Quiz 2

Nadezhda Gesheva

18 Feb 2018

Text problems

1. Explain in your words what the unnest_token function does.

The unnest_token function explodes or splits each item from a selected column to one-item-per column frame.

2. Explain your words what the gutenbergr package does.

The gutenbergr package provides public access to various texts from the Project Gutenberg (works no longer patented). It also contains metadata for each work, author and subject.

3. Explain in your words how sentiment lexicon work.

Sentiment lexicon provides us with the functionality to obtain the overall sentiment of a text We do that by comparing the words most frequently used to words which we regards positive/negative("bing" lexicon); by employing various words expressing emotions such as joy, sadness, fear, etc. ("nrc" lexicon); or by scaling them between -5 and 5 showing overall positiveness/neutrality/negativity of the words in a text.

4. How does inner_join provide sentiment analysis functionality.

Inner join enables us to match the words that both exist in a certain lexicon and the words present in the text.

5. Explain in your words what tf-idf does.

Tf stands for term frequency and represents the frequency in which words occur in a text. Idf stands for inverse documented frequency, which is a useful technique that gives you actual value that you get out of a token within a document. It does that by assigning weights to the words that are used frequently everywhere, while boosting the weight of frequent words used only in particular chunks.

6. Explain why you may want to do tokenization by bigram.

Bigrams are a useful tool to detect the bias that you might get from text analysis using monograms. For instance, the word "happy" might be the most frequent within a text, and hence we would conclude that the text is positive. Nevertheless, the author of this particular text might use the bigram "not happy" instead of "unhappy". Hence, our initial conclusion might be wrong.

```r
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(tidyverse)
```

```
## -- Attaching packages ----------------------------------------------------------------- tidyverse 1.2
## v ggplot2 3.1.0     v readr   1.1.1
## v tibble  1.4.2     v purrr   0.2.5
## v tidyr   0.8.1     v stringr 1.3.1
## v ggplot2 3.1.0     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------------------------------ tidyverse_conflicts
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidytext)
library(gutenbergr)
library(ggplot2)
```

Most frequent words

```
wilde_works <-  gutenberg_download(c(174, 301, 773, 774, 790, 844, 854, 873, 875, 885, 887,
                                    902, 921, 1017, 1031, 1057, 1141, 1308, 1338, 14062,
                                    14240, 14522, 23229, 26494, 30120, 30191, 33979, 41806,
                                    42104, 42704, 51563),
                                  meta_fields = "title"
                                  )
```

```
## Determining mirror for Project Gutenberg from http://www.gutenberg.org/robot/harvest
```

```
## Using mirror http://aleph.gutenberg.org
```

```
london_works <- gutenberg_download(c( 215, 310, 318, 710, 746, 788, 910, 911, 1029, 1056, 1074,
                                     1075, 1089, 1096, 1160, 1161, 1162, 1163, 1164, 1187, 1208,
                                     1449, 1596, 1655, 1669, 1688, 1730, 2377, 2415, 2416, 2429,
                                     2512, 2545, 4953, 6455, 10736, 11051, 12336, 14449, 14654,
                                     14658, 16257, 18062, 19678, 21936, 21970, 21971, 22104, 48474),
                                     meta_fields = "title"
                                     )
```

```
## Warning in .f(.x[[i]], ...): Could not download a book at http://
## aleph.gutenberg.org/1/9/6/7/19678/19678.zip
```

```
wilde_works %>%
  count(title)
```

```
## # A tibble: 31 x 2
##    title                                                         n
##    <chr>                                                     <int>
##  1 A Critic in Pall Mall: Being Extracts from Reviews and Miscellan~  6643
##  2 A Florentine Tragedy; La Sainte Courtisane                 1636
##  3 A House of Pomegranates                                    3329
##  4 A Woman of No Importance                                   3274
##  5 "A Woman of No Importance\nAudio performance"               124
##  6 An Ideal Husband                                           4464
##  7 Charmides, and Other Poems                                 2064
##  8 Children in Prison and Other Cruelties of Prison Life       403
##  9 De Profundis                                               1500
## 10 Essays and Lectures                                        5062
## # ... with 21 more rows
```

```r
# Our tibble contains 21 works of Wilde

london_works %>%
  count(title)
```

```
## # A tibble: 48 x 2
##    title                                                              n
##    <chr>                                                          <int>
##  1 A Daughter of the Snows                                        10360
##  2 A Son Of The Sun                                                6963
##  3 Adventure                                                       8007
##  4 Before Adam                                                     3711
##  5 "Brown Wolf and Other Jack London Stories\nChosen and Edited By ~  6546
##  6 Burning Daylight                                               11769
##  7 Children of the Frost                                           5573
##  8 Dutch Courage and Other Stories                                 3730
##  9 Jerry of the Islands                                            7131
## 10 John Barleycorn                                                 6236
## # ... with 38 more rows
```

```r
# Our tibble contains 48 works of London

# Wilde - Transform the tibble to a tidy-text dataset
tidy_wilde <- wilde_works %>%
  group_by(title) %>%
  mutate(line_number = row_number()) %>%
  unnest_tokens(word, text) %>%
  ungroup()

# The most frequently used words by Wilde
tidy_wilde %>%
  count(word, sort = TRUE) %>%
  head(10)
```

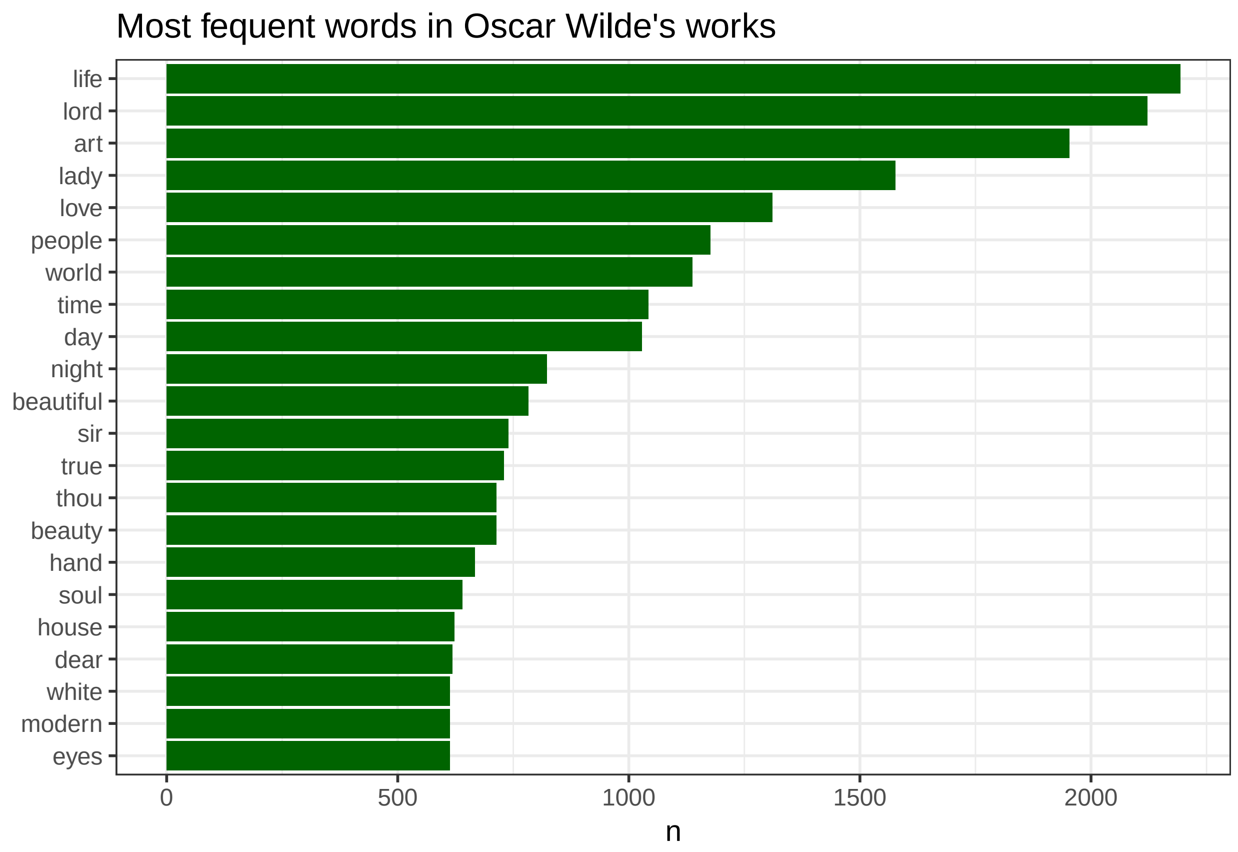
```
## # A tibble: 10 x 2
##    word      n
##    <chr> <int>
##  1 the   58693
##  2 of    36992
##  3 and   32858
##  4 to    23067
##  5 a     20222
##  6 in    16567
##  7 is    16359
##  8 that  13411
##  9 i     12415
## 10 it    11480
```

```r
# mostly stop words

tidy_wilde %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  head(10)
```

```
## Joining, by = "word"
## # A tibble: 10 x 2
##    word       n
##    <chr>  <int>
##  1 life    2193
##  2 lord    2122
##  3 art     1953
##  4 lady    1577
##  5 love    1311
##  6 people  1176
##  7 world   1138
##  8 time    1042
##  9 day     1028
## 10 night    823
```

```
tidy_wilde %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  filter(n > 600) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col(fill = "darkgreen") +
  xlab(NULL) +
  ggtitle("Most fequent words in Oscar Wilde's works") +
  theme_bw() +
  coord_flip()
```

```
## Joining, by = "word"
```



```r
# London - Transform the tibble to a tidy-text dataset
tidy_london <- london_works %>%
  group_by(title) %>%
  mutate(line_number = row_number()) %>%
  unnest_tokens(word, text) %>%
  ungroup()

# The most frequently used words by London
tidy_london %>%
  count(word, sort = TRUE) %>%
  head(10)
```

```
## # A tibble: 10 x 2
##    word       n
##    <chr>  <int>
##  1 the   184905
##  2 and   113750
##  3 of     80128
##  4 to     65104
##  5 a      61256
##  6 he     48596
##  7 was    47827
##  8 in     45733
##  9 i      41912
## 10 it     35354
```

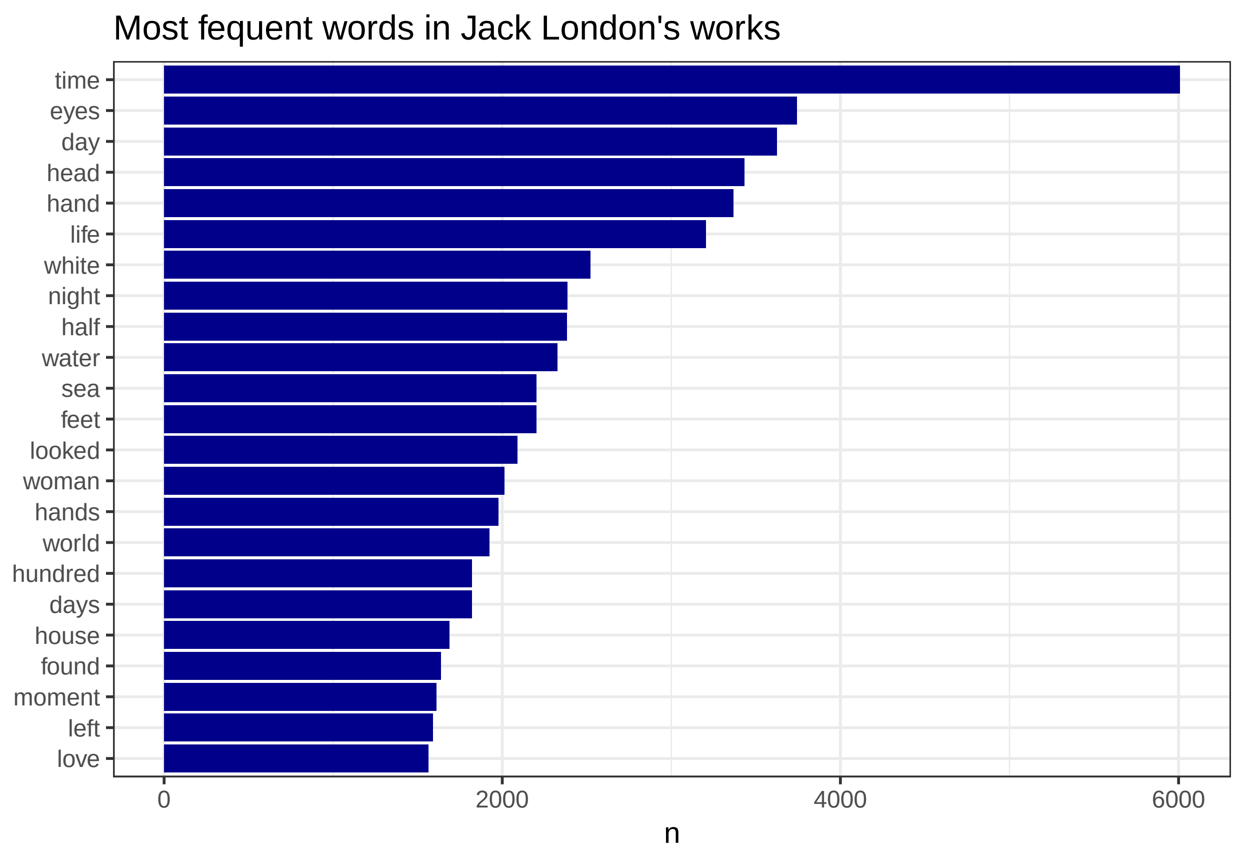
```r
# mostly stop words

tidy_london %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  head(10)
```

```
## Joining, by = "word"
```

```
## # A tibble: 10 x 2
##    word      n
##    <chr> <int>
##  1 time   6009
##  2 eyes   3743
##  3 day    3624
##  4 head   3432
##  5 hand   3368
##  6 life   3203
##  7 white  2521
##  8 night  2386
##  9 half   2381
## 10 water  2326
```

```r
# Life, time and night are in both of top ten most frequent words used by both Oscar and Jack.

tidy_london %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  filter(n > 1500) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col(fill = "darkblue") +
  xlab(NULL) +
  ggtitle("Most fequent words in Jack London's works") +
  theme_bw() +
  coord_flip()
```

```
## Joining, by = "word"
```



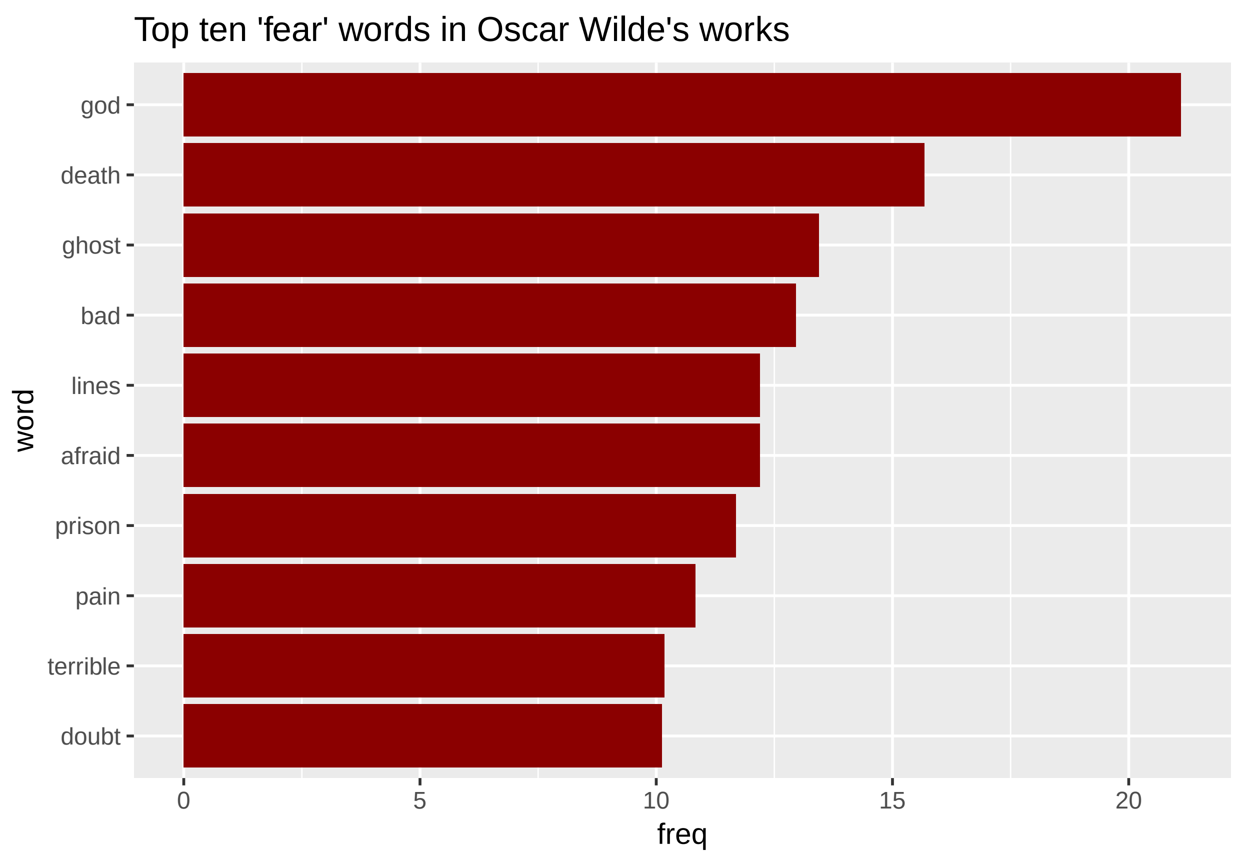
Sentiment Analysis

```
t_wilde <- wilde_works %>%
  unnest_tokens(word, text) %>%
  count(title, word, sort = TRUE) %>%
  ungroup()

sentim_wilde <- t_wilde %>%
  inner_join(get_sentiments("nrc"))
```

```
## Joining, by = "word"
```

```
fear_words_wilde <- sentim_wilde %>%
  filter(sentiment == "fear") %>%
  group_by(word) %>%
  summarize(freq = mean(n)) %>%
  arrange(desc(freq))

fear_words_wilde %>%
  top_n(10) %>%
  mutate(word = reorder(word, freq)) %>%
  ggplot(aes(word, freq)) +
  geom_col(fill = "darkred") +
  ggtitle("Top ten 'fear' words in Oscar Wilde's works") +
  coord_flip()
```

```
## Selecting by freq
```



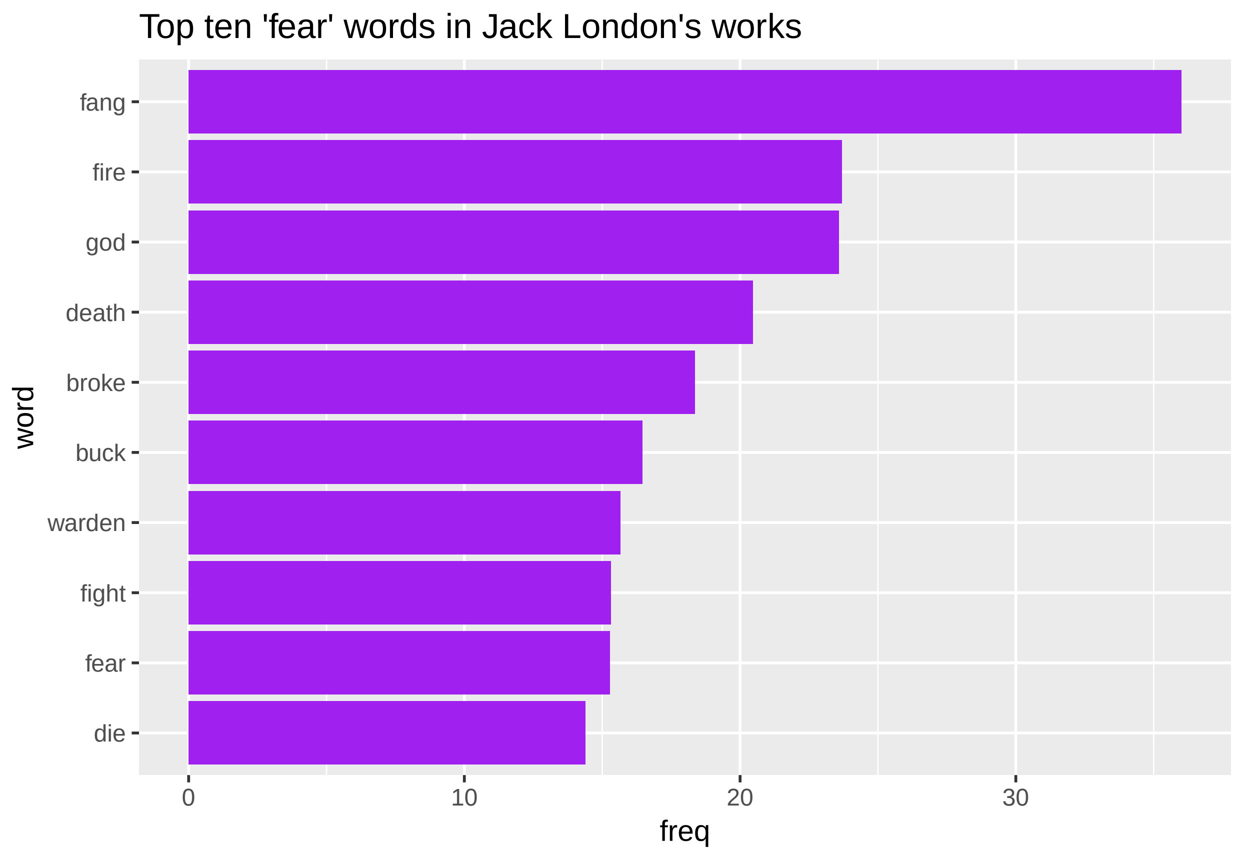
```
# Similarly for Jack London
t_london <- london_works %>%
  unnest_tokens(word, text) %>%
  count(title, word, sort = TRUE) %>%
  ungroup()

sentim_london <- t_london %>%
  inner_join(get_sentiments("nrc"))
```

```
## Joining, by = "word"
```

```
fear_words_london <- sentim_london %>%
  filter(sentiment == "fear") %>%
  group_by(word) %>%
```

```r
  summarize(freq = mean(n)) %>%
  arrange(desc(freq))

fear_words_london %>%
  top_n(10) %>%
  mutate(word = reorder(word, freq)) %>%
  ggplot(aes(word, freq)) +
  geom_col(fill = "purple") +
  ggtitle("Top ten 'fear' words in Jack London's works") +
  coord_flip()
```

```
## Selecting by freq
```



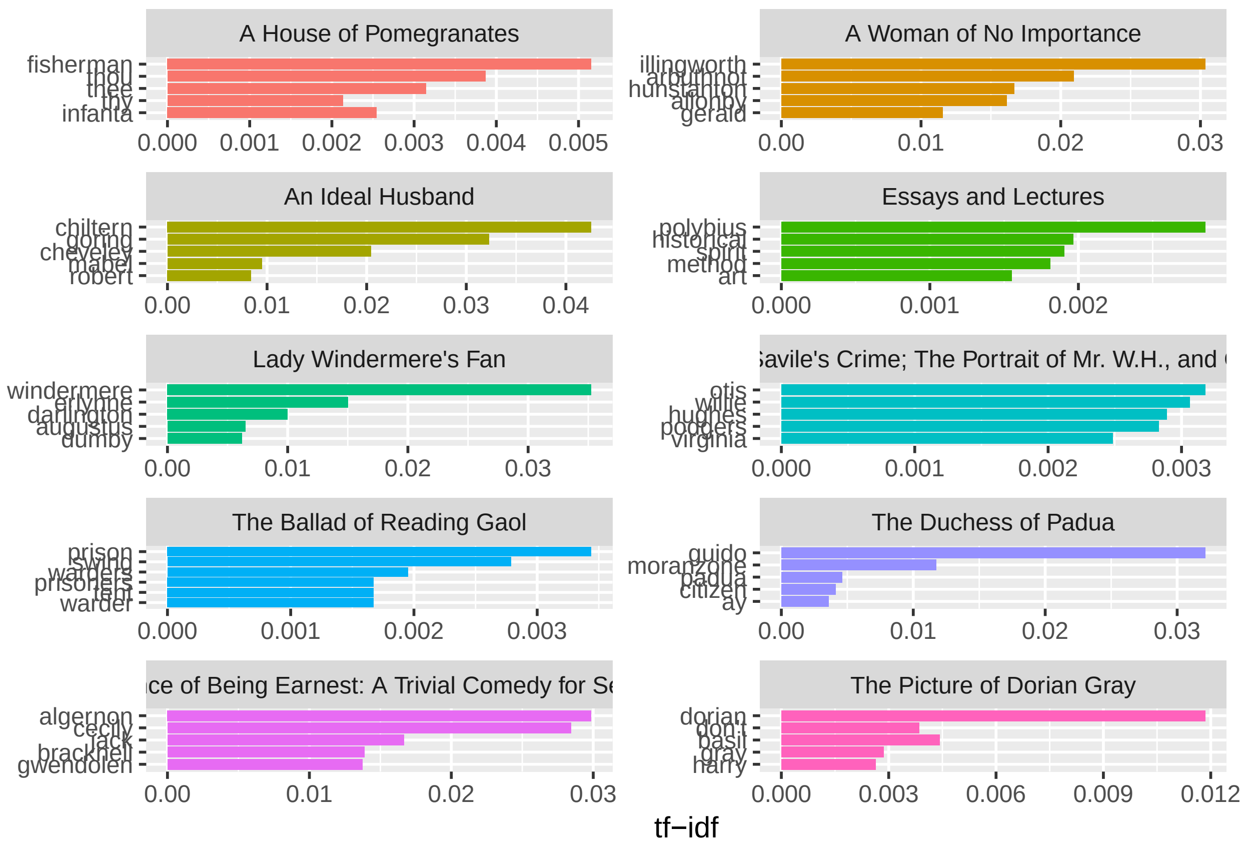
Tf-idf

```r
# For the tf-idf analysis we will just pick 10 works per author

# Start with Oscar Wilde
wilde_10_works <-  gutenberg_download(c(174, 301, 773, 774, 790, 844, 854, 873, 875, 885),
                                meta_fields = "title")

book_words_wilde <- wilde_10_works %>%
  unnest_tokens(word, text) %>%
  count(title, word, sort = TRUE) %>%
  ungroup()
total_words_wilde <- book_words_wilde %>%
  group_by(title) %>%
  summarize(total = sum(n))

book_words_wilde <- left_join(book_words_wilde, total_words_wilde)
```

```
## Joining, by = "title"
```

```r
#book_words_wilde

book_words_wilde <- book_words_wilde %>%
  bind_tf_idf(word, title, n)

book_words_wilde %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  group_by(title) %>%
  top_n(5) %>%
  ungroup %>%
  ggplot(aes(word, tf_idf, fill = title)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~title, ncol = 2, scales = "free") +
  coord_flip()
```

```
## Selecting by tf_idf
```

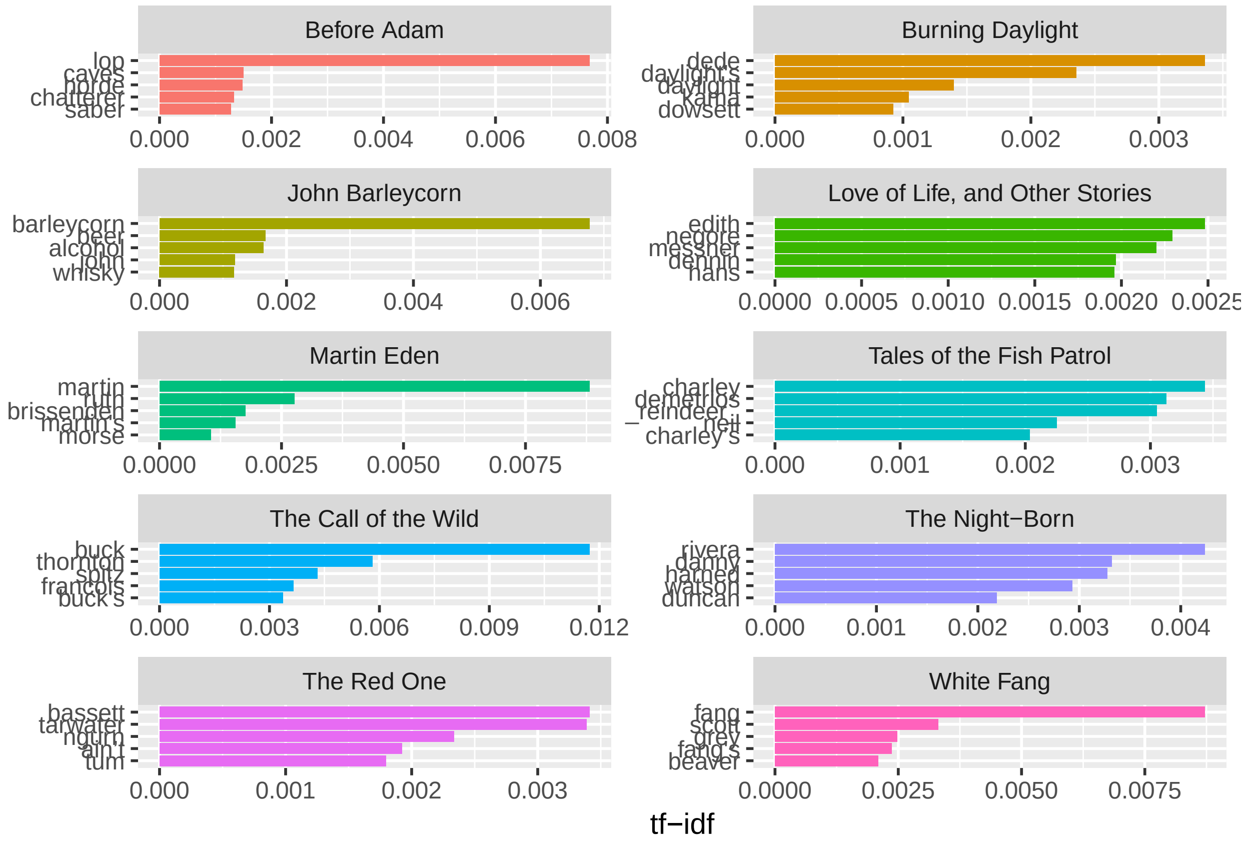


```r
# Similar analysis tf-idf for Jack London's works

london_10_works <- gutenberg_download(c( 215, 310, 318, 710, 746, 788, 910, 911, 1029, 1056),
                                      meta_fields = "title")

book_words_london <- london_10_works %>%
  unnest_tokens(word, text) %>%
  count(title, word, sort = TRUE) %>%
  ungroup()
total_words_london <- book_words_london %>%
  group_by(title) %>%
  summarize(total = sum(n))

book_words_london <- left_join(book_words_london, total_words_london)
```

```
## Joining, by = "title"
```

```r
#book_words_london

book_words_london <- book_words_london %>%
  bind_tf_idf(word, title, n)

book_words_london %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  group_by(title) %>%
  top_n(5) %>%
  ungroup %>%
  ggplot(aes(word, tf_idf, fill = title)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~title, ncol = 2, scales = "free") +
  coord_flip()
```

```
## Selecting by tf_idf
```



Before Adam
lop
caves
horde
chatterer
saber
0.000
0.002
0.004
0.006
0.008
Burning Daylight
dede
daylight's
daylight
kama
dowsett
0.000
0.001
0.002
0.003
John Barleycorn
barleycorn
beer
alcohol
john
whisky
0.000
0.002
0.004
0.006
Love of Life, and Other Stories
edith
negore
messner
dennin
hans
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
Martin Eden
martin
ruth
brissenden
martin's
morse
0.0000
0.0025
0.0050
0.0075
Tales of the Fish Patrol
charley
demetrios
_reindeer
neil
charley's
0.000
0.001
0.002
0.003
The Call of the Wild
buck
thornton
spitz
francois
buck's
0.000
0.003
0.006
0.009
0.012
The Night−Born
rivera
danny
harned
watson
duncan
0.000
0.001
0.002
0.003
0.004
The Red One
bassett
tarwater
ngurn
ain't
tum
0.000
0.001
0.002
0.003
White Fang
fang
scott
grey
fang's
beaver
0.0000
0.0025
0.0050
0.0075
tf−idf