Sentiment Analysis Project in R

LANGAT ERICK

R Project – Sentiment Analysis

The aim of this project is to build a sentiment analysis model which will allow us to categorize words based on their sentiments, that is whether they are positive, negative and also the magnitude of it. Before we start with our R project, let us understand sentiment analysis in detail.

What is Sentiment Analysis?

Sentiment Analysis is a process of extracting opinions that have different polarities. By polarities, we mean positive, negative or neutral. It is also known as opinion mining and polarity detection. With the help of sentiment analysis, you can find out the nature of opinion that is reflected in documents, websites, social media feed, etc. Sentiment Analysis is a type of classification where the data is classified into different classes. These classes can be binary in nature (positive or negative) or, they can have multiple classes (happy, sad, angry, etc.).

Developing our Sentiment Analysis Model in R

We will carry out sentiment analysis with R in this project. The dataset that we will use will be provided by the R package '**janeaustenR**'.

In order to build our project on sentiment analysis, we will make use of the tidytext package that comprises of sentiment lexicons that are present in the dataset of 'sentiments'.

```r
#load the libraries
library(bookdown)
suppressPackageStartupMessages(require(tidyverse))
library(tidytext)
#check sentiments
dim(sentiments)
```

```
## [1] 6786    2
```

```r
head(sentiments)
```

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

We will make use of three general purpose lexicons like –

- AFINN
- bing
- loughran

These three lexicons make use of the unigrams. Unigrams are a type of n-gram model that consists of a sequence of 1 item, that is, a word collected from a given textual data. In the AFINN lexicon model scores the words in a range from -5 to 5. The increase in negativity corresponds the negative sentiment whereas an increase in positivity corresponds the positive one. The bing lexicon model on the other hand, classifies the sentiment into a binary category of negative or positive. And finally, the loughran model that performs analysis of the shareholder's reports. In this project, we will make use of the bing lexicons to extract the sentiments out of our data. We can retrieve these lexicons using the get_sentiments() function. We can implement this as follows

```
get_sentiments("bing") %>% head()
```

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

Performing Sentiment Analysis with the Inner Join

In this step, we will import our libraries **'janeaustenr', 'stringr'** as well as **'tidytext'.** The janeaustenr package will provide us with the textual data in the form of books authored by the novelist Jane Austen. Tidytext will allow us to perform efficient text analysis on our data. We will convert the text of our books into a tidy format using **unnest_tokens**() function.

```
library(janeaustenr)
library(stringr)
library(tidytext)
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)
head(tidy_data)
```

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

```r
dim(tidy_data)
```

```
## [1] 725055      4
```

We have performed the tidy operation on our text such that each row contains a single word. We will now make use of the "bing" lexicon to and implement filter() over the words that correspond to joy. We will use the book Sense and Sensibility and derive its words to implement out sentiment analysis model.

```r
suppressPackageStartupMessages(library(dplyr))

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

```
## # A tibble: 6 x 2
##   word       sentiment
##   <chr>      <chr>
## 1 abound     positive
## 2 abounds    positive
## 3 abundance  positive
## 4 abundant   positive
## 5 accessable positive
## 6 accessible positive
```

```r
#filter
tidy_data %>%
 filter(book == "Emma") %>%
 semi_join(positive_senti) %>%
 count(word, sort = TRUE) %>% head(15) %>%  arrange(desc(n))
```

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

```
## # A tibble: 15 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
## 11 best        85
## 12 happiness    76
## 13 pretty       68
## 14 perfectly    67
## 15 ready        66
```

From the above result, we observe many positive words like "good", "happy", "love" etc. In the next step, we will use spread() function to segregate our data into separate columns of positive and negative sentiments. We will then use the mutate() function to calculate the total sentiment, that is, the difference between positive and negative sentiment.

```
suppressPackageStartupMessages(require(tidyr))

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

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

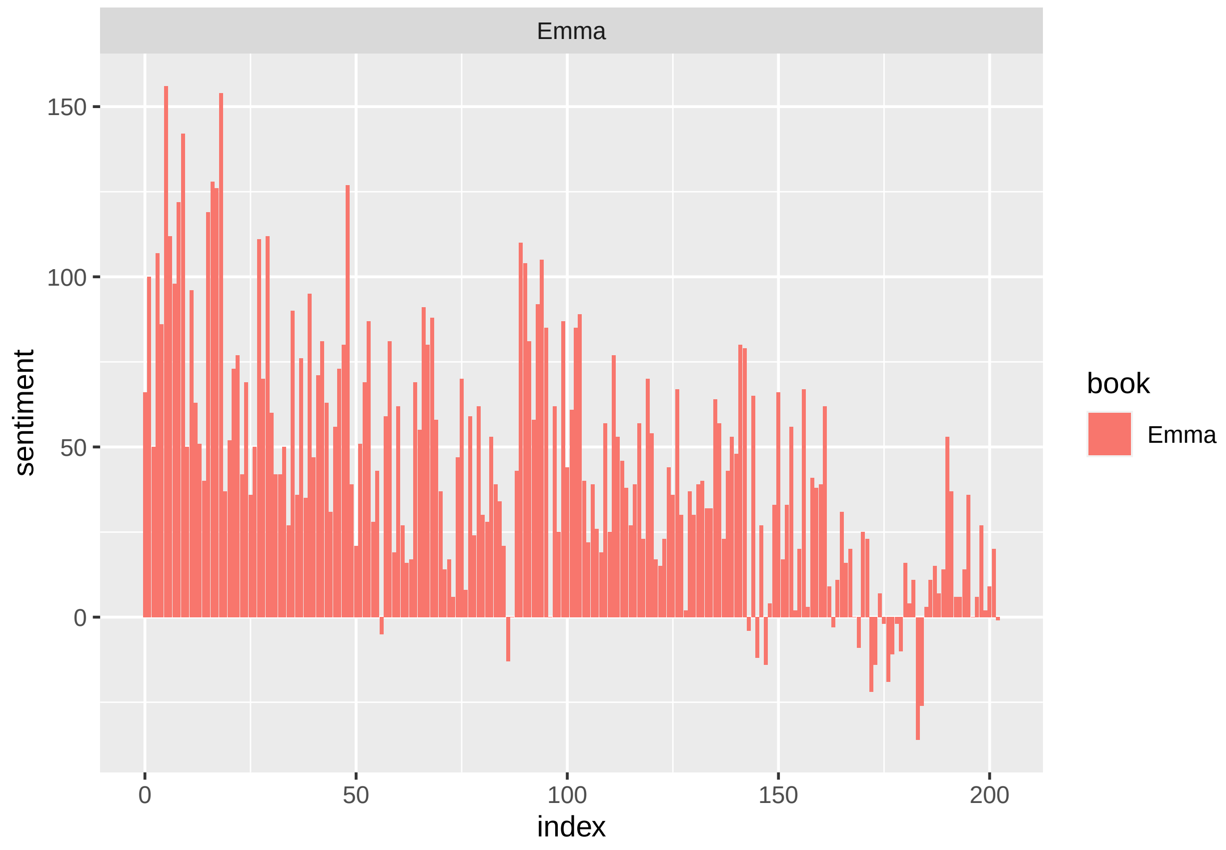
```
## Warning in inner_join(., bing): Detected an unexpected many-to-many relationship between `x` and `y`
## i Row 435434 of `x` matches multiple rows in `y`.
## i Row 5051 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
head(Emma_sentiment)
```

```
## # A tibble: 6 x 5
##   book  index negative positive sentiment
##   <chr> <dbl>    <dbl>    <dbl>     <dbl>
## 1 Emma      0      130      196        66
## 2 Emma      1      130      230       100
## 3 Emma      2      135      185        50
## 4 Emma      3      138      245       107
## 5 Emma      4      141      227        86
## 6 Emma      5       98      254       156
```

In the next step, we will visualize the words present in the book "Emma" based on their corrosponding positive and negative scores.

```
library(ggplot2)

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



Let us now proceed towards counting the most common positive and negative words that are present in the novel.

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

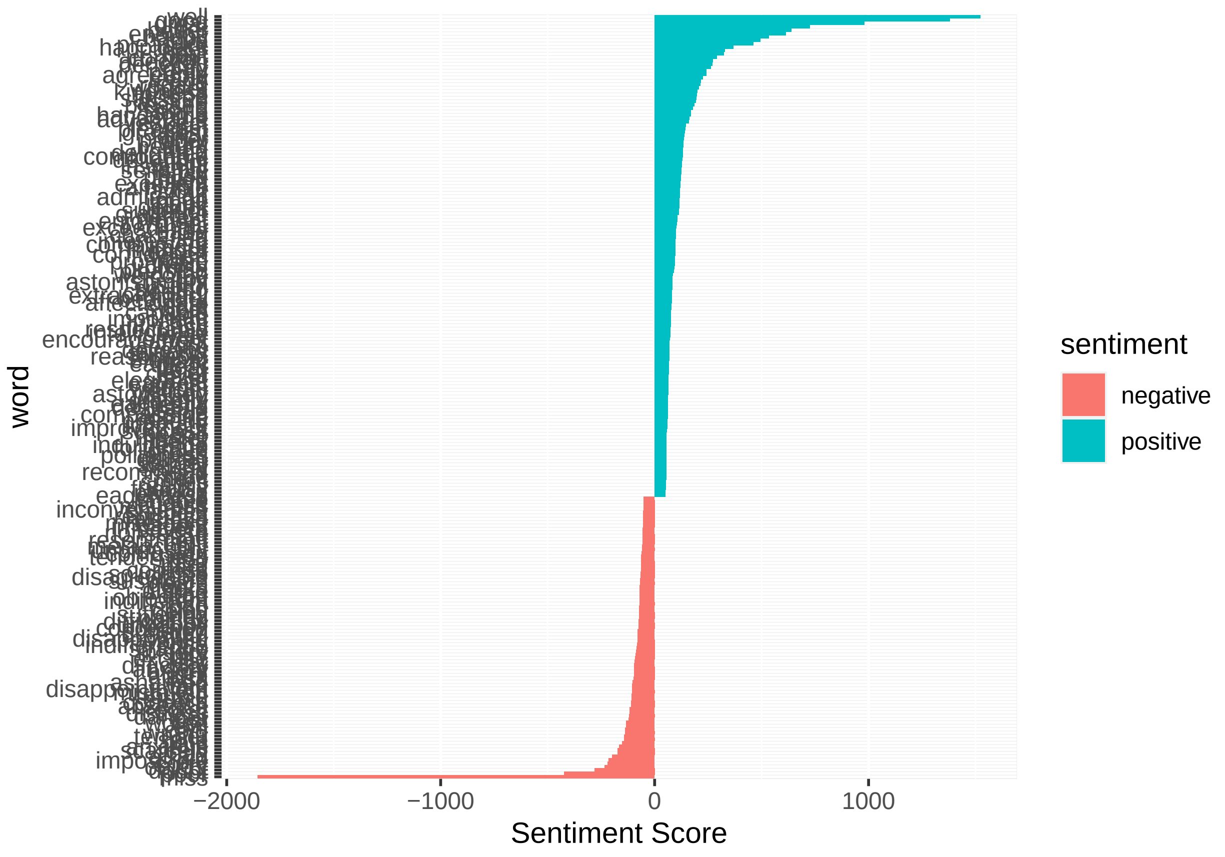
```
## Joining with ‘by = join_by(word)‘

## Warning in inner_join(., bing): Detected an unexpected many-to-many relationship between ‘x‘ and ‘y‘
## i Row 435434 of ‘x‘ matches multiple rows in ‘y‘.
## i Row 5051 of ‘y‘ matches multiple rows in ‘x‘.
## i If a many-to-many relationship is expected, set ‘relationship =
##   "many-to-many"‘ to silence this warning.
```

```
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 next step, we will perform visualization of our sentiment score. We will plot the scores along the axis that is labeled with both positive as well as negative words. We will use ggplot() function to visualize our data based on their scores.

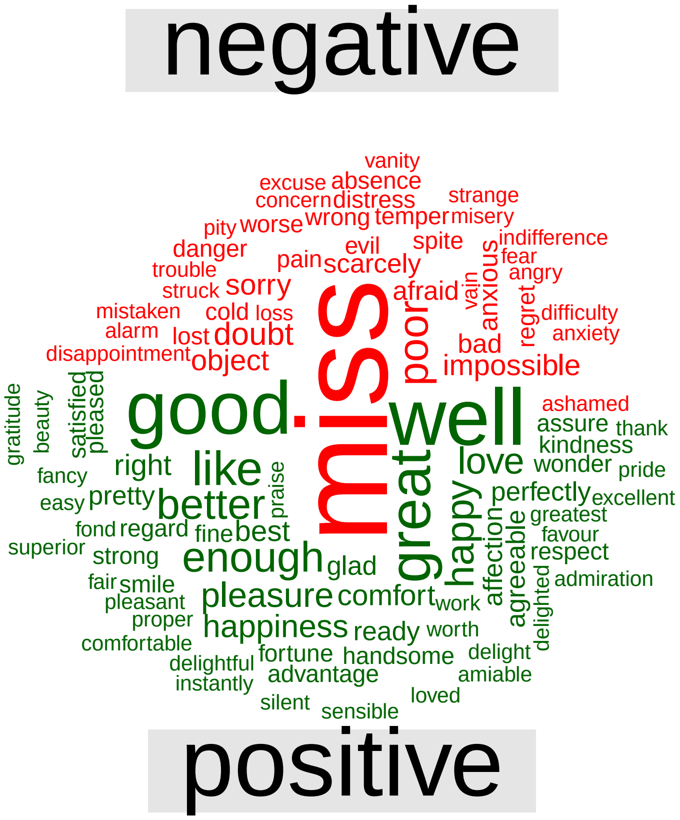
```
counting_words %>%  filter(n>50) %>%
          mutate(n=ifelse(sentiment=="negative", -n, n)) %>%
          mutate(word=reorder(word, n)) %>%
          ggplot(aes(word, n, fill= sentiment))+
          geom_col()+
          coord_flip()+
          labs(y="Sentiment Score")
```



In the final visualization, let us create a wordcloud that will delineate the most recurring positive and negative words. In particular, we will use the comparision.cloud() function to plot both negative and positive words in a single wordcloud as follows:

```
#packages
suppressPackageStartupMessages(require(reshape2))
suppressPackageStartupMessages(require(wordcloud))

#plot
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)‘

## Warning in inner_join(., bing): Detected an unexpected many-to-many relationship between ‘x‘ and ‘y‘
## i Row 435434 of ‘x‘ matches multiple rows in ‘y‘.
## i Row 5051 of ‘y‘ matches multiple rows in ‘x‘.
## i If a many-to-many relationship is expected, set ‘relationship =
##   "many-to-many"‘ to silence this warning.
```



This word cloud will enable us to efficiently visualize the negative as well as positive groups of data. Therefore, we are now able to see the different groups of data based on their corresponding sentiments. You can experiment with several other datasets like tweets in order to gain a comprehensive insight into sentiment analysis.

Summary

In this blog, we went through our project of sentiment analysis in R. We learnt about the concept of sentiment analysis and implemented it over the dataset of Jane Austen's books. We were able to delineate it through various visualizations after we performed data wrangling on our data. We used a lexical analyzer – 'bing' in this instance of our project. Furthermore, we also represented the sentiment score through a plot and also made a visual report of wordcloud.