SentAnalysisData607

Mathew Katz

2022-11-06

Firstly I am going to post a bunch of code from Text Mining with R, Chapter 2 (Sentiment Analysis.) {https://www.tidytextmining.com/sentiment.html}

```
library(tidytext)
get_sentiments("afinn")
```

```
## # A tibble: 2,477 x 2
##    word       value
##    <chr>      <dbl>
##  1 abandon       -2
##  2 abandoned     -2
##  3 abandons      -2
##  4 abducted      -2
##  5 abduction     -2
##  6 abductions    -2
##  7 abhor         -3
##  8 abhorred      -3
##  9 abhorrent     -3
## 10 abhors        -3
## # ... with 2,467 more rows
```

```
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
```

```
get_sentiments("nrc")
```

```
## # A tibble: 13,872 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 abacus      trust
##  2 abandon     fear
##  3 abandon     negative
##  4 abandon     sadness
##  5 abandoned   anger
##  6 abandoned   fear
##  7 abandoned   negative
##  8 abandoned   sadness
##  9 abandonment anger
## 10 abandonment fear
## # ... with 13,862 more rows
```

```r
library(janeaustenr)
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(stringr)

tidy_books <- austen_books() %>%
  group_by(book) %>%
  mutate(
    linenumber = row_number(),
    chapter = cumsum(str_detect(text,
                                regex("^chapter [\\divxlc]",
                                      ignore_case = TRUE)))) %>%
  ungroup() %>%
  unnest_tokens(word, text)
```

```r
nrc_joy <- get_sentiments("nrc") %>%
  filter(sentiment == "joy")

tidy_books %>%
  filter(book == "Emma") %>%
  inner_join(nrc_joy) %>%
  count(word, sort = TRUE)
```

```
## Joining, by = "word"
## # A tibble: 301 x 2
##    word          n
##    <chr>     <int>
##  1 good        359
##  2 friend      166
##  3 hope        143
##  4 happy       125
##  5 love        117
##  6 deal         92
##  7 found        92
##  8 present      89
##  9 kind         82
## 10 happiness    76
## # ... with 291 more rows
```

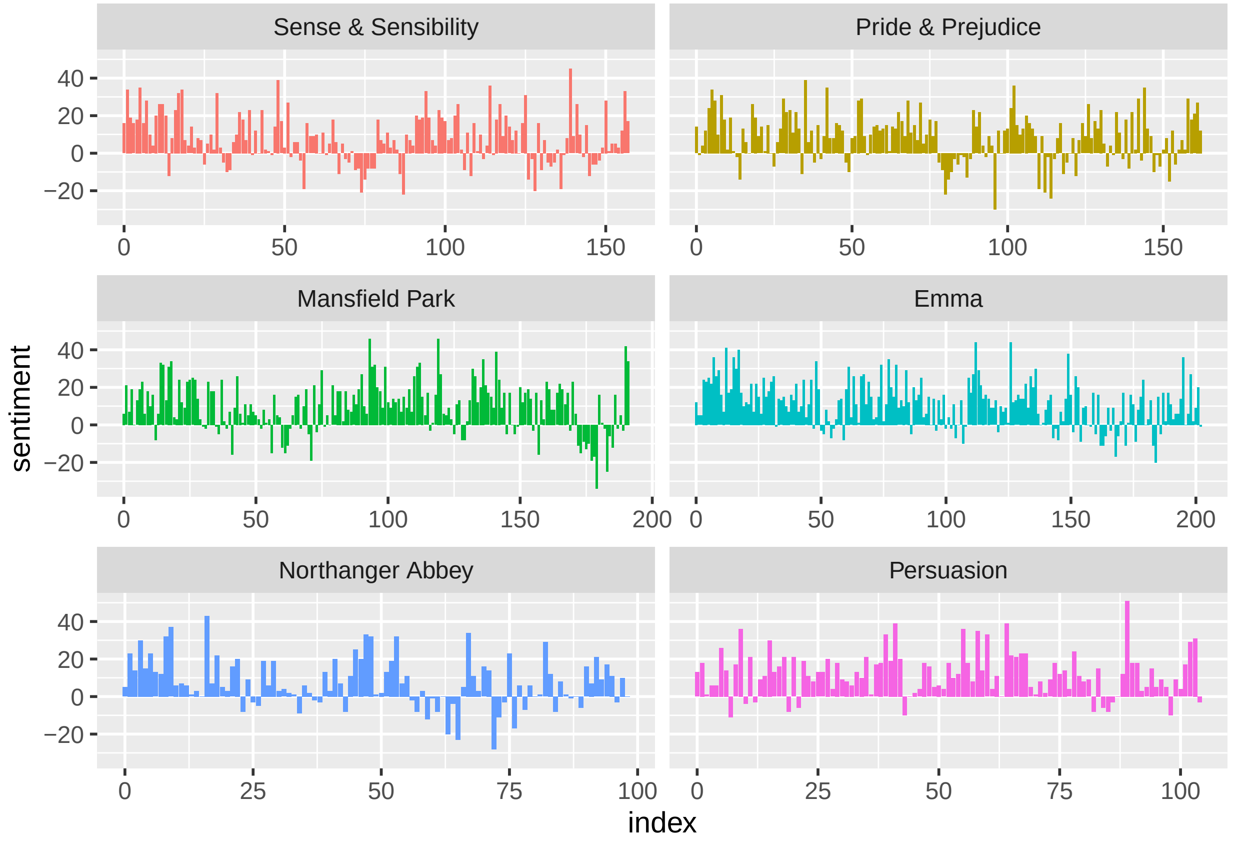
```
library(tidyr)

jane_austen_sentiment <- tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(book, index = linenumber %/% 80, sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(sentiment = positive - negative)
```

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

```
library(ggplot2)

ggplot(jane_austen_sentiment, aes(index, sentiment, fill = book)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~book, ncol = 2, scales = "free_x")
```



```
pride_prejudice <- tidy_books %>%
  filter(book == "Pride & Prejudice")

pride_prejudice
```

```
## # A tibble: 122,204 x 4
##    book              linenumber chapter word
##    <fct>                  <int>   <int> <chr>
##  1 Pride & Prejudice          1       0 pride
##  2 Pride & Prejudice          1       0 and
##  3 Pride & Prejudice          1       0 prejudice
##  4 Pride & Prejudice          3       0 by
##  5 Pride & Prejudice          3       0 jane
##  6 Pride & Prejudice          3       0 austen
##  7 Pride & Prejudice          7       1 chapter
##  8 Pride & Prejudice          7       1 1
##  9 Pride & Prejudice         10       1 it
## 10 Pride & Prejudice         10       1 is
## # ... with 122,194 more rows
```

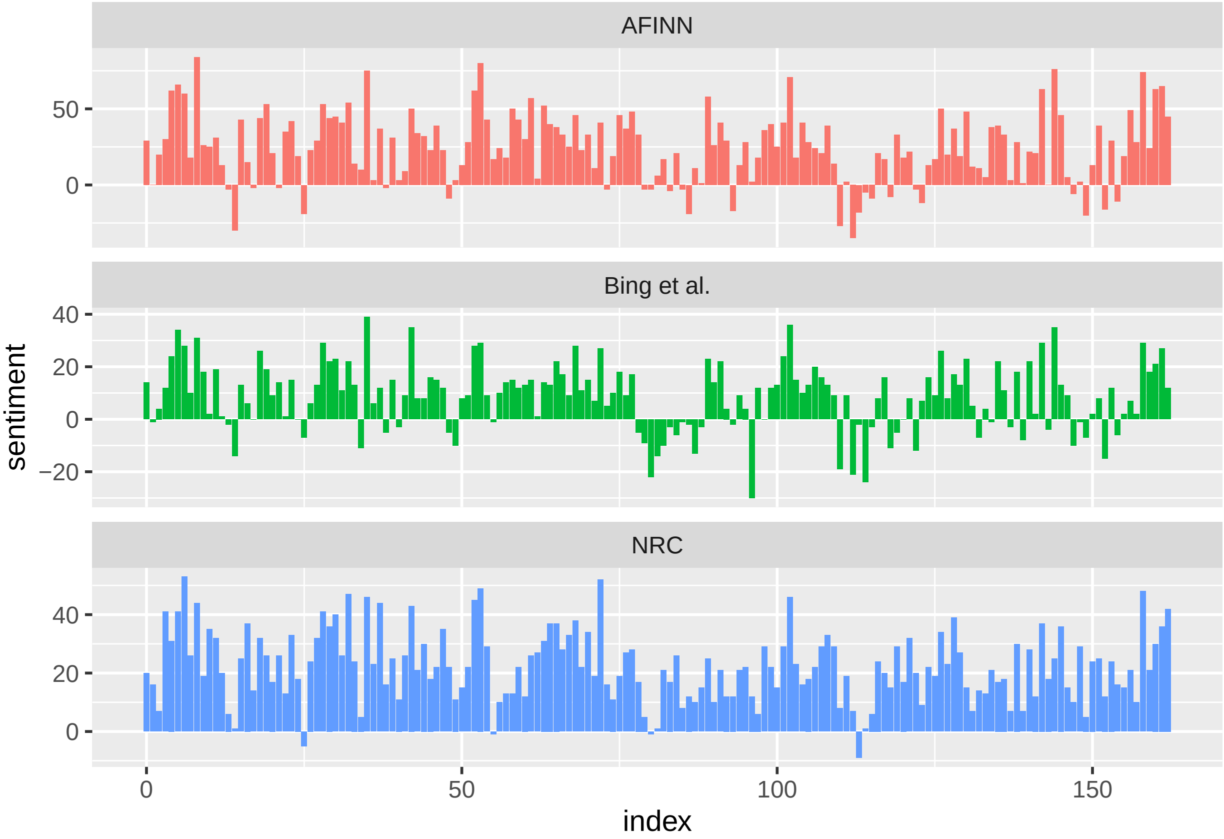
```
afinn <- pride_prejudice %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(index = linenumber %/% 80) %>%
  summarise(sentiment = sum(value)) %>%
  mutate(method = "AFINN")
```

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

```
bing_and_nrc <- bind_rows(
  pride_prejudice %>%
    inner_join(get_sentiments("bing")) %>%
    mutate(method = "Bing et al."),
  pride_prejudice %>%
    inner_join(get_sentiments("nrc") %>%
                 filter(sentiment %in% c("positive",
                                         "negative"))
    ) %>%
    mutate(method = "NRC")) %>%
  count(method, index = linenumber %/% 80, sentiment) %>%
  pivot_wider(names_from = sentiment,
              values_from = n,
              values_fill = 0) %>%
  mutate(sentiment = positive - negative)
```

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

```
bind_rows(afinn,
          bing_and_nrc) %>%
  ggplot(aes(index, sentiment, fill = method)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~method, ncol = 1, scales = "free_y")
```



```
get_sentiments("nrc") %>%
  filter(sentiment %in% c("positive", "negative")) %>%
  count(sentiment)
```

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative   3316
## 2 positive   2308
```

```
get_sentiments("bing") %>%
  count(sentiment)
```

```
## # A tibble: 2 x 2
##   sentiment     n
##   <chr>     <int>
## 1 negative   4781
## 2 positive   2005
```

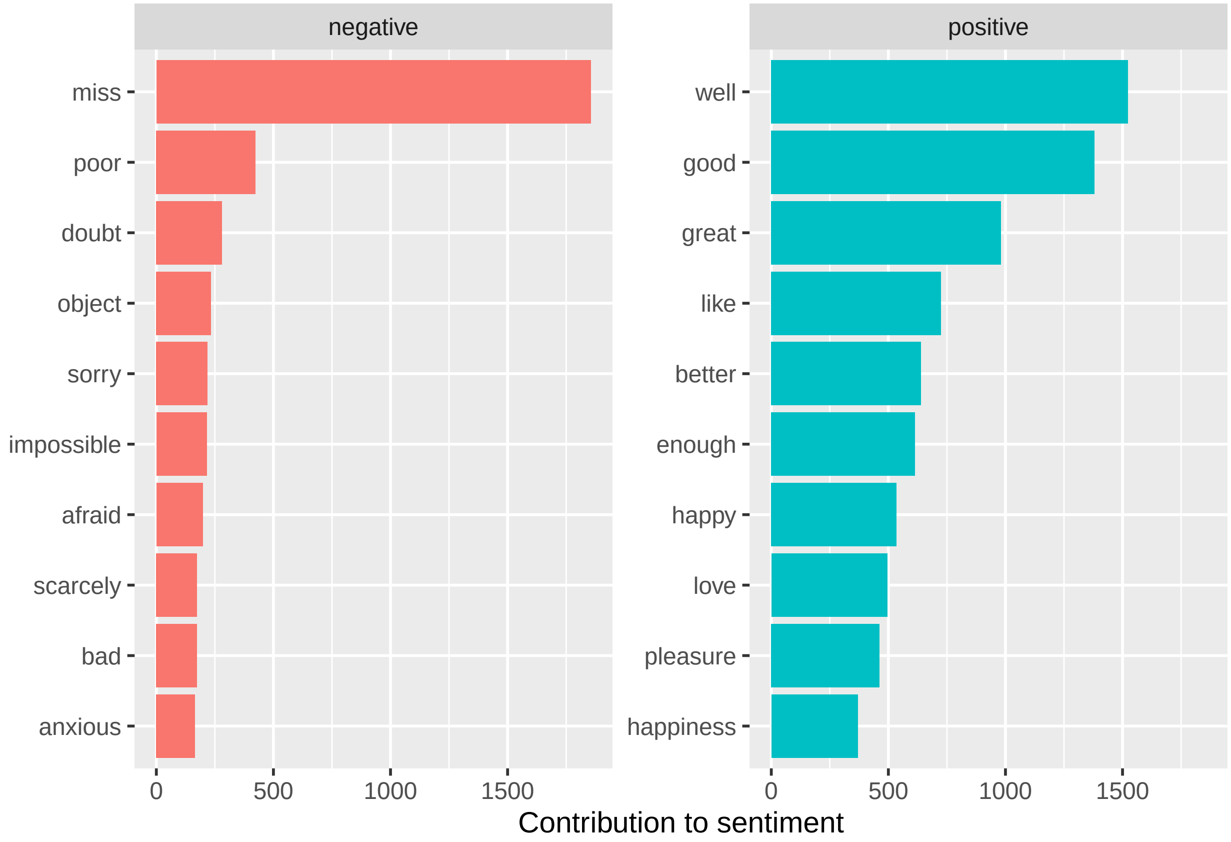
```
bing_word_counts <- tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

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

```
bing_word_counts
```

```
## # A tibble: 2,585 x 3
##    word     sentiment     n
##    <chr>    <chr>     <int>
##  1 miss     negative   1855
##  2 well     positive   1523
##  3 good     positive   1380
##  4 great    positive    981
##  5 like     positive    725
##  6 better   positive    639
##  7 enough   positive    613
##  8 happy    positive    534
##  9 love     positive    495
## 10 pleasure positive    462
## # ... with 2,575 more rows
```

```
bing_word_counts %>%
  group_by(sentiment) %>%
  slice_max(n, n = 10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(x = "Contribution to sentiment",
       y = NULL)
```



```
custom_stop_words <- bind_rows(tibble(word = c("miss"),
                                      lexicon = c("custom")),
                               stop_words)

custom_stop_words
```

```
## # A tibble: 1,150 x 2
##    word        lexicon
##    <chr>       <chr>
##  1 miss        custom
##  2 a           SMART
##  3 a's         SMART
##  4 able        SMART
##  5 about       SMART
##  6 above       SMART
##  7 according   SMART
##  8 accordingly SMART
##  9 across      SMART
## 10 actually    SMART
## # ... with 1,140 more rows
```

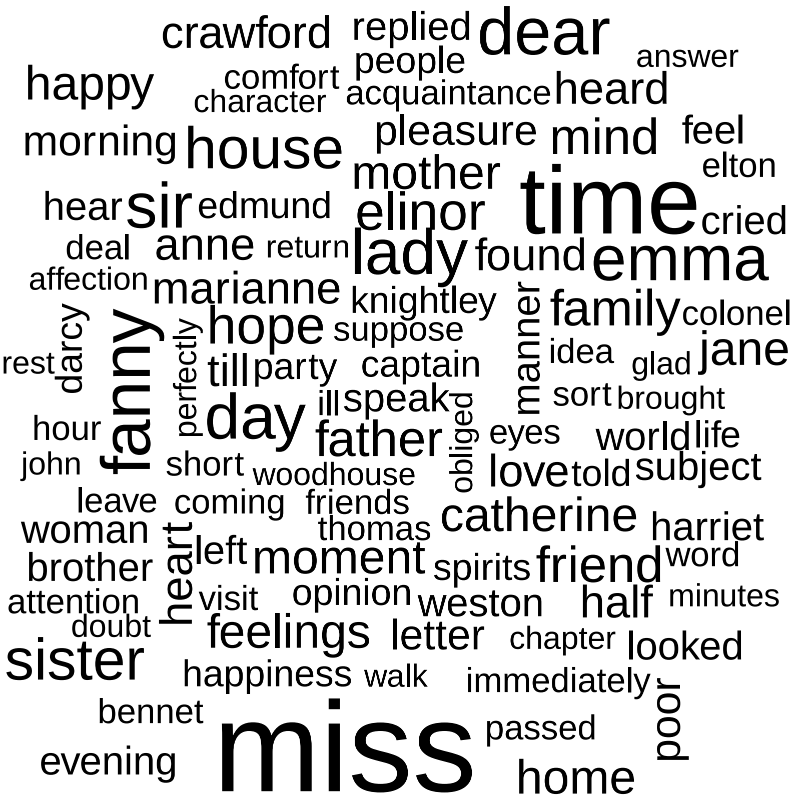
```
library(wordcloud)
```

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

```
tidy_books %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100))
```

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

```
## Warning in wordcloud(word, n, max.words = 100): elizabeth could not be fit on
## page. It will not be plotted.
```



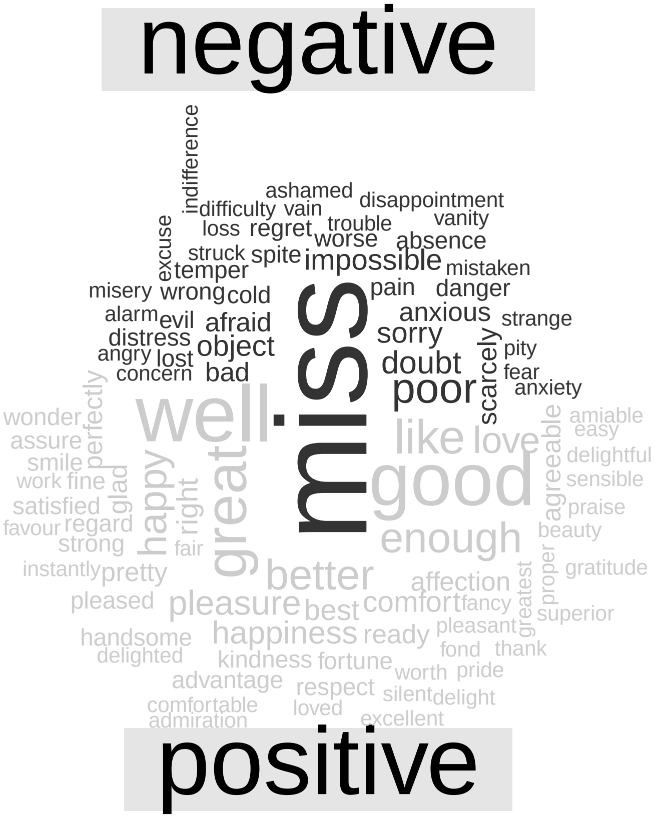
```
library(reshape2)
```

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

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

```
tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("gray20", "gray80"),
                   max.words = 100)
```

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



```
p_and_p_sentences <- tibble(text = prideprejudice) %>%
  unnest_tokens(sentence, text, token = "sentences")
```

```
p_and_p_sentences$sentence[2]
```

```
## [1] "by jane austen"
```

```
austen_chapters <- austen_books() %>%
  group_by(book) %>%
  unnest_tokens(chapter, text, token = "regex",
                pattern = "Chapter|CHAPTER [\\dIVXLC]") %>%
  ungroup()

austen_chapters %>%
  group_by(book) %>%
  summarise(chapters = n())
```

```
## # A tibble: 6 x 2
##   book                chapters
##   <fct>                  <int>
## 1 Sense & Sensibility       51
## 2 Pride & Prejudice         62
## 3 Mansfield Park              49
## 4 Emma                        56
## 5 Northanger Abbey            32
## 6 Persuasion                  25
```

```
bingnegative <- get_sentiments("bing") %>%
  filter(sentiment == "negative")

wordcounts <- tidy_books %>%
  group_by(book, chapter) %>%
  summarize(words = n())
```

```
## `summarise()` has grouped output by 'book'. You can override using the
## `.groups` argument.
```

```
tidy_books %>%
  semi_join(bingnegative) %>%
  group_by(book, chapter) %>%
  summarize(negativewords = n()) %>%
  left_join(wordcounts, by = c("book", "chapter")) %>%
  mutate(ratio = negativewords/words) %>%
  filter(chapter != 0) %>%
  slice_max(ratio, n = 1) %>%
  ungroup()
```

```
## Joining, by = "word"
## `summarise()` has grouped output by 'book'. You can override using the
## `.groups` argument.

## # A tibble: 6 x 5
##   book                chapter negativewords words  ratio
##   <fct>                 <int>         <int> <int>  <dbl>
## 1 Sense & Sensibility      43           161  3405 0.0473
## 2 Pride & Prejudice        34           111  2104 0.0528
## 3 Mansfield Park           46           173  3685 0.0469
## 4 Emma                     15           151  3340 0.0452
## 5 Northanger Abbey         21           149  2982 0.0500
## 6 Persuasion                4            62  1807 0.0343
```

Let's extend the code in two ways: Work with a different corpus of our choosing, and Incorporate at least one additional sentiment lexicon (possibly from another R package that I've found through research).

```
library('readr')
scripts <- read_csv("RickAndMortyScripts.csv")
```

```
## Rows: 1905 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): episode name, name, line
## dbl (3): index, season no., episode no.
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
nrc <- get_sentiments("nrc")
bing <- get_sentiments("bing")
afinn <- get_sentiments("afinn")
```

```
scripts = scripts %>% rename(Index = "index",
                   Season.No = "season no.",
                   Episode.No = "episode no.",
                   Episode.Name = "episode name",
                   Character.Name = "name",
                   Dialog = "line")
```

```
# Head of the table
head(scripts, 4)
```

```
## # A tibble: 4 x 6
##   Index Season.No Episode.No Episode.Name Character.Name Dialog
##   <dbl>     <dbl>      <dbl> <chr>        <chr>          <chr>
## 1     0         1          1 Pilot        Rick           Morty! You gotta come ~
## 2     1         1          1 Pilot        Morty          What, Rick? What's goi~
## 3     2         1          1 Pilot        Rick           I got a surprise for y~
## 4     3         1          1 Pilot        Morty          It's the middle of the~
```

```
# Tail of the table
tail(scripts, 4)
```

```
## # A tibble: 4 x 6
##   Index Season.No Episode.No Episode.Name           Character.Name Dialog
##   <dbl>     <dbl>      <dbl> <chr>                  <chr>          <chr>
## 1  2484         3          7 Tales From the Citadel Rick           Got some of ~
## 2  2485         3          7 Tales From the Citadel Morty          I'm really h~
## 3  2486         3          7 Tales From the Citadel Rick           Pssh! Not at~
## 4  2487         3          7 Tales From the Citadel Morty          Whoo! Yeah! ~
```

```
# Summary
summary(scripts)
```

```
##      Index        Season.No       Episode.No     Episode.Name
##  Min.   :   0   Min.   :1.000   Min.   : 1.000   Length:1905
##  1st Qu.: 548   1st Qu.:1.000   1st Qu.: 1.000   Class :character
##  Median :1164   Median :2.000   Median : 3.000   Mode  :character
##  Mean   :1190   Mean   :2.155   Mean   : 3.208
##  3rd Qu.:1844   3rd Qu.:3.000   3rd Qu.: 5.000
##  Max.   :2487   Max.   :3.000   Max.   :10.000
##  Character.Name        Dialog
##  Length:1905        Length:1905
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Clean Corpus Function: This predefined function is going to clean the text from:

the punctuation - removePunctuation extra white space - stripWhitespace transforms to lower case - tolower stopwords (common words that should be ignored) - stopwords numbers - removeNumbers

```
cleanCorpus <- function(text){
  # punctuation, whitespace, lowercase, numbers
  text.tmp <- tm_map(text, removePunctuation)
  text.tmp <- tm_map(text.tmp, stripWhitespace)
  text.tmp <- tm_map(text.tmp, content_transformer(tolower))
  text.tmp <- tm_map(text.tmp, removeNumbers)

  # removes stopwords
  stopwords_remove <- c(stopwords("en"), c("thats","weve","hes","theres","ive","im",
                                           "will","can","cant","dont","youve","us",
                                           "youre","youll","theyre","whats","didnt"))
  text.tmp <- tm_map(text.tmp, removeWords, stopwords_remove)

  return(text.tmp)
}
```

These predefined functions will process the text depending on the case:

Unigrams take only 1 word at a time Bigrams take 2 sequential words at a time Trigrams (you guessed) take 3 sequential words at a time Eg. text: "come on morty"

Unigram: "come", "on", "morty" Bigram: "come on", "on morty" Trigram: "come on morty" Term Document Matrix: it's a mathematical matrix that describes the frequency of terms that occur in a collection of documents. More simply put, is a matrix that has on:

rows - words that can be found in the analysed documents columns - the documents in order values - the frequency of each word in each document

Unigram:

```
frequentTerms <- function(text){

  # create the matrix
  s.cor <- VCorpus(VectorSource(text))
  s.cor.cl <- cleanCorpus(s.cor)
  s.tdm <- TermDocumentMatrix(s.cor.cl)
  s.tdm <- removeSparseTerms(s.tdm, 0.999)
  m <- as.matrix(s.tdm)
  word_freqs <- sort(rowSums(m), decreasing = T)

  # change to dataframe
  dm <- data.frame(word=names(word_freqs), freq=word_freqs)

  return(dm)
}
```

Bigram:

```
# Bigram tokenizer
tokenizer_2 <- function(x){
  NGramTokenizer(x, Weka_control(min=2, max=2))
```

```r
}

# Bigram function
frequentBigrams <- function(text){

  s.cor <- VCorpus(VectorSource(text))
  s.cor.cl <- cleanCorpus(s.cor)
  s.tdm <- TermDocumentMatrix(s.cor.cl, control=list(tokenize=tokenizer_2))
  s.tdm <- removeSparseTerms(s.tdm, 0.999)
  m <- as.matrix(s.tdm)
  word_freqs <- sort(rowSums(m), decreasing=T)
  dm <- data.frame(word=names(word_freqs), freq=word_freqs)

  return(dm)
}
```

Trigram:

```r
# Trigram tokenizer
tokenizer_3 <- function(x){
  NGramTokenizer(x, Weka_control(min=3, max=3))
}

# Trigram function
frequentTrigrams <- function(text){

  s.cor <- VCorpus(VectorSource(text))
  s.cor.cl <- cleanCorpus(s.cor)
  s.tdm <- TermDocumentMatrix(s.cor.cl, control=list(tokenize=tokenizer_3))
  s.tdm <- removeSparseTerms(s.tdm, 0.999)
  m <- as.matrix(s.tdm)
  word_freqs <- sort(rowSums(m), decreasing=T)
  dm <- data.frame(word=names(word_freqs), freq=word_freqs)

  return(dm)
}
```

Bing Lexicon cathegorizes the words into positives and negatives.

To be able to do so in our data, first we make a dataframe that splits all the words in 1 dialogue onto rows. Afterwards, we can join our data with the lexicon, leaving us with a beautiful classification of our words.

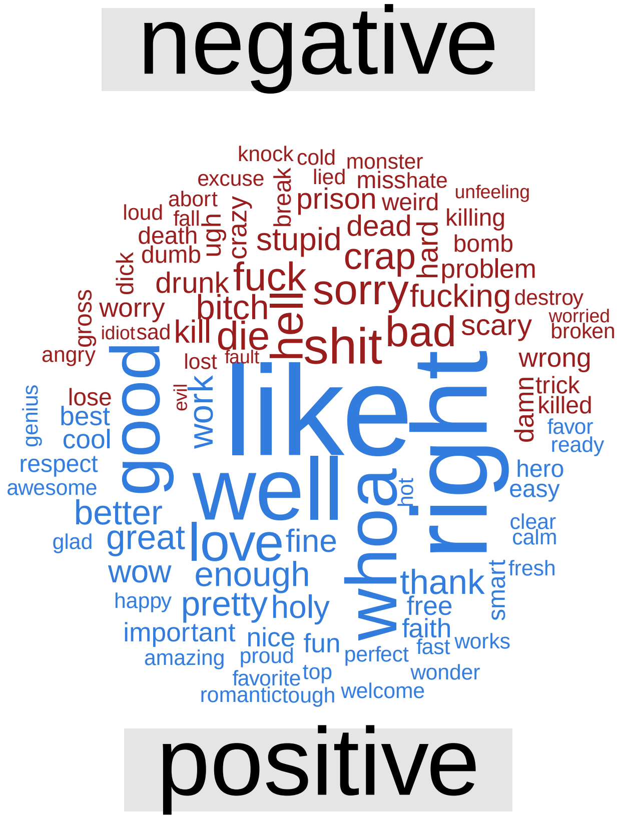
```r
# Creating our tokens
tokens <- scripts %>%
  mutate(dialogue = as.character(scripts$Dialog)) %>%
  unnest_tokens(word, dialogue)

tokens %>% head(5) %>% select(Character.Name, word)
```

```
## # A tibble: 5 x 2
##   Character.Name word
##   <chr>          <chr>
## 1 Rick           morty
## 2 Rick           you
## 3 Rick           gotta
## 4 Rick           come
## 5 Rick           on
```

```
tokens %>%
  # append the bing sentiment and prepare the data
  inner_join(bing, "word") %>%
  count(word, sentiment, sort=T) %>%
  acast(word ~ sentiment, value.var = "n", fill=0) %>%

  # wordcloud
  comparison.cloud(colors=c("#991D1D", "#327CDE"), max.words = 100)
```



How is the overall mood in Rick & Morty?

The nrc lexicon cathegorizes the words in 10 moods:

positive negative anger anticipation disgust fear joy sadness surprise trust Let's look at how these sentiments rank in out data:

```
sentiments <- tokens %>%
  inner_join(nrc, "word") %>%
  count(sentiment, sort=T)

sentiments
```

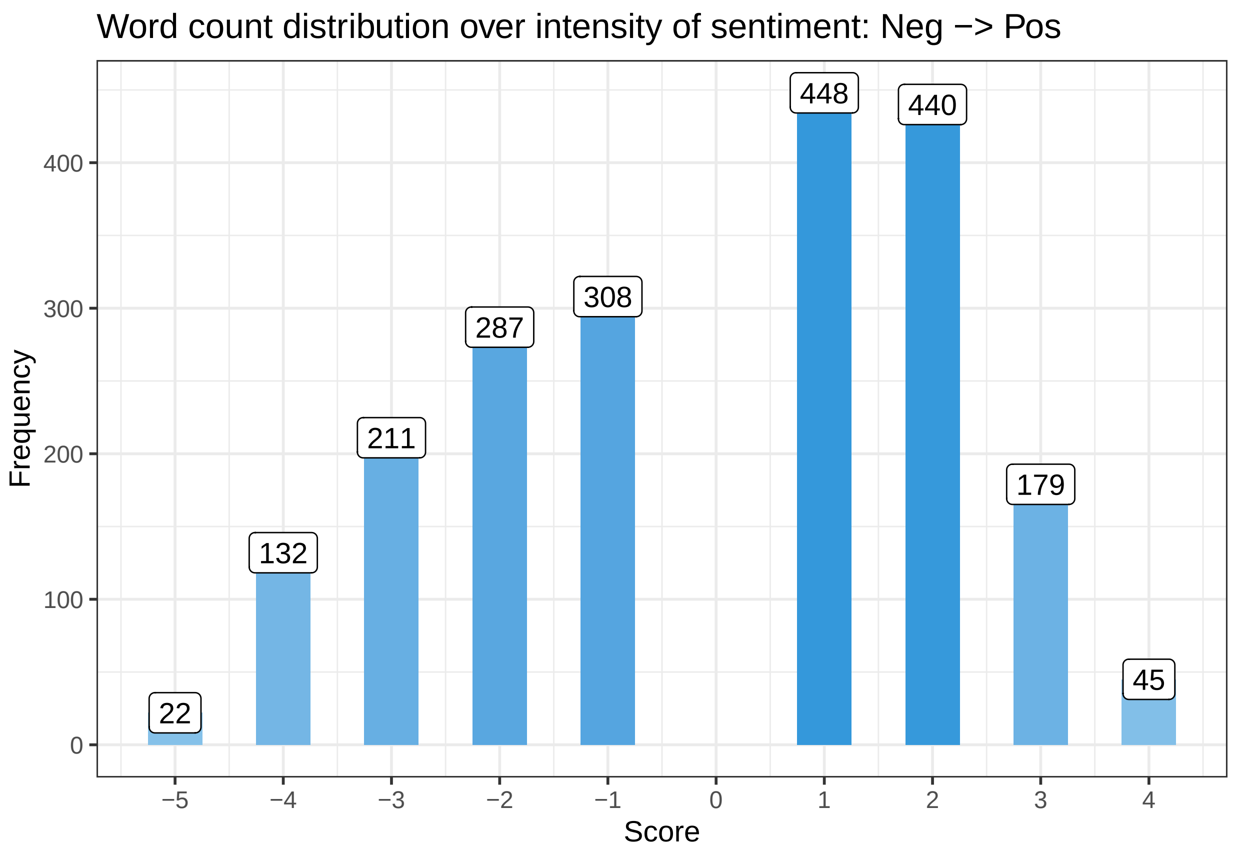
```
## # A tibble: 10 x 2
##    sentiment        n
##    <chr>        <int>
##  1 positive       977
##  2 negative       901
##  3 trust          645
##  4 anticipation   591
##  5 fear           567
##  6 joy            494
##  7 anger          415
##  8 sadness        414
##  9 disgust        312
## 10 surprise       266
```

Afinn Lexicon ranks every word from -5 to 5, where:

-5 being the most negative +5 being the most positive

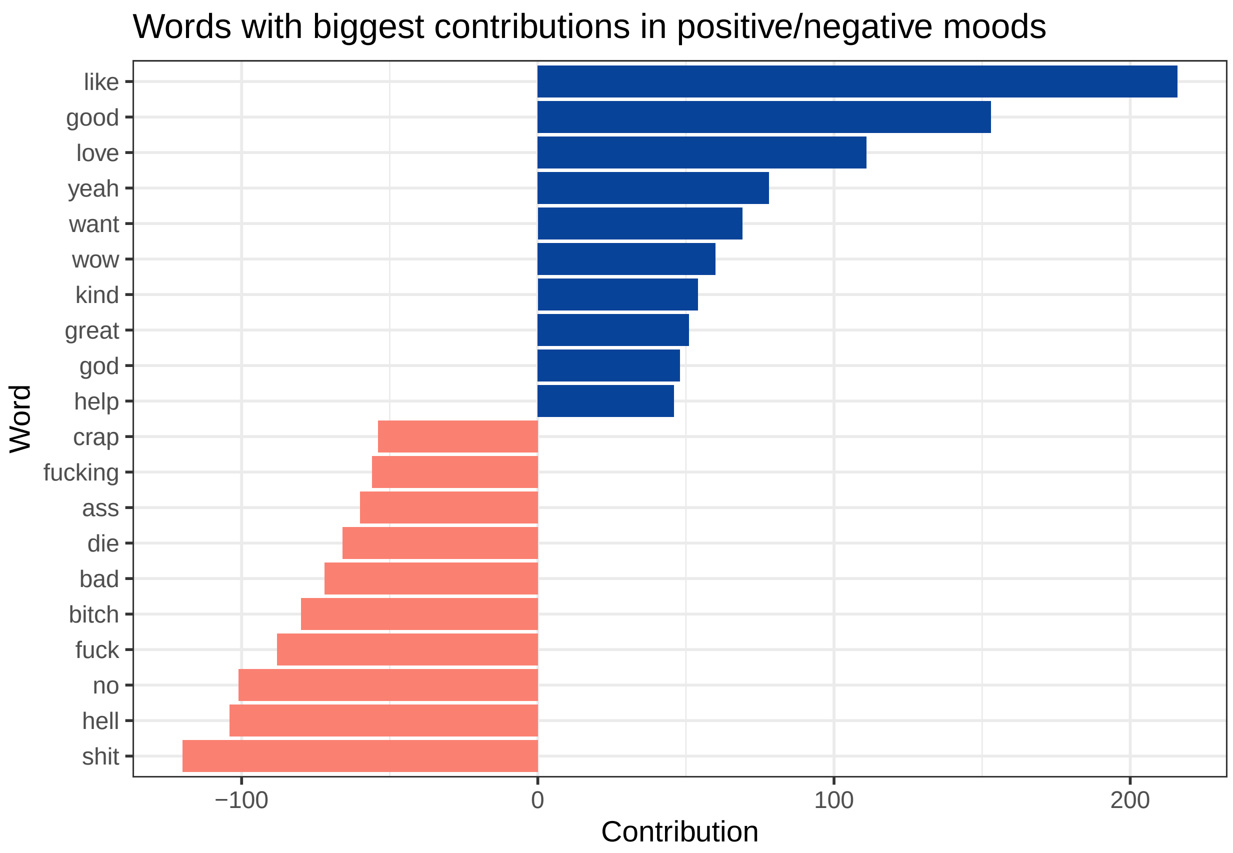
```
tokens %>%
  # Count how many word per value
  inner_join(afinn, "word") %>%
  count(value, sort=T) %>%

  # Plot
  ggplot(aes(x=value, y=n)) +
  geom_bar(stat="identity", aes(fill=n), show.legend = F, width = 0.5) +
  geom_label(aes(label=n)) +
  scale_fill_gradient(low="#85C1E9", high="#3498DB") +
  scale_x_continuous(breaks=seq(-5, 5, 1)) +
  labs(x="Score", y="Frequency", title="Word count distribution over intensity of sentiment: Neg -> Pos
  theme_bw()
```



```
tokens %>%
  # by word and value count number of occurences
  inner_join(afinn, "word") %>%
  count(word, value, sort=T) %>%
  mutate(contribution = n * value,
         sentiment = ifelse(contribution<=0, "Negative", "Positive")) %>% #another variable
  arrange(desc(abs(contribution))) %>%
  head(20) %>%

  # plot
  ggplot(aes(x=reorder(word, contribution), y=contribution, fill=sentiment)) +
  geom_col(aes(fill=sentiment), show.legend = F) +
  labs(x="Word", y="Contribution", title="Words with biggest contributions in positive/negative moods")
  coord_flip() +
  scale_fill_manual(values=c("#FA8072", "#08439A")) +
  theme_bw()
```



```
library("stopwords")

# Create a dataframe with stopwords
stopwords_script <- tibble(word = c(stopwords("en"), c("thats","weve","hes","theres","ive","im",
                                                           "will","can","cant","dont","youve","us",
                                                           "youre","youll","theyre","whats","didnt")))
print(stopwords_script)
```

```
## # A tibble: 192 x 1
##    word
##    <chr>
##  1 i
##  2 me
##  3 my
##  4 myself
##  5 we
##  6 our
##  7 ours
##  8 ourselves
##  9 you
## 10 your
## # ... with 182 more rows
```

```
# Create the dataframe of tokens
scripts %>%
  mutate(dialogue = as.character(scripts$Dialog)) %>%
  filter(Character.Name %in% c("Rick","Morty","Beth","Jerry","Summer")) %>%

  # removes stopwords
  unnest_tokens(word, dialogue) %>%
  anti_join(stopwords_script, by="word") %>%

  # top N frequent words per character
  count(Character.Name, word) %>%
  group_by(Character.Name) %>%
  arrange(desc(n)) %>%
  slice(1:10) %>%

  mutate(word2 = factor(paste(word, Character.Name, sep="__"),
                        levels = rev(paste(word, Character.Name, sep="__"))))
```

```
## # A tibble: 50 x 4
## # Groups:   Character.Name [5]
##    Character.Name word       n word2
##    <chr>          <chr>  <int> <fct>
##  1 Beth           jerry     22 jerry__Beth
##  2 Beth           dad       12 dad__Beth
##  3 Beth           oh        12 oh__Beth
##  4 Beth           summer    12 summer__Beth
##  5 Beth           know      11 know__Beth
##  6 Beth           morty     10 morty__Beth
##  7 Beth           want      10 want__Beth
##  8 Beth           like       9 like__Beth
##  9 Beth           mean       9 mean__Beth
## 10 Beth           get        8 get__Beth
## # ... with 40 more rows
```

Rick and Morty's script made this project very enjoyable.