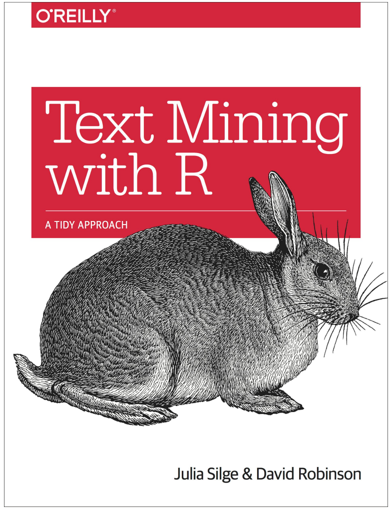
Introduction to Text Mining

Dr Andrew J. Stewart

E: drandrewjstewart@gmail.com
T: @ajstewart_lang
G: ajstewartlang

Text Mining in R

In any set of texts (such as books, interview transcripts etc.) it’s often useful to be able to quantify key aspects of the constituent parts (e.g., words, phrases). For example, some types of language may be more common in one interview transcript vs. another, and it can be useful to visualise the content of a particular text to compare it with others.



What we’ll cover in this introduction...

Summarising text data.

Sentiment analysis.

Extracting frequency information (and demonstrating Zipf's law).

Characterising text that makes a unique contribution to a particular instance.

N-gram analysis.

The Packages We’ll Be Using

We’ll use the `{tidyverse}` as we’ll need to do some data wrangling and visualisation. We’ll also use `{tidytext}` for working with text in a tidy format, and `{gutenbergr}` which allows us to connect to Project Gutenberg in order to download public domain texts.

```
library(tidyverse)
library(tidytext)
library(gutenbergr)
```

We'll use the texts of some books by HG Wells in our examples...

We are going to download from Project Gutenberg the text of four books by HG Wells. We will combine these four books into a dataframe called `books`

```
titles <- c("The War of the Worlds",
           "The Time Machine",
           "Twenty Thousand Leagues under the Sea",
           "The Invisible Man: A Grotesque Romance")

books <- gutenberg_works(title %in% titles) %>%
  gutenberg_download(meta_fields = "title")
```

```
str(books)
tibble [27,540 × 3] (S3: tbl_df/tbl/data.frame)
 $ gutenberg_id: int [1:27540] 35 35 35 35 35 35 35 35 35 35 ...
 $ text        : chr [1:27540] "The Time Machine" "" "An Invention" "" ...
 $ title       : chr [1:27540] "The Time Machine" "The Time Machine" "The Time Machine" "The Time
Machine" ...
```

```
head(books, n = 8)
# A tibble: 15 x 3
   gutenberg_id text                                    title
          <int> <chr>                                    <chr>
 1           35 "The Time Machine"                      The Time Machine
 2           35 ""                                      The Time Machine
 3           35 "An Invention"                          The Time Machine
 4           35 ""                                      The Time Machine
 5           35 "by H. G. Wells"                        The Time Machine
 6           35 ""                                      The Time Machine
 7           35 ""                                      The Time Machine
 8           35 "CONTENTS"                              The Time Machine
```

```
books %>% distinct(title)
# A tibble: 4 x 1
  title
  <chr>
1 The Time Machine
2 The War of the Worlds
3 Twenty Thousand Leagues under the Sea
4 The Invisible Man: A Grotesque Romance
```

Examining rows 31:40 of the `text` column of our `books` tibble:

```
books$text[31:40]
 [1] " I."
 [2] " Introduction"
 [3] ""
 [4] ""
 [5] "The Time Traveller (for so it will be convenient to speak of him) was"
 [6] "expounding a recondite matter to us. His pale grey eyes shone and"
 [7] "twinkled, and his usually pale face was flushed and animated. The fire"
 [8] "burnt brightly, and the soft radiance of the incandescent lights in the"
 [9] "lilies of silver caught the bubbles that flashed and passed in our"
[10] "glasses. Our chairs, being his patents, embraced and caressed us rather"
```

Currently the text is all in one column in our dataframe - we need to transform it into tidy format such that one word appears in each row. We do this by 'unnesting' the text column and removing 'stop words'. These are common words (e.g., function words like 'the' and 'of').

```
all_text <- books %>%
 unnest_tokens(word, text) %>%
 anti_join(stop_words)
```

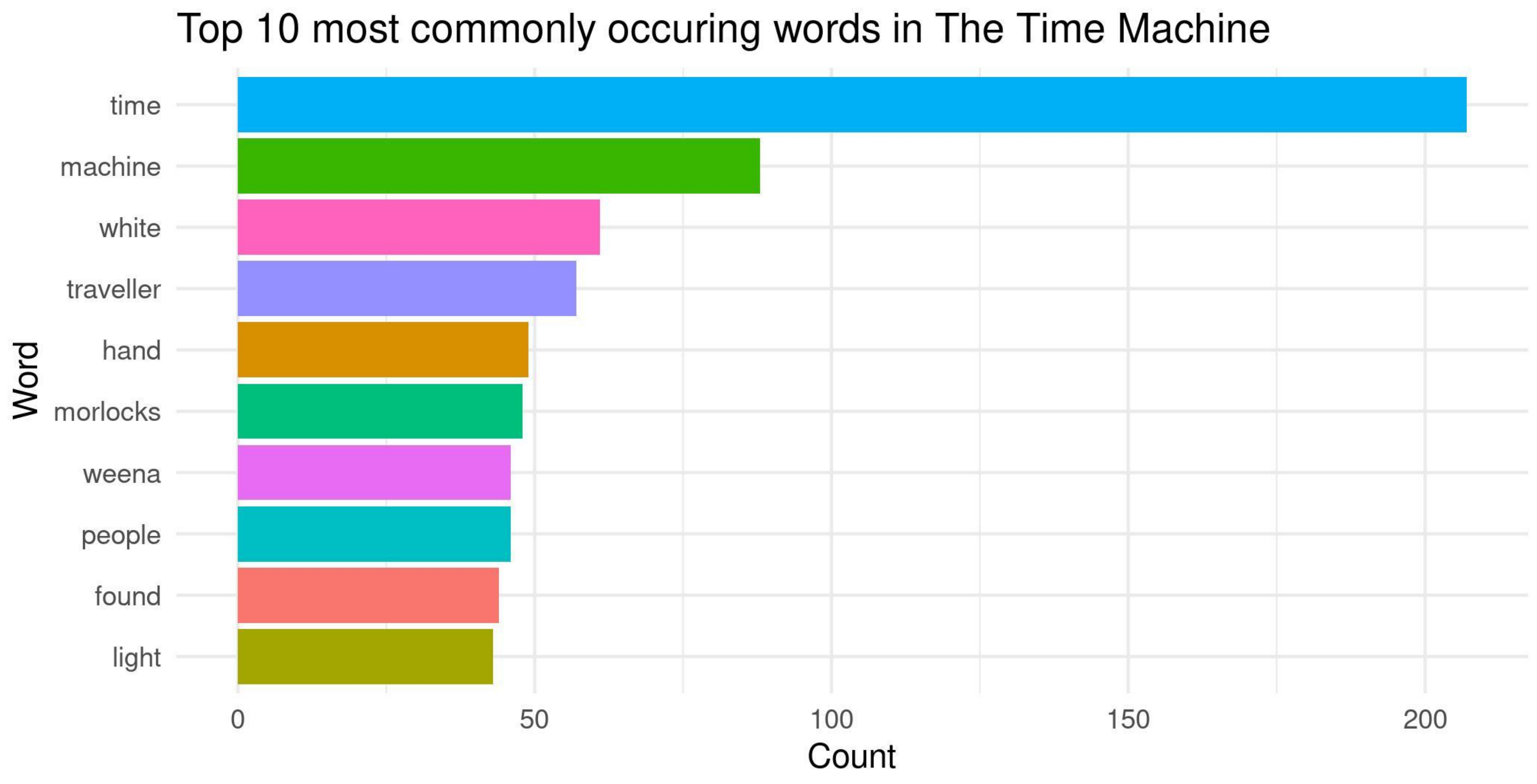
```
all_text
# A tibble: 91,676 x 3
   gutenberg_id title            word
          <int> <chr>            <chr>
 1           35 The Time Machine time
 2           35 The Time Machine machine
 3           35 The Time Machine invention
 4           35 The Time Machine contents
 5           35 The Time Machine introduction
 6           35 The Time Machine ii
 7           35 The Time Machine machine
 8           35 The Time Machine iii
 9           35 The Time Machine time
10           35 The Time Machine traveller
# … with 91,666 more rows
```

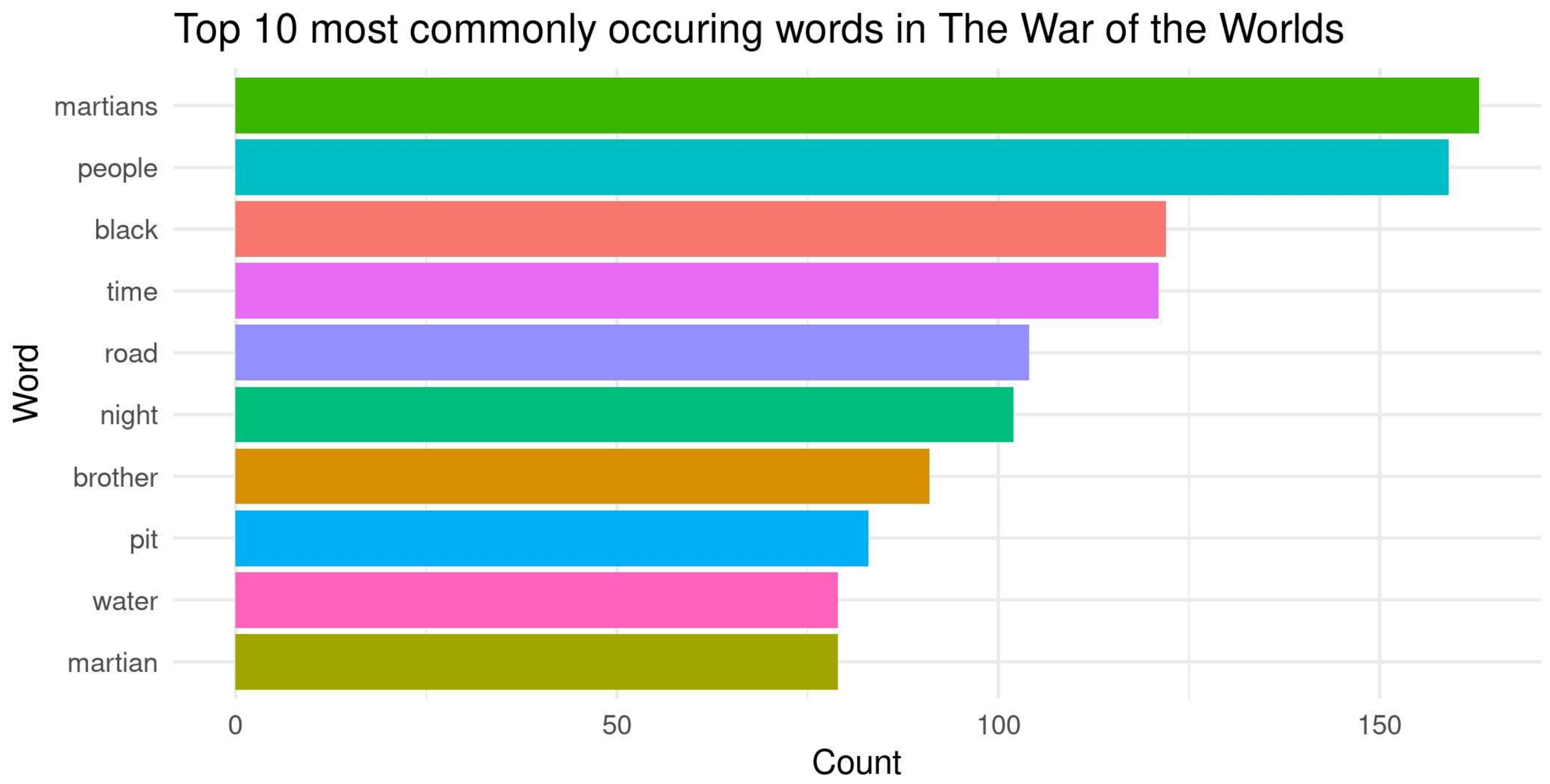
Summary Data of “The Time Machine”

```
all_text %>%
  filter(title == "The Time Machine") %>%
  count(word, sort = TRUE) %>%
  top_n(10)

Selecting by n
# A tibble: 10 x 2
   word          n
   <chr>     <int>
 1 time        207
 2 machine      88
 3 white        61
 4 traveller    57
 5 hand         49
 6 morlocks     48
 7 people       46
 8 weena        46
 9 found        44
10 light        43
```

Summary Data of “The War of the Worlds”

```
all_text %>%
  filter(title == "The War of the Worlds") %>%
  count(word, sort = TRUE) %>%
  top_n(10)

Selecting by n
# A tibble: 10 x 2
   word          n
   <chr>     <int>
 1 martians    163
 2 people      159
 3 black       122
 4 time        121
 5 road        104
 6 night       102
 7 brother      91
 8 pit          83
 9 martian      79
10 water        79
```


Top 10 most commonly occuring words in The Time Machine
time
machine
white
traveller
hand
morlocks
weena
people
found
light
Word
0
50
100
150
200
Count


Top 10 most commonly occuring words in The War of the Worlds
martians
people
black
time
road
night
brother
pit
water
martian
Word
0
50
100
150
Count

Sentiment Analysis

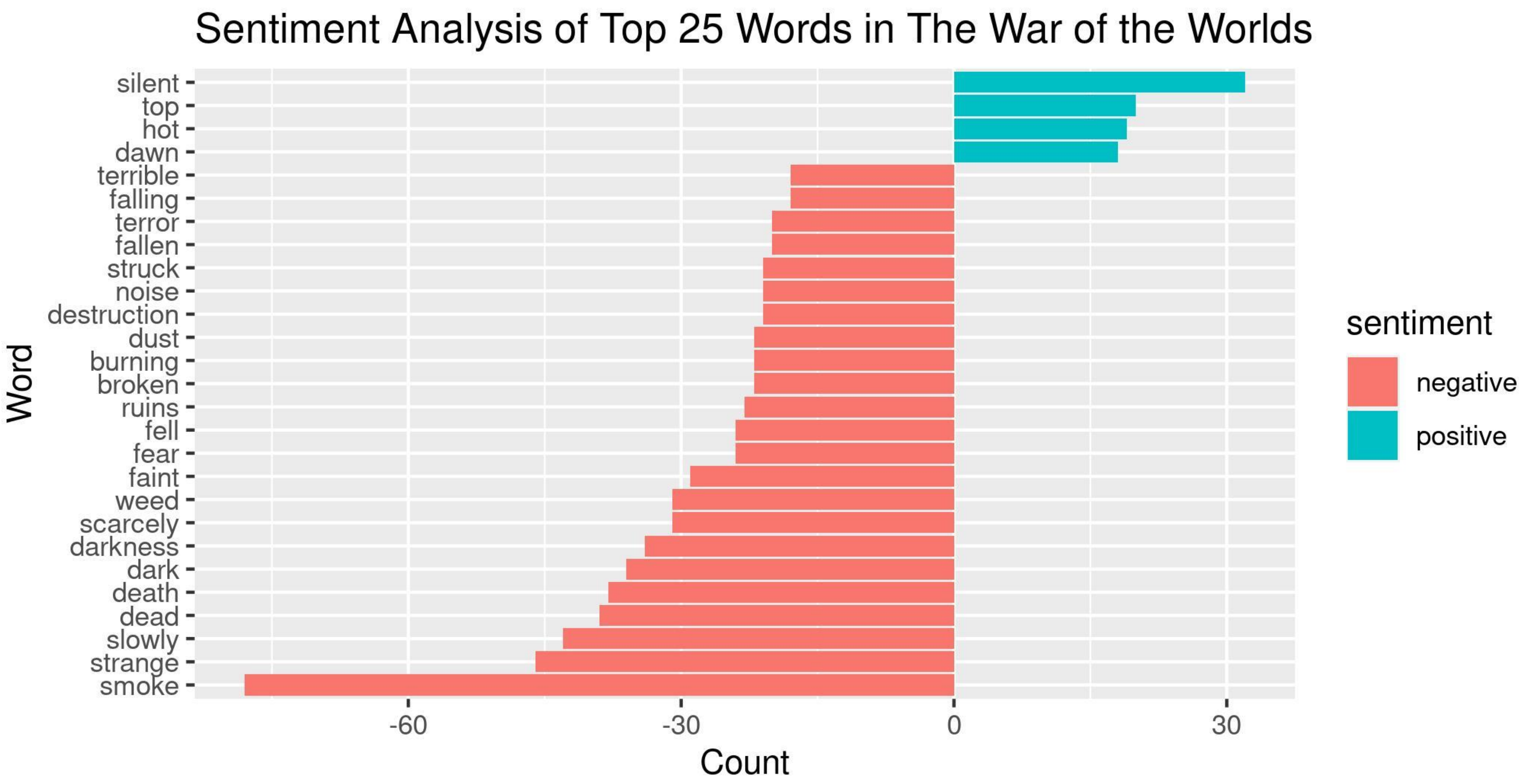
We can use one of the sentiment databases built-in to the tidytext package. The 'bing' database has sentiment ratings (positive vs. negative) for almost 7,000 words.

```
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
# … with 6,776 more rows
```

Sentiment Analysis

We can 'join' our `all_text` data to the sentiment dataset using the `inner_join()` function from `{dplyr}`

```
all_text_sentiments <- all_text %>%
  inner_join(get_sentiments("bing"))

head(all_text_sentiments)

# A tibble: 6 x 4
  gutenberg_id title            word          sentiment
         <int> <chr>            <chr>         <chr>
1          35 The Time Machine golden        positive
2          35 The Time Machine shock         negative
3          35 The Time Machine darkness      negative
4          35 The Time Machine trap          negative
5          35 The Time Machine convenient    positive
6          35 The Time Machine pale          negative
```

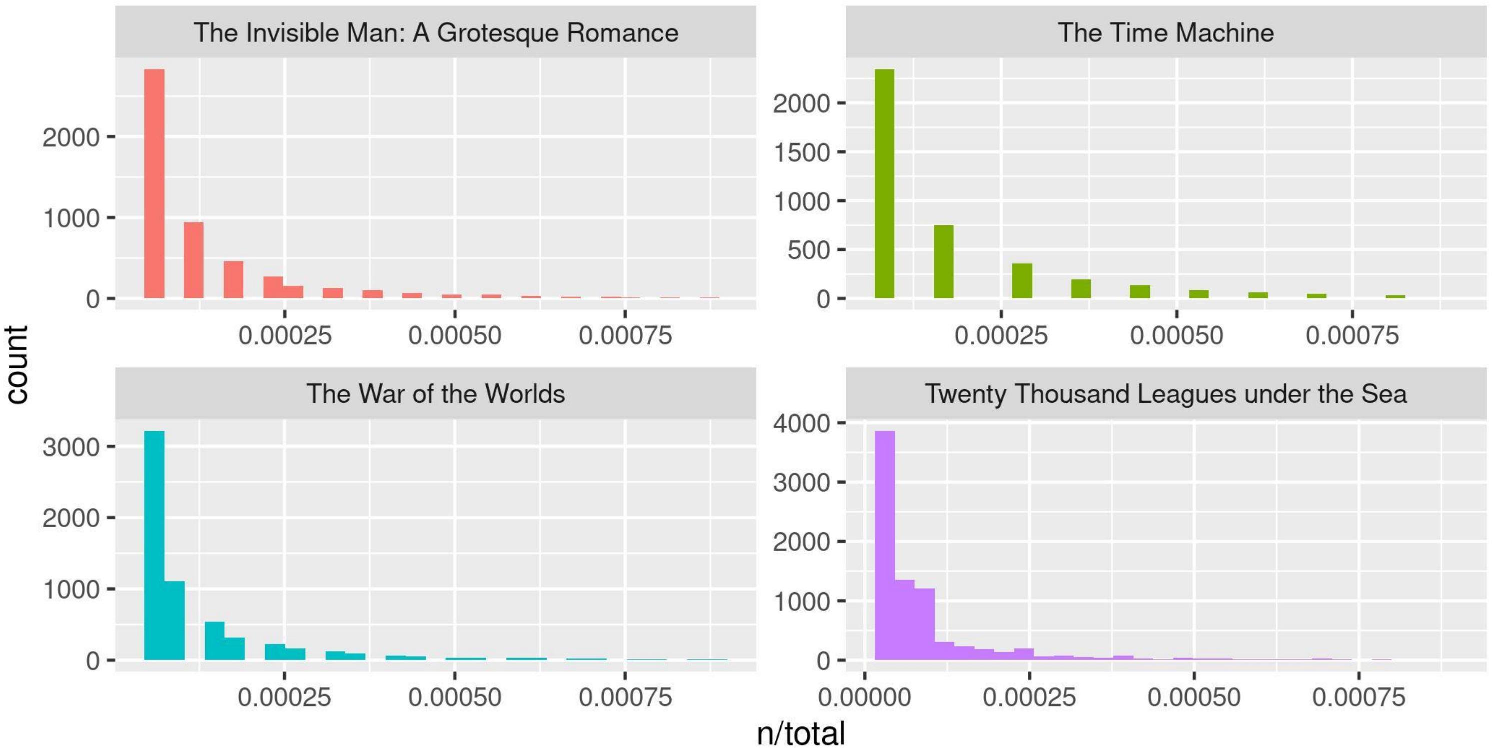

Sentiment Analysis of Top 25 Words in The War of the Worlds
silent
top
hot
dawn
terrible
falling
terror
fallen
struck
noise
destruction
dust
burning
broken
ruins
fell
fear
faint
weed
scarcely
darkness
dark
death
dead
slowly
strange
smoke
Word
-60
-30
0
30
Count
sentiment
negative
positive

Examining the proportion of useage of each word in each book

```
book_words <- all_text %>%
  group_by(title) %>%
  count(title, word, sort = TRUE)

total_words <- book_words %>%
  group_by(title) %>%
  summarise(total = sum(n))

book_words <- left_join(book_words, total_words)

book_words %>%
  mutate(proportion = n/total) %>%
  group_by(title) %>%
  arrange(desc(title, proportion)) %>%
  top_n(3) %>%
  select(-n, -total)
```

```
Selecting by proportion
# A tibble: 12 x 3
# Groups:   title [4]
   title                                   word       proportion
   <chr>                                   <chr>           <dbl>
 1 Twenty Thousand Leagues under the Sea  captain        0.0153
 2 Twenty Thousand Leagues under the Sea  nautilus       0.0131
 3 Twenty Thousand Leagues under the Sea  sea            0.00880
 4 The War of the Worlds                   martians   0.00722
 5 The War of the Worlds                   people     0.00704
 6 The War of the Worlds                   black      0.00540
 7 The Time Machine                        time       0.0184
 8 The Time Machine                        machine    0.00781
 9 The Time Machine                        white      0.00541
10 The Invisible Man: A Grotesque Romance kemp           0.0117
11 The Invisible Man: A Grotesque Romance invisible      0.00990
12 The Invisible Man: A Grotesque Romance door           0.00930
```

Visualizing the data - Zipf's Law

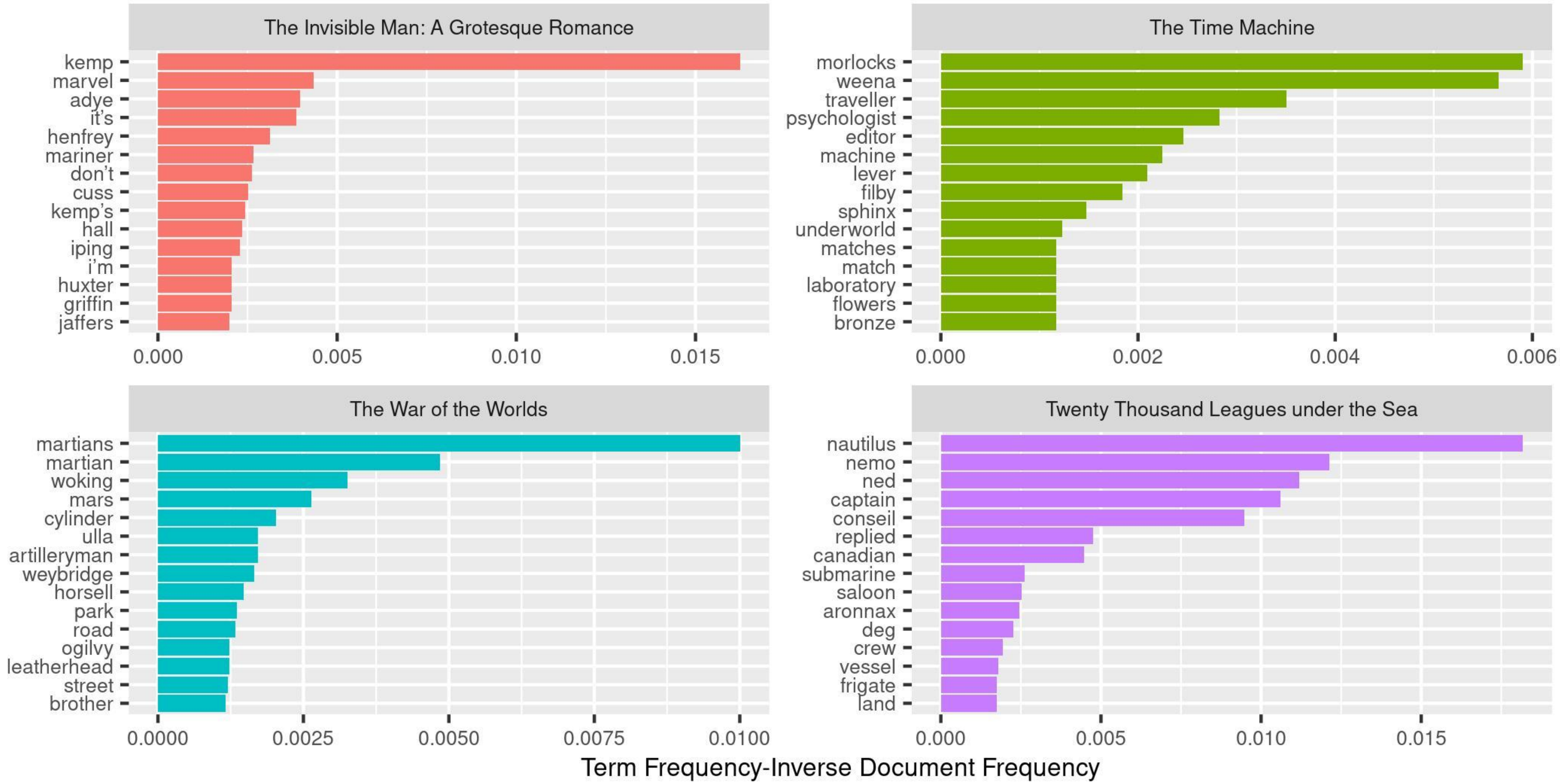


Which words are most important (and most unique) to each book?

The `bind_tf_idf()` function works out the important words for each book by adding a weighting to each word - decreasing the weight for commonly used words and increasing the weight for words not used much in the overall corpus. This is the term frequency-inverse document frequency measure used widely in text analysis.

This allows us to identify what words tend to be uniquely associated with each of the four books. This is known as the term frequency-inverse document frequency statistic.

```
book_words_tf_idf <- book_words %>%
  bind_tf_idf(word, title, n)
```


The Invisible Man: A Grotesque Romance
kemp
marvel
adye
it's
henfrey
mariner
don't
cuss
kemp's
hall
iping
i'm
huxter
griffin
jaffers
0.000
0.005
0.010
0.015
The Time Machine
morlocks
weena
traveller
psychologist
editor
machine
lever
filby
sphinx
underworld
matches
match
laboratory
flowers
bronze
0.000
0.002
0.004
0.006
The War of the Worlds
martians
martian
woking
mars
cylinder
ulla
artilleryman
weybridge
horsell
park
road
ogilvy
leatherhead
street
brother
0.0000
0.0025
0.0050
0.0075
0.0100
Twenty Thousand Leagues under the Sea
nautilus
nemo
ned
captain
conseil
replied
canadian
submarine
saloon
aronnax
deg
crew
vessel
frigate
land
0.000
0.005
0.010
0.015
Term Frequency-Inverse Document Frequency

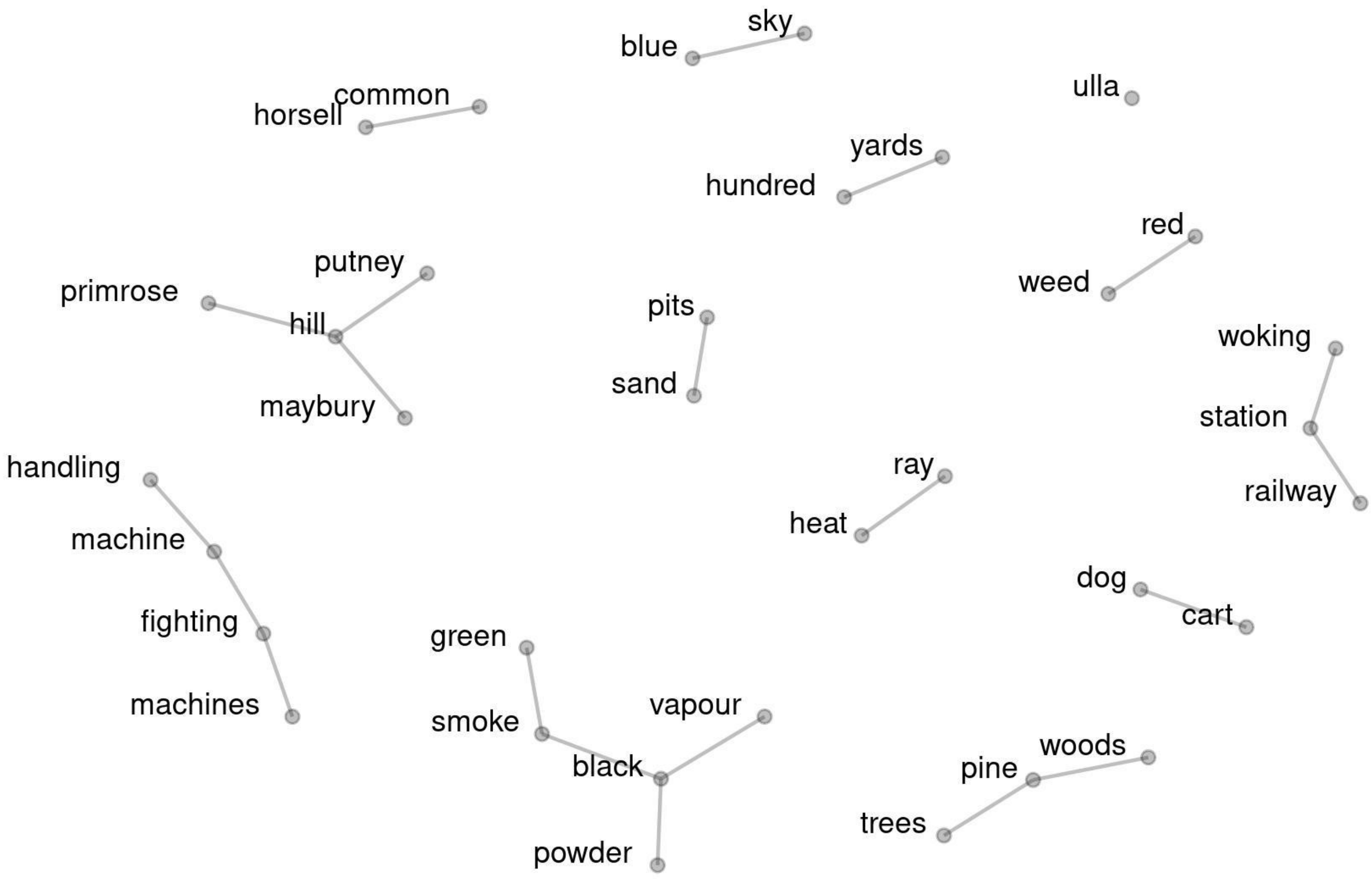
N-gram tokenizing

So far we've unnested such that each word is separate. But we can also unnest by n-grams to capture *sequences* of words. In this example, let's look at tokenizing by bigram.

```
wotw_bigrams <- books %>%
  filter(title == "The War of the Worlds") %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
  separate(col = bigram, into = c("word1", "word2", sep = " ")) %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  count(word1, word2, sort = TRUE)
```

Plotting a network graph of bigrams

```
bigram_graph <- wotw_bigrams %>%
  filter(n > 5) %>%
  graph_from_data_frame()

set.seed(1234)
ggraph(bigram_graph, layout = "fr") +
  geom_edge_link(alpha = .25) +
  geom_node_point(alpha = .25) +
  geom_node_text(aes(label = name), vjust = -.1, hjust = 1.25, size = 3) +
  guides(size = FALSE) +
  xlim(10, 22) +
  theme_void()
```


sky
blue
ulla
common
horsell
yards
hundred
red
weed
putney
primrose
hill
pits
woking
sand
maybury
station
ray
handling
railway
heat
machine
dog
cart
fighting
green
vapour
machines
smoke
woods
pine
black
trees
powder

Summary

With `{tidytext}` in R you can extract a lot of information about different texts - you might consider applying the approach to interview transcripts (for example) as a way of providing quantitative insights in addition to qualitative approaches.

You might even want to use the term frequency-inverse document frequency measure as a way of understanding what words or n-grams are associated with particular interviews (or sets of interviews) and not with others.