Project – Sentiment Analysis of Jane Austen's Books

we will develop our very own project of sentiment analysis using R. We will make use of the tiny text package to analyze the data and provide scores to the corresponding words that are present in the dataset.

Aim of the project:

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, we 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'.

First, we will import the needed packages, then we will go through data cleaning and data preprocessing steps. Pre-processing of text data involves; to help remove the noise from the data, converting the text data to all lower cases, eliminating all punctuations and eradicating stop words such as "the", "is" "and" and others. The preprocessing step is followed by tokenizing the text, which involves the decomposition of text into manageable words or terms on which sentiment can be determined.

After data preparation, sentiment lexicons will be adopted so as to score the words respectively. Lexicons are the word list that have been preprocessed and each word has been assigned sentiment polarity scores for it. These scores aid in establishing the degree of positivity or negativity of a piece of text. For instance, happy, joy will be positive while sad, anger will be negative. The addition of these scores makes it possible to measure the polarity of a rather large amount of text such as a paragraph and other documents.

```r
#install.packages('tidytext')
library(janeaustenr)
library(stringr)
library(tidytext)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
```

```r
sentiments
```

```
## # 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
## # i 6,776 more rows
```

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 –

```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
## # i 6,776 more rows
```

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.

```r
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)
```

As we have performed the tidy operation on our text such that each row contains a single word. Now we will 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.

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

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

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

```
## # A tibble: 593 x 2
##    word          n
##    <chr>     <int>
##  1 well        240
##  2 good        177
##  3 great       149
##  4 enough      103
##  5 happy       100
##  6 like         83
##  7 affection    79
##  8 better       78
##  9 love         77
## 10 pleasure     67
## # i 583 more rows
```

From our 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.

```
SS_sentiment <- tidy_data %>%
  filter(book == "Sense & Sensibility") %>%  # Select only the book Sense & Sensibility
  inner_join(get_sentiments("bing"), by = "word") %>%  # Join with the Bing sentiment lexicon
  count(index = linenumber %/% 80, sentiment) %>%  # Count sentiments for chunks of 80 lines
  spread(sentiment, n, fill = 0) %>%  # Separate into positive and negative columns
  mutate(sentiment = positive - negative)  # Calculate the total sentiment

# View the resulting data
print(SS_sentiment)
```

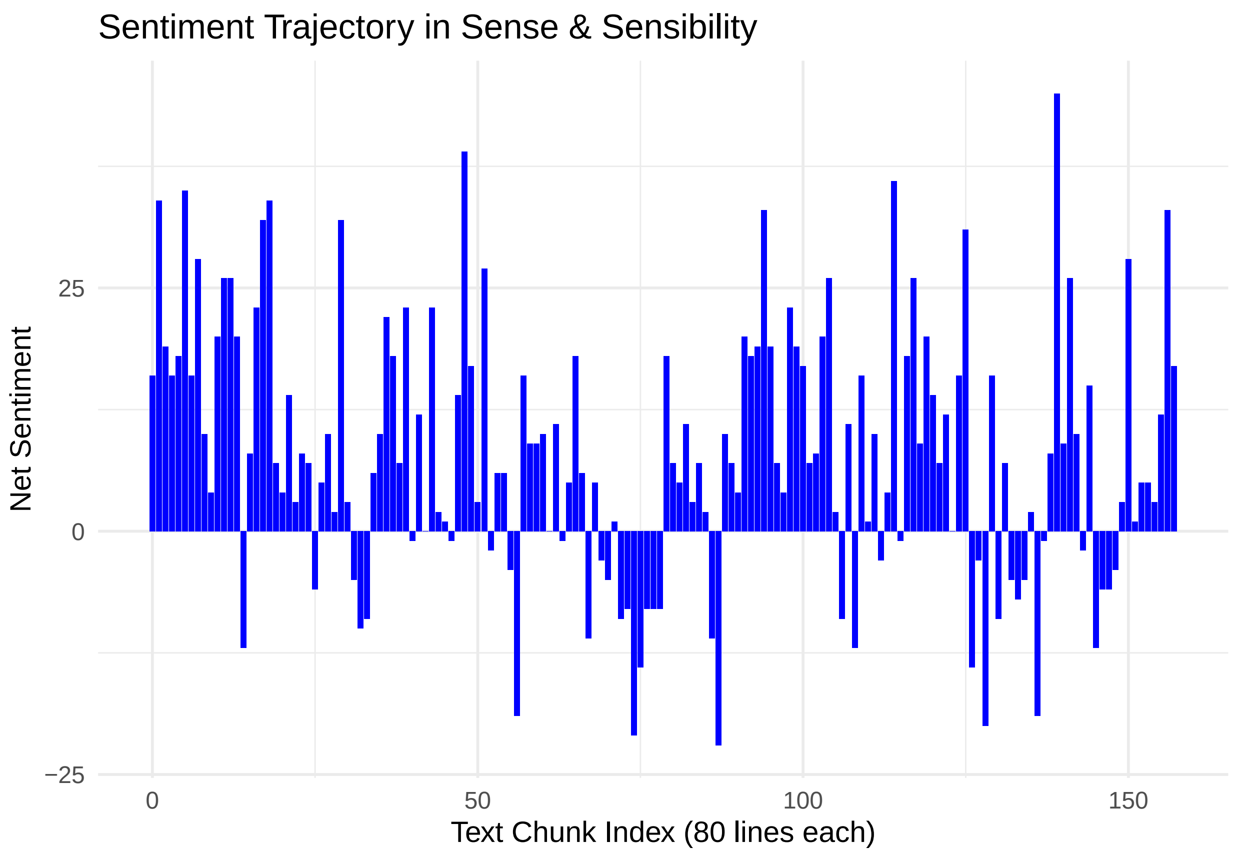
```
## # A tibble: 158 x 4
##    index negative positive sentiment
##    <dbl>    <dbl>    <dbl>     <dbl>
##  1     0       16       32        16
##  2     1       19       53        34
##  3     2       12       31        19
##  4     3       15       31        16
##  5     4       16       34        18
##   6     5        16        51         35
##   7     6        24        40         16
##   8     7        23        51         28
##   9     8        30        40         10
##  10     9        15        19          4
## # i 148 more rows
```

Visualization

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

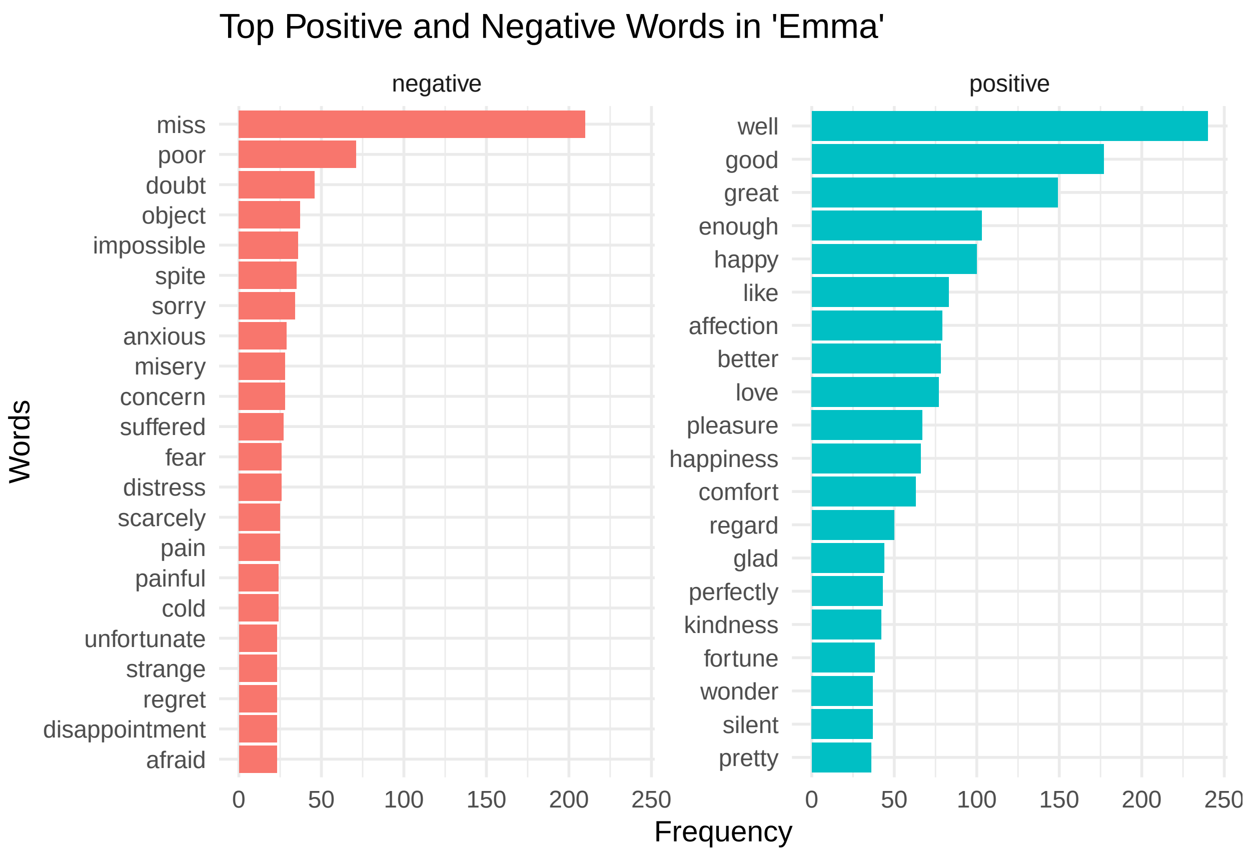
```r
library(ggplot2)

ggplot(SS_sentiment, aes(x = index, y = sentiment)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(
    title = "Sentiment Trajectory in Sense & Sensibility",
    x = "Text Chunk Index (80 lines each)",
    y = "Net Sentiment"
  ) +
  theme_minimal()
```



```r
# Prepare data for visualization
SS_words <- tidy_data %>%
  filter(book == "Sense & Sensibility") %>%  # Filter for Emma
  inner_join(get_sentiments("bing"), by = "word") %>%  # Match with Bing lexicon
  count(word, sentiment, sort = TRUE)  # Count occurrences by word and sentiment
```

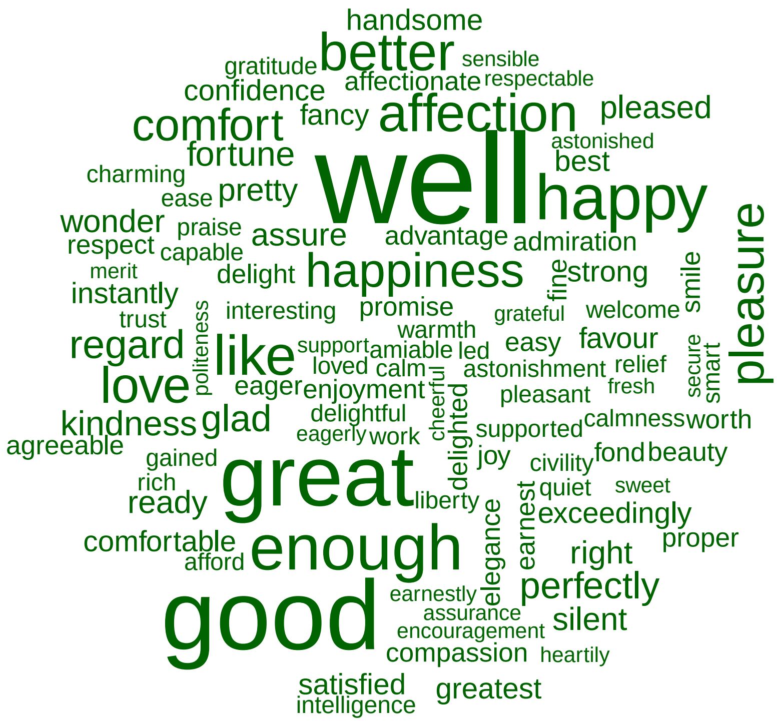
```
# Bar chart for top words contributing to sentiment
SS_words %>%
  group_by(sentiment) %>%
  top_n(20, n) %>%  # Select the top 10 words for each sentiment
  ungroup() %>%
  mutate(word = reorder_within(word, n, sentiment)) %>%  # Reorder for better visualization
  ggplot(aes(x = word, y = n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +  # Separate panels for positive and negative
  coord_flip() +
  scale_x_reordered() +  # Use reordered scale
  labs(
    title = "Top Positive and Negative Words in 'Emma'",
    x = "Words",
    y = "Frequency"
  ) +
  theme_minimal()
```



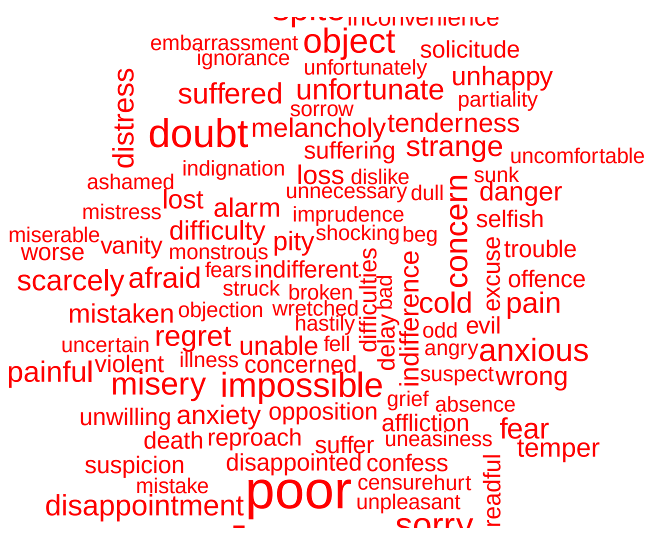
```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
# Separate positive and negative words
positive_words <- SS_words %>%
  filter(sentiment == "positive") %>%
  with(wordcloud(word, n, max.words = 100, colors = "darkgreen"))
```



```r
negative_words <- SS_words %>%
  filter(sentiment == "negative") %>%
  with(wordcloud(word, n, max.words = 100, colors = "red"))
```



Summary

In this project, we went through sentiment analysis in R. Then 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.

Hope you enjoyed this R Sentiment Analysis Project.