An introduction to text mining

Ashish Dutt

February 21, 2018

An introduction to text mining

Let's use the text of Jane Austen's 6 completed, published novels from the `janeaustenr` package, and transform them into a tidy format. The `janeaustenr` package provides these texts in a one-row-per-line format, where a line is this context is analogous to a literal printed line in a physical book. Let's start with that, and also use `mutate()` to annotate a linenumber quantity to keep track of lines in the original format

Let's start by installing and loading the required packages in the R environment.

```
# install the following required packages
# install.packages("janeaustenr", dependencies=TRUE)
# install.packages("dplyr", dependencies=TRUE)
# install.packages("stringr", dependencies=TRUE)
# install.packages("utf8", dependencies=TRUE)
# install.packages("wordcloud", dependencies=TRUE)
# install.packages("reshape2", dependencies=TRUE)

# Load the package
library(janeaustenr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(stringr)
library(tidytext)
library(ggplot2)
library(utf8)
library(wordcloud)
```

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

```
library(reshape2)
```

Step 1: We will now look at the jane austen's books.

```
original_books <- austen_books() %>%
  group_by(book) %>%
  mutate(linenumber = row_number(),
         chapter = cumsum(str_detect(text, regex("^chapter[\\divxlc]",ignore_case = TRUE)))) %>%
  ungroup()
# Show the books
head(original_books, 5)
```

```
## # A tibble: 5 x 4
##   text                  book                linenumber chapter
##   <chr>                 <fct>                    <int>   <int>
## 1 SENSE AND SENSIBILITY Sense & Sensibility          1       0
## 2 ""                    Sense & Sensibility          2       0
## 3 by Jane Austen        Sense & Sensibility          3       0
## 4 ""                    Sense & Sensibility          4       0
## 5 (1811)                Sense & Sensibility          5       0
```

Step 2: To work with this as a tidy dataset, we need to restructure it in the one-token-per-row format, which is done with `the unnest_tokens()` function.

```
tidy_books<- original_books %>%
  unnest_tokens(word, text)
head(tidy_books, 5)
```

```
## # A tibble: 5 x 4
##   book                linenumber chapter word
##   <fct>                    <int>   <int> <chr>
## 1 Sense & Sensibility          1       0 sense
## 2 Sense & Sensibility          1       0 and
## 3 Sense & Sensibility          1       0 sensibility
## 4 Sense & Sensibility          3       0 by
## 5 Sense & Sensibility          3       0 jane
```

Now that the data is is in one-word-per-row format, we can manipulate it with tidy tools like `dplyr`. Often in text analysis, we will want to remove stop words; stop words are words that are not useful for an analysis, typically extremely common words such as “the”, “of”, “to”, and so forth in English. We can remove stop words (kept in the tidytext dataset `stop_words`) with an `anti_join()`.

```
data("stop_words")
tidy_books <- tidy_books %>%
  anti_join(stop_words)
```

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

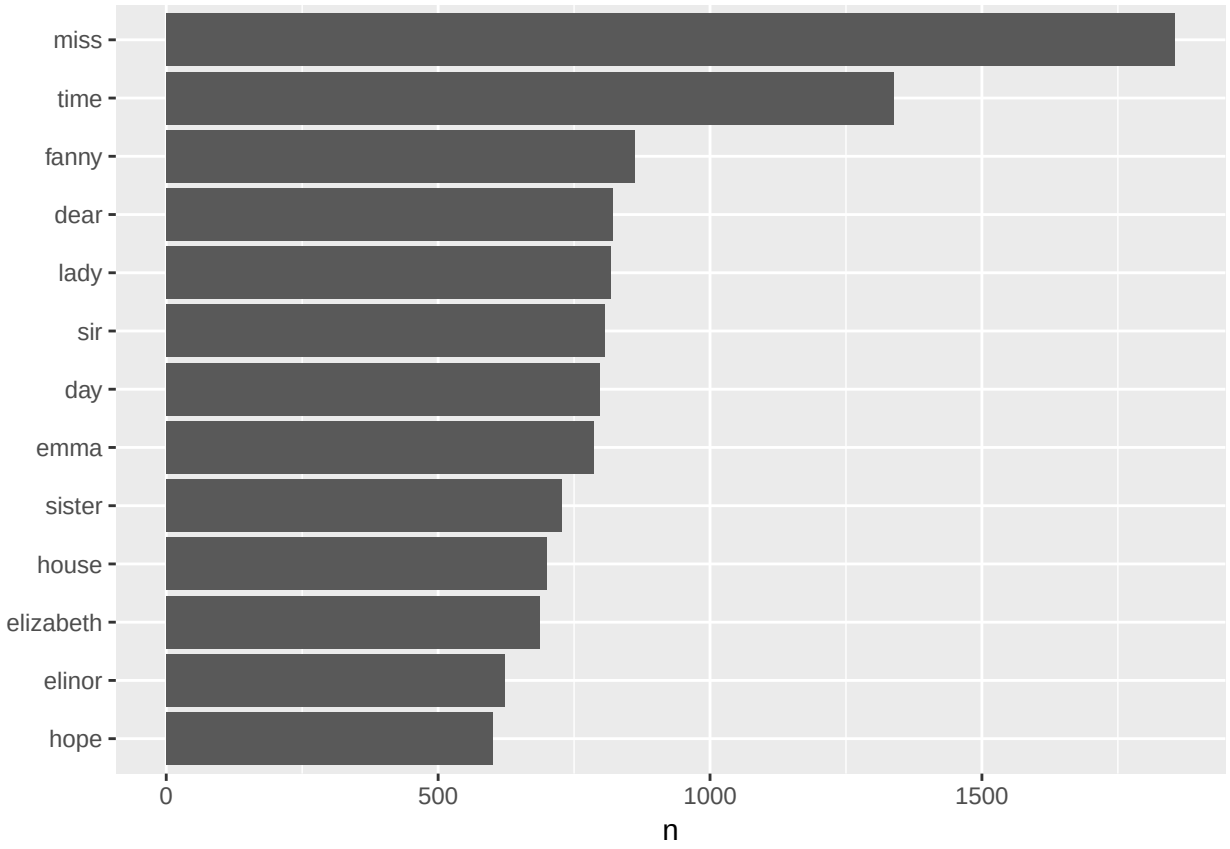
We can also use dplyr’s `count()` to find the most common words in all the books as a whole.

```
tidy_books %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 13,914 x 2
##    word       n
##    <chr>  <int>
##  1 miss    1855
##  2 time    1337
##  3 fanny    862
##  4 dear     822
##  5 lady     817
##  6 sir      806
##  7 day      797
##  8 emma     787
##  9 sister   727
## 10 house    699
## # ... with 13,904 more rows
```

Let’s plot the common occuring words

```
tidy_books %>%
  count(word, sort = TRUE)%>%
  dplyr::filter(n>600) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```



Sentiment analysis with tidy data

One way to analyze the sentiment of a text is to consider the text as a combination of its individual words and the sentiment content of the whole text as the sum of the sentiment content of the individual words. This isn't the only way to approach sentiment analysis, but it is an often-used approach, and an approach that naturally takes advantage of the tidy tool ecosystem.

As discussed above, there are a variety of methods and dictionaries that exist for evaluating the opinion or emotion in text. The tidytext package contains several sentiment lexicons in the `sentiments` dataset.

```
sentiments
```

```
## # A tibble: 27,314 x 4
##    word        sentiment lexicon score
##    <chr>       <chr>     <chr>   <int>
##  1 abacus      trust     nrc        NA
##  2 abandon     fear      nrc        NA
```

```
##  3 abandon     negative  nrc        NA
##  4 abandon     sadness   nrc        NA
##  5 abandoned   anger     nrc        NA
##  6 abandoned   fear      nrc        NA
##  7 abandoned   negative  nrc        NA
##  8 abandoned   sadness   nrc        NA
##  9 abandonment anger     nrc        NA
## 10 abandonment fear      nrc        NA
## # ... with 27,304 more rows
```

The three general-purpose lexicons are

- `AFINN` from Finn Årup Nielsen,
- `bing` from Bing Liu and collaborators, and
- `nrc` from Saif Mohammad and Peter Turney.

All three of these lexicons are based on unigrams, i.e., single words. These lexicons contain many English words and the words are assigned scores for positive/negative sentiment, and also possibly emotions like joy, anger, sadness, and so forth.

```
get_sentiments("afinn")
```

```
## # A tibble: 2,476 x 2
##    word       score
##    <chr>      <int>
##  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 2,466 more rows
```

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

```
## # A tibble: 6,788 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 2-faced     negative
##  2 2-faces     negative
##  3 a+          positive
##  4 abnormal    negative
##  5 abolish     negative
##  6 abominable  negative
##  7 abominably  negative
##  8 abominate   negative
##  9 abomination negative
## 10 abort       negative
## # ... with 6,778 more rows
```

Sentiment analysis with inner join

With data in a tidy format, sentiment analysis can be done as an inner join. This is another of the great successes of viewing text mining as a tidy data analysis task; much as removing stop words is an antijoin operation, performing sentiment analysis is an inner join operation.

Let's look at the joy words in Emma? First, we need to take the text of the novels and convert the text to the tidy format using `unnest_tokens()`, just as we did above. Let's also 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.

```
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)
head(tidy_books, 5)
```

```
## # A tibble: 5 x 4
##   book                linenumber chapter word
##   <fct>                    <int>   <int> <chr>
## 1 Sense & Sensibility          1       0 sense
## 2 Sense & Sensibility          1       0 and
## 3 Sense & Sensibility          1       0 sensibility
## 4 Sense & Sensibility          3       0 by
## 5 Sense & Sensibility          3       0 jane
```

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`.

```
nrcjoy <- get_sentiments("nrc") %>%
  dplyr::filter(sentiment == "joy")

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

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

```
## # A tibble: 303 x 2
##    word        n
##    <chr>   <int>
##  1 good      359
##  2 young     192
##  3 friend    166
##  4 hope      143
##  5 happy     125
##  6 love      117
##  7 deal       92
##  8 found      92
##  9 present    89
## 10 kind       82
```

```
## # ... with 293 more rows
```

We see many positive, happy words about hope, friendship, and love here.

Most common positive and negative words

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

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

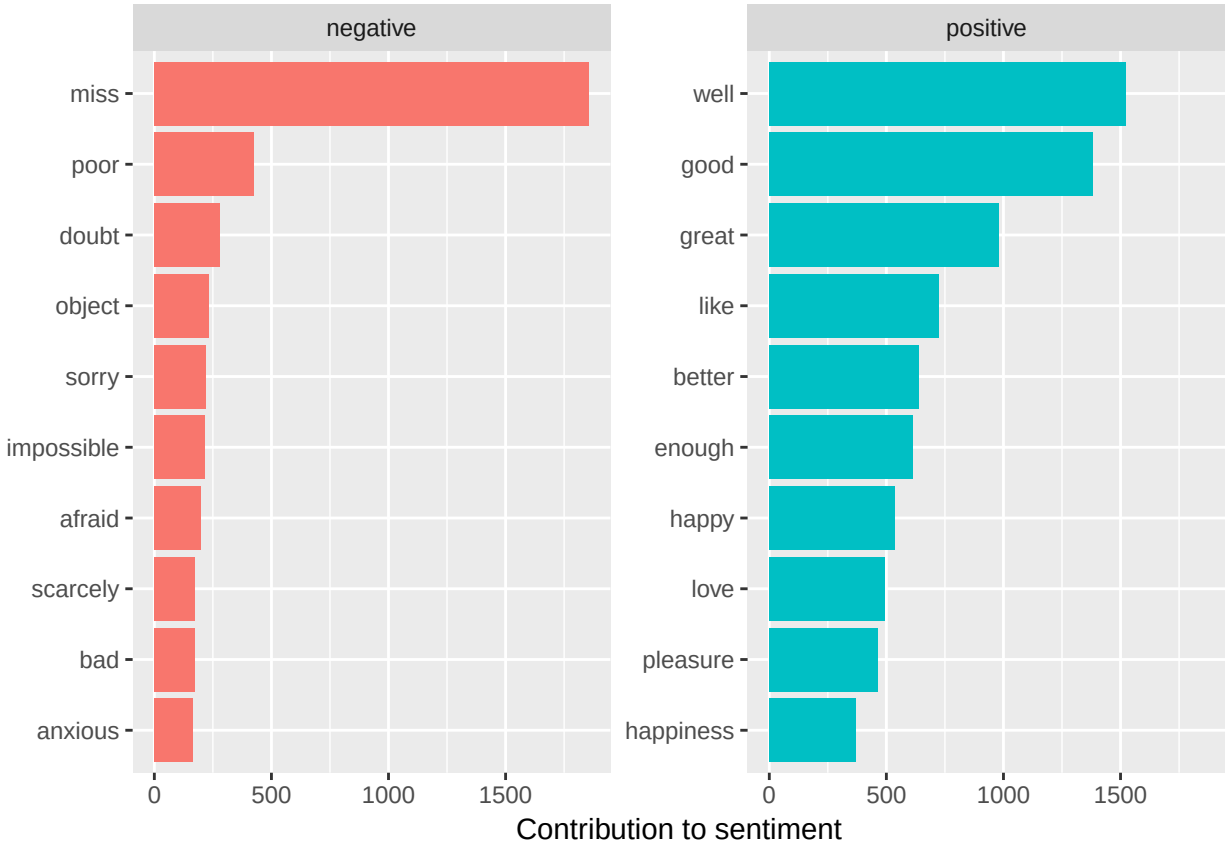
```
bing_word_counts
```

```
## # A tibble: 2,585 x 3
##    word      sentiment     n
##    <chr>     <chr>     <int>
##  1 miss      negative   1855
##  2 well      positive   1523
##  3 good      positive   1380
##  4 great     positive    981
##  5 like      positive    725
##  6 better    positive    639
##  7 enough    positive    613
##  8 happy     positive    534
##  9 love      positive    495
## 10 pleasure  positive    462
## # ... with 2,575 more rows
```

This can be shown visually, and we can pipe straight into ggplot2, if we like, because of the way we are consistently using tools built for handling tidy data frames.

```
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
```



Lets look at wordcloud now

Consider the `wordcloud` package, which uses base R graphics. Let's look at the most common words in Jane Austen's works as a whole again, but this time as a wordcloud.

```
tidy_books %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 80))
```

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

```
## Warning in wordcloud(word, n, max.words = 80): miss could not be fit on
## page. It will not be plotted.
```

Let's do the sentiment analysis to tag positive and negative words using an inner join, then find the most common positive and negative words.

```
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"
```

This should be enough to get you started with text mining in R. If you are interested to read further, I implore you to read the following text;

- Mastering Text Mining with R by Ashish Kumar, Avinash Paul, Packt Publication 2016