ST720 Data Science

Sentiment analysis with tidy data

Seung Jun Shin (sjshin@krea.ac.kr)

Department of Statistics, Korea University

Introduction

- **Word frequency** allowed us to analyze which words are used most frequently in documents and to compare documents.
- When human readers approach a text, we use our understanding of the emotional intent of words to infer whether a section of text is positive or negative.
- We can use the tools of text mining to approach the emotional content of text programmatically.
- 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.

`sentiments` Dataset

- The tidytext package contains several sentiment lexicons
  - `AFINN`: assigns words with a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment.
  - `bing` : categorizes words in a binary fashion into positive and negative categories.
  - `nrc`: categorizes words in a binary fashion (“yes”/“no”) into categories of positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust.
- 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.

`sentiments` Dataset

- All of this information is tabulated in the `sentiments` dataset, and tidytext provides a function `get_sentiments()` to get specific sentiment lexicons.

```r
library(tidytext)
print(get_sentiments("afinn"), n = 5)
```

```
## # 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
## # ... with 2,472 more rows
```

sentiments Dataset

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

sentiments Dataset

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

```
## # A tibble: 13,901 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 13,891 more rows
```

Dictionary-based Sentiment analysis

- 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.

Sentiment analysis with inner join

```r
library(janeaustenr)
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)
```

Sentiment analysis with inner join

```
nrc_joy <- get_sentiments("nrc") %>%
  filter(sentiment == "joy")
tidy_books %>%
  filter(book == "Emma") %>%
  inner_join(nrc_joy) %>%
  count(word, sort = TRUE) %>%
  print(n = 5)
```

```
## 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
## # ... with 298 more rows
```

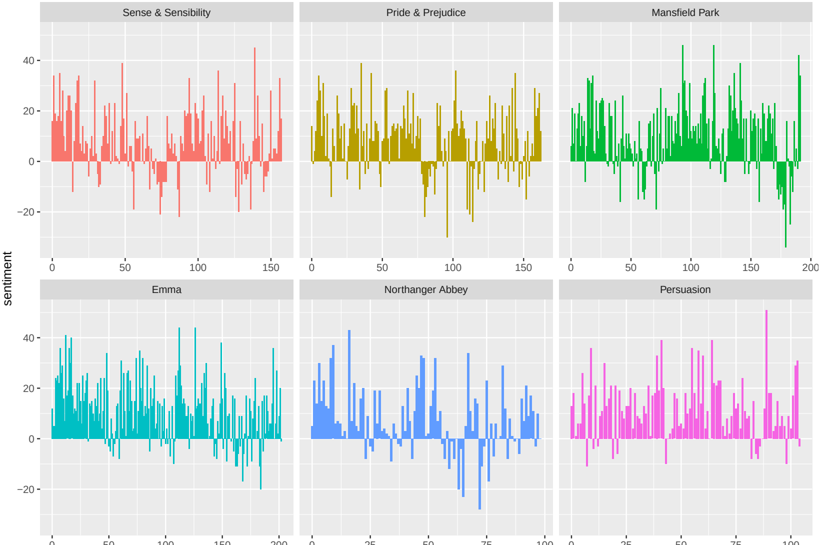
Sentiment analysis with inner join

```r
jane_austen_sentiment <- tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(book, index = linenumber %/% 80, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)
print(jane_austen_sentiment, n = 5)
```

```
## # A tibble: 920 x 5
##   book                index negative positive sentiment
##   <fct>               <dbl>    <dbl>    <dbl>     <dbl>
## 1 Sense & Sensibility     0       16       32        16
## 2 Sense & Sensibility     1       19       53        34
## 3 Sense & Sensibility     2       12       31        19
## 4 Sense & Sensibility     3       15       31        16
## 5 Sense & Sensibility     4       16       34        18
## # ... with 915 more rows
```

Sentiment analysis with inner join

```
ggplot(jane_austen_sentiment, aes(index, sentiment, fill = book)
  geom_col(show.legend = FALSE) +
  facet_wrap(~book, ncol = 3, scales = "free_x")
```



Most common positive and negative words

- Analyze word counts that contribute to each sentiment.

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

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

```r
print(bing_word_counts, n = 5)
```

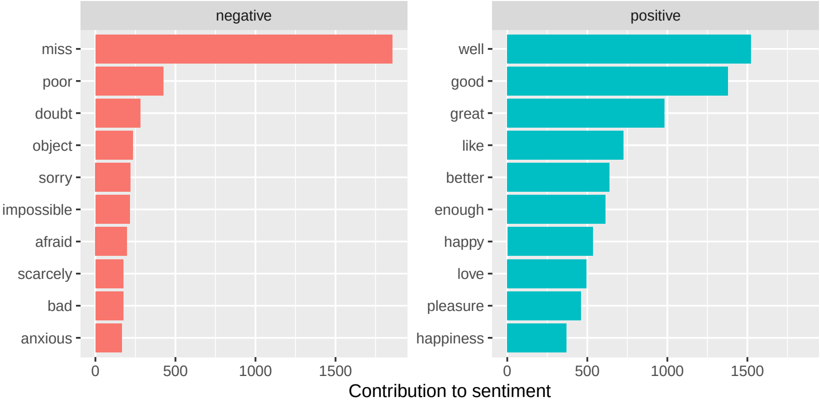
```
## # A tibble: 2,585 x 3
##   word  sentiment     n
##   <chr> <chr>     <int>
## 1 miss  negative   1855
## 2 well  positive   1523
## 3 good  positive   1380
## 4 great positive    981
## 5 like  positive    725
## # ... with 2,580 more rows
```

Most common positive and negative words

- Let's Visulaize it

```r
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()
```

Most common positive and negative words



Stop Words Editing

- miss is coded as negative but it is used as a title for young.
- 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)
print(custom_stop_words, n = 5)
```

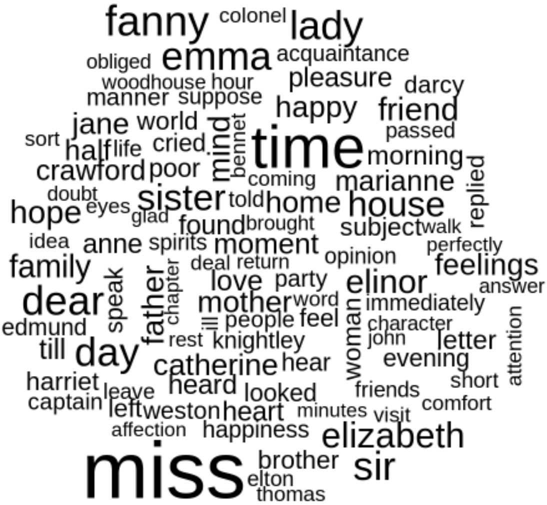
```
## # A tibble: 1,150 x 2
##   word  lexicon
##   <chr> <chr>
## 1 miss  custom
## 2 a     SMART
## 3 a's   SMART
## 4 able  SMART
## 5 about SMART
## # ... with 1,145 more rows
```

Wordclouds

- Tidy format is useful for wordclouds in `wordcloud` package.

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

Wordclouds

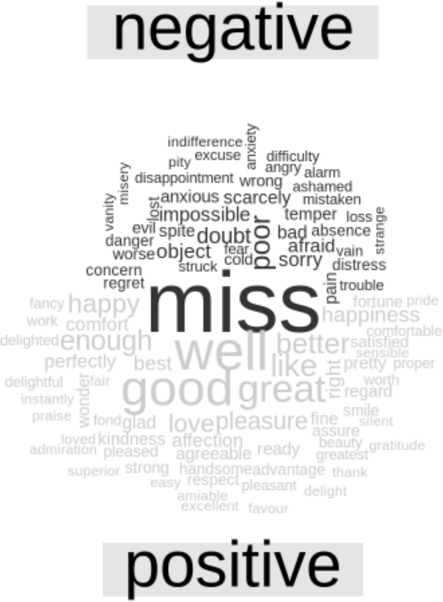

fanny
colonel
lady
obliged
emma
acquaintance
woodhouse
hour
pleasure
darcy
manner
suppose
happy
friend
jane
world
mind
bennet
time
passed
sort
half
life
cried
morning
crawford
poor
coming
marianne
replied
doubt
sister
told
home
house
hope
eyes
glad
found
brought
subject
walk
idea
anne
spirits
moment
perfectly
opinion
deal
return
feelings
family
speak
father
chapter
love
party
elinor
answer
dear
mother
word
immediately
edmund
ill
people
feel
woman
character
attention
rest
knightley
john
letter
till
day
catherine
hear
evening
harriet
heard
short
leave
looked
friends
captain
left
weston
heart
minutes
visit
comfort
affection
happiness
elizabeth
miss
brother
sir
elton
thomas

Wordclounds: Comparison

```
library(reshape2)
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)
```

- Check `acast` function in `reshape2` package.

Wordclounds: Comparison



Looking at units beyond just words

- Sometimes it is useful or necessary to look beyond single words.
- For example, one can tokenize text in a sentence level:

```
PandP_sentences <- tibble(text = prideprejudice) %>%
  unnest_tokens(sentence, text, token = "sentences")
PandP_sentences$sentence[2]
```

```
## [1] "however little known the feelings or views of such a man
```

Looking at units beyond just words

- Token can be expressed as regex.

```
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()) %>% print(n = 3)
```

```
## # A tibble: 6 x 2
##   book                chapters
##   <fct>                  <int>
## 1 Sense & Sensibility       51
## 2 Pride & Prejudice         62
## 3 Mansfield Park            49
## # ... with 3 more rows
```

Looking at units beyond just words

- Which chapter is most negative in each of Jane Austen's novels?

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

wordcounts <- tidy_books %>%
  group_by(book, chapter) %>%
  summarize(words = n())
print(wordcounts, n  = 5)
```

```
## # A tibble: 275 x 3
## # Groups:   book [6]
##   book                chapter words
##   <fct>                 <int> <int>
## 1 Sense & Sensibility       0     7
## 2 Sense & Sensibility       1  1571
## 3 Sense & Sensibility       2  1970
## 4 Sense & Sensibility       3  1538
## 5 Sense & Sensibility       4  1952
## # ... with 270 more rows
```

Looking at units beyond just words

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

```
## # 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.
    - how a narrative arc changes throughout its course or what words with emotional and opinion content are important for a particular text.
- With tidy text data, sentiment analysis can be implemented as an `inner_join`.