exploration.R

Gabriel

2020-03-11

```
# Loading in packages to be used
packages <- c("tidyverse", "ggplot2", "stringr", "quanteda",
              "tidytext", "textdata", "wordcloud", "wordcloud2",
              "reshape2", "tm", "igraph", "ggraph")
invisible(lapply(packages, library, character.only = TRUE))
```

```
## -- Attaching packages ------------------------------------------------------------------ tidyverse 1
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

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

```
## Package version: 1.5.2
```

```
## Parallel computing: 2 of 4 threads used.
```

```
## See https://quanteda.io for tutorials and examples.
```

```
##
## Attaching package: 'quanteda'
```

```
## The following object is masked from 'package:utils':
##
##     View
```

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

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
##
## Attaching package: 'tm'
```

```
## The following objects are masked from 'package:quanteda':
##
##     as.DocumentTermMatrix, stopwords

##
## Attaching package: 'igraph'

## The following object is masked from 'package:quanteda':
##
##     as.igraph

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:purrr':
##
##     compose, simplify

## The following object is masked from 'package:tidyr':
##
##     crossing

## The following object is masked from 'package:tibble':
##
##     as_data_frame

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
# loading in the dataset
read_data <- function(num) {
  input_file <- "data/Gungor_2018_VictorianAuthorAttribution_data-train.csv"
  if(!is.null(input_file)) {
    dataset <- read.csv(input_file, header = T, stringsAsFactors = FALSE,
                        nrows = num)
  } else stop("Incorrect File Path", call. = TRUE)
}

# Sampling 2 authors
df <- read_data(1294)

# sampling 3 authors
df <- read_data(1507)

# Check for missing data: we have no missing data
colSums(is.na(df))
```

```
##   text author
##      0      0
```

```
# rename author column to label and convert to factor
df <- df %>% rename(label = author)
df$label <- as.factor(df$label)
```

```r
# Feature engineering - calculate character length
char_length <- function(df, text) {
  df %>% mutate(char_length =
                  str_count(text, pattern = boundary(type = "character")))
}
df <- char_length(df, text)

# Probability class distribution
prob_dist <- function(x) {
  prop.table(summary(x))
}
prob_dist(df$label)
```

```
##         1         2         3
## 0.6051758 0.2534837 0.1413404
```

```r
# Visualization: distribution between Authors and Character length
char_dist <- function(df, char_length, label) {
  ggplot(df, aes(x = char_length, fill = label)) +
    scale_fill_discrete(name = "Authors",
                        labels = c("Arthur Conan Doyle", "Charles Darwin")) +
    geom_histogram(binwidth = 10) + theme_bw() +
    labs(y = "Author ID", x = "Char Length",
         title="Frequency of Character Length with Author's Label",
         caption = "Data from UCI Machine learning Repository")
}

char_dist(df, char_length, label)
```


```
# changing author name
authors_name <- function(df, label) {
  df <- df %>% mutate(author_name = case_when(label == 1 ~ "Arthur Conan Doyle",
                                              label == 2 ~ "Charles Darwin",
                                              TRUE ~ "Missing" ))
}
df <- authors_name(df, label)

# Authors with the highest text length
df %>% group_by(author_name) %>%
  tally(sort = TRUE)
```

```
## # A tibble: 3 x 2
##   author_name            n
##   <chr>              <int>
## 1 Arthur Conan Doyle   912
## 2 Charles Darwin       382
## 3 Missing              213
```

```
#Class inbalance
df %>% group_by(label) %>%
  tally(sort=TRUE)
```

```
## # A tibble: 3 x 2
##   label     n
##   <fct> <int>
## 1 1       912
## 2 2       382
```

```
## 3 3       213
```

```
# Stratified sampling
split <- function(data) {
  splits <- sample(1:3, size = nrow(df), prob = c(.5, .2, .3), replace = T)
  training <<- df[splits == 1,]
  validation <<- df[splits == 2,]
  testing <<- df[splits == 3,]
}
split(df)

# Checking class distributions
prob_dist(training$label)
```

```
##         1         2         3
## 0.5843293 0.2841965 0.1314741
```

```
prob_dist(validation$label)
```

```
##         1         2         3
## 0.6245614 0.2245614 0.1508772
```

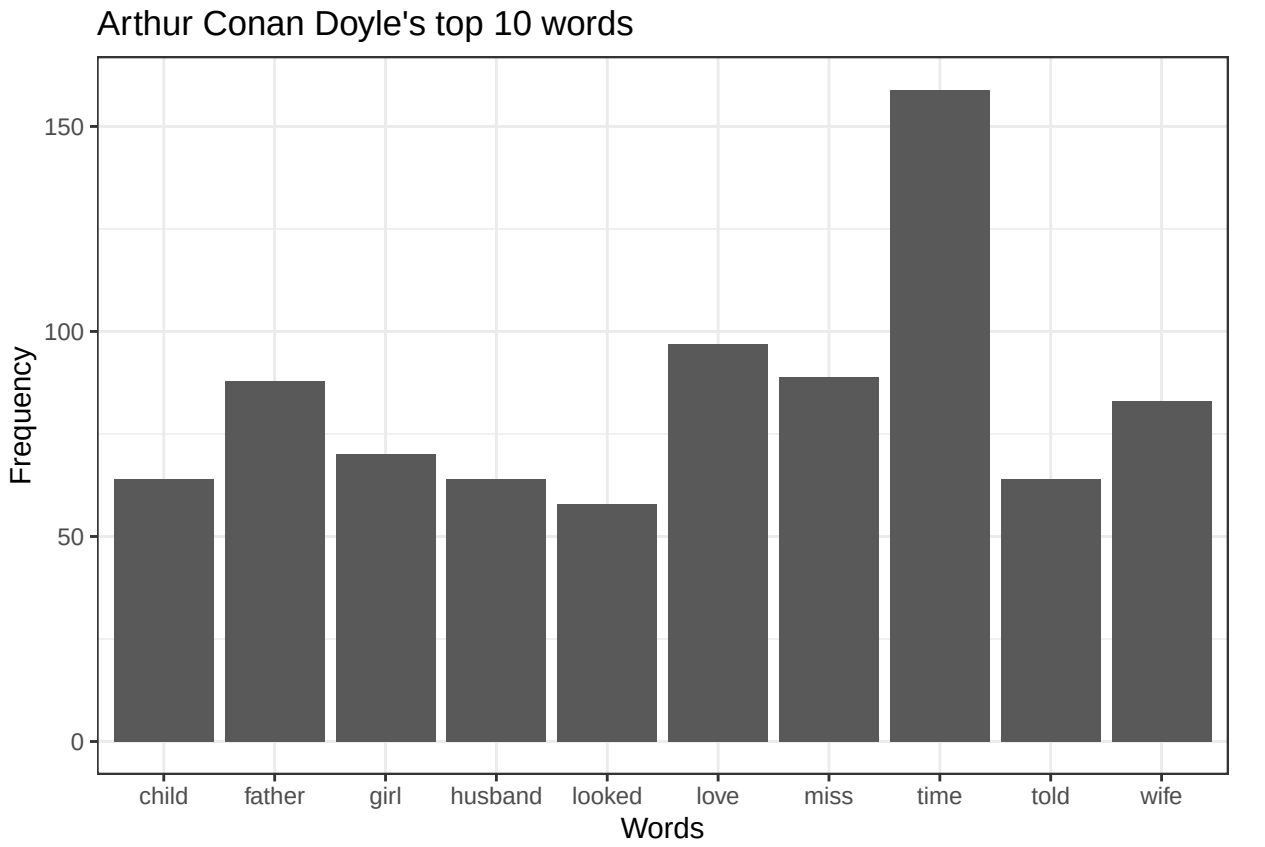
```
prob_dist(testing$label)
```

```
##         1         2         3
## 0.6268657 0.2217484 0.1513859
```

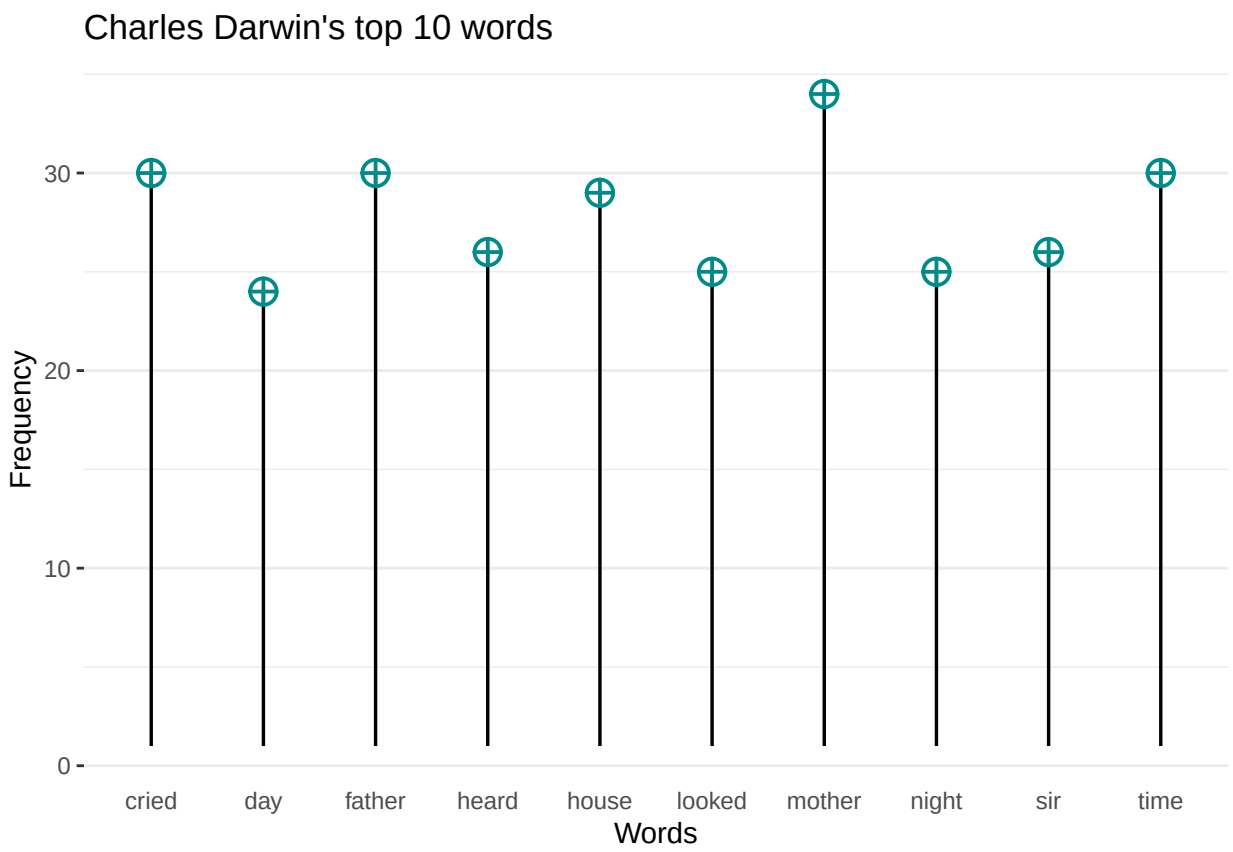
```
# shuffle data: in order to reduce variance
shuffleRows <- function(df){
  return(df[sample(nrow(df)),])
}

training <- shuffleRows(training)
validation <- shuffleRows(validation)
testing <- shuffleRows(testing)
```

```
# Data preparation for sentiment analysis
# Using the 50% of our data and a representative sampling of 100 rows
sent_df <- as_tibble(training[1:100, c("text", "label")])
sent_tokens <- unnest_tokens(sent_df, w_tokens, text, token = "words",
                             to_lower = TRUE)
sent_tokens <- sent_tokens %>% anti_join(stop_words,
                                         by = c("w_tokens" = "word"))

# Top 10 words of author
top_words <- sent_tokens %>% group_by(w_tokens, label) %>%
  tally(sort = T,  name = "freq")
top10_l1 <- sent_tokens %>% group_by(w_tokens) %>%
  filter(label == 1) %>%
  tally(sort = T,  name = "freq") %>% top_n(10, freq)
top10_l2 <- sent_tokens %>% group_by(w_tokens) %>%
  filter(label == 2) %>%
  tally(sort = T,  name = "freq") %>% top_n(10, freq)

# Arthur Conan Doyle top 10 words
ggplot(top10_l1, aes(x = w_tokens, y = freq)) + theme_bw() +
  geom_bar(stat="Identity") + labs(y = "Frequency", x = "Words",
```

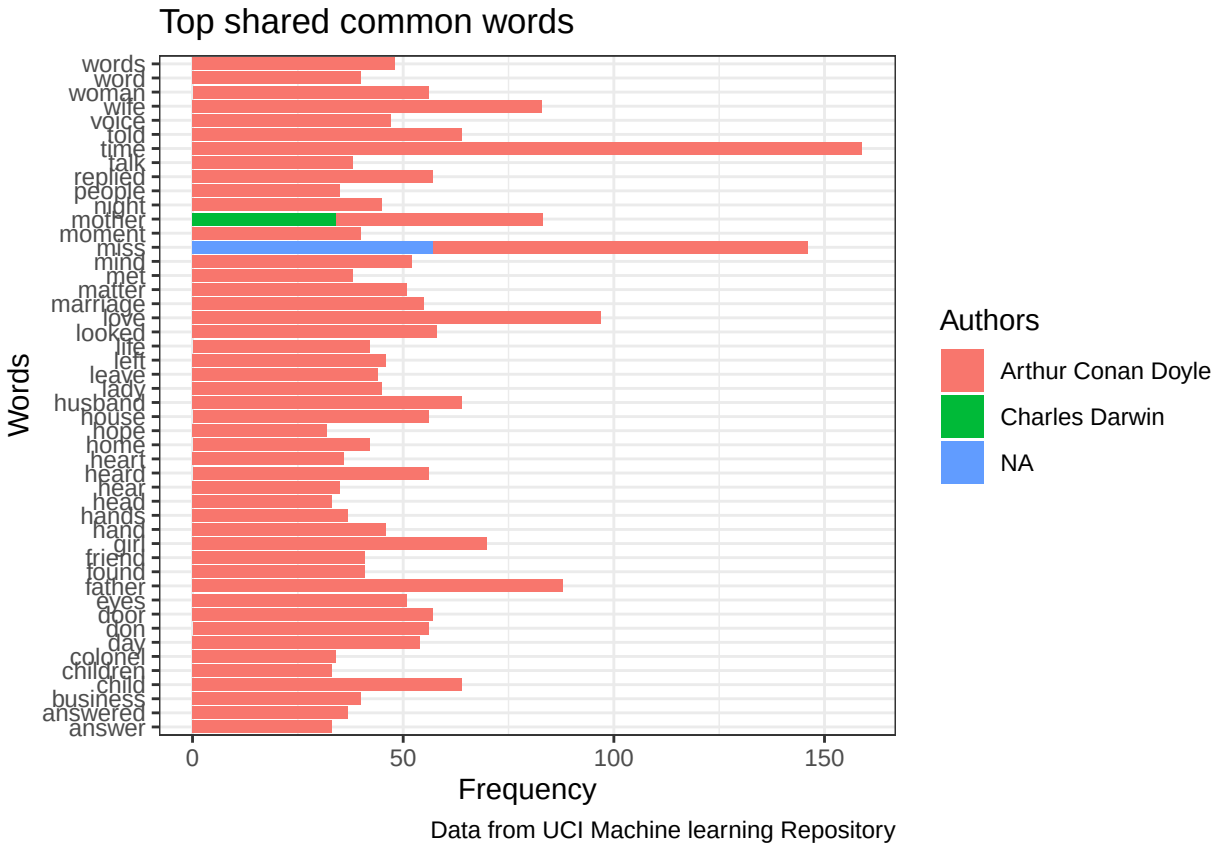
```
                                  title = "Arthur Conan Doyle's top 10 words")
```




```
# Charles Darwin top 10 words
ggplot(top10_l2, aes(x = w_tokens, y = freq)) +
  geom_segment(aes(x = w_tokens, xend = w_tokens, y = 1, yend = freq),
               size = 0.6) +
  geom_point(size = 4, color="cyan4", shape=10, stroke=1) + theme_bw() +
  theme(panel.grid.major.x = element_blank(), panel.border = element_blank(),
        axis.ticks.x = element_blank()) +
  labs(y = "Frequency", x = "Words", title = "Charles Darwin's top 10 words")
```




```
# from the top 100 words; which words do the authors share in common?
ggplot(top_words[1:50,], aes(x = w_tokens, y = freq, fill = label)) +
  geom_bar(stat="Identity") + coord_flip() + theme_bw() +
  labs(y = "Frequency", x = "Words", title="Top shared common words",
       caption = "Data from UCI Machine learning Repository") +
  scale_fill_discrete(name = "Authors",
                      labels = c("Arthur Conan Doyle", "Charles Darwin"))
```



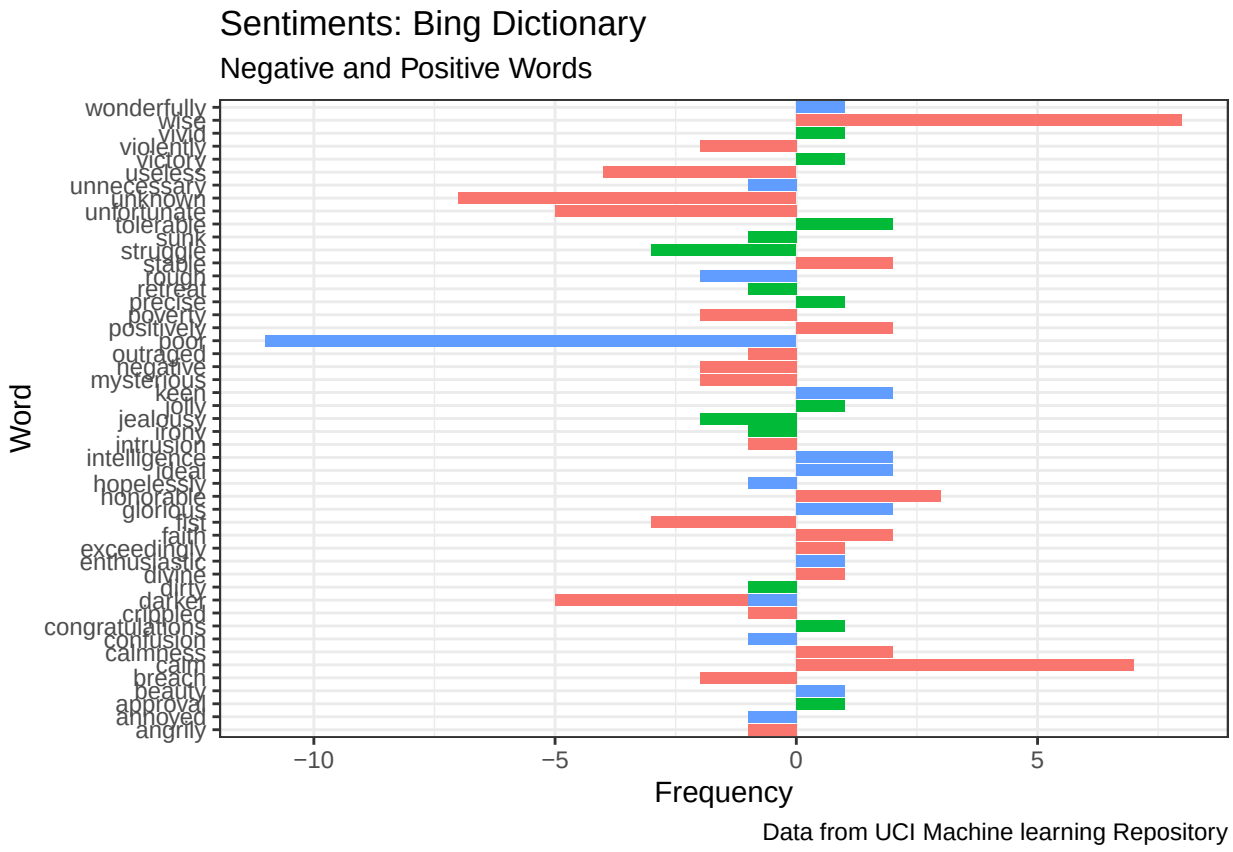

```
positive <- get_sentiments(lexicon = "bing") %>%
  filter(sentiment == "positive")

# positive sentiment analysis for Arthur Conan Doyle
top_words %>%
  filter(label == 1) %>%
  semi_join(positive, by = c("w_tokens" = "word"))
```

```
## # A tibble: 415 x 3
## # Groups:   w_tokens [415]
##    w_tokens label  freq
##    <chr>    <fct> <int>
##  1 love     1        97
##  2 glad     1        23
##  3 loved    1        18
##  4 happy    1        17
##  5 pretty   1        17
##  6 ready    1        16
##  7 smile    1        15
##  8 worth    1        15
##  9 promise  1        14
## 10 won      1        14
## # ... with 405 more rows
```

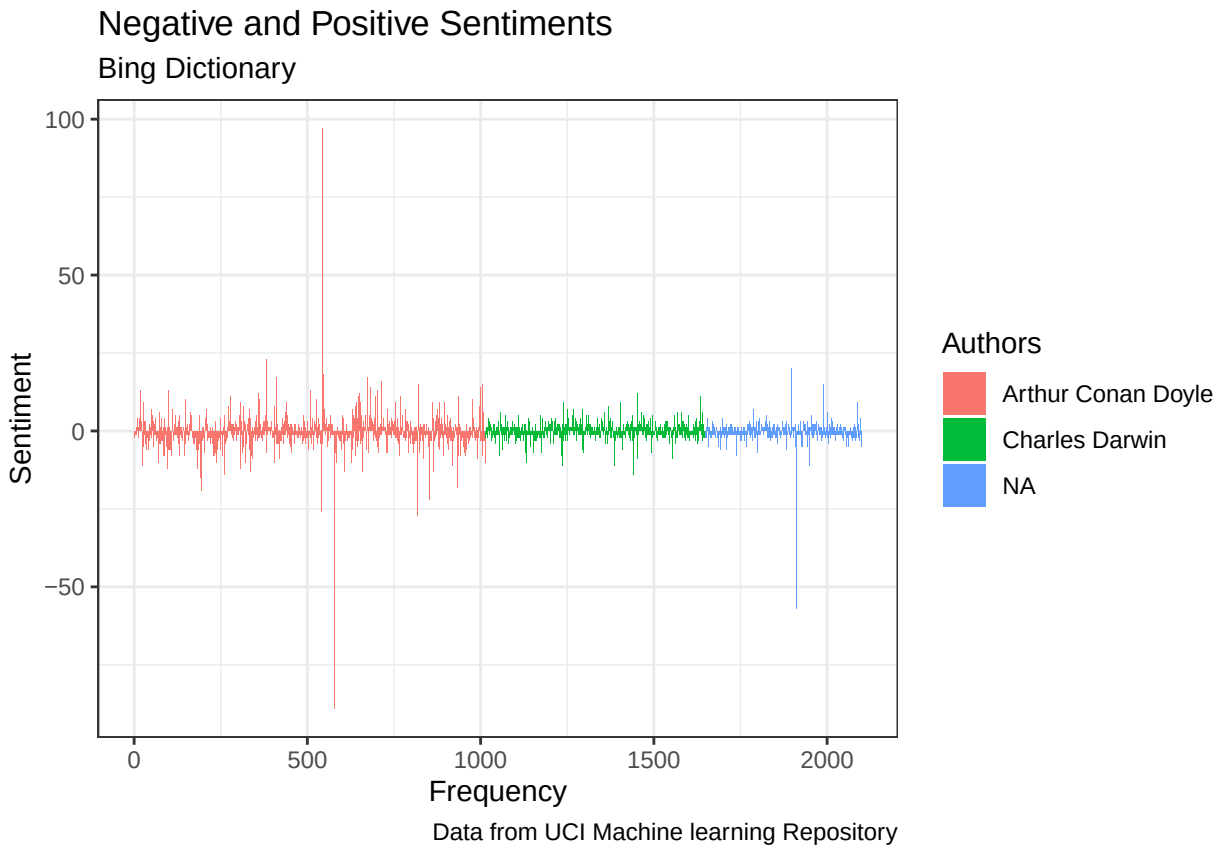
```
# Positive sentiment analysis for charles darwin
top_words %>%
  filter(label == 2) %>%
  semi_join(positive, by = c("w_tokens" = "word"))
```

```
## # A tibble: 299 x 3
## # Groups:   w_tokens [299]
##    w_tokens  label  freq
##    <chr>     <fct> <int>
##  1 pretty    2        12
##  2 won       2        11
##  3 fine      2         9
##  4 noble     2         9
##  5 love      2         8
##  6 nobly     2         8
##  7 free      2         7
##  8 glad      2         7
##  9 happy     2         7
## 10 beautiful 2         6
## # ... with 289 more rows
```

```
# Sentiment score for words
bing <- get_sentiments(lexicon = "bing")
bing <- tibble::rowid_to_column(bing, "id")
bing_s <- sent_tokens %>%
  group_by(label) %>%
  inner_join(bing, by = c("w_tokens" = "word")) %>%
  count(w_tokens,  sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)
bing_s <- tibble::rowid_to_column(bing_s, "id")

# Using the bing dictionary to compare top 50 words negative and positive sentiments
set.seed(12456)
bing_s <- shuffleRows(bing_s)
ggplot(bing_s[1:50,], aes(x = w_tokens, y = sentiment, fill = label)) +
  coord_flip() +
  geom_col(position = "stack", show.legend = F) +
  theme_bw() +
  labs(y = "Frequency", x = "Word", title="Sentiments: Bing Dictionary",
       subtitle = "Negative and Positive Words",
       caption = "Data from UCI Machine learning Repository")
```

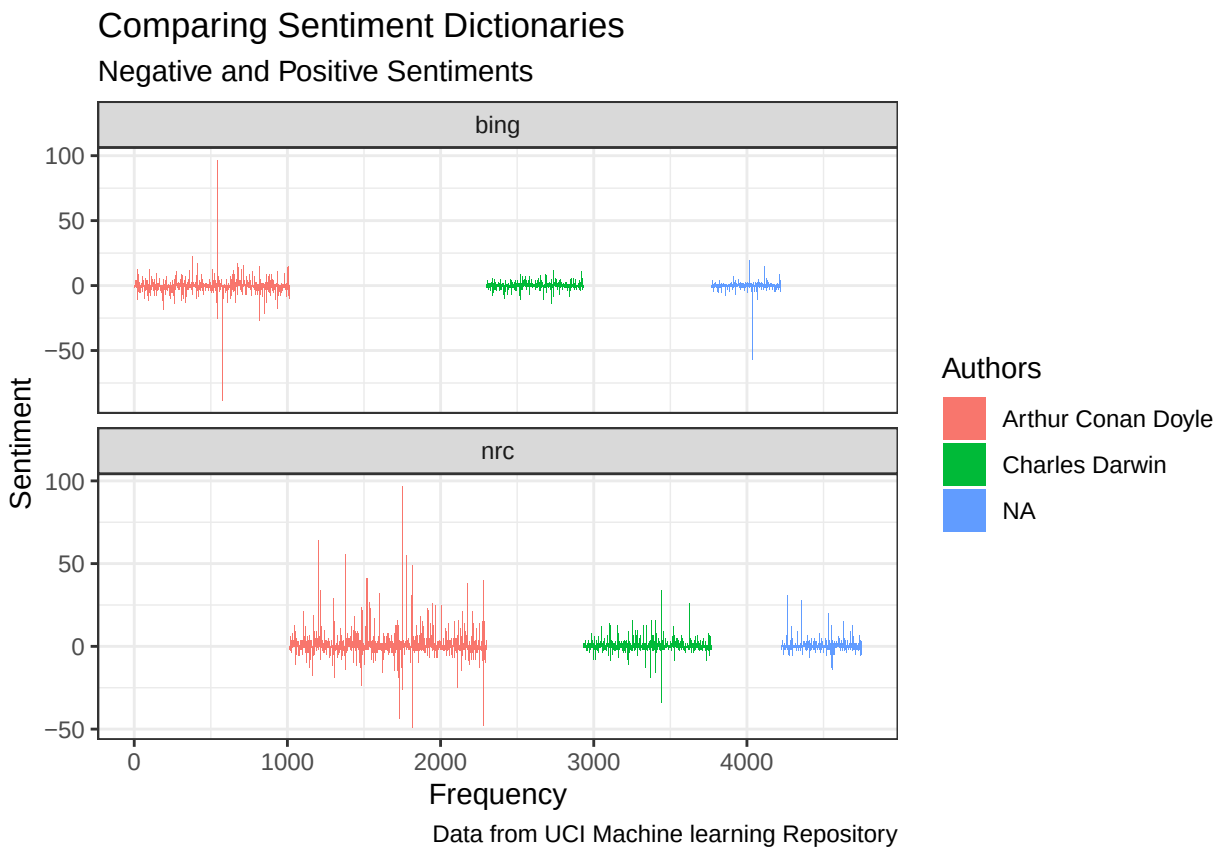



```
# Using the bing dictionary to compare the negative and positive sentiments author's words
ggplot(bing_s, aes(x = id, y = sentiment, fill = label)) +
  geom_col() +
  theme_bw() +
  labs(y = "Sentiment", x = "Frequency",
       title="Negative and Positive Sentiments",
       subtitle = "Bing Dictionary",
       caption = "Data from UCI Machine learning Repository") +
  scale_fill_discrete(name = "Authors",
                      labels = c("Arthur Conan Doyle", "Charles Darwin"))
```



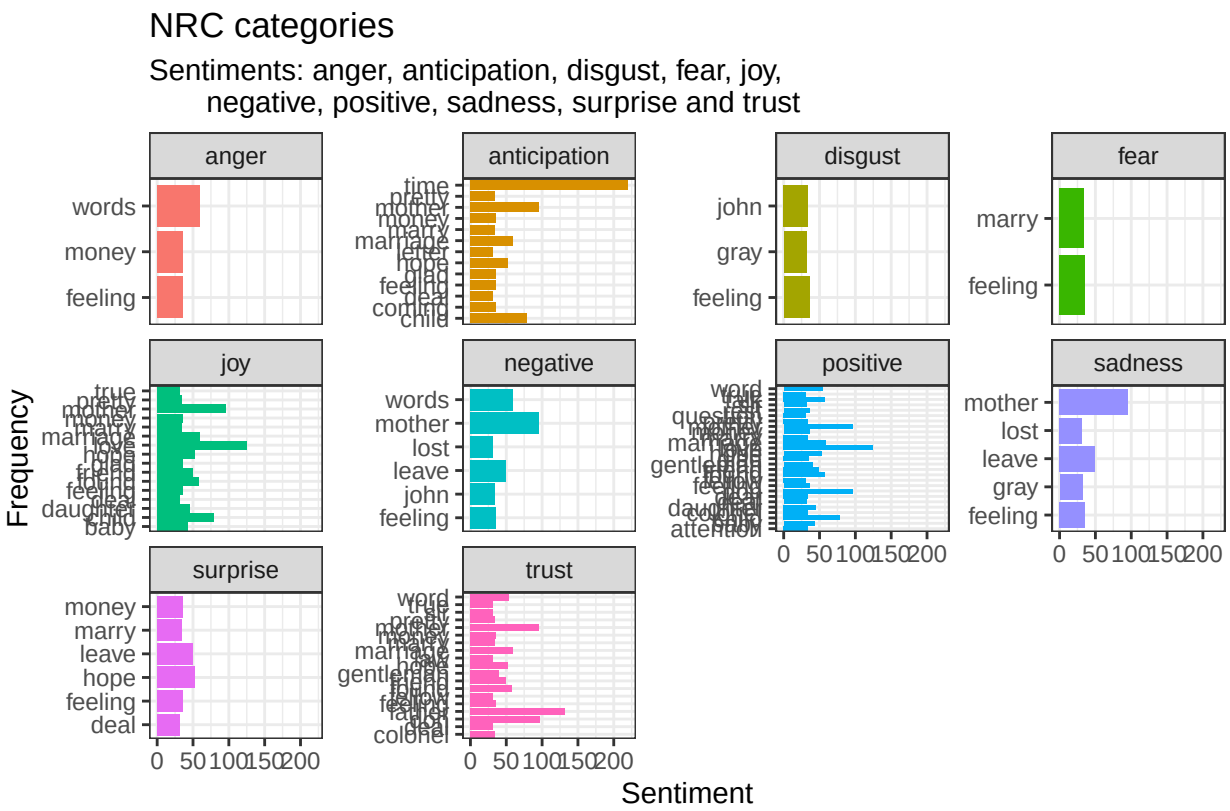

```
# Comparing sentiment dictionaries for differences in categorization of sentiments
afinn <- get_sentiments(lexicon = "afinn")
afinn <- sent_tokens %>%
  inner_join(afinn, by = c("w_tokens" = "word")) %>%
  mutate(method = "afinn") %>%
  rename(sentiment = value)
afinn <- tibble::rowid_to_column(afinn, "id")

# nrc dictionary
nrc <- get_sentiments(lexicon = "nrc")
nrc <- tibble::rowid_to_column(nrc, "id")

# bing and nrc dictionaries combined
bing_and_nrc <- bind_rows(sent_tokens %>%
                            group_by(label) %>%
                            inner_join(bing, by = c("w_tokens" = "word")) %>%
                            mutate(method = "bing"), sent_tokens %>%
                             group_by(label) %>%
                            inner_join(nrc, by = c("w_tokens" = "word")) %>%
                            filter(sentiment %in% c("positive", "negative")) %>%
                            mutate(method = "nrc")) %>%
  count(method, id, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)


bing_and_nrc <- bing_and_nrc %>% select(-id)
```

```
bing_and_nrc <- tibble::rowid_to_column(bing_and_nrc, "id")
bing_and_nrc <- shuffleRows(bing_and_nrc)

ggplot(bing_and_nrc, aes(x = id, y = sentiment, fill = label)) +
  geom_col() +
  theme_bw() +
  labs(y = "Sentiment", x = "Frequency",
       title="Comparing Sentiment Dictionaries",
       subtitle = "Negative and Positive Sentiments",
       caption = "Data from UCI Machine learning Repository") +
  scale_fill_discrete(name = "Authors",
                      labels = c("Arthur Conan Doyle", "Charles Darwin")) +
  facet_wrap(~method, ncol = 1, scales = "free_y")
```

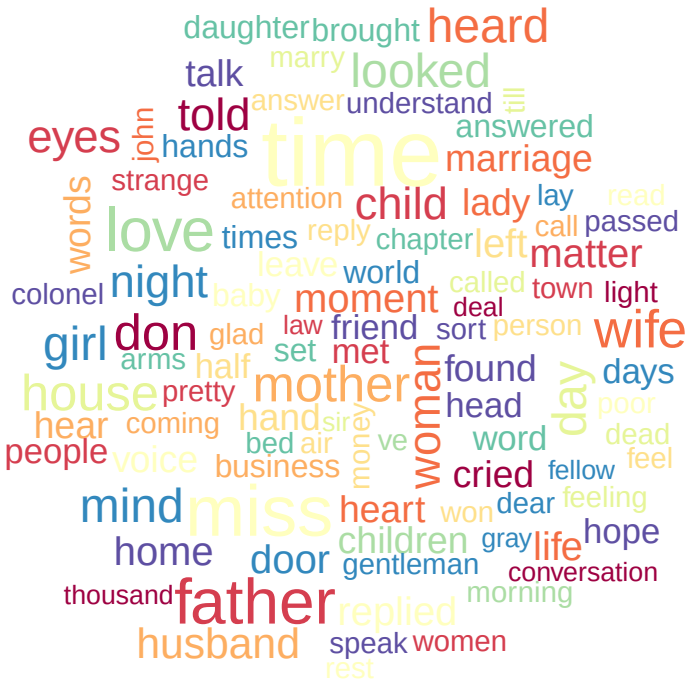



```
# nrc
nrc_counts <- sent_tokens %>%
  inner_join(nrc, by = c("w_tokens" = "word")) %>%
  count(w_tokens, sentiment, sort = T) %>%
  ungroup()

# nrc sentiment categories
ggplot(nrc_counts[1:100,], aes(x = w_tokens, y = n, fill = sentiment))  +
  geom_col(show.legend = F) +
  theme_bw() +
  labs(y = "Sentiment", x = "Frequency", title="NRC categories",
       subtitle = "Sentiments: anger, anticipation, disgust, fear, joy,
       negative, positive, sadness, surprise and trust",
```

```
      caption = "Data from UCI Machine learning Repository") +
  facet_wrap(~sentiment, scales = "free_y") + coord_flip()
```




```
# word cloud plot 1
sent_tokens %>%
  count(w_tokens) %>%
  with(wordcloud(w_tokens, n, max.words = 100, random.order=TRUE,
                 random.color = TRUE, rot.per=0.1, scale = c(3.0,0.5),
                 colors = brewer.pal(11, "Spectral")))
```




```
# Analysing shingles
# ngram = 2; seperate filter and combine individual stopwords
ngram_2 <- function(sent_df, shingles, text, shingle1, stop_words,
                    shingle2, num, label) {
  ngram <- unnest_tokens(sent_df, shingles, text, token = "ngrams", n = num)
  bi_sep <<- ngram %>%
    separate(shingles, c("shingle1", "shingle2"), sep = " ",
             remove = T, convert = F)
  bi_fil <- bi_sep %>%
    filter(!shingle1 %in% stop_words$word) %>%
    filter(!shingle2 %in% stop_words$word)
  bi_count <- bi_fil %>% group_by(label) %>%
    count(shingle1, shingle2, sort = T, name = "total")
  bigram <- bi_fil %>%
    unite(shingles, shingle1, shingle2, sep = " ", remove = T)
  bigram %>% group_by(label) %>%
    count(shingles, sort = T, name = "freq")
}

ngram_2(sent_df, shingles, text, shingle1, stop_words, shingle2, num = 2, label)
```

```
## # A tibble: 4,625 x 3
## # Groups:   label [3]
##    label shingles        freq
##    <fct> <chr>          <int>
##  1 1     marriage bond     21
##  2 1     sugar princess    20
##  3 2     ic chronicle      10
##  4 1     book keeper        7
##  5 1     miss giddy         7
##  6 2     noble purpose      7
##  7 2     nobly won          7
```

```
##  8 3     dearest nay        7
##  9 3     miss skin          7
## 10 3     nay dearest        7
## # ... with 4,615 more rows
```

```
# ngrams = 3. # not alot going here.
ngram_3 <- function(sent_df, shingles, text, shingle1, stop_words, shingle2,
                    shingle3, shingle4, num, label) {
  ngram <- unnest_tokens(sent_df, shingles, text, token = "ngrams", n = num)
  sep <- ngram %>%
    separate(shingles, c("shingle1", "shingle2", "shingle3", "shingle4"),
             sep = " ", remove = T, convert = F)
  fil <- sep %>%
    filter(!shingle1 %in% stop_words$word) %>%
    filter(!shingle2 %in% stop_words$word) %>%
    filter(!shingle3 %in% stop_words$word) %>%
    filter(!shingle4 %in% stop_words$word)
  ngram <- fil %>%
    unite(shingles, shingle1, shingle2, shingle3, shingle4, sep = " ",
          remove = T, na.rm = T)
  ngram %>%
    group_by(label) %>%
    count(shingles, sort = T, name = "freq")
}

ngram_3(sent_df, shingles, text, shingle1, stop_words, shingle2,
        shingle3, shingle4, num = 3, label)
```

```
## Warning: Expected 4 pieces. Missing pieces filled with `NA` in 99748 rows [1, 2,
## 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```
## # A tibble: 753 x 3
## # Groups:   label [3]
##    label shingles                        freq
##    <fct> <chr>                          <int>
##  1 3     nay dearest nay                    7
##  2 1     hundred thousand dollars           2
##  3 1     sugar princess chapter             2
##  4 3     deep drawn sigh                    2
##  5 1     acquainted miss confessed          1
##  6 1     african experience looked          1
##  7 1     answers refer wholly               1
##  8 1     apparently senseless form          1
##  9 1     approaching dinner overpowered     1
## 10 1     arm belonged feeling               1
## # ... with 743 more rows
```

```
# Remeber from our previous sentiment anlysis, the word "mother" seem to have
# huge importance in positive and negative sentiments
# lets see what that is all about in a bigram settings

bi_sep %>%
  filter(shingle2 == "mother") %>%
  count(shingle1, shingle2, sort = TRUE)
```

```
## # A tibble: 30 x 3
```

```
##    shingle1 shingle2     n
##    <chr>    <chr>    <int>
##  1 her      mother      19
##  2 the      mother      12
##  3 my       mother       9
##  4 his      mother       8
##  5 and      mother       7
##  6 a        mother       6
##  7 your     mother       5
##  8 oh       mother       3
##  9 s        mother       3
## 10 dead     mother       2
## # ... with 20 more rows
```

```
s_girl <- bi_sep %>%
  filter(shingle2 == "girl") %>%
  inner_join(nrc, by = c("shingle1" = "word")) %>%
  count(shingle1, shingle2, sentiment, sort = TRUE)

s_girl
```

```
## # A tibble: 38 x 4
##    shingle1 shingle2 sentiment        n
##    <chr>    <chr>    <chr>        <int>
##  1 blind    girl     negative         2
##  2 young    girl     anticipation     2
##  3 young    girl     joy              2
##  4 young    girl     positive         2
##  5 young    girl     surprise         2
##  6 baby     girl     joy              1
##  7 baby     girl     positive         1
##  8 fair     girl     positive         1
##  9 good     girl     anticipation     1
## 10 good     girl     joy              1
## # ... with 28 more rows
```

```
# igraph df - using the full dataset
igraph <- function(df, shingles, text, shingle1, stop_words, shingle2, label) {
  igraph_df <- as_tibble(df, c("text", "label"))
  ngram <- unnest_tokens(igraph_df, shingles, text, token = "ngrams", n = 2)
  bi_sep <- ngram %>%
    separate(shingles, c("shingle1", "shingle2"), sep = " ",
             remove = T, convert = F)
  bi_fil <- bi_sep %>%
    filter(!shingle1 %in% stop_words$word) %>%
    filter(!shingle2 %in% stop_words$word)
  igraph_c <- bi_fil %>% group_by(label) %>%
    count(shingle1, shingle2, sort = T, name = "total")
}

igraph_count <- igraph(df, shingles, text, shingle1, stop_words, shingle2, label)

# top combinations: igrpah counts + data frame to plot
igraph_pp <- function(igraph_count, total) {
  igraph_pp <- igraph_count %>%
```

```
    filter(total > 10) %>%
    graph_from_data_frame()
}
igraph_pp <- igraph_pp(igraph_count, total)

# network graph
net_g <- function(grid, arrow, igraph_pp, total, name) {
  set.seed(0988)
  ar <- grid::arrow(type = "closed", length = unit(.15, "inches"))
  ggraph(igraph_pp, layout = "fr") +
    geom_edge_link(aes(edge_alpha = total, edge_width = total),
                   show.legend = FALSE, arrow = ar, edge_colour = "cyan4",
                   end_cap = circle(.07, 'inches')) +
    geom_node_point(color = "red", size = 3) +
    geom_node_text(aes(label = name), repel = T,
                   point.padding = unit(0.2, "lines")) +
    theme_void() + ggtitle("Word Network")
}

net_g(grid, arrow, igraph_pp, total, name)
```

Word Network