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. The data can be downloaded from here. You can either read it into R directly from the URL, or read in a locally saved copy. Here, I've saved a copy called `nyt_titles.tsv` in a directory one level up from my R script.

The data is saved as a tab-separated values (.tsv) file, and we can read it into R using the `read_tsv()` function from the {readr} package.

```
library(readr)
nyt_titles <- read_tsv("../nyt_titles.tsv")
```

id	title	author	year	total_weeks	first_week	debut_rank	best_rank
0	"H" IS FOR HOMICIDE	Sue Grafton	1991	15	1991-05-05	1	2
1	"I" IS FOR INNOCENT	Sue Grafton	1992	11	1992-04-26	14	2
10	''G'' IS FOR GUMSHOE	Sue Grafton	1990	6	1990-05-06	4	8
100	A DOG'S JOURNEY	W. Bruce Cameron	2012	1	2012-05-27	3	14
1000	CHANGING FACES	Kimberla Lawson Roby	2006	1	2006-02-19	11	14

id	title	author	year	total_weeks	first_week	debut_rank	best_rank
1001	CHAOS	Patricia Cornwell	2016	3	2016-12-04	1	7

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, the book's debut rank, and it's best rank.

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}. `slice_max()` lets you extract the `n` rows for the largest values of a particular variable. We set `with_ties = FALSE` to ensure we get exactly 1000 rows returned. For the 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 - 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")
```

word	value
abandon	-2
abandoned	-2
abandons	-2
abducted	-2
abduction	-2
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 - this removes any rows from the titles data for which there is a corresponding row in the `stop_words` data set.

```
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()` - this keeps only words which appear in both the titles data set and the afinn sentiments data set.

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

word	value
consent	2
heaven	2
kill	-3
ugly	-3
lovely	3

word	value
war	-2

To calculate the average sentiment across all titles, we extract the sentiment `value` column as a vector using `pull()`, and then calculate the mean. 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 sentiments, and number of instances of words in textual data. First, we can calculate the number of times each word is used in the titles. 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 using `distinct()`.

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

To make our word cloud a bit nicer 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

common words.

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

word	count	value
time	12	NA
love	12	3
house	10	NA
night	10	NA
world	9	NA
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. The limits of the colour scale are set to match the limits of the sentiment data - between -5 and 5. The `theme_void()` function removes elements such axes, gridlines, and ticks to make our wordcloud appear cleaner.

```
library(ggplot2)
library(ggwordcloud)
set.seed(42)
ggplot(data = plot_data,
       mapping = 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 = "#dedede",
                         limits = c(-5, 5)) +
  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.

You can download the data used in this analysis from here, and the R script is available on GitHub.