Lab 8: Text Mining and NLP

Alex Ptacek

Overview

In this lab, we will use tidy text techniques to analyze a dataset of amazon reviews. Each problem utilizes the tidy text mining techniques described in either chapter 2 (Problem 1), chapter 3 (Problem 2), or chapter 4 (Problem 3) of the tidy text mining with r textbook. Note: the dataset for this assignment is a bit bigger than what we have typically worked with in the class. On my computer everything worked fast enough, but if your computer is older and you find the computations intolerably slow you may reduce the size of the dataset by 90% by taking only the first 10% of reviews. If you do this make sure it is clearly stated. I have also listed a second shorter version of the file.

Problem 1: Sentiment and Review Score

- Download "simple_reviews.json" from the following google drive link: https://drive.google.com/drive/folders/1bk_2ihR5gQ8k6Tkn0NNpB58K1efruKnX?usp=sharing, read it into R, and rectangle it so that it is a dataframe where each row contains a single amazon review. If this file is too large, click here to download a shorter version where 90% of the reviews have been dropped].

Load Packages

```r
library(tidyverse)
library(jsonlite)
library(tidytext)
```

```r
reviews_json <- tibble(json = read_json("~/Downloads/simple_reviews.json"))

reviews <- reviews_json |>
  unnest_wider(json)
```

- To make sentiment analysis possible, add an index variable to the review data frame so that each review is uniquely identified by an integer. Then tokenize the review data frame using words as the tokens, and remove all stop words from the data set.

```r
data(stop_words)

reviews <- reviews |>
  mutate(index = row_number(), .before = 1) |>
  unnest_tokens(word, reviewText) |>
  anti_join(stop_words)
```

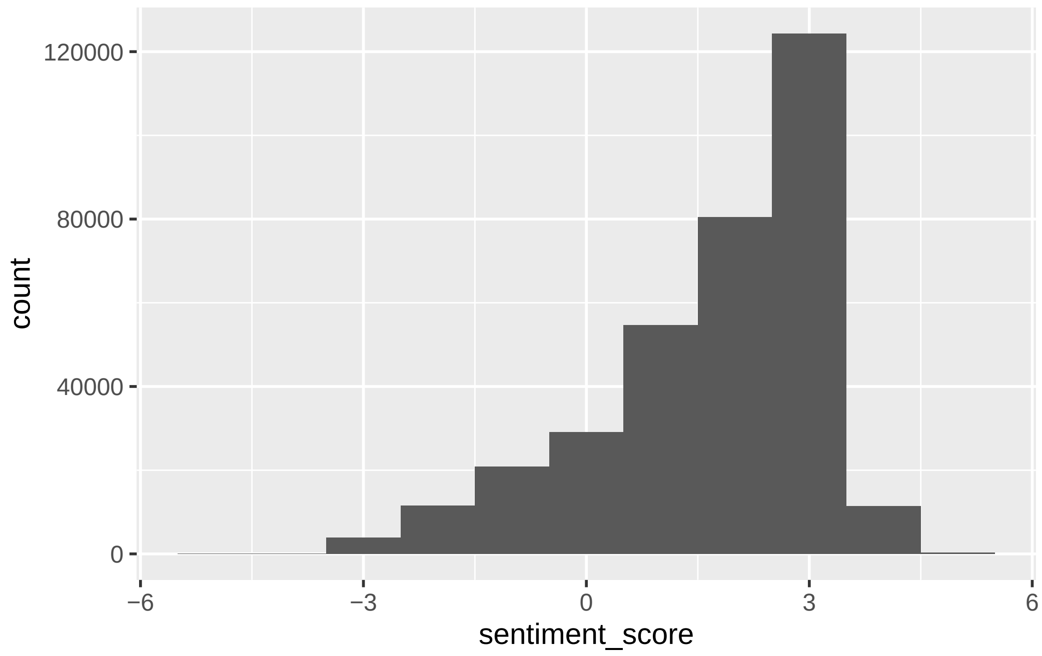
- Does sentiment correlate with reviews? Use the `afinn` lexicon to calculate a sentiment score for each review, normalizing by the number of lexicon words in each review. Visualize the distribution of sentiment scores for each rating and calculate the mean sentiment score for each review category. What do you observe?

Answer: Based on the distribution of `sentiment_score` by review, most reviews seem to be at least slightly positive. Based on the average `sentiment_score` by `overall` (rating), `sentiment_score` seems to correlate positively with the 1-5 rating. As rating score increases, `sentiment_score` tends to increase.

```r
afinn <- get_sentiments("afinn")

review_sentiment_scores <- reviews |>
  inner_join(afinn) |>
  group_by(index) |>
  summarise(n = n(),
            sentiment_score = mean(value))

review_sentiment_scores |>
  ggplot(aes(x = sentiment_score)) +
    geom_histogram(binwidth = 1)
```



```
reviews |>
  inner_join(review_sentiment_scores) |>
  group_by(overall) |>
  summarise(n = n(),
            sentiment_score = mean(sentiment_score))
```

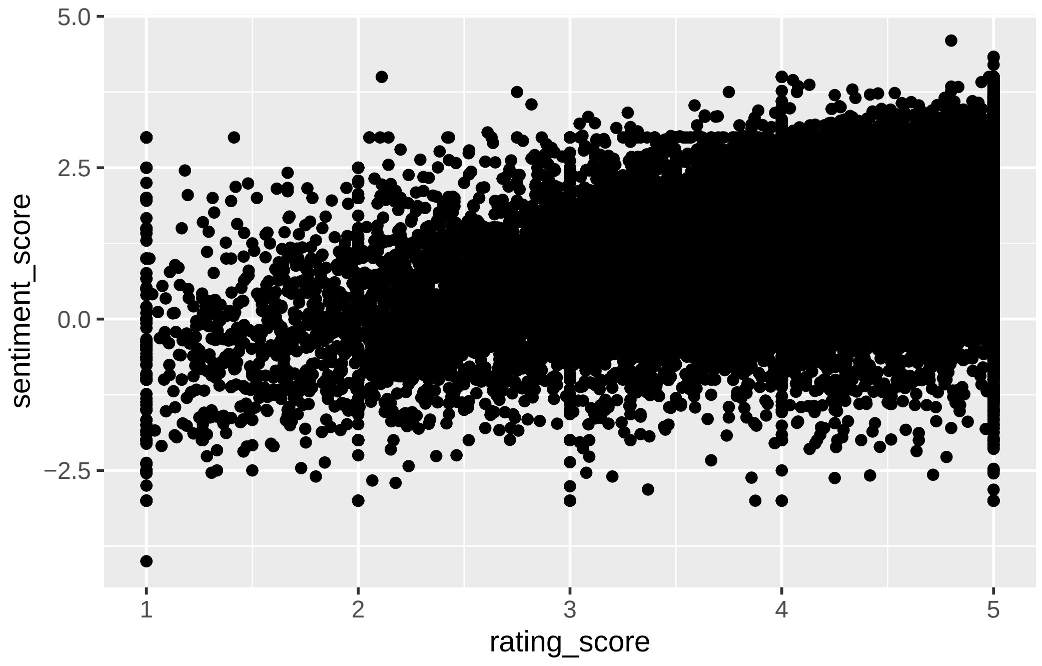
```
# A tibble: 5 x 3
  overall       n sentiment_score
    <int>   <int>           <dbl>
1       1  215570         -0.357
2       2  188861          0.0941
3       3  369373          0.456
4       4  758861          1.01
5       5 3285608          1.58
```

- Reviewer Personalities: For each reviewer, compute the number of ratings, the mean sentiment, and the mean review score. Filter for reviewers who have written more than 10 reviews, and plot the relationship between mean rating and mean sentiment. What do you observe?

Answer: The relationship between `rating` and `sentiment_score` by reviewer seems slightly positive. As average `rating` increases, average `sentiment_score` increases slightly, as well.

However, there is a significant ceiling and floor due to the finite 1-5 rating range, which is increasing the uncertainty of any visible correlation.

```
reviews |>
  inner_join(review_sentiment_scores) |>
  group_by(reviewerID) |>
  summarise(n = n(),
            sentiment_score = mean(sentiment_score),
            rating_score = mean(overall)) |>
  filter(n > 10) |>
  ggplot(aes(x = rating_score, y = sentiment_score)) +
  geom_point()
```



Problem 2: Words with high relative frequency

- As your starting point, take the tokenized data frame that has been filtered to remove stop words, but hasn't been joined with the sentiment lexicon data. For each item (`asin`), use the `bind_tf_idf` function to find the word that occurs in the reviews of that item with the highest frequency relative to the frequency of words in the entire review test dataset.

```r
reviews_tf_idf <- reviews |>
  group_by(asin, word) |>
  summarise(n = n()) |>
  bind_tf_idf(word, asin, n) |>
  arrange(desc(tf_idf))
```

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

- Select five items from (either at random or by hand) and look up the `asin` code on Amazon.com. In each of these cases, does the highest relative frequency word correspond to the identify or type of the item that you chose? You may not be able to find every single item, but I was able to find a solid majority of the ones I searched for by searching amazon.com for the `asin`.

Answer: I searched about 30 `asins` *and was only able to get one result, so I think most of these products have been taken off Amazon. I found one where the highest tf_idf was for "oven", but the product is paint, so they are not related.*

Problem 3: Bigrams and Sentiment

- Consider the two negative words `not` and `don't`. Starting from the original dataset, tokenize the data into bigrams. Then calculate the frequency of bigrams that start with either `not` or `don't`. What are the 10 most common words occurring after `not` and after `don't`? What are their sentiment values according to the `afinn` lexicon?

```r
n_gram_reviews <- reviews_json |>
  unnest_wider(json) |>
  mutate(index = row_number(), .before = 1) |>
  unnest_tokens(bigram, reviewText, token = "ngrams", n = 2) |>
  separate(bigram, c("word1", "word2"), sep = " ") |>
  filter(!word1 %in% stop_words$word) |>
  filter(!word2 %in% stop_words$word)


bigram_negatives <- n_gram_reviews |>
  filter(word1 == "not" |
         word1 == "don't" |
         word1 == "dont") |>
  count(word1, word2, sort = TRUE)

bigram_negatives |>
```

```
  head(10) |>
  inner_join(afinn, by = join_by(word2 == word))
```

```
# A tibble: 5 x 4
  word1 word2      n value
  <chr> <chr>  <int> <dbl>
1 dont  waste     24    -1
2 dont  pay        5    -1
3 dont  forget     4    -1
4 dont  bother     3    -2
5 dont  cut        2    -1
```

- Pick the most commonly occurring bigram where the first word is `not` or `don't` and the `afinn` sentiment of the second word is 2 or greater. Compute the mean rating of the reviews containing this bigram. How do they compare the average review score over the entire dataset?

Answer: The mean rating of reviews containing the bigram "dont recommend" (3) is 1.4 points lower on average than the mean rating of all reviews in the dataset (4.4).

```
bigram_negatives |>
  inner_join(afinn, by = join_by(word2 == word)) |>
  filter(value >= 2)
```

```
# A tibble: 3 x 4
  word1 word2         n value
  <chr> <chr>     <int> <dbl>
1 dont  recommend     2     2
2 dont  worth         2     2
3 dont  clean         1     2
```

```
n_gram_reviews |>
  filter(word1 == "dont", word2 == "recommend") |>
  group_by(word1, word2) |>
  summarise(mean_rating = mean(overall))
```

```
`summarise()` has grouped output by 'word1'. You can override using the
`.groups` argument.
```

```
# A tibble: 1 x 3
# Groups:   word1 [1]
  word1 word2     mean_rating
  <chr> <chr>           <dbl>
1 dont  recommend           3
```

```r
reviews |>
  summarise(mean_rating = mean(overall))
```

```
# A tibble: 1 x 1
  mean_rating
        <dbl>
1        4.39
```