Sentiment Analysis in R

AUTHOR
Nicola Rennie

Sentiment analysis is a natural language processing technique used to analyse whether textual data is positive, negative, or neutral. A common application is the analysis of customer feedback. This tutorial serves as a brief introduction to performing sentiment analysis in R.

Data

The data used here is from Week 19 of #TidyTuesday. This data set contains data on books that have been on the New York Times Bestsellers List. Information is included on the title, author, the year of release, the number of weeks spent on the list, the first week on the list, debut rank, and best rank.

```
nyt_titles <- readr::read_csv("nyt_titles.csv")
head(nyt_titles)
```

```
# A tibble: 6 × 8
     id title           author  year total_weeks first_week debut_rank
best_rank
  <dbl> <chr>           <chr>  <dbl>       <dbl> <date>          <dbl>
<dbl>
1     0 "\"H\" IS FOR … Sue G…  1991          15 1991-05-05          1
2
2     1 "\"I\" IS FOR … Sue G…  1992          11 1992-04-26         14
2
3    10 "'G' IS FOR … Sue G…  1990           6 1990-05-06          4
8
4   100 "A DOG'S JOURN… W. Br…  2012           1 2012-05-27          3
14
5  1000 "CHANGING FACE… Kimbe…  2006           1 2006-02-19         11
14
```

```
6  1001 "CHAOS"          Patri…  2016           3 2016-12-04          1
7
```

For this analysis we limit ourselves to the 1,000 most popular titles, in terms of the total number of weeks spent on the list. To extract these 1,000 titles, we use `slice_max()` from {dplyr}. For the sentiment analysis here, we will analyse the sentiment of the titles. Therefore we extract only this column using `select()`.

```
library(dplyr)
text_data <- nyt_titles %>%
  slice_max(total_weeks, n = 1000, with_ties = FALSE) %>%
  select(title)
```

Sentiment analysis

In R, the main package for carrying out sentiment analysis is {tidytext}. There are two main data sets we need to download. First, we load the `stop_words` data included with {tidytext}. This data set contains a list of stop words. Stop words are commonly used words such as “the”, “is” or “and”. We will later use the `stop_words` data to eliminate these unimportant words.

Secondly, we obtain a list of sentiments using the `get_sentiments()` function. In this example, we use the *afinn* data. This data gives a list of known words, and their sentiment score. For the afinn data, the sentiment is given as a value between -5 and 5, where -5 is very negative, 0 is neutral, and 5 is very positive.

```
library(tidytext)
data(stop_words)
afinn_df <- get_sentiments("afinn")
head(afinn_df)
```

```
# A tibble: 6 × 2
  word       value
  <chr>      <dbl>
```

```
1 abandon       -2
2 abandoned     -2
3 abandons      -2
4 abducted      -2
5 abduction     -2
6 abductions    -2
```

We use `unnest_tokens()` to process the titles data. This function splits each string into separate words, converts them to lowercase, and flattens the data such that each row represents a separate word. We then use `anti_join()` to remove the stop words.

```
text_data <- text_data %>%
  unnest_tokens(word, title) %>%
  anti_join(stop_words, by = "word")
```

Now, we can join the words in the titles to the words in the sentiments data set. Note that not all words in our list will have a sentiment score in the database, so we lose some data. We join the two data sets together using `inner_join()`.

```
sentiments <- text_data %>%
  inner_join(afinn_df, by = "word")
head(sentiments)
```

```
# A tibble: 6 × 2
  word    value
  <chr>   <dbl>
1 consent     2
2 heaven      2
3 kill       -3
4 ugly       -3
5 lovely      3
6 war        -2
```

We can use more functions from {dplyr} to extract the sentiment value column, an calculate the mean sentiment across all titles. The average sentiment is 0.14 - overall slightly positive.

```
sentiments %>%
  pull(value) %>%
  mean()
```

```
[1] 0.1355932
```

Word clouds

Word clouds are commonly used to visualise the sentiment, and number of instances of words in text. First, we can calculate the number of times each word is used in a title. We use `group_by()` to group together repetitions of the same word, and add the number of occurrences as a new column using `mutate()`.

```
counts <- text_data %>%
  group_by(word) %>%
  mutate(count = n())
```

We can then join the sentiments data set to the counts data set using `left_join()`, and we remove any duplicate rows where words are represented multiple times.

```
plot_data <- counts %>%
  left_join(sentiments, by = "word") %>%
  distinct()
```

To make our word cloud a bit nice looking, we filter out any words that only occur once or twice. This leaves us with 104 unique words. Unfortunately, most of these don't have an associated sentiment score. If we also use

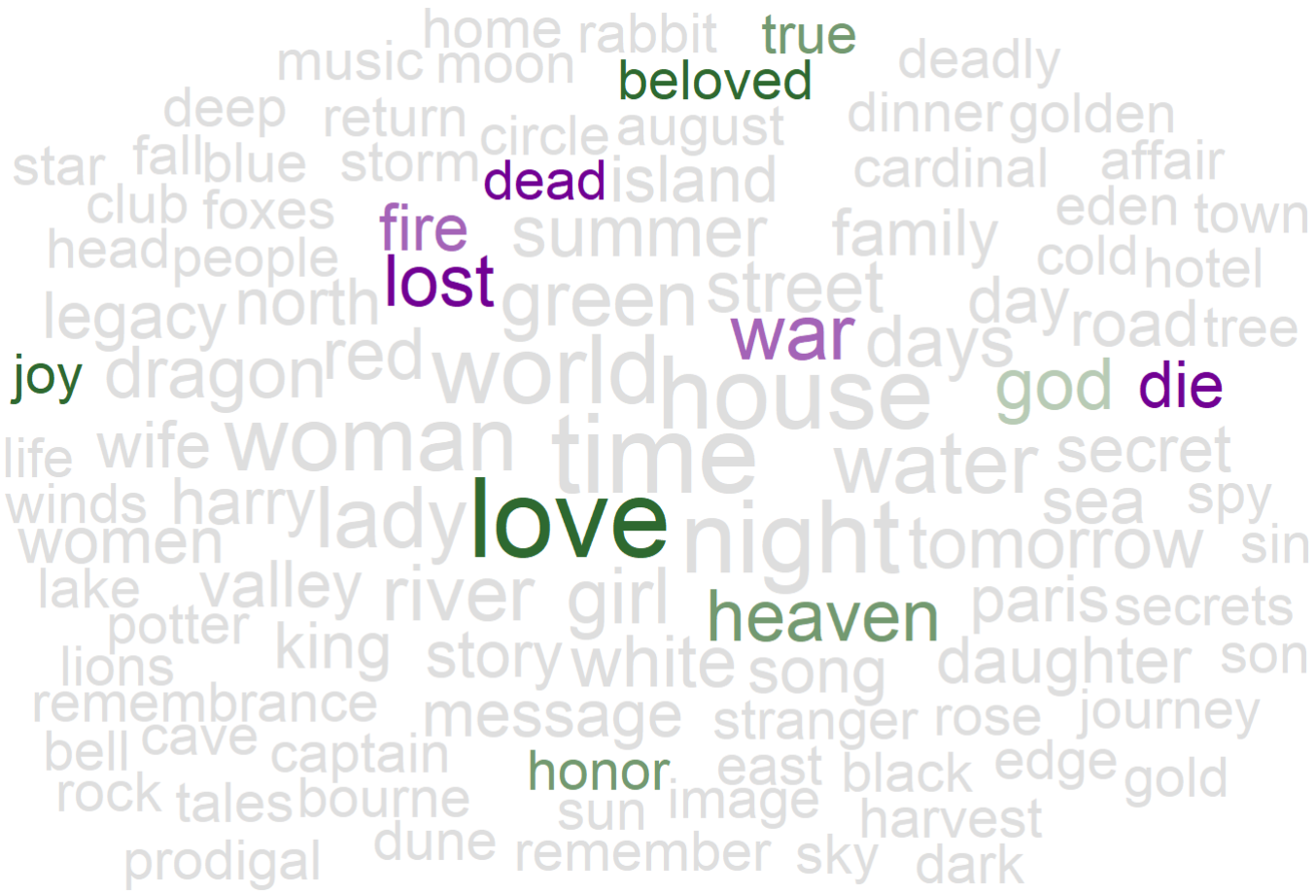
`arrange()` to sort our data from most to least common words, we can see that *time* and *love* are the most common words.

```
plot_data <- plot_data %>%
  filter(count > 2) %>%
  arrange(desc(count))
head(plot_data)
```

```
# A tibble: 6 × 3
# Groups:   word [6]
  word  count value
  <chr> <int> <dbl>
1 time     12    NA
2 love     12     3
3 house    10    NA
4 night    10    NA
5 world     9    NA
6 woman     8    NA
```

In R, {ggplot2} is the most common data visualisation package. Unfortunately it does not have built-in functionality to create word clouds. Therefore, we load the {ggwordcloud} package in addition. The `geom_text_wordcloud()` function is the key function that constructs the word cloud. We colour the words based on their sentiment score - green for positive words, and purple for negative words. Words that do not have a sentiment score are coloured light gray. The size of the words are given by how often they occur.

```
library(ggplot2)
library(ggwordcloud)
set.seed(42)
ggplot(plot_data, aes(label = word, size = count, colour = value)) +
  geom_text_wordcloud() +
  scale_size_area(max_size = 15) +
  scale_colour_gradient2(low = "#710193", high = "#2e6930", na.value =
  theme_void()
```



And that's it! This was a simple introduction to sentiment analysis in R. There are many more extensions to what's covered here. For this data set, I'd be really interested in looking at the changing sentiments of titles over time.