Sentiment Analysis and Topic Modeling

Kylie Ariel Bemis

11/6/2018

Sentiment analysis

Sentiment analysis allows us to attach opinions and sentiments to text data in order to analyze its emotional intent or impressions on human readers.

One common way to approach sentiment analysis is by attaching sentiments to individuals words with a common lexicon. The `tidytext` package offers three lexicons for sentiment analysis.

AFINN sentiments

Positive or negative sentiment score:

```r
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
```

bing sentiments

“Positive” or “negative” only:

```
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
```

nrc sentiments

10 distinct sentiments:

```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
```

Analyze sentiment in Jane Austen

Suppose we want to analyze sentiment in Jane Austen's novels.

First, we load the text of her novels into R as a tidy text data frame:

```
library(stringr)
library(janeaustenr)
tidy_austen <- 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)
```

```r
tidy_austen
```

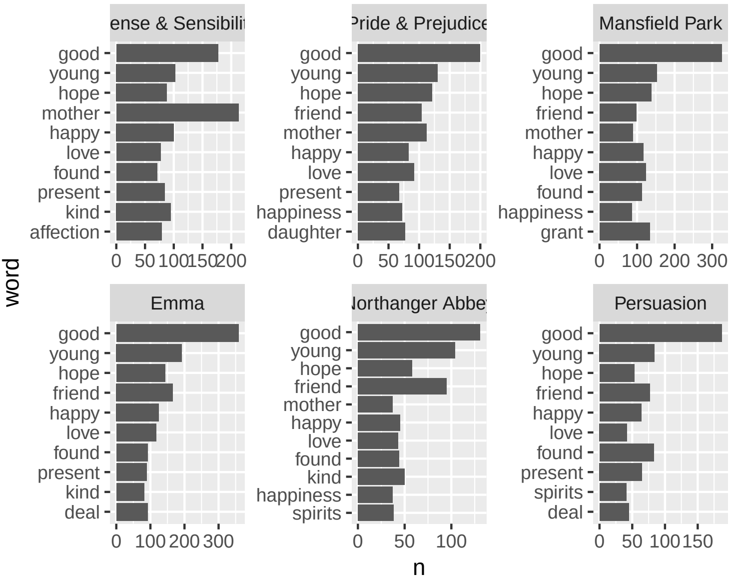
```
## # A tibble: 725,055 x 4
##    book                linenumber chapter word
##    <fct>                    <int>   <int> <chr>
##  1 Sense & Sensibility          1       0 sense
##  2 Sense & Sensibility          1       0 and
##  3 Sense & Sensibility          1       0 sensibility
##  4 Sense & Sensibility          3       0 by
##  5 Sense & Sensibility          3       0 jane
##  6 Sense & Sensibility          3       0 austen
##  7 Sense & Sensibility          5       0 1811
##  8 Sense & Sensibility         10       1 chapter
##  9 Sense & Sensibility         10       1 1
## 10 Sense & Sensibility         13       1 the
## # ... with 725,045 more rows
```

Find words associated with “joy”

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

tidy_austen %>%
  inner_join(nrcjoy, by="word") %>%
  count(book, word, sort=TRUE) %>%
  mutate(word = reorder(word, n)) %>%
  group_by(book) %>%
  top_n(10) %>%
  ggplot(aes(x=word, y=n)) +
  geom_col(show.legend=FALSE) +
  facet_wrap(~book, scales = "free") +
  coord_flip()
```

```
## Selecting by n
```

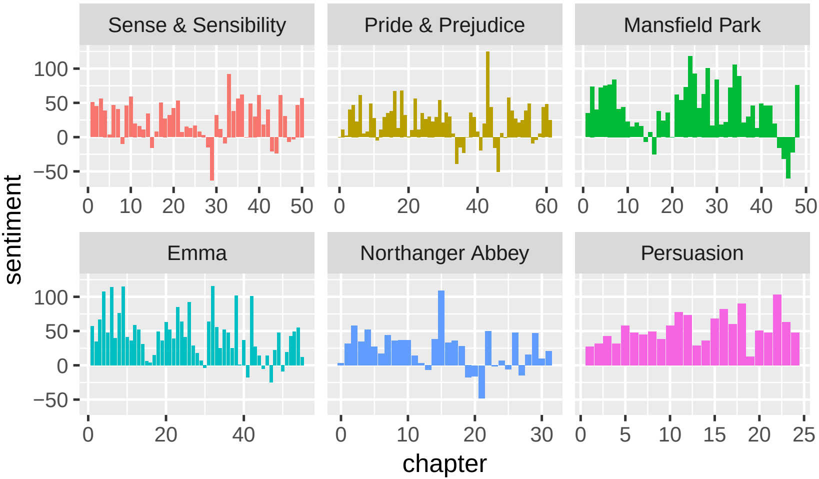


Visualize pos/neg sentiment chapter-by-chapter

To visualize the sentiment chapter-by-chapter, we can sum the count of positive and negative sentiments in each chapter.

```r
austen_sentiment <- tidy_austen %>%
  inner_join(get_sentiments("bing"), by="word") %>%
  count(book, chapter, sentiment) %>%
  spread(sentiment, n, fill = 0L) %>%
  mutate(sentiment = positive - negative)

austen_sentiment %>%
  ggplot(aes(chapter, sentiment, fill = book)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~book, ncol = 3, scales = "free_x")
```



Sense & Sensibility
Pride & Prejudice
Mansfield Park
Emma
Northanger Abbey
Persuasion
sentiment
chapter
100
50
0
−50
0
10
20
30
40
50
60
5
15
25

Get context for sentiment analysis with bigrams

A major drawback of attaching sentiment to individual words is that we lose the context surrounding each word.

By considering multiple words and words that co-occur with each other, we can introduce an important context for sentiment analysis.

One way to get context for sentiment analysis is by analyzing bigrams.

```
austen_bigrams <- austen_books() %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
  separate(bigram, c("word1", "word2"), sep = " ")
```

Words commonly preceded by “not”

```r
austen_bigrams %>%
  filter(word1 == "not", !word2 %in% stop_words$word) %>%
  count(word1, word2, sort = TRUE)
```

```
## # A tibble: 988 x 3
##    word1 word2            n
##    <chr> <chr>        <int>
##  1 not   hear            39
##  2 not   speak           35
##  3 not   expect          34
##  4 not   bear            33
##  5 not   imagine         26
##  6 not   understand      26
##  7 not   suppose         25
##  8 not   care            23
##  9 not   feel            22
## 10 not   immediately     22
## # ... with 978 more rows
```

Analyze comonly negated words in Jane Austen

"Not understand" and "not care" should have a very different sentiment attributed to them than "understand" or "care" should.

Let's analyze the most commonly negated words in Jane Austen's novels and attempt to measure their contribution to a sentiment analysis based only on individual words.

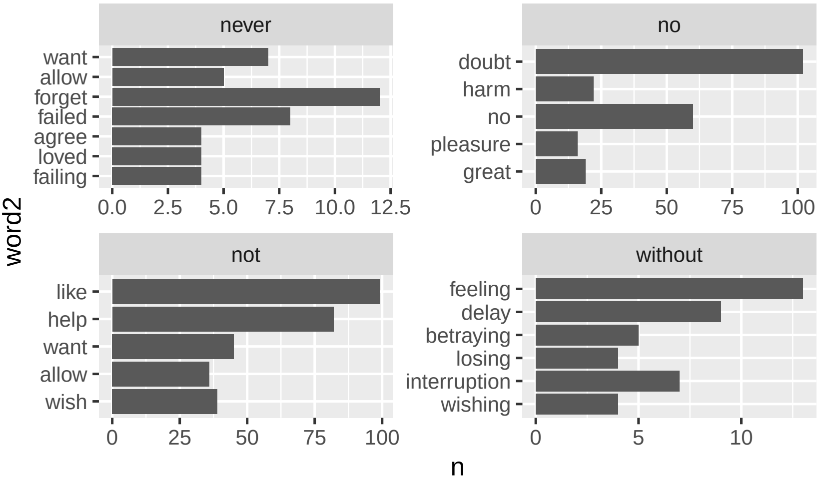
```
negation_words <- c("not", "no", "never", "without")

austen_neg <- austen_bigrams %>%
  filter(word1 %in% negation_words) %>%
  inner_join(get_sentiments("afinn"), by = c("word2" = "word"))
  count(word1, word2, score, sort = TRUE) %>%
  ungroup()
```

Plot most commonly negated words in Jane Austen

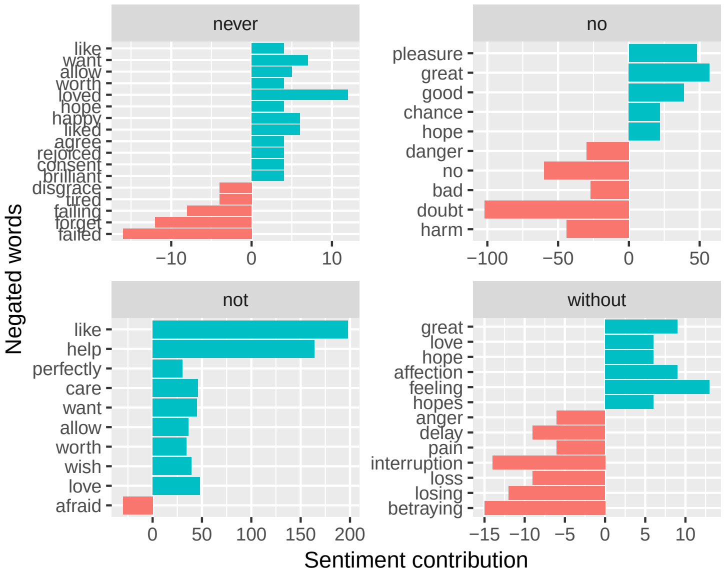
```r
austen_neg %>%
  arrange(desc(n)) %>%
  mutate(word2 = reorder(word2, n)) %>%
  group_by(word1) %>%
  top_n(5) %>%
  ggplot(aes(word2, n)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~word1, scales="free") +
  coord_flip()
```

```
## Selecting by n
```



Negated words by contribution to sentiment score

```
austen_neg %>%
  mutate(contribution = n * score) %>%
  arrange(desc(abs(contribution))) %>%
  mutate(word2 = reorder(word2, contribution)) %>%
  group_by(word1) %>%
  top_n(10, wt=abs(contribution)) %>%
  ggplot(aes(word2, contribution, fill = contribution > 0)) +
  geom_col(show.legend = FALSE) +
  xlab("Negated words") +
  ylab("Sentiment contribution") +
  facet_wrap(~word1, scales="free") +
  coord_flip()
```



never
like
want
allow
worth
loved
hope
happy
liked
agree
rejoiced
consent
brilliant
disgrace
tired
failing
forget
failed
−10
0
10
no
pleasure
great
good
chance
hope
danger
no
bad
doubt
harm
−100
−50
0
50
not
like
help
perfectly
care
want
allow
worth
wish
love
afraid
0
50
100
150
200
without
great
love
hope
affection
feeling
hopes
anger
delay
pain
interruption
loss
losing
betraying
−15
−10
−5
0
5
10
Negated words
Sentiment contribution

Topic modeling

In text mining, it is common to have a collection of documents like tweets or emails that we'd like to categorize. *Topic modeling* is the unsupervised classification of text data into discovered categories or topics.

Latent Dirichlet Allocation (LDA) is a common method for fitting topic models. LDA treats each document as a mixture of topics, and each topic as a mixture of words or terms. Rather than being assigned to a distinct topic, this allows documents to overlap in topic.

The `topicmodels` package provides methods for fitting topic models.

Associate Press data

The `topicmodels` package expects data in the form of a *document-term matrix* rather than the tidy text format.

Fortunately, it is easy to convert between the two formats, but for now we will investigate the Associated Press data provided by the `topicmodels` package, which is already in the correct format.

```
library(topicmodels)

data("AssociatedPress")
AssociatedPress
```

```
## <<DocumentTermMatrix (documents: 2246, terms: 10473)>>
## Non-/sparse entries: 302031/23220327
## Sparsity           : 99%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

This data is a collection of 2246 news articles. We would like to divide the articles into topics.

Fit a topic model to AP data

We can use the `LDA()` function from the `topicmodels` package to fit the topic model to the data.

The `tidytext` package provides a `tidy()` function for tidying the results of the fitted model for each parameter for interest.

```
# set a seed so the results are reproducible
ap_lda <- LDA(AssociatedPress, k = 2, control = list(seed = 1234))
ap_topics <- tidy(ap_lda, matrix = "beta")
```

```r
ap_topics
```

```
## # A tibble: 20,946 x 3
##    topic term            beta
##    <int> <chr>          <dbl>
##  1     1 aaron       1.69e-12
##  2     2 aaron       3.90e- 5
##  3     1 abandon     2.65e- 5
##  4     2 abandon     3.99e- 5
##  5     1 abandoned   1.39e- 4
##  6     2 abandoned   5.88e- 5
##  7     1 abandoning 2.45e-33
##  8     2 abandoning 2.34e- 5
##  9     1 abbott      2.13e- 6
## 10     2 abbott      2.97e- 5
## # ... with 20,936 more rows
```

Interpretting topic models

Fitted LDA models have two important parameters of interest:

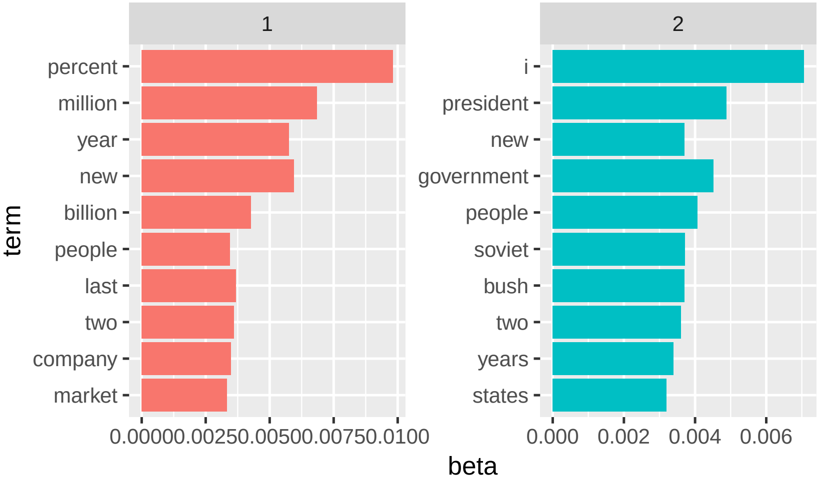
- **beta** gives the *"word-topic"* probabilities; they are calculated for each word-topic combination, and give the relative importance of each word for that topic
- **gamma** gives the *"document-topic"* probabilities; they are calculated for each document-topic combination, and give the probability that a document belongs to a certain topic

Plot the most important terms for each topic

```
ap_top_terms <- ap_topics %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

ap_top_terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```

Finance vs. politics?



Classify Jane Austen novels with topic modeling

Suppose we shuffled the individual chapters from all of Jane Austen's six novels together. Could we figure out which novel each chapter came from?

```
austen_chapters <- tidy_austen %>%
  unite(document, book, chapter) %>%
  anti_join(stop_words) %>%
  count(document, word, sort=TRUE) %>%
  ungroup()
```

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

```r
austen_chapters
```

```
## # A tibble: 145,200 x 3
##    document            word         n
##    <chr>               <chr>    <int>
##  1 Persuasion_21       elliot      62
##  2 Emma_47             harriet     52
##  3 Emma_21             miss        44
##  4 Persuasion_12       anne        43
##  5 Persuasion_12       captain     42
##  6 Persuasion_11       captain     41
##  7 Mansfield Park_19 sir         40
##  8 Persuasion_21       smith       38
##  9 Mansfield Park_35 fanny       37
## 10 Persuasion_22       anne        37
## # ... with 145,190 more rows
```

Convert to DocumentTermMatrix

We can use the cast_dtm() function to convert our tidy text data frame to the document-term matrix format.

```
austen_dtm <- austen_chapters %>% cast_dtm(document, word, n)
austen_dtm
```

```
## <<DocumentTermMatrix (documents: 275, terms: 13914)>>
## Non-/sparse entries: 145200/3681150
## Sparsity           : 96%
## Maximal term length: 19
## Weighting          : term frequency (tf)
```

Fit a topic model to Jane Austen's chapters

Now we fit a topic model to the Jane Austen chapters, attempting to categorize the data into 6 topics.

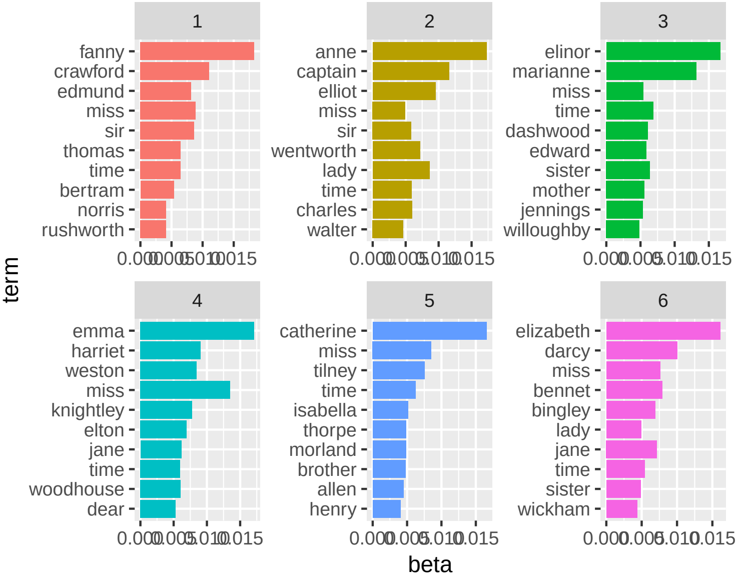
```
library(topicmodels)
austen_lda <- LDA(austen_dtm, k=6, control=list(seed=5678))
```

And we find the most important terms for each topic:

```
top_terms <- austen_lda %>%
  tidy() %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)
```

Plot the top terms associated with each topic

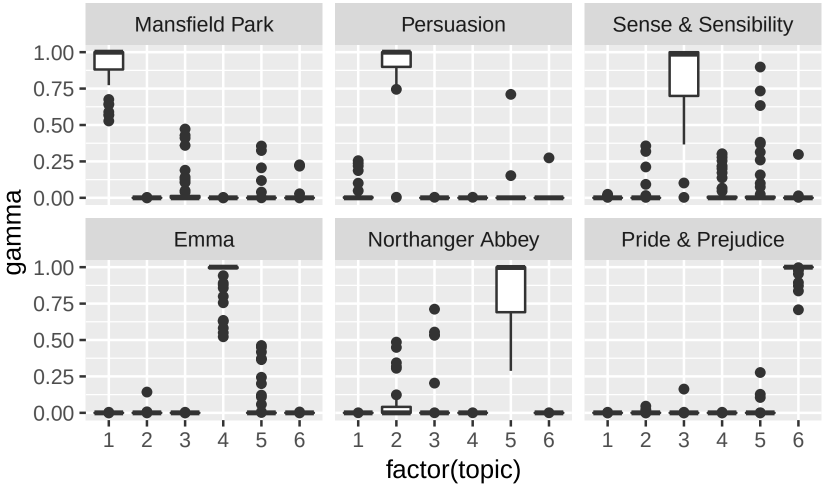
```
top_terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~topic, scales = "free") +
  coord_flip()
```


1
fanny
crawford
edmund
miss
sir
thomas
time
bertram
norris
rushworth
2
anne
captain
elliot
miss
sir
wentworth
lady
time
charles
walter
3
elinor
marianne
miss
time
dashwood
edward
sister
mother
jennings
willoughby
4
emma
harriet
weston
miss
knightley
elton
jane
time
woodhouse
dear
5
catherine
miss
tilney
time
isabella
thorpe
morland
brother
allen
henry
6
elizabeth
darcy
miss
bennet
bingley
lady
jane
time
sister
wickham
0.000
0.005
0.010
0.015
term
beta

How often were chapters grouped with their novels?

```r
chapters_gamma <- austen_lda %>%
  tidy(matrix = "gamma")

chapters_gamma %>%
  separate(document, c("book", "chapter"), sep="_") %>%
  mutate(book = reorder(book, gamma * topic)) %>%
  ggplot(aes(factor(topic), gamma)) +
  geom_boxplot() +
  facet_wrap(~book)
```


Mansfield Park
Persuasion
Sense & Sensibility
Emma
Northanger Abbey
Pride & Prejudice
gamma
1.00
0.75
0.50
0.25
0.00
1 2 3 4 5 6
factor(topic)