Sentiment Analysis

David Moste

3/31/2020

Introduction

For this assignment, I am tasked with getting an example from *Text Mining with R* running and then extending the example to a new corpus and a neew sentiment lexicon. Sections 1-6 are directly from *Text Mining with R*[1]. Section 7 is my extension.

1. Sentiments Dataset

```r
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 3.6.3
```

```r
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
```

```r
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
```

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

```
## # A tibble: 13,901 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 13,891 more rows
```

2. Sentiment Analysis with Inner Join

```r
library(janeaustenr)
```

```
## Warning: package 'janeaustenr' was built under R version 3.6.3
```

```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(stringr)

tidy_books <- austen_books() %>%
  group_by(book) %>%
```

```r
  mutate(linenumber = row_number(),
         chapter = cumsum(str_detect(text, regex("^chapter [\\divxlc]",
                                                 ignore_case = TRUE)))) %>%
  ungroup() %>%
  unnest_tokens(word, text)

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

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

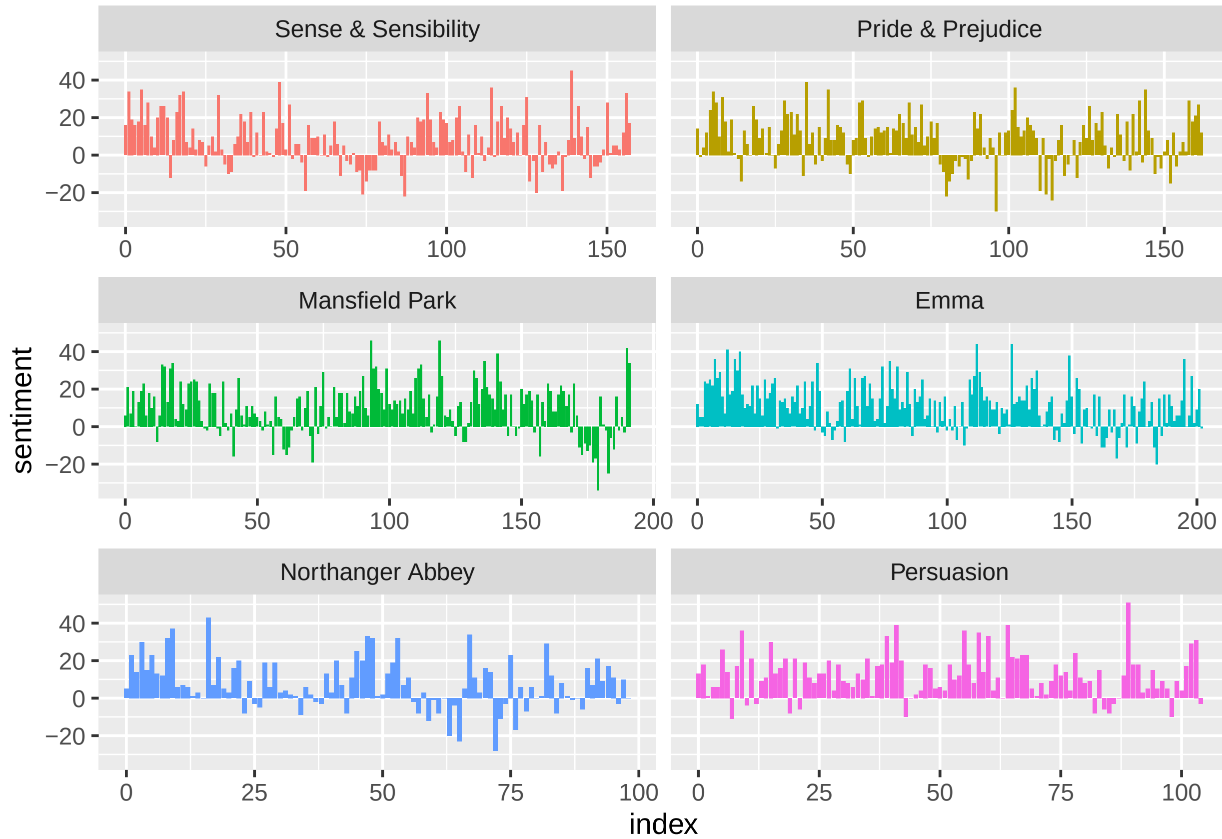
```
## Joining, by = "word"
```

```
## # A tibble: 303 x 2
##    word        n
##    <chr>   <int>
##  1 good      359
##  2 young     192
##  3 friend    166
##  4 hope      143
##  5 happy     125
##  6 love      117
##  7 deal       92
##  8 found      92
##  9 present    89
## 10 kind       82
## # ... with 293 more rows
```

```r
library(tidyr)

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

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

```r
library(ggplot2)

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



3. Comparing the Three Dictionaries

```r
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
```

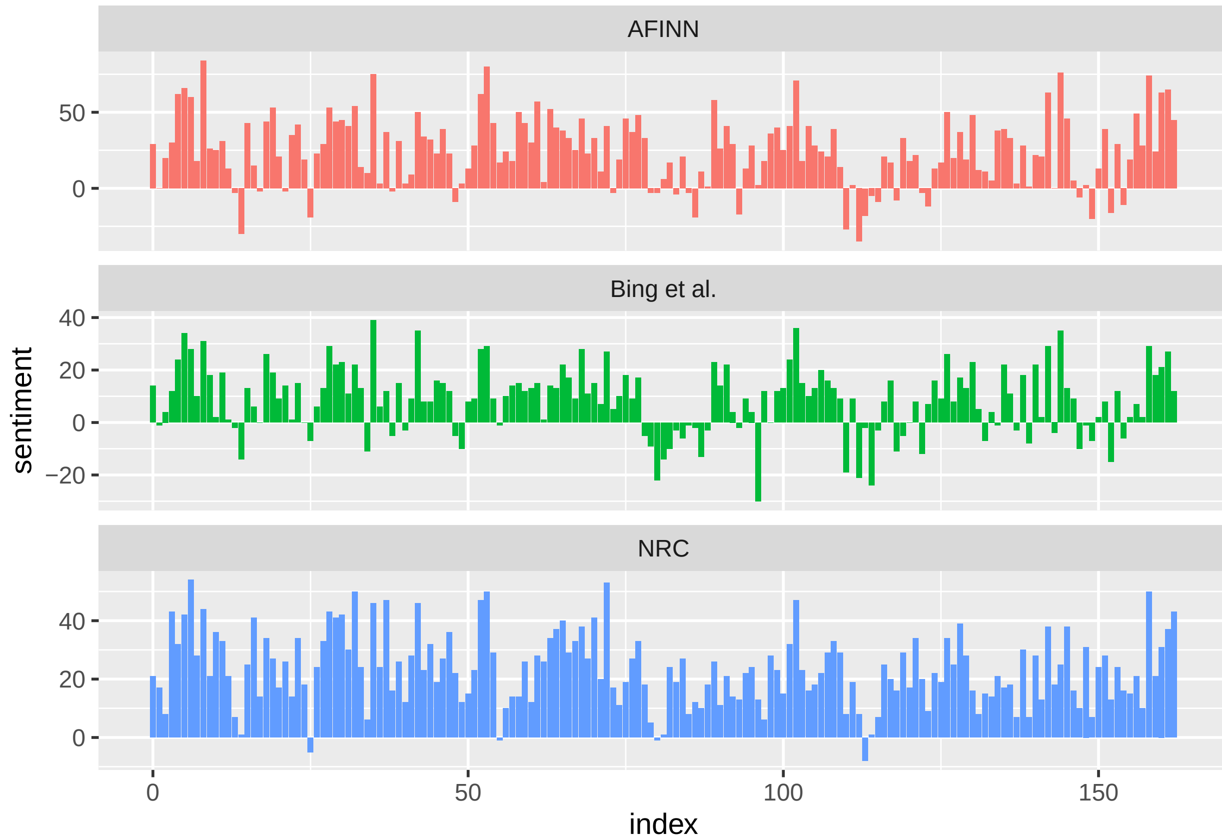
```r
afinn <- pride_prejudice %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(index = linenumber %/% 80) %>%
  summarise(sentiment = sum(value)) %>%
  mutate(method = "AFINN")
```

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

```r
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) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)
```

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

```r
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   3324
## 2 positive   2312
```

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

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

4. Most Common Positive & Negative Words

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

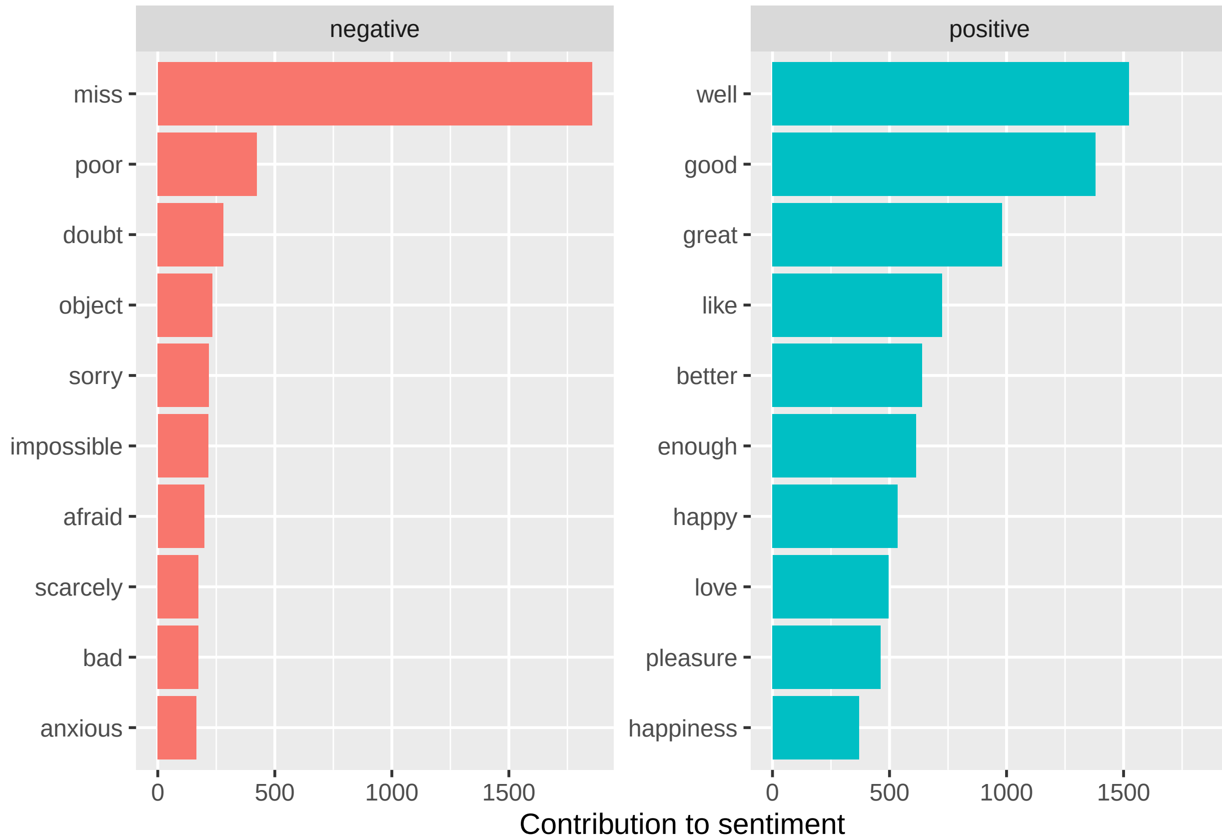
```
## Joining, by = "word"
```

```r
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
```

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

```
## Selecting by n
```




```
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
```

5. Wordclouds

```r
library(wordcloud)
```

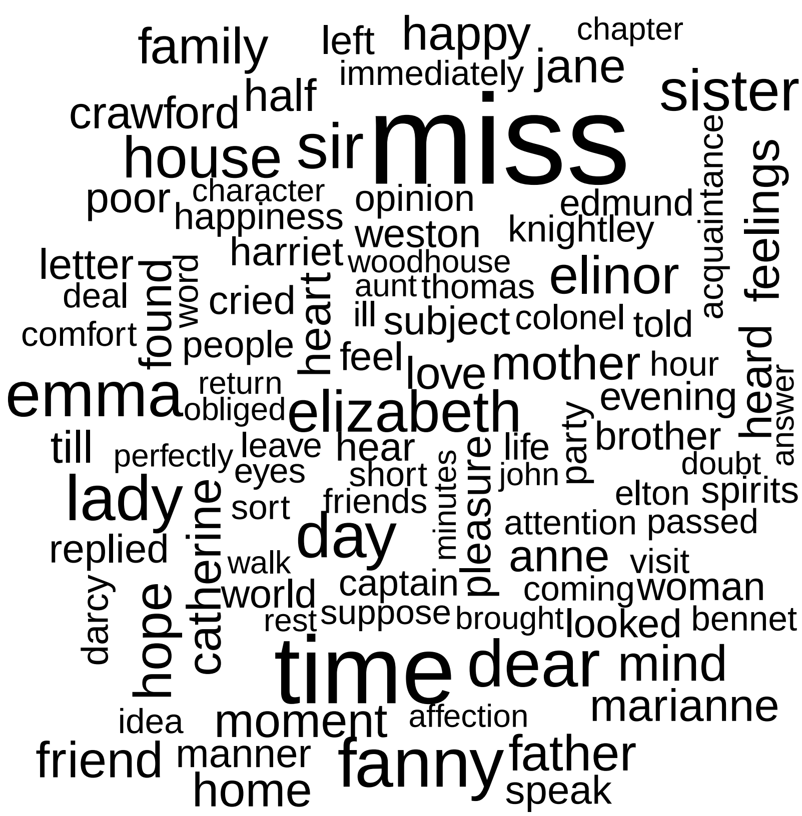
```
## Warning: package 'wordcloud' was built under R version 3.6.3
```

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

```r
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): morning could not be fit on
## page. It will not be plotted.
```




```r
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 3.6.3
```

```
##
## 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"
```

6. Looking Beyond Words

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

PandP_sentences$sentence[2]
```

```
## [1] "however little known the feelings or views of such a man may be on his first entering a neighbou
```

```r
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
```

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

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

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) %>%
  top_n(1) %>%
  ungroup()
```

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

## Selecting by ratio

## # 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
```

7. My Extension

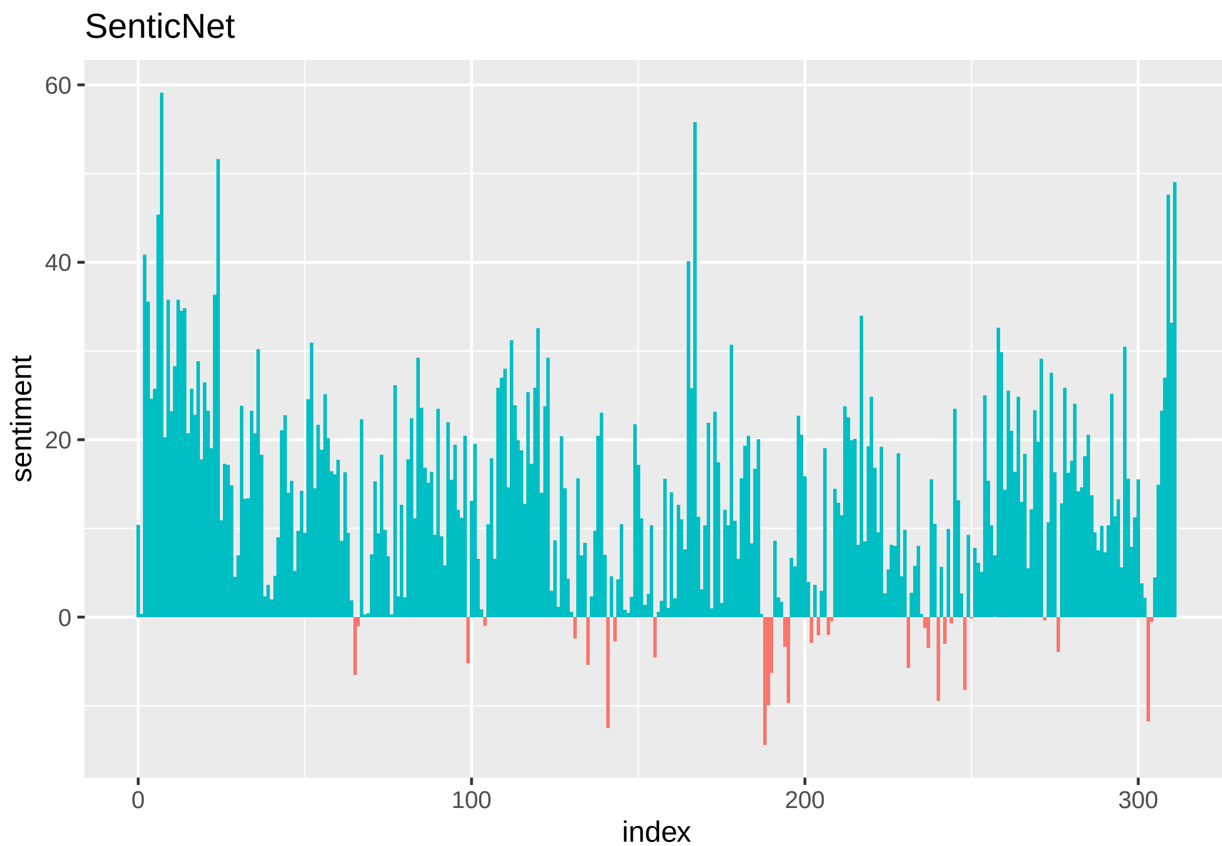
For this section, I analyzed *The Iliad* using four different sentiment lexicons: SenticNet[2], SentiWordNet[3], Hu Liu[4], and SlangSD[5].

```r
# open libraries
library(lexicon)
```

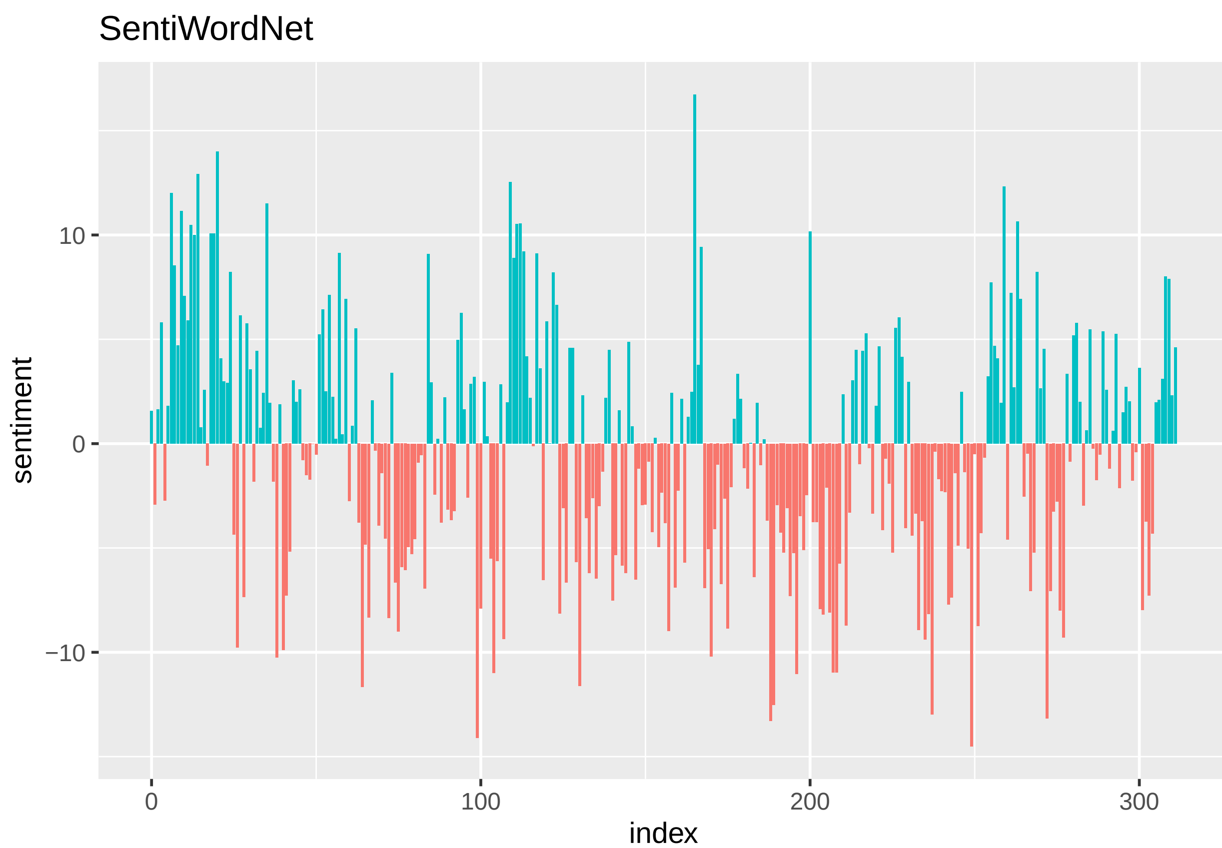
```
## Warning: package 'lexicon' was built under R version 3.6.3
```

```r
library(readr)

# open four lexicons: SenticNet, SentiWordNet, Hu Liu, and SlangSD
senticnet <- hash_sentiment_senticnet
names(senticnet) <- c("word","score")

sentiword <- hash_sentiment_sentiword
names(sentiword) <- c("word","score")

huliu <- hash_sentiment_huliu
names(huliu) <- c("word","score")

slangsd <- hash_sentiment_slangsd
names(slangsd) <- c("word","score")

# open The Ilad text file and tidy it up with tidytext
iliad <- tibble(line = 1:length(read_lines("TheIliad.txt")),
                    text = c(read_lines("TheIliad.txt"))) %>%
  unnest_tokens(word, text)

# create sentiments based on each of the lexicons
senticnet_sentiment <- iliad %>%
  inner_join(senticnet, by = "word") %>%
  group_by(index = line %/% 80) %>%
  summarise(sentiment = sum(score)) %>%
  mutate(overall = ifelse(sentiment > 0, "Positive", "Negative"))

sentiword_sentiment <- iliad %>%
  inner_join(sentiword, by = "word") %>%
  group_by(index = line %/% 80) %>%
  summarise(sentiment = sum(score)) %>%
  mutate(overall = ifelse(sentiment > 0, "Positive", "Negative"))

huliu_sentiment <- iliad %>%
  inner_join(huliu, by = "word") %>%
  group_by(index = line %/% 80) %>%
  summarise(sentiment = sum(score)) %>%
  mutate(overall = ifelse(sentiment > 0, "Positive", "Negative"))

slangsd_sentiment <- iliad %>%
  inner_join(slangsd, by = "word") %>%
  group_by(index = line %/% 80) %>%
  summarise(sentiment = sum(score)) %>%
```

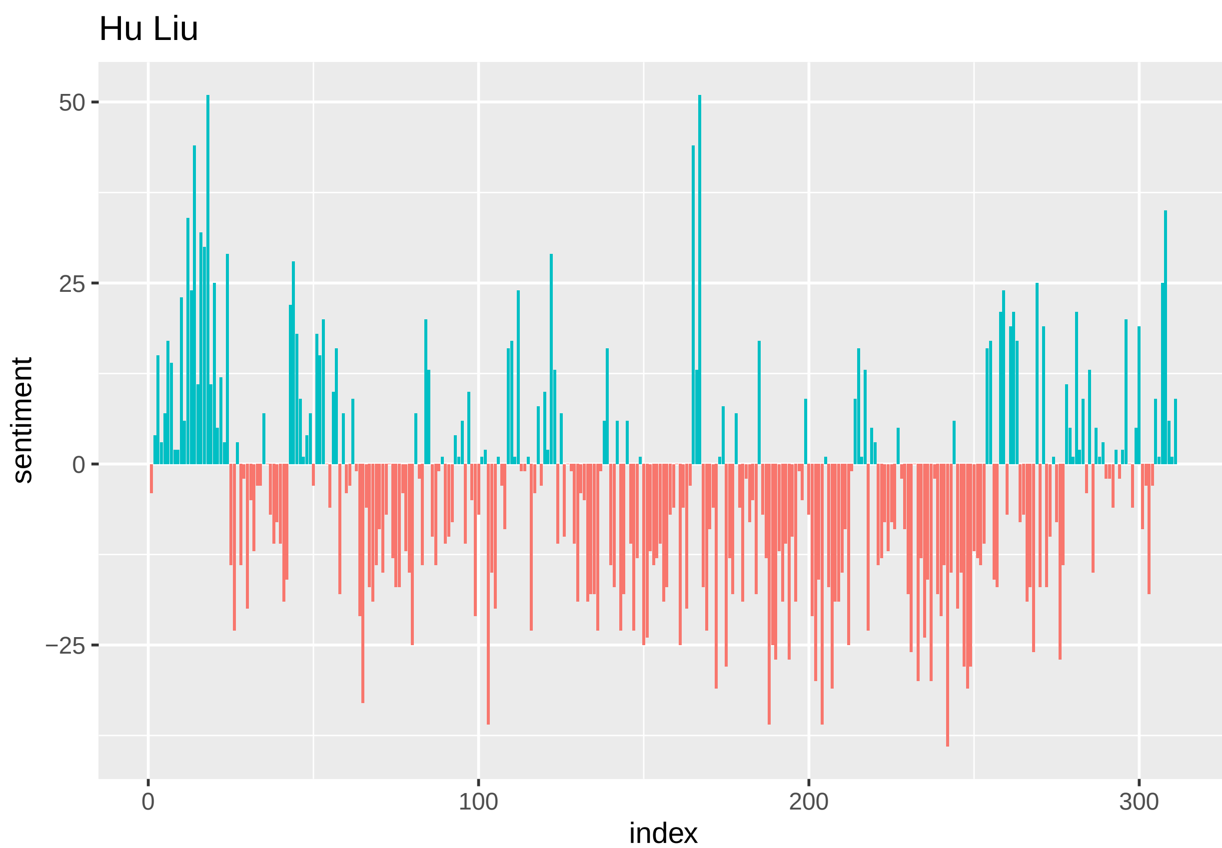
```r
  mutate(overall = ifelse(sentiment > 0, "Positive", "Negative"))

# plot each lexicon sentiment
ggplot(senticnet_sentiment, aes(index, sentiment, fill = overall)) +
  geom_col(show.legend = FALSE) + labs(title = "SenticNet")
```



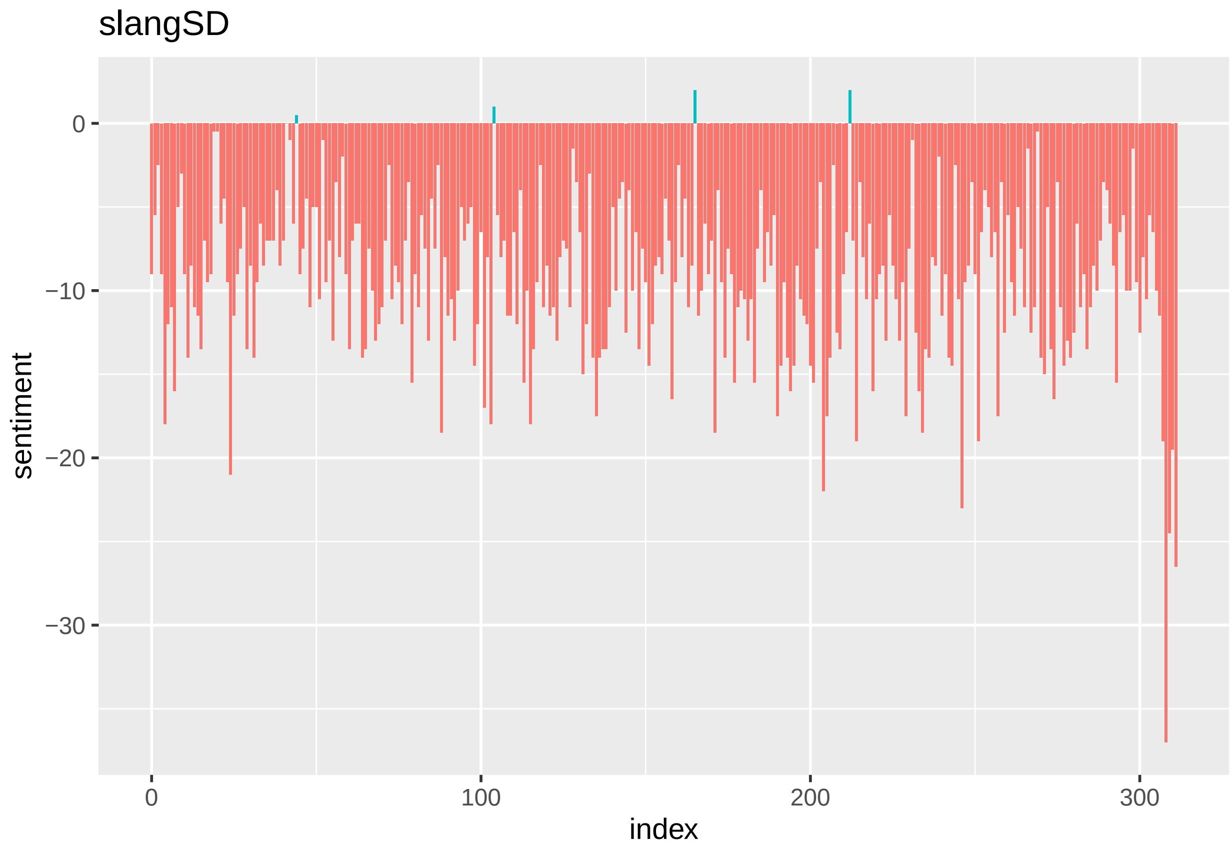

```r
ggplot(sentiword_sentiment, aes(index, sentiment, fill = overall)) +
  geom_col(show.legend = FALSE) + labs(title = "SentiWordNet")
```




```
ggplot(huliu_sentiment, aes(index, sentiment, fill = overall)) +
  geom_col(show.legend = FALSE) + labs(title = "Hu Liu")
```




```
ggplot(slangsd_sentiment, aes(index, sentiment, fill = overall)) +
  geom_col(show.legend = FALSE) + labs(title = "slangSD")
```

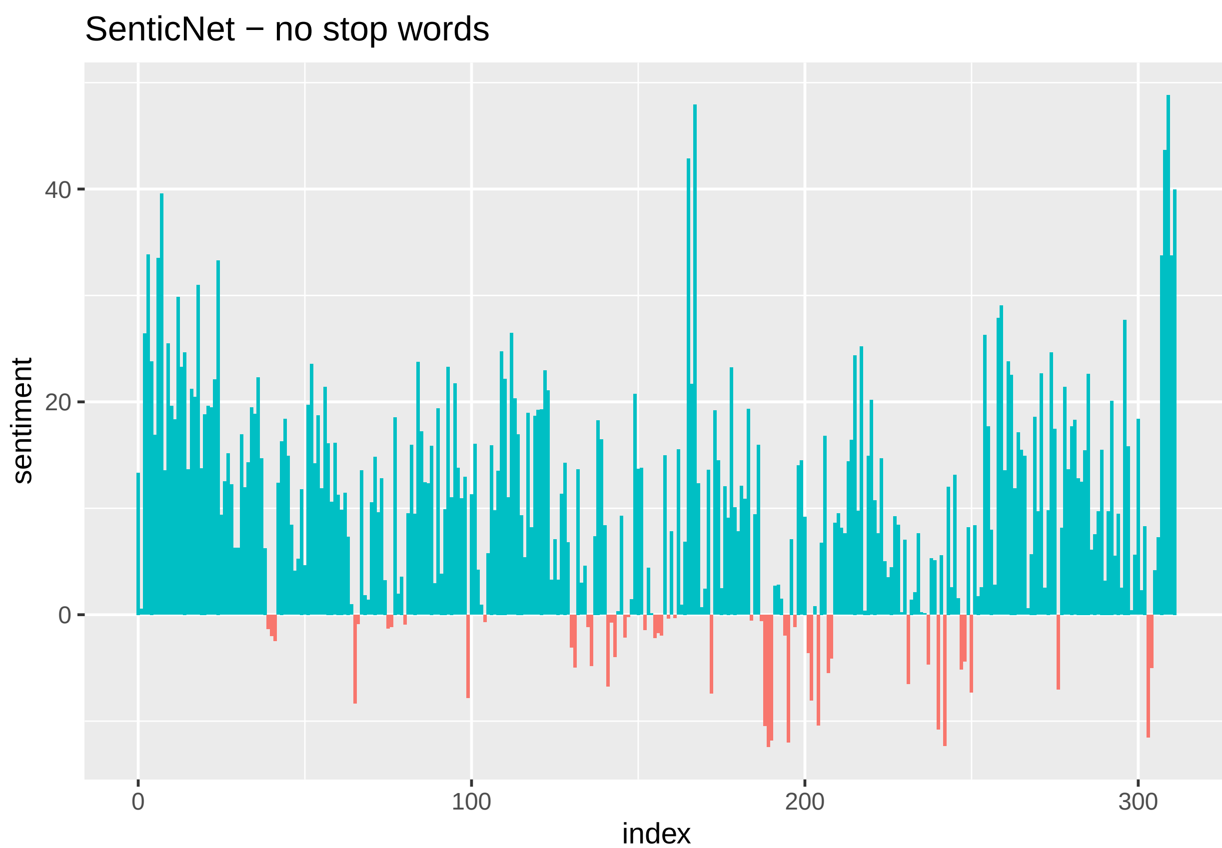


After looking at the sentiment of *The Iliad* using four different sentiment lexicons, I found it incredible how different the lexicons rated the same text. A text such as *The Iliad* should certainly come out as negative, but it was scored as positive (*SenticNet*), neutral (*Hu Liu* and *SentiWordNet*), and negative (*slangSD*) depending on the sentiment lexicon used.

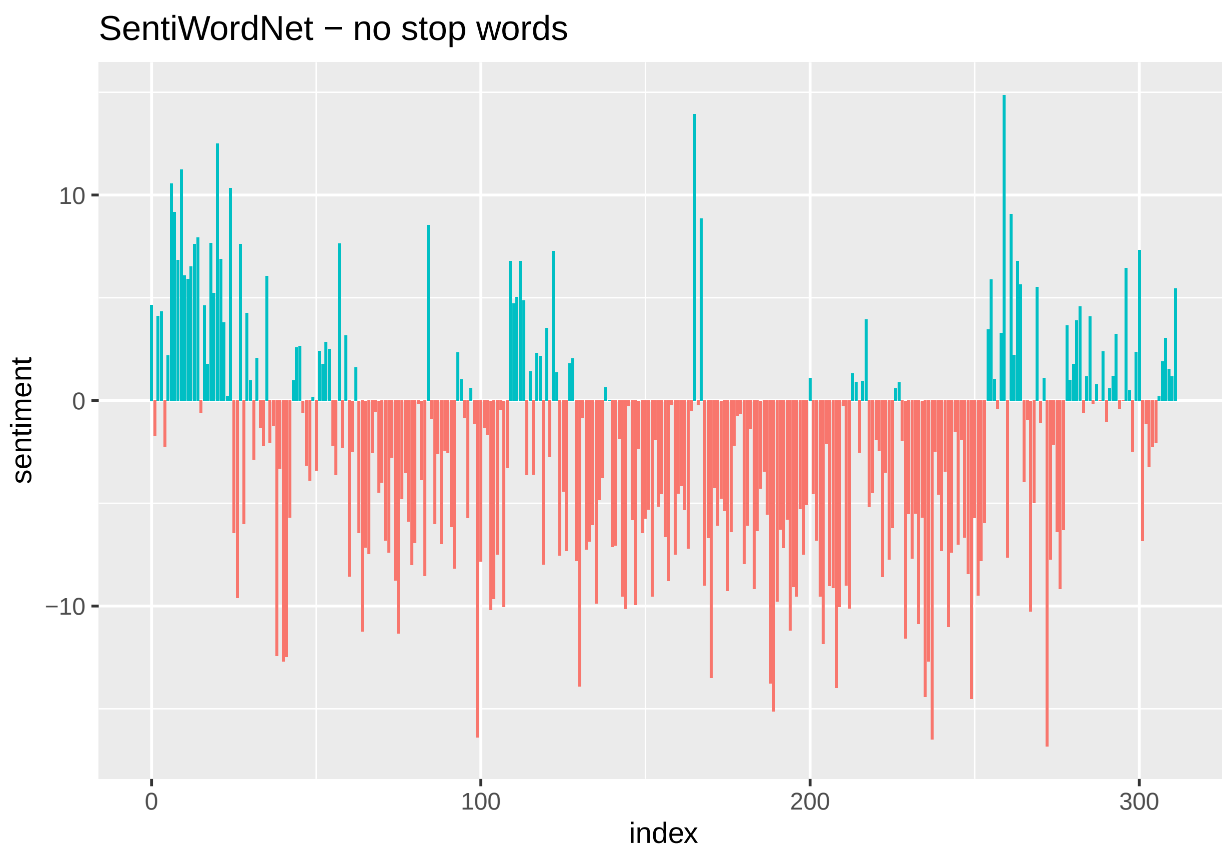
I decided to remove stop words from the sentiments to see what impact those might be having.

```
# remove stop words from each sentiment
senticnet_sentiment_go <- iliad %>%
  inner_join(senticnet, by = "word") %>%
  anti_join(stop_words, by = "word") %>%
  group_by(index = line %/% 80) %>%
  summarise(sentiment = sum(score)) %>%
  mutate(overall = ifelse(sentiment > 0, "Positive", "Negative"))

sentiword_sentiment_go <- iliad %>%
  inner_join(sentiword, by = "word") %>%
  anti_join(stop_words, by = "word") %>%
  group_by(index = line %/% 80) %>%
  summarise(sentiment = sum(score)) %>%
  mutate(overall = ifelse(sentiment > 0, "Positive", "Negative"))

huliu_sentiment_go <- iliad %>%
  inner_join(huliu, by = "word") %>%
  anti_join(stop_words, by = "word") %>%
  group_by(index = line %/% 80) %>%
  summarise(sentiment = sum(score)) %>%
```

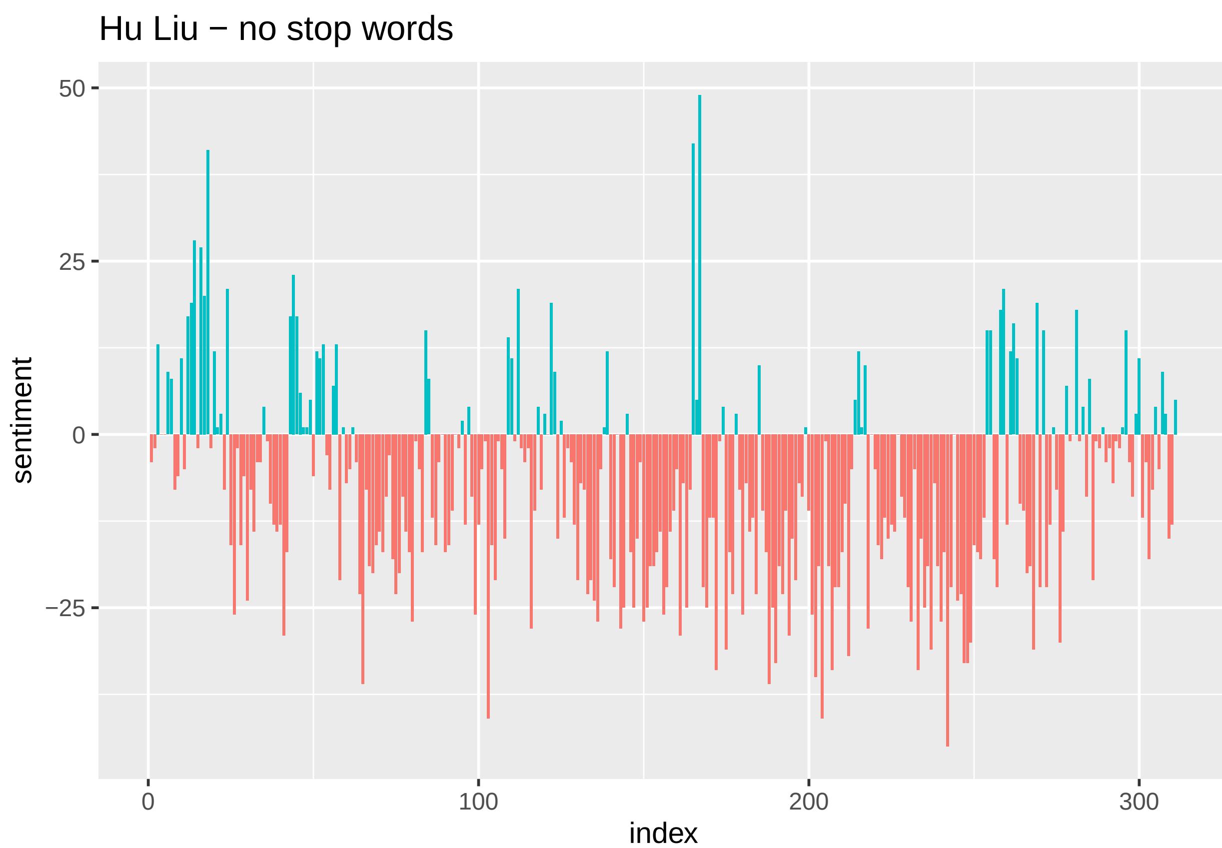
```
  mutate(overall = ifelse(sentiment > 0, "Positive", "Negative"))

slangsd_sentiment_go <- iliad %>%
  inner_join(slangsd, by = "word") %>%
  anti_join(stop_words, by = "word") %>%
  group_by(index = line %/% 80) %>%
  summarise(sentiment = sum(score)) %>%
  mutate(overall = ifelse(sentiment > 0, "Positive", "Negative"))

# plot each lexicon sentiment
ggplot(senticnet_sentiment_go, aes(index, sentiment, fill = overall)) +
  geom_col(show.legend = FALSE) + labs(title = "SenticNet - no stop words")
```



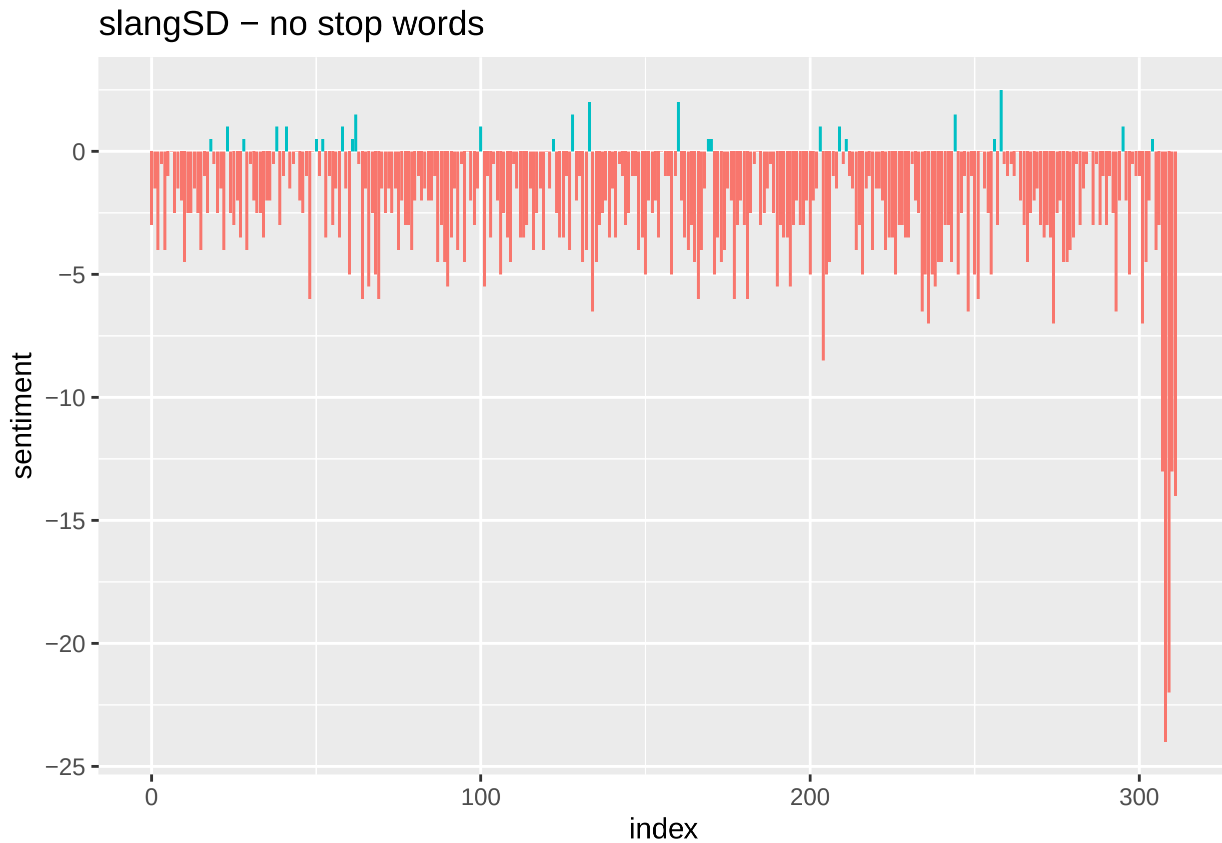

```
ggplot(sentiword_sentiment_go, aes(index, sentiment, fill = overall)) +
  geom_col(show.legend = FALSE) + labs(title = "SentiWordNet - no stop words")
```




```r
ggplot(huliu_sentiment_go, aes(index, sentiment, fill = overall)) +
  geom_col(show.legend = FALSE) + labs(title = "Hu Liu - no stop words")
```

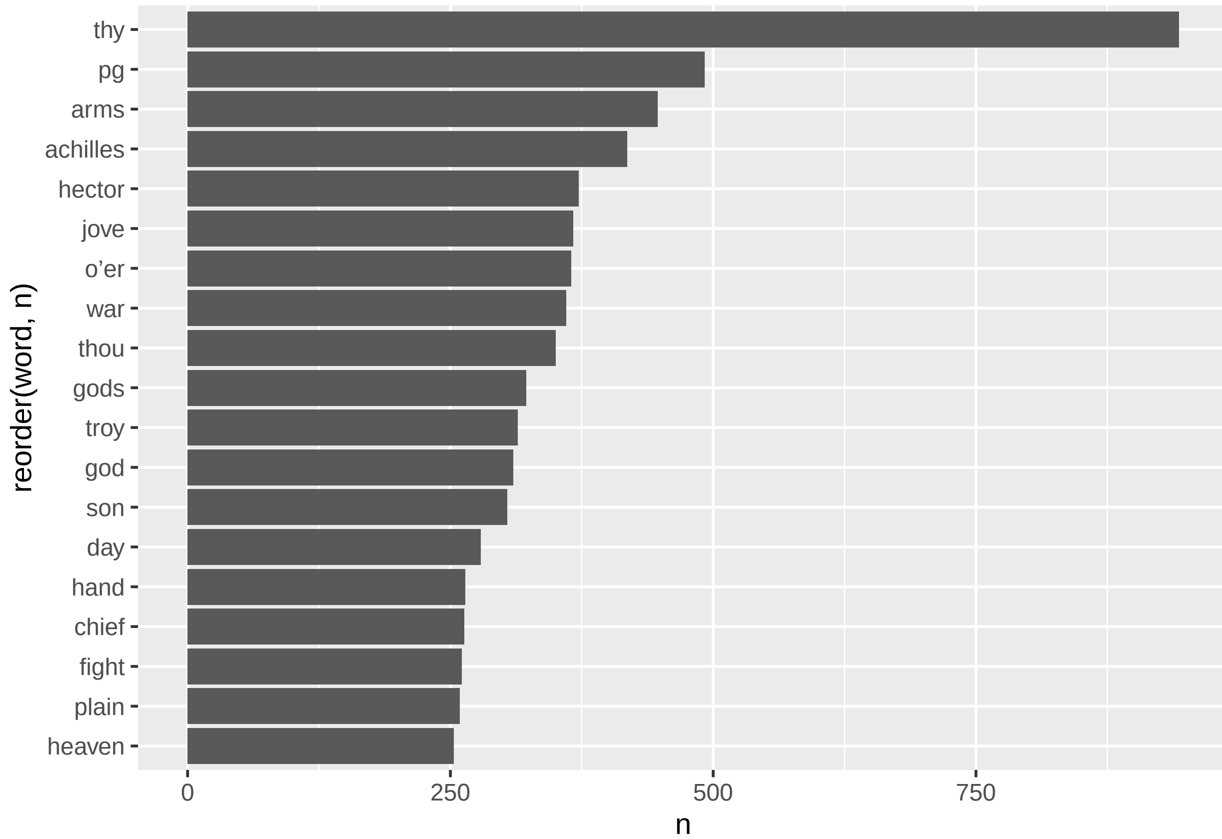



```
ggplot(slangsd_sentiment_go, aes(index, sentiment, fill = overall)) +
  geom_col(show.legend = FALSE) + labs(title = "slangSD - no stop words")
```



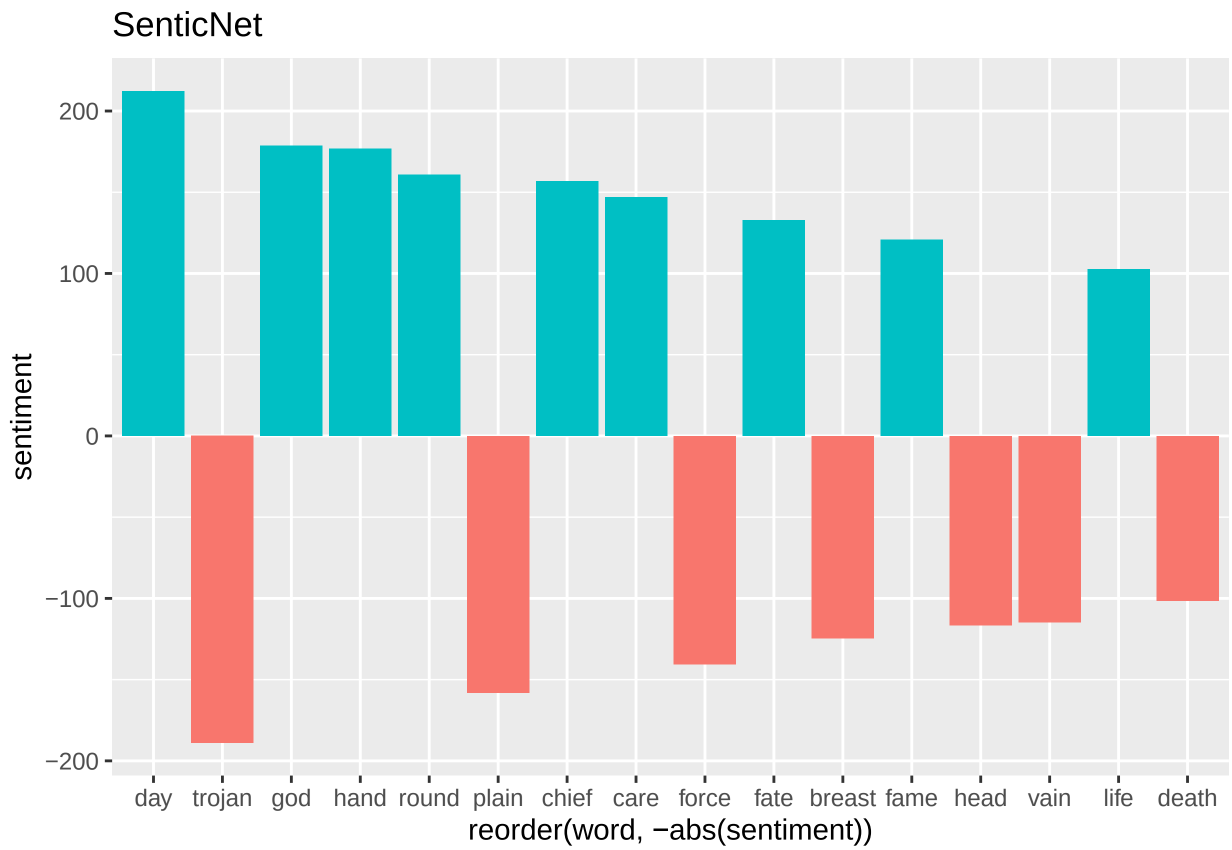
All of these plots made more sense for a story such as *The Iliad*, but I was curious now about which words were the largest contributors to the overall sentiment of the story.

```
iliad_words <- iliad %>%
  count(word, sort = TRUE) %>%
  anti_join(stop_words, by = "word") %>%
  filter(n > 250)

ggplot(iliad_words, aes(x= reorder(word, n), y= n)) + geom_bar(stat = "identity", show.legend = FALSE) +
```

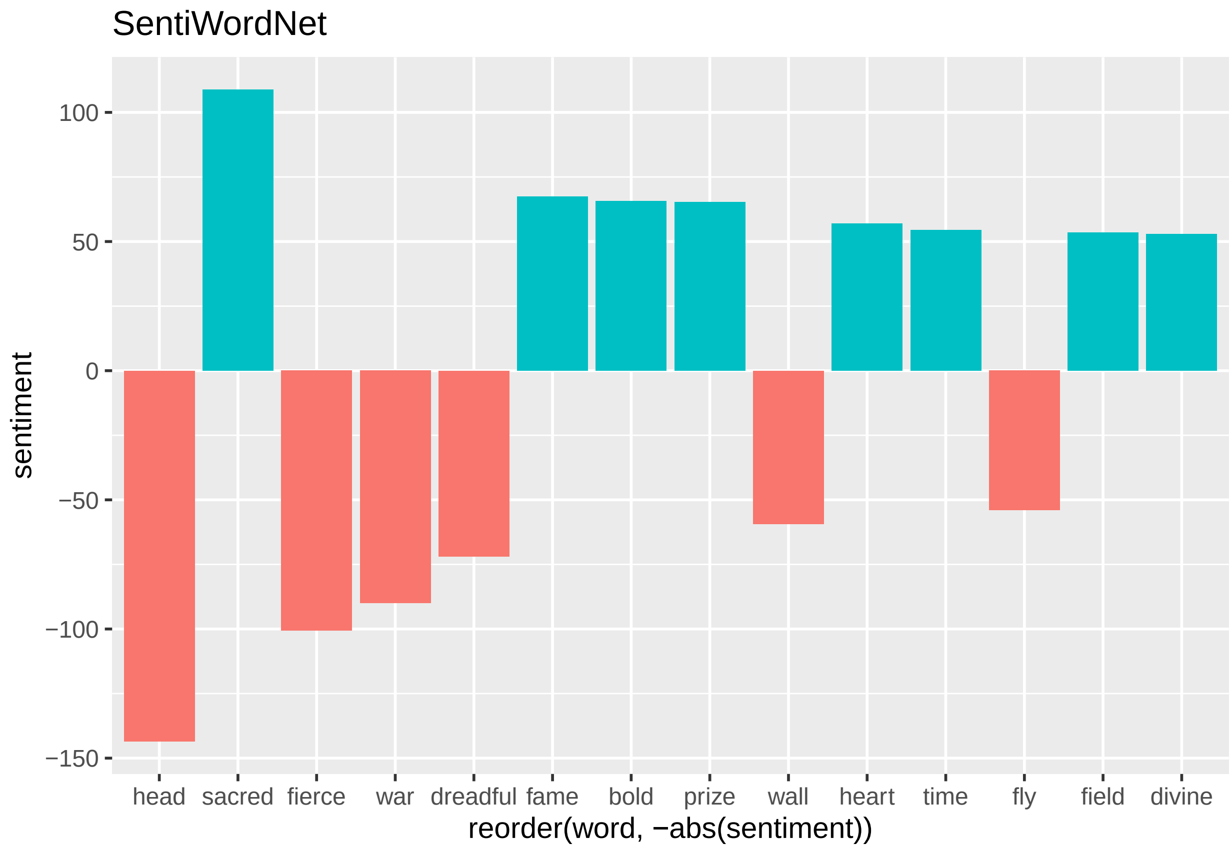



```
senticnet_words <- iliad %>%
  inner_join(senticnet, by = "word") %>%
  anti_join(stop_words, by = "word") %>%
  group_by(word) %>%
  summarise(sentiment = sum(score)) %>%
  mutate(overall = ifelse(sentiment > 0, "Positive", "Negative")) %>%
  filter(abs(sentiment) > 100)

sentiword_words <- iliad %>%
  inner_join(sentiword, by = "word") %>%
  anti_join(stop_words, by = "word") %>%
  group_by(word) %>%
  summarise(sentiment = sum(score)) %>%
  mutate(overall = ifelse(sentiment > 0, "Positive", "Negative")) %>%
  filter(abs(sentiment) > 50)

huliu_words <- iliad %>%
  inner_join(huliu, by = "word") %>%
  group_by(word) %>%
  summarise(sentiment = sum(score)) %>%
  mutate(overall = ifelse(sentiment > 0, "Positive", "Negative")) %>%
  filter(abs(sentiment) > 150)

slangsd_words <- iliad %>%
  inner_join(slangsd, by = "word") %>%
  anti_join(stop_words, by = "word") %>%
```

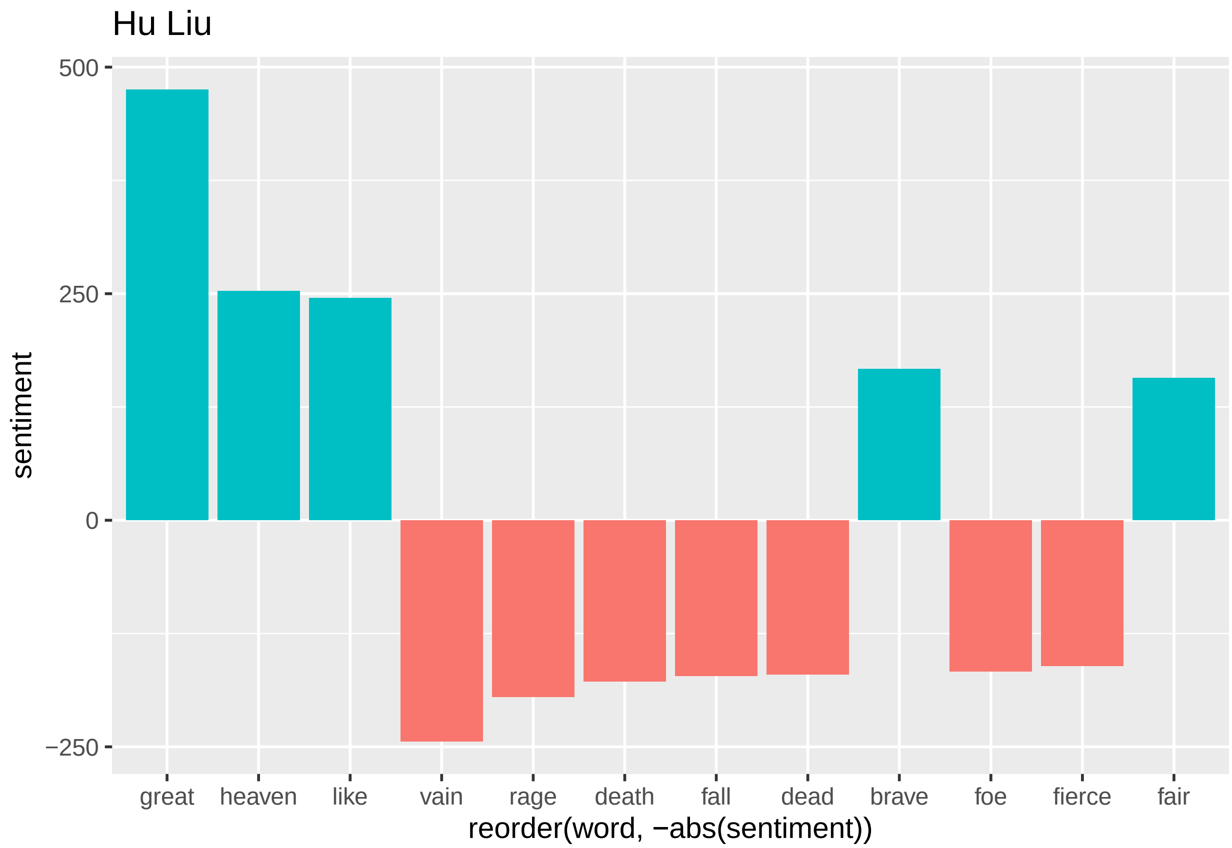
```
  group_by(word) %>%
  summarise(sentiment = sum(score)) %>%
  mutate(overall = ifelse(sentiment > 0, "Positive", "Negative")) %>%
  filter(abs(sentiment) > 50)

ggplot(senticnet_words, aes(x= reorder(word, -abs(sentiment)), y= sentiment, fill = overall)) + geom_ba
```



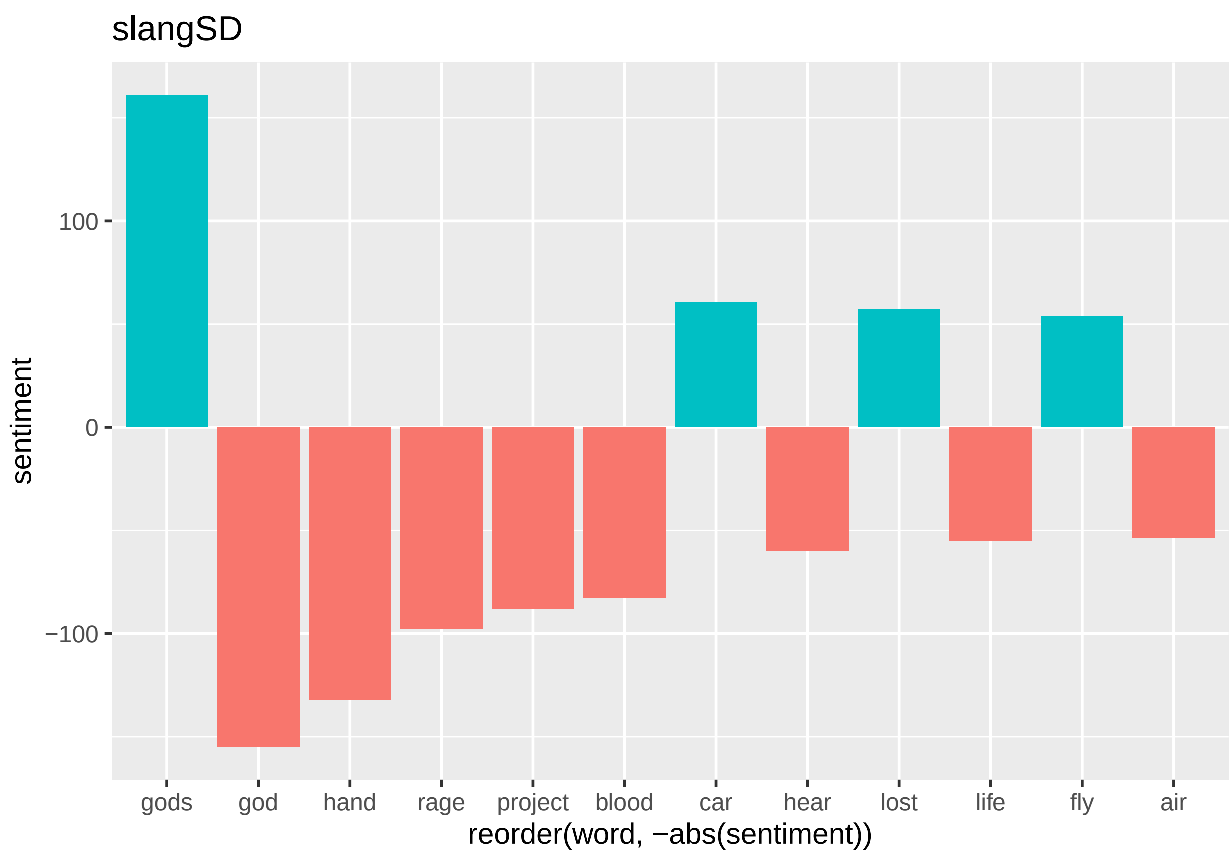

```
ggplot(sentiword_words, aes(x= reorder(word, -abs(sentiment)), y= sentiment, fill = overall)) + geom_ba
```




```
ggplot(huliu_words, aes(x= reorder(word, -abs(sentiment)), y= sentiment, fill = overall)) + geom_bar(sta
```




```
ggplot(slangsd_words, aes(x= reorder(word, -abs(sentiment)), y= sentiment, fill = overall)) + geom_bar(s
```



Conclusion

After looking at the sentiment of *The Iliad* using four different sentiment lexicons, both with and without stop words, I find it incredible how differently stop words are treated by each lexicon. Some lexicons (*SenticNet*) swayed drastcally based on stop words, going from a positive reading of *The Iliad* to a negative one when stop words were removed, while others (*slangSD*) had no discernible change when stop words were removed.

After looking at the most impactful words from each sentiment lexicon, I believe the Hu Liu lexicon is the most approriate for this text. The most impactful words are meaningful even before stop words are removed and the overall sentiment outcome looks similar to what I would expect.

Citations

1. Silge, Julia, and David Robinson. "Text Mining with R." *2 Sentiment Analysis with Tidy Data*, 7 Mar. 2020, www.tidytextmining.com/sentiment.html.

2. Cambria, E., Poria, S., Bajpai, R. and Schuller, B. SenticNet 4: A semantic resource for sentiment analysis based on conceptual primitives. In: COLING, pp. 2666-2677, Osaka (2016) http:// sentic.net/downloads

3. Baccianella S., Esuli, A. and Sebastiani, F. (2010). SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. International Conference on Language Resources and Evaluation.

4. Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2004). Seattle, Washington. Hu, M., & Liu, B. (2004). Mining opinion features in customer reviews. National Conference on Artificial Intelligence.
5. Wu, L., Morstatter, F., and Liu, H. (2016). SlangSD: Building and using a sentiment dictionary of slang words for short-text sentiment classification. CoRR. abs/1168.1058. 1-15.