02 Sentiment analysis with tidy data

H. David Shea

29 Jul 2021

Contents

Opinion mining or **sentiment analysis** refers to determining the emotional intent of words - whether the text is positive or negative, say. The following examples show the use of text mining tools to "approach the emotional content of text programmatically."

The `sentiments` datasets

The `tidytext` package provides access to several sentiment lexicons. These are accessible via the `get_sentiments()` function.

```
kable(get_sentiments("afinn")[1:10,])
```

word	value
abandon	-2
abandoned	-2
abandons	-2
abducted	-2
abduction	-2
abductions	-2
abhor	-3
abhorred	-3
abhorrent	-3
abhors	-3

```
kable(get_sentiments("bing")[1:10,])
```

word	sentiment
2-faces	negative
abnormal	negative
abolish	negative
abominable	negative
abominably	negative
abominate	negative
abomination	negative
abort	negative
aborted	negative
aborts	negative

```
kable(get_sentiments("nrc")[1:10,])
```

word	sentiment
abacus	trust
abandon	fear
abandon	negative
abandon	sadness
abandoned	anger
abandoned	fear
abandoned	negative
abandoned	sadness
abandonment	anger
abandonment	fear

"Dictionary-based methods like the ones we are discussing find the total sentiment of a piece of text by adding up the individual sentiment scores for each word in the text."

Some limitations of this approach:

- must be sure that the lexicon and the text being analyzed are similar in style and time reference
- qualifier words - like 'not' in 'not happy' - are treated separately from their qualified term
- in narrative context, concepts like sarcasm and irony will not be captured
- the size of the analyzed text can complicate things if the sentiment changes in different sections of the text

Sentiment analysis with inner join

Here we will use the `nrc` lexicon and filter words that connotate 'joy'. Doing an `inner_join()` of the book text with these 'joy' words, we can determine how often the 'joy' words appeared in the text.

NOTE: for these analyses, we will not remove any stop words as words like 'good' are included in them.

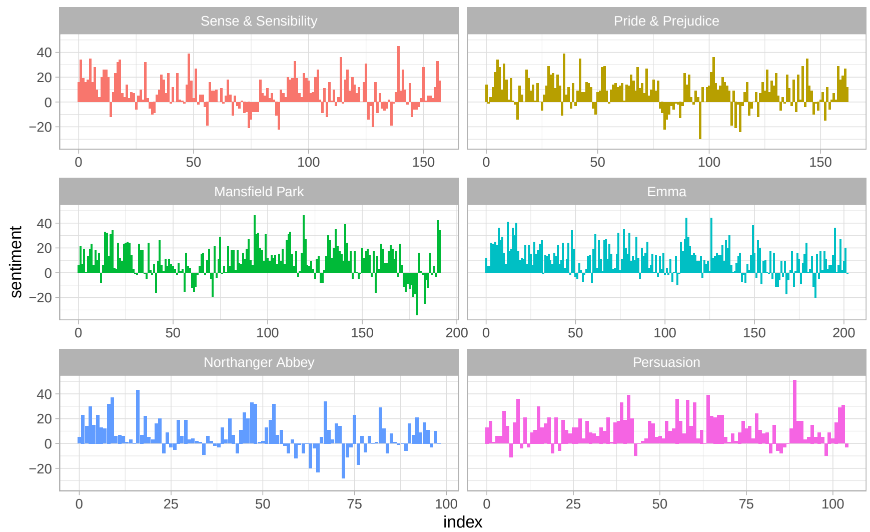
```
tidy_books <- austen_books() %>%
    group_by(book) %>%
    mutate(linenumber = row_number(),
           chapter = cumsum(str_detect(
               text,
               regex("^chapter [\\divxlc]",
```

```r
                        ignore_case = TRUE)
            ))) %>%
    ungroup() %>%
    unnest_tokens(word, text)

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

tidy_books %>%
    filter(book == "Emma") %>%
    inner_join(nrc_joy, by = "word") %>%
    count(word, sort = TRUE) %>%
    filter(n >= 80) %>%
    kable()
```

word	n
good	359
young	192
friend	166
hope	143
happy	125
love	117
deal	92
found	92
present	89
kind	82

Next, we track the change in sentiment through the Jane Austen books. This is done by joining the words to the `bing` lexicon which denotes words as either negative or positive. Then each book is broken up into consecutive 80 line chunks (see `index` in the `count` below). For each 80 line chunk, calculate `sentiment` as the number of `positive` words minus the number of `negative` words. Then plot the `sentiment` versus `index` for each book.

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

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



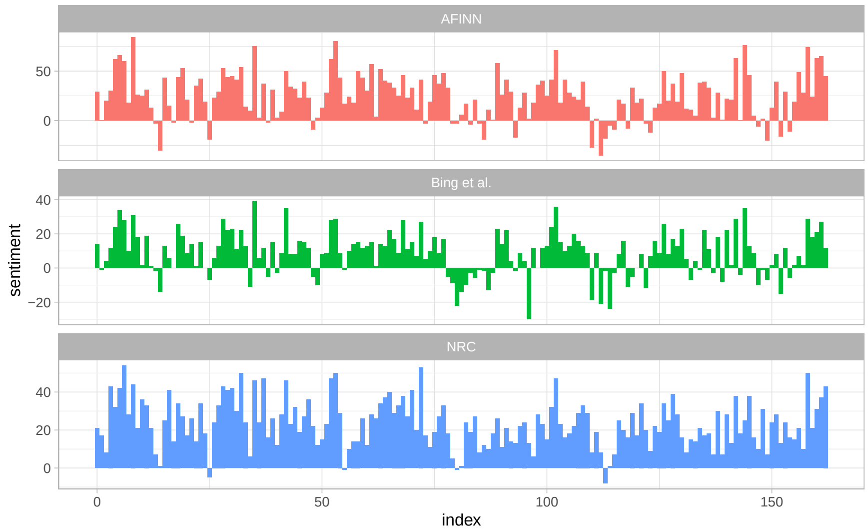
Comparing the three sentiment dictionaries

The three sentiment lexicons that we have reviewed all have variations in the way they sourced their sentimeent indications. And, the `afinn` lexicon uses a `-5` to `+5` scale for sentiment rather than the binary `negative` or `positive` indicators used by then `bing` and `nrc` lexicons. Nonetheless, with a little work, we can compare the sentiment indication change for each of the lexicons across a single work to see where there are variations.

Using *Pride and Prejudice*, as the comparison text. The `afinn` sentiment is calculated as the sum of `-5` to `+5` scores for each word within each text chunk. The `bing` and `nrc` sentiment is calculated as the difference between the number of `positive` and `negative` words within each text chunk.

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

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

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

```
                  values_fill = 0) %>%
    mutate(sentiment = positive - negative)

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") +
    theme_light()
```



There is some variation - especially in scale - however, the general trend of sentiment through times seems to be similar across the lexicons.

To see what drives some of the variation, we can look at `positive` versus `negative` frequency for the lexicons that have a binary coding - `bing` and `nrc`.

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

sentiment	n
negative	3324
positive	2312

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

sentiment	n
negative	4781

sentiment	n
positive	2005

```
comp_nrc_bing <- get_sentiments("nrc") %>%
    filter(sentiment %in% c("positive", "negative")) %>%
    inner_join(get_sentiments("bing"), by = "word") %>%
    transmute(word = word, nrc = sentiment.x, bing = sentiment.y)

table(comp_nrc_bing$nrc, comp_nrc_bing$bing, dnn = c("nrc", "bing")) %>%
    kable()
```

	negative	positive
negative	1675	21
positive	19	681

Both have more negatives than positives, but the ratio of negative to positive is higher in `bing` - which also has more words categorized on positive / negative sentiment. This contributes to the `nrc` scores being generally higher than the `bing` scores on similar texts.

Note, however, that in the cases where they do match on words, they tend to agree on the sentiment as seen in the bottom table.

Most common positive and negative words

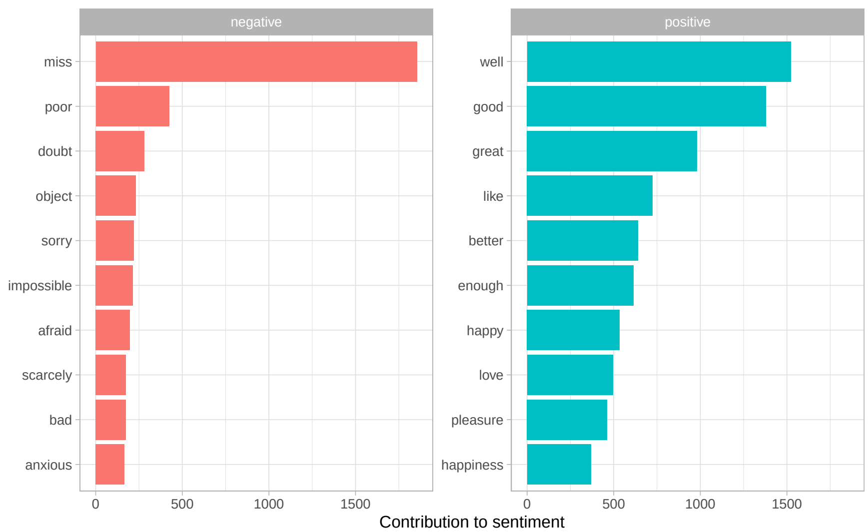
Another aspet that we can investigate is the impact that any word had on the overall sentiment. Here we show that in the following table and plots.

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

kable(bing_word_counts[1:15, ])
```

word	sentiment	n
miss	negative	1855
well	positive	1523
good	positive	1380
great	positive	981
like	positive	725
better	positive	639
enough	positive	613
happy	positive	534
love	positive	495
pleasure	positive	462
poor	negative	424
happiness	positive	369
right	positive	329

word	sentiment	n
best	positive	323
comfort	positive	292

```
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) +
    theme_light()
```



This does show a potential anomaly with the word 'miss' - which is coded as negative sentiment. If the meaning in the text is a reference to a young woman, that would not necessarily be correct.

One way to handle this is with a custom stop words list.

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

kable(custom_stop_words[1:10, ])
```

word	lexicon
miss	custom
a	SMART
a's	SMART
able	SMART

word	lexicon
about	SMART
above	SMART
according	SMART
accordingly	SMART
across	SMART
actually	SMART

Wordclouds

Here we review some alternative graphic views from those used already (i.e., `ggplot`).

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

Figure 1: The most common words in Jane Austen's novels - Wordcloud view

Some functions require a matrix. The package `reshape2` provides the function `acast()` to do this. Then we can use `comparison.cloud()` to see a Wordcloud segragated by positive and negative.

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

Looking at units beyond just words

Here we investigate tokenizing text a levels larger than single words.

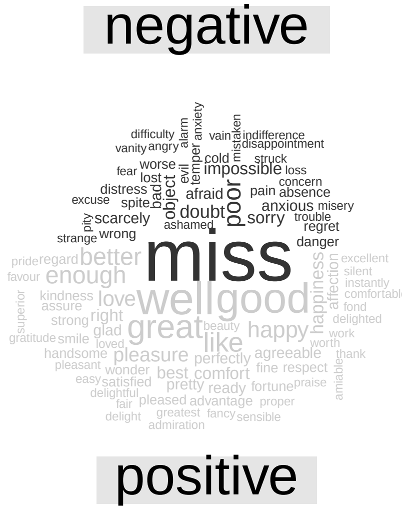


Figure 2: The most common positive versus negative words in Jane Austen's novels - Wordcloud view

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

kable(p_and_p_sentences[1:10,])
```

sentence
pride and prejudice
by jane austen
chapter 1
it is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want of a wife.
however little known the feelings or views of such a man may be on his first entering a neighbourhood, this truth is so well fixed in the minds of the surrounding families, that he is considered the rightful property of some one or other of their daughters.
"my dear mr.

We see that the tokenizing algorithm does have some trouble with the text encoding. One solution is to try changing the encoding.

```
p_and_p_sentences <- tibble(text = prideprejudice) %>%
    mutate(text = iconv(text, to = 'latin1')) %>%
    unnest_tokens(sentence, text, token = "sentences")

kable(p_and_p_sentences[1:10,])
```

sentence
pride and prejudice
by jane austen

sentence
chapter 1
it is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want of a wife.
however little known the feelings or views of such a man may be on his first entering a neighbourhood, this truth is so well fixed in the minds of the surrounding families, that he is considered the rightful property of some one or other of their daughters.
"my dear mr.

But we see in this case that it doesn't really help.

Another option is to use a regular expression to tokenize the text - for instance chapters in a book.

```
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()) %>%
    kable()
```

book	chapters
Sense & Sensibility	51
Pride & Prejudice	62
Mansfield Park	49
Emma	56
Northanger Abbey	32
Persuasion	25

Then, for instance, we can look at sentiment by chapter. The following finds the chapter in each Jane Austen book with the highest negative word ratio in the book.

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

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

tidy_books %>%
    semi_join(bingnegative) %>%
    group_by(book, chapter) %>%
    summarize(negativewords = n()) %>%
    left_join(wordcounts, by = c("book", "chapter")) %>%
    mutate(pct_neg = round(100 * negativewords / words, 2)) %>%
    filter(chapter != 0) %>%
    slice_max(pct_neg, n = 1) %>%
```

```
ungroup() %>%
kable()
```

book	chapter	negativewords	words	pct_neg
Sense & Sensibility	43	161	3405	4.73
Pride & Prejudice	34	111	2104	5.28
Mansfield Park	46	173	3685	4.69
Emma	15	151	3340	4.52
Northanger Abbey	21	149	2982	5.00
Persuasion	4	62	1807	3.43