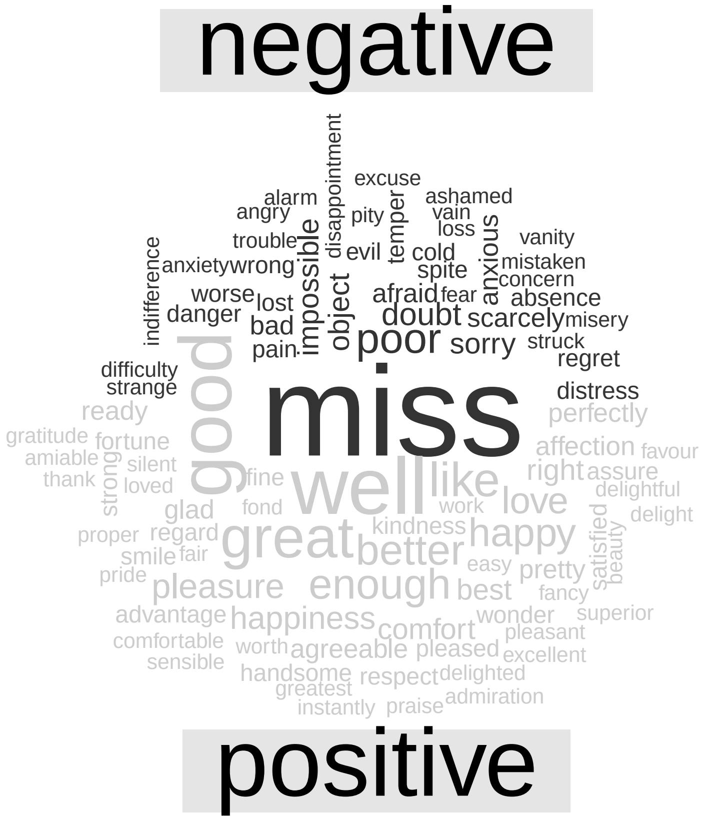
The size of a word's text is in proportion to its frequency within its sentiment. We see the most important positive and negative words, but the sizes of the words are not comparable across sentiments.

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

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

```
tidy_books %>%
  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"
```



### LOOKING AT UNITS BEYOND JUST WORDS

```
p_and_p_sentences <- tibble(text = prideprejudice) %>%
  unnest_tokens(sentence, text, token = "sentences")
```

```
p_and_p_sentences$sentence[2]
```

Some sentiment analysis algorithms look beyond only unigrams (i.e. single words) to try to understand the sentiment of a sentence as a whole. R packages included coreNLP (T. Arnold and Tilton 2016), cleanNLP (T. B. Arnold 2016), and sentimentr (Rinker 2017) are examples of such sentiment analysis algorithms.

```
## [1] "by jane austen"
```

The sentence tokenizing seem to have a bit of trouble with UTF-8 encoded text, especially with sections of dialogue; it does much better with punctuation in ASCII. Before unnesting, we use the function unnest_tokens() to split into tokens using a regex pattern. We could use this to split the text of Jane Austen's novels into a data frame by chapter.

```r
austen_chapters <- austen_books() %>%
  group_by(book) %>%
  unnest_tokens(chapter, text, token = "regex",
                pattern = "Chapter|CHAPTER [\\dIVXLC]") %>%
  ungroup()

austen_chapters %>%
  group_by(book) %>%
  summarise(chapters = n())
```

We have recovered the correct number of chapters in each novel (plus an "extra" row for each novel title). In the austen_chapters data frame, each row corresponds to one chapter.

```
## # A tibble: 6 x 2
##   book                chapters
##   <fct>                  <int>
## 1 Sense & Sensibility       51
## 2 Pride & Prejudice         62
## 3 Mansfield Park            49
## 4 Emma                      56
## 5 Northanger Abbey          32
## 6 Persuasion                25
```

We use tidy text analysis to ask questions such as what are the most negative chapters in each of Jane Austen's novels?

First, get the list of negative words from the Bing lexicon.

Second, make a data frame of how many words are in each chapter so we can normalize for the length of chapters.

Then, find the number of negative words in each chapter and divide by the total words in each chapter.

These are the chapters with the most sad words in each book, normalized for number of words in the chapter.

```r
bingnegative <- get_sentiments("bing") %>%
  filter(sentiment == "negative")
```

```r
wordcounts <- tidy_books %>%
  group_by(book, chapter) %>%
  summarize(words = n())
```

Chapter 34 in Pride and Justice has the highest proportion of negative words with ratio of .053.

```
## `summarise()` has grouped output by 'book'. You can override using the
## `.groups` argument.
```

```r
tidy_books %>%
  semi_join(bingnegative) %>%
  group_by(book, chapter) %>%
  summarize(negativewords = n()) %>%
  left_join(wordcounts, by = c("book", "chapter")) %>%
  mutate(ratio = negativewords/words) %>%
  filter(chapter != 0) %>%
  slice_max(ratio, n = 1) %>%
  ungroup()
```

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

```
## # A tibble: 6 x 5
##   book                chapter negativewords words  ratio
##   <fct>                 <int>         <int> <int>  <dbl>
## 1 Sense & Sensibility      43           161  3405 0.0473
## 2 Pride & Prejudice        34           111  2104 0.0528
## 3 Mansfield Park           46           173  3685 0.0469
## 4 Emma                     15           151  3340 0.0452
## 5 Northanger Abbey         21           149  2982 0.0500
## 6 Persuasion                4            62  1807 0.0343
```

###**SUMMARY**

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.