Sentiment Analysis with R

Lucas Varela

2024-02-01

Sentiment Analysis with R

The goal is to create a model which allow us to categorize words based on their emotions, that is weather they are positive or negative, while at the same time measuring the magnitude of each one.

Analyzing sentiments involves extracting diverse opinions, categorized by polarities such as positive, negative, or neutral. This process, also referred to as opinion mining and polarity detection, discerns the nature of opinions within documents, websites, social media feeds, etc. It's a classification method that sorts data into various classes, which can be binary (positive or negative) or multi-class (happy, sad, angry, etc.).

Model

In this project, we'll conduct sentiment analysis using R. The dataset, sourced from the 'janeaustenR' package, will serve as our foundation. To facilitate sentiment analysis, we'll leverage the 'tidytext' package, which includes sentiment lexicons found in the 'sentiments' dataset.

```r
glimpse(sentiments)
```

```
## Rows: 6,786
## Columns: 2
## $ word      <chr> "2-faces", "abnormal", "abolish", "abominable", "abominably"~
## $ sentiment <chr> "negative", "negative", "negative", "negative", "negative", ~
```

We will use three general lexicons:

- bing
- loughran
- AFINN

These three lexicons utilize unigrams, which are single items or words extracted from textual data in a type of n-gram model. The AFINN lexicon assigns scores to words on a scale from -5 to 5, with an increase in negativity indicating a negative sentiment and an increase in positivity corresponding to a positive sentiment. In contrast, the bing lexicon classifies sentiment into a binary category of negative or positive. The Loughran model specializes in analyzing shareholder reports.

For our project, we'll employ the bing lexicons to extract sentiments from our data, and we can retrieve these lexicons using the **`get_sentiments()`** function:

```
get_sentiments("loughran")
```

Analysis with the Inner Join

The 'janeaustenr' package furnishes us with textual data from books written by Jane Austen. Leveraging 'tidytext,' we can conduct effective text analysis on our data. To structure the text of our books systematically, we'll employ the `unnest_tokens()` function to convert it into a tidy format.

```
tidy_data <- 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)
```

After tidying our text, with each row representing a single word, we'll utilize the "bing" lexicon. Implementing the `filter()` function, we'll isolate the words associated with joy from the book "Sense and Sensibility" to construct our sentiment analysis model.

```
positive_senti <- get_sentiments("bing") %>%
 filter(sentiment == "positive")

tidy_data %>%
 filter(book == "Emma") %>%
 semi_join(positive_senti) %>%
 count(word, sort = TRUE)
```

```
## Joining with ‘by = join_by(word)‘
```

```
## # A tibble: 668 x 2
##    word         n
##    <chr>    <int>
##  1 well       401
##  2 good       359
##  3 great      264
##  4 like       200
##  5 better     173
##  6 enough     129
##  7 happy      125
##  8 love       117
##  9 pleasure   115
## 10 right       92
## # i 658 more rows
```

After examining our previous outcome, which revealed numerous positive words such as "well" "good" and "great", we'll proceed to organize our data further.

Employing the `spread()` function, we'll categorize our data into distinct columns for positive and negative sentiments. Subsequently, using the `mutate()` function, we'll calculate the overall sentiment, representing the difference between positive and negative sentiment.

```r
bing <- get_sentiments("bing")
Emma_sentiment <- tidy_data %>%
  inner_join(bing, relationship = "many-to-many") %>%
  count(book = "Emma" , index = linenumber %/% 80, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)
```

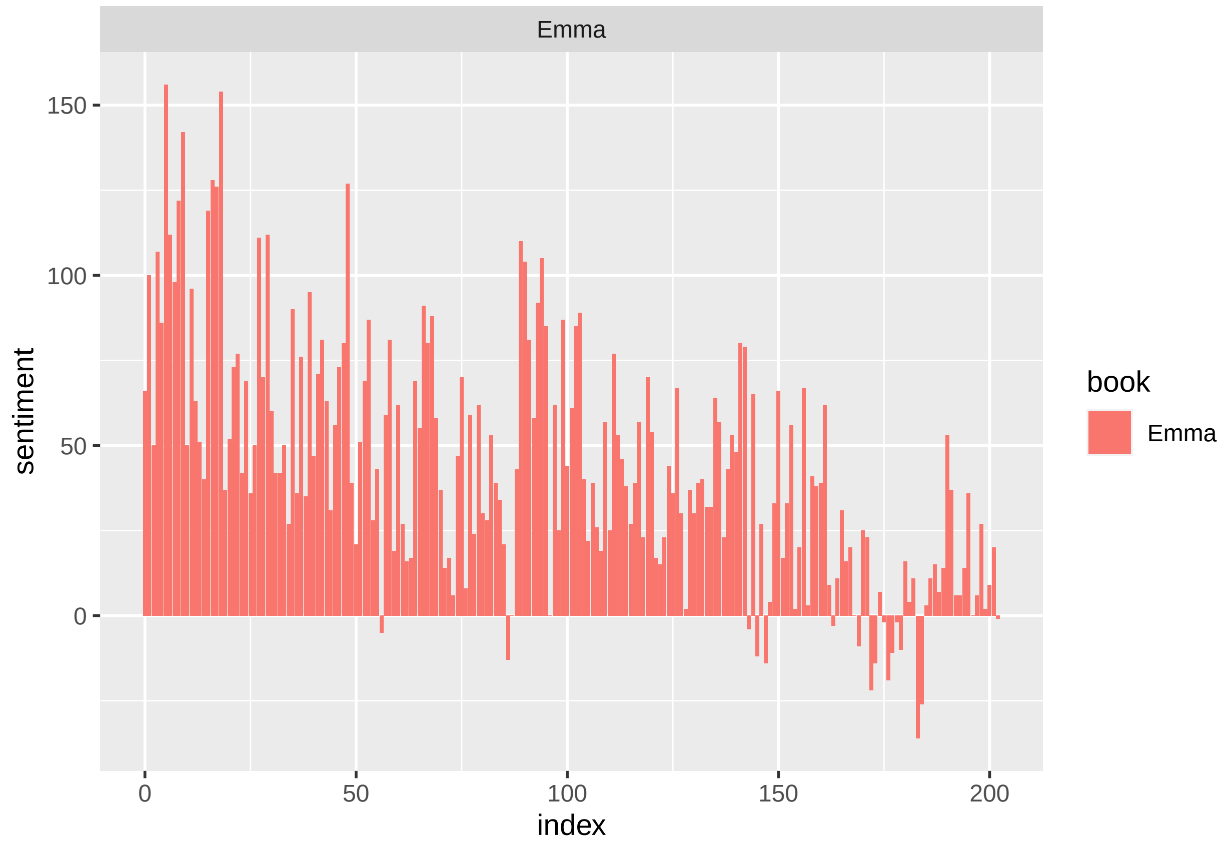
```
## Joining with ‘by = join_by(word)‘
```

```r
glimpse(Emma_sentiment)
```

```
## Rows: 203
## Columns: 5
## $ book      <chr> "Emma", "Emma", "Emma", "Emma", "Emma", "Emma", "Emma", "Emm~
## $ index     <dbl> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17~
## $ negative  <dbl> 130, 130, 135, 138, 141, 98, 120, 149, 119, 93, 138, 117, 14~
## $ positive  <dbl> 196, 230, 185, 245, 227, 254, 232, 247, 241, 235, 188, 213, ~
## $ sentiment <dbl> 66, 100, 50, 107, 86, 156, 112, 98, 122, 142, 50, 96, 63, 51~
```

Now we will visualize the words present in the book “Emma” based on their corresponding positive and negative scores:

```r
ggplot(Emma_sentiment, aes(index, sentiment, fill = book)) +
 geom_bar(stat = "identity", show.legend = TRUE) +
 facet_wrap(~book, ncol = 2, scales = "free_x")
```



To identify the most frequently occurring positive and negative words in the novel, we'll proceed with the counting process.

```
counting_words <- tidy_data %>%
 inner_join(bing) %>%
 count(word, sentiment, sort = TRUE)
```

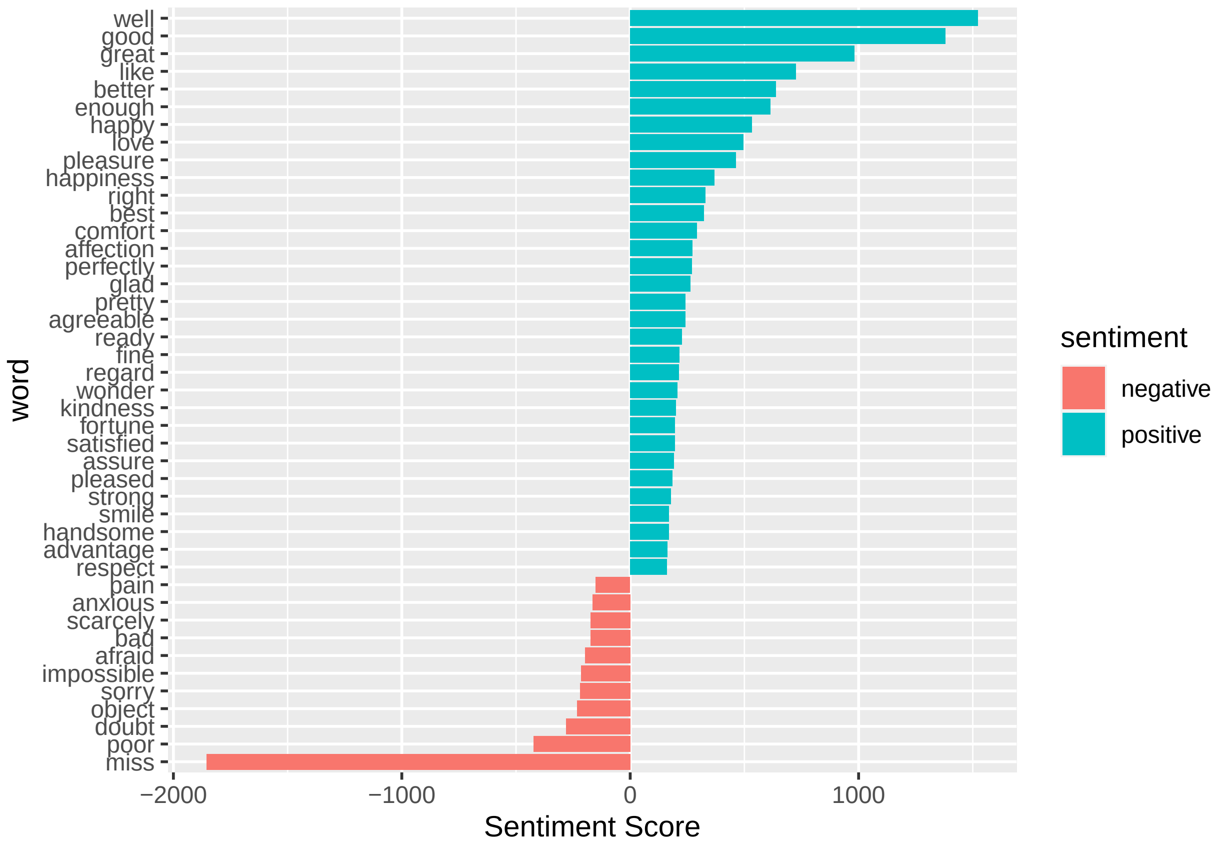
```
## Joining with `by = join_by(word)`
```

```
head(counting_words)
```

```
## # A tibble: 6 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
```

In the upcoming step, we'll visualize our sentiment scores by plotting them along an axis labeled with both positive and negative words. The ggplot() function will be employed to create a visual representation of our data based on their scores.

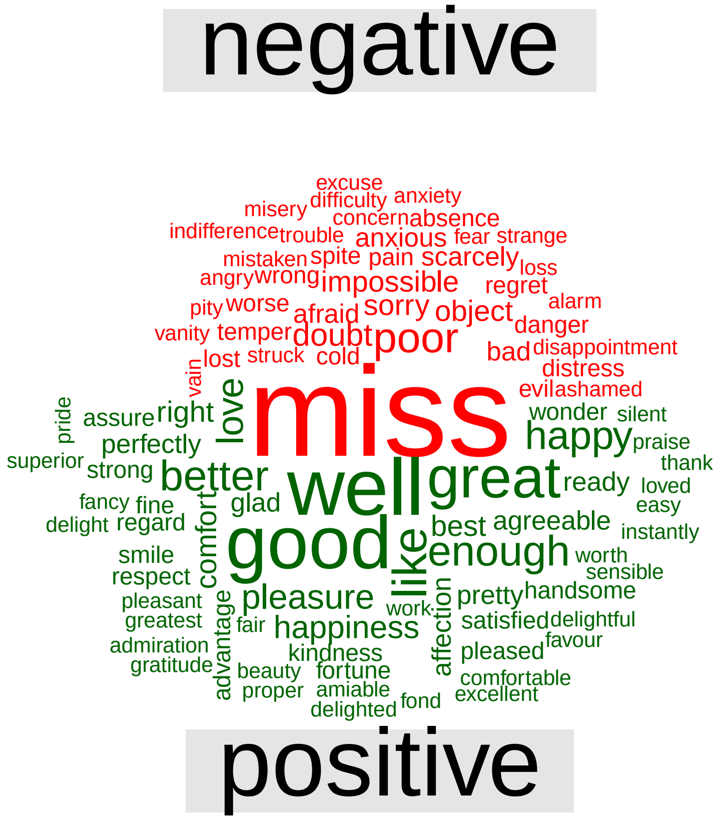
```
counting_words %>%
 filter(n > 150) %>%
 mutate(n = ifelse(sentiment == "negative", -n, n)) %>%
 mutate(word = reorder(word, n)) %>%
 ggplot(aes(word, n, fill = sentiment))+
 geom_col() +
 coord_flip() +
 labs(y = "Sentiment Score")
```



For the concluding visualization, we'll generate a word cloud to illustrate the most frequently occurring positive and negative words. Specifically, we'll utilize the `comparison.cloud()` function to plot both negative and positive words in a unified word cloud.

```r
tidy_data %>%
 inner_join(bing) %>%
 count(word, sentiment, sort = TRUE) %>%
 acast(word ~ sentiment, value.var = "n", fill = 0) %>%
 comparison.cloud(colors = c("red", "dark green"),
          max.words = 100)
```

```
## Joining with `by = join_by(word)`
```



The word cloud serves as an effective tool to visually comprehend the distinct groups of data categorized by their sentiments, both negative and positive. This allows us to gain insights into the different sentiments present in the data.

Conclusion

Tis sentiment analysis project successfully leveraged R and various lexicons to categorize words based on their emotional tones in Jane Austen's novel "Emma." The analysis involved the use of the 'tidytext' package, exploration of different sentiment lexicons like bing, loughran, and AFINN, and visualization techniques such as bar plots and word clouds. The results provided a nuanced understanding of sentiments, revealing the most frequently occurring positive and negative words. The visualizations, including bar plots and word clouds, offer a comprehensive view of sentiment distribution throughout the text, aiding in the interpretation of emotional tones within the novel.