Analysis of Stack Overflow Q&A

Semih Barutcu

12/11/2020

I used a kaggle dataset about the topic StackSample: 10% of Stack Overflow Q&A. The link for the dataset: https://www.kaggle.com/stackoverflow/stacksample

This dataset includes questions, answers and tags as text of 10% of questions and answers from the Stack Overflow. I would like to do data exploration over these datasets and study tf-idf. I would like to work on an appropriate machine learning algorithm for these dataset lastly.

I have done an introductory data explorations until now using Text Mining with R: A Tidy Approach by Julia Silge and David Robinson book as a reference.

Preprocessing

```
library(pacman)

p_load(dplyr, tidytext, ggplot2, wordcloud, stringr, lubridate)
```

```
questions <- read.csv("data/Questions.csv")

tags <- read.csv("data/Tags.csv")

answers <- read.csv("data/Answers.csv")
```

Date are formatted by lubridate package function ymd_hms().

```
questions$CreationDate <- ymd_hms(questions$CreationDate)
questions$ClosedDate <- ymd_hms(questions$ClosedDate)
```

I used the first 100,000 observations to work at the beginning. I used smaller datasets where my 16 GB RAM is not sufficient for size of datasets.

```
set.seed(123)
tag_small <- sample_n(tags, 100000)

q_small <- sample_n(questions, 30000)
```

Exploratory Data Analysis (EDA)

Tags

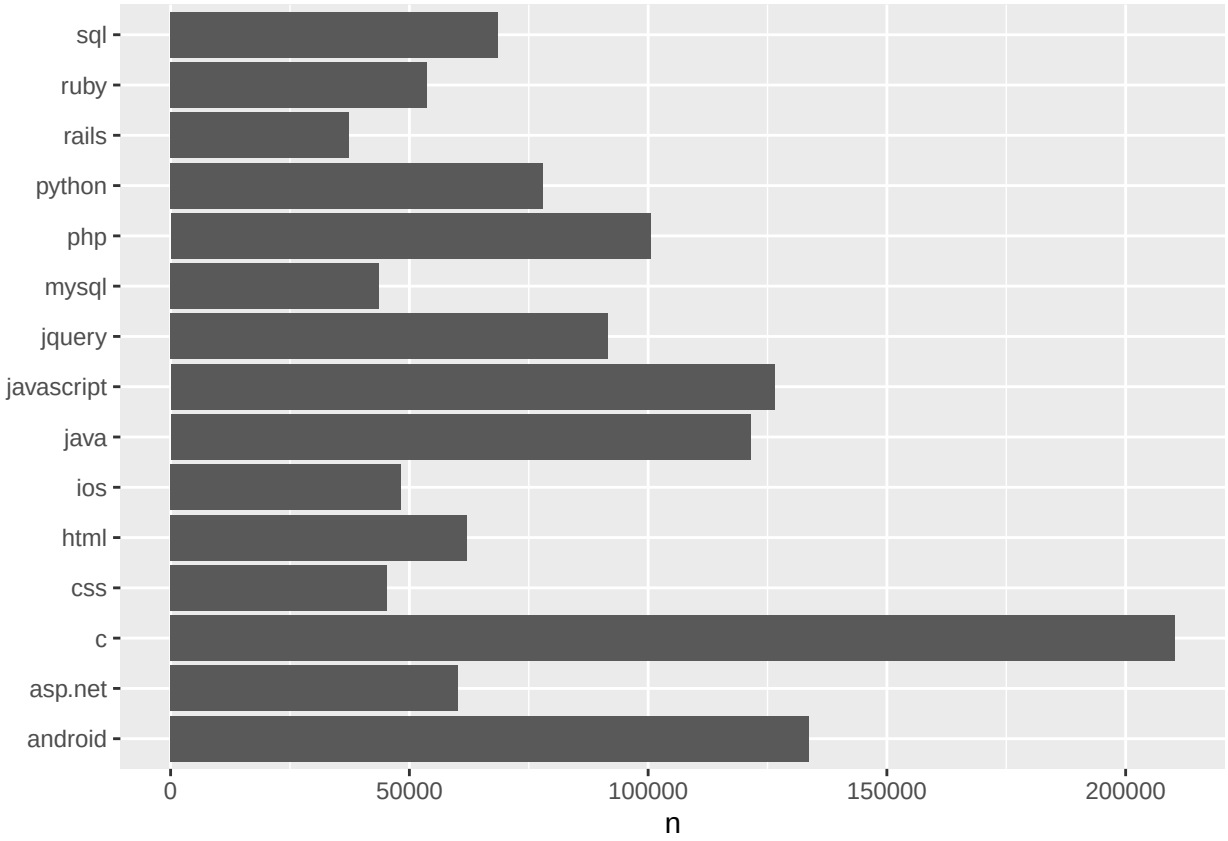
Tags are formed by unnesting tokens.

```
tags <- tags %>%
  unnest_tokens(tag, Tag)

tag_small <- tag_small %>%
  unnest_tokens(tag, Tag)
```

You can see counts of 15 most common the tags dataset below.

```
tags %>%
  group_by(tag) %>%
  summarize(n = n()) %>%
  top_n(15) %>%
  ggplot(aes(tag, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```



Questions

Body of questions are formed as singular words by unnesting tokens after some signs and stop words are removed. Stop words are some of most common words such as "the," "of," "to," and so forth in English which are not identifying and not useful for an analysis.

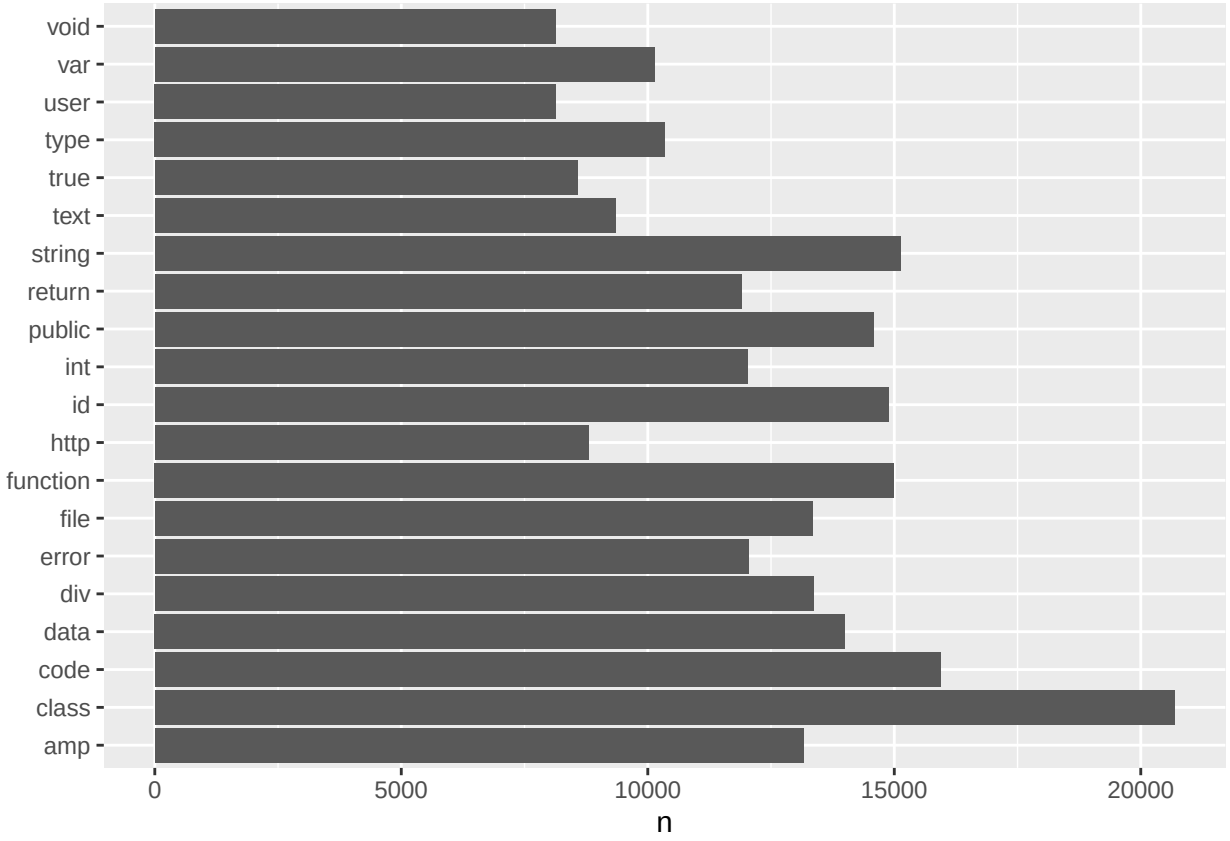
```
# I used same stop words and a similar querying on q_word with work of Julia Silge
# The notebook can be found here: https://www.kaggle.com/juliasilge/tf-idf-of-stack-overflow-questions
# eliminate the words gt and lt which are used for greater than and less than.
my_stop_words <- bind_rows(stop_words %>%
                             filter(lexicon == "snowball"),
                           tibble(word = c("gt", "lt"),
                                  lexicon = rep("custom", 2)))

q_small <- q_small %>%
```

```
  mutate(Body = str_replace_all(Body, "<[^>]*>", "")) %>%
  unnest_tokens(word, Body) %>%
  filter(str_detect(word, "^[a-z]")) %>%
  filter(str_detect(word, "[0-9]", negate = T)) %>%
  filter(!word %in% c("ccccccccccccccc", "welcome.seamlessoffers.com", "interfashionadmin.theindiestudi
  anti_join(my_stop_words, by = "word")
```

You can see the most common words on the Body part of small questions dataset. code, class and file are top 3 words. Most of the words are not useful to identify the tags because they are not belong to a specific language while some of them can help us eliminating some tags. For example, class word is used for an Object-oriented programming (OOP) language most probably or amp is an open-source custom web development framework created to speed up the loading time of web pages on mobile devices and the word is related with this framework.

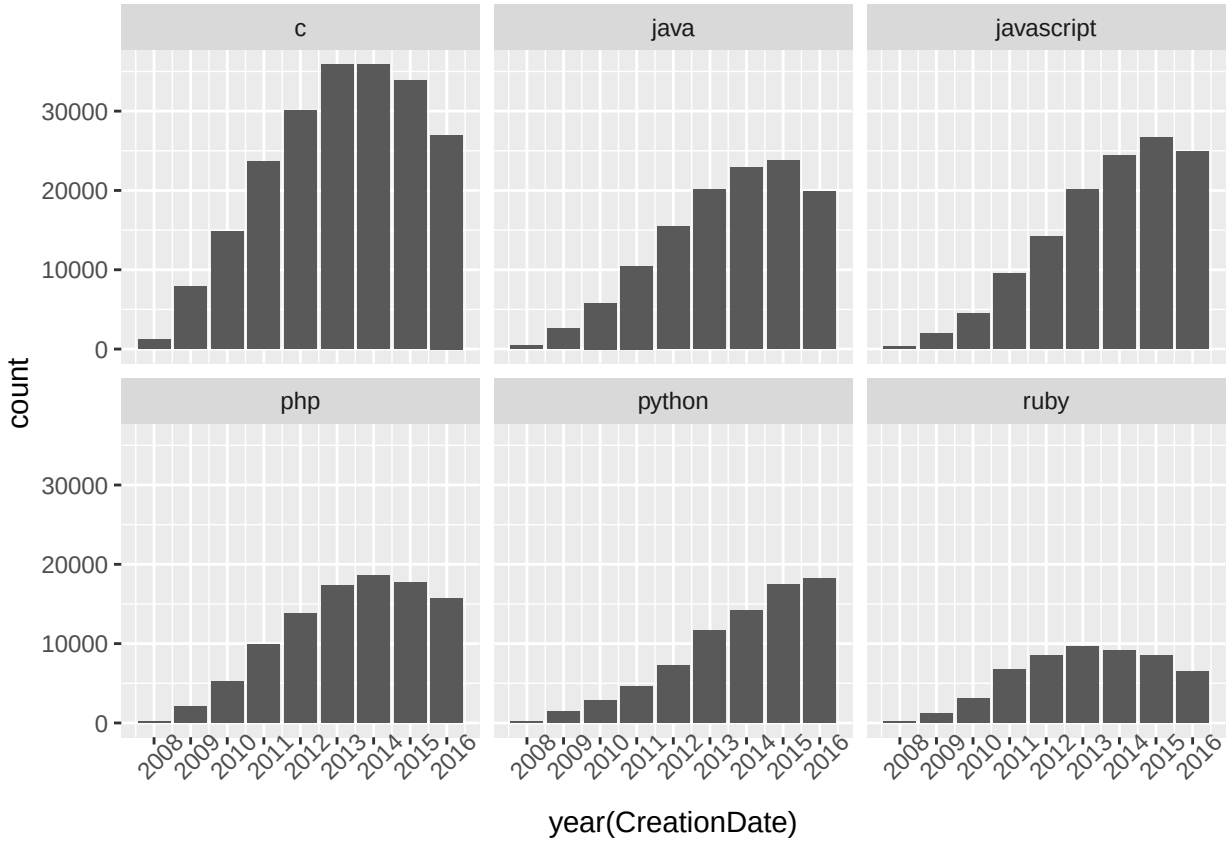
```
q_small %>%
  group_by(word) %>%
  anti_join(stop_words, by = "word") %>%
  summarize(n = n(), .groups = 'drop') %>%
  top_n(20) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```



I joined the questions and tags datasets and filtered the search for some popular languages. Then, I plotted a bar graph count of each year's observations with respect to these languages. We can see the rise of total tags used by users until 2012. Java and javascript were used each year more than previous year thru 2015 while

python is the only language that the total counts were increased every year. Still, c and c++ has most tags but trends show that python and javascript are the most promising languages.

```
questions %>% select(Id, CreationDate) %>%
  left_join(tags, by = "Id") %>%
  filter(tag %in% c( "ruby", "python", "php", "javascript", "java", "c")) %>%
  ggplot(aes(year(CreationDate))) +
  geom_bar() +
  facet_wrap(~tag) +
  scale_x_continuous(breaks = 2007:2016, ) +
  theme(axis.text.x = element_text(angle = 45))
```



We can see that ruby language topics have least closed dates ratio by far for the selected 6 languages.

```
questions %>% select(Id, ClosedDate) %>%
  left_join(tags, by = "Id") %>%
  filter(tag %in% c( "ruby", "python", "php", "javascript", "java", "c")) %>%
  group_by(tag) %>%
  summarise(Closed_Counts = sum(!is.na(ClosedDate)), Closed_Ratio = sum(!is.na(ClosedDate))/ n() ) %>%
  arrange(desc(Closed_Ratio))
```

```
## # A tibble: 6 x 3
##   tag        Closed_Counts Closed_Ratio
##   <chr>              <int>        <dbl>
## 1 php                 7260       0.0722
## 2 java                8165       0.0672
## 3 c                  12759       0.0607
## 4 python              4530       0.0581
```

```
## 5 javascript          6566       0.0519
## 6 ruby                1243       0.0232
```

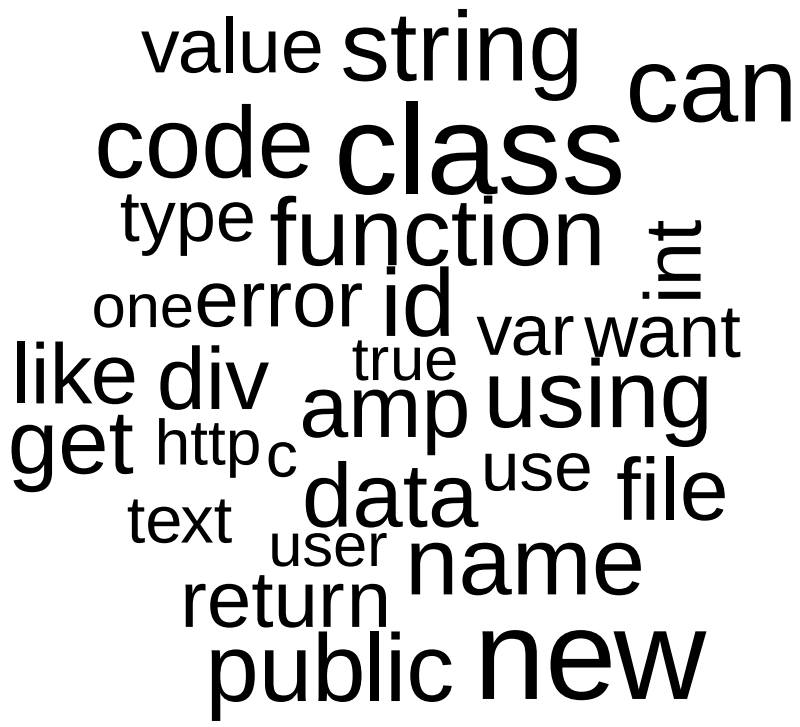
You can see score statistics of questions with respect to 6 computer programming languages. C has the highest average and php has the least. By looking minimum and maximum scores we can observe far away than averages, especially for maximum values. But you can see the 0.025 and 0.975 which make sense with the average. So, I can say that most of the topic are not voted much because the 95% range is narrow with an average between 1 and 2.1 while some questions are aroused interest especially in a positive way.

```
questions %>% select(Id, Score) %>%
  left_join(tags, by = "Id") %>%
  filter(tag %in% c( "ruby", "python", "php", "javascript", "java", "c")) %>%
  group_by(tag) %>%
  summarise(Avg = round(mean(Score), 2), Min = min(Score), Max = max(Score), Q_025 = quantile(Score, 0.
  arrange(desc(Avg))
```

```
## # A tibble: 6 x 6
##   tag          Avg   Min   Max Q_025 Q_975
##   <chr>      <dbl> <int> <int> <dbl> <dbl>
## 1 c           2.13   -20  1473    -2    13
## 2 python      1.98   -23   824    -2    12
## 3 ruby        1.94   -11   648    -1    12
## 4 java        1.78   -45  3613    -2    11
## 5 javascript  1.67   -18  2363    -2     9
## 6 php         0.99   -13  1760    -2     6
```

WordCloud

```
q_small %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 30))
```




```
tag_small %>%
  count(tag) %>%
  with(wordcloud(tag, n, max.words = 25))
```

Sentiment Analysis

I used afinn sentiments to make sentimental analysis on the questions. The AFINN lexicon assigns words with a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment. You can see the scores of the all questions of the small questions dataset.

```
q_small %>%
  select(Id, word) %>%
  inner_join(get_sentiments("afinn"), by = "word") %>%
  group_by(Id) %>%
  summarise(sentiment = sum(value), .groups = 'drop')
```

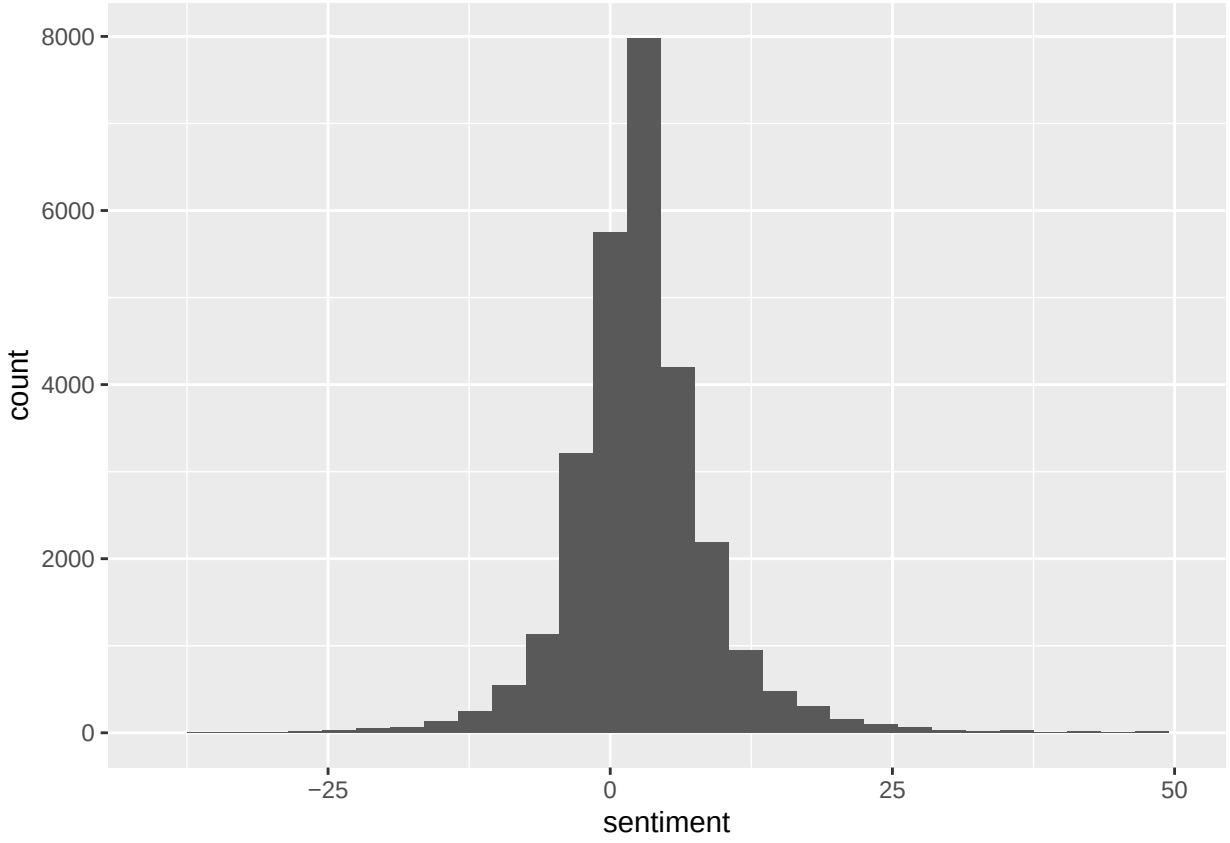
```
## # A tibble: 27,803 x 2
##       Id sentiment
##    <int>     <dbl>
##  1  3790         9
##  2 11740        13
##  3 15390         3
##  4 17870         6
##  5 18450         2
##  6 44980         9
##  7 48070         8
##  8 49890         0
##  9 58520         2
## 10 64570         0
## # ... with 27,793 more rows
```

You can see the histogram of the scores below.

```
q_small %>%
  select(Id, word) %>%
  inner_join(get_sentiments("afinn"), by = "word") %>%
  group_by(Id) %>%
  summarise(sentiment = sum(value), .groups = 'drop') %>%
  ggplot(aes(sentiment)) +
  geom_histogram(binwidth = 3) +
  xlim(-40, 50)
```

```
## Warning: Removed 77 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```



I calculated the count of positive and negative words and ordered them in descending order.

```
q_small_afinn_count <- q_small %>%
  select(Id, word) %>%
  inner_join(get_sentiments("afinn"), by = "word") %>%
  mutate(sentiment = ifelse(value > 0, "positive", "negative")) %>%
  count(word, sentiment) %>%
  arrange(desc(n)) %>%
  top_n(24)
```

```
## Selecting by n
```

```
q_small_afinn_count
```

```
##           word sentiment     n
## 1         like  positive 12964
## 2        error  negative 12045
## 3         want  positive 10681
## 4         true  positive  8571
## 5      problem  negative  6482
## 6         help  positive  5598
## 7       thanks  positive  5114
## 8       please  positive  3545
## 9         fine  positive  2640
## 10         top  positive  2435
## 11       wrong  negative  2265
## 12    solution  positive  2097
## 13       block  negative  1560
## 14      failed  negative  1461
## 15        save  positive  1455
## 16 appreciated  positive  1427
## 17     extends  positive  1354
## 18      errors  negative  1312
## 19       thank  positive  1287
## 20        best  positive  1279
## 21       alert  negative  1270
## 22        join  positive  1243
## 23       empty  negative  1235
## 24      better  positive  1222
```

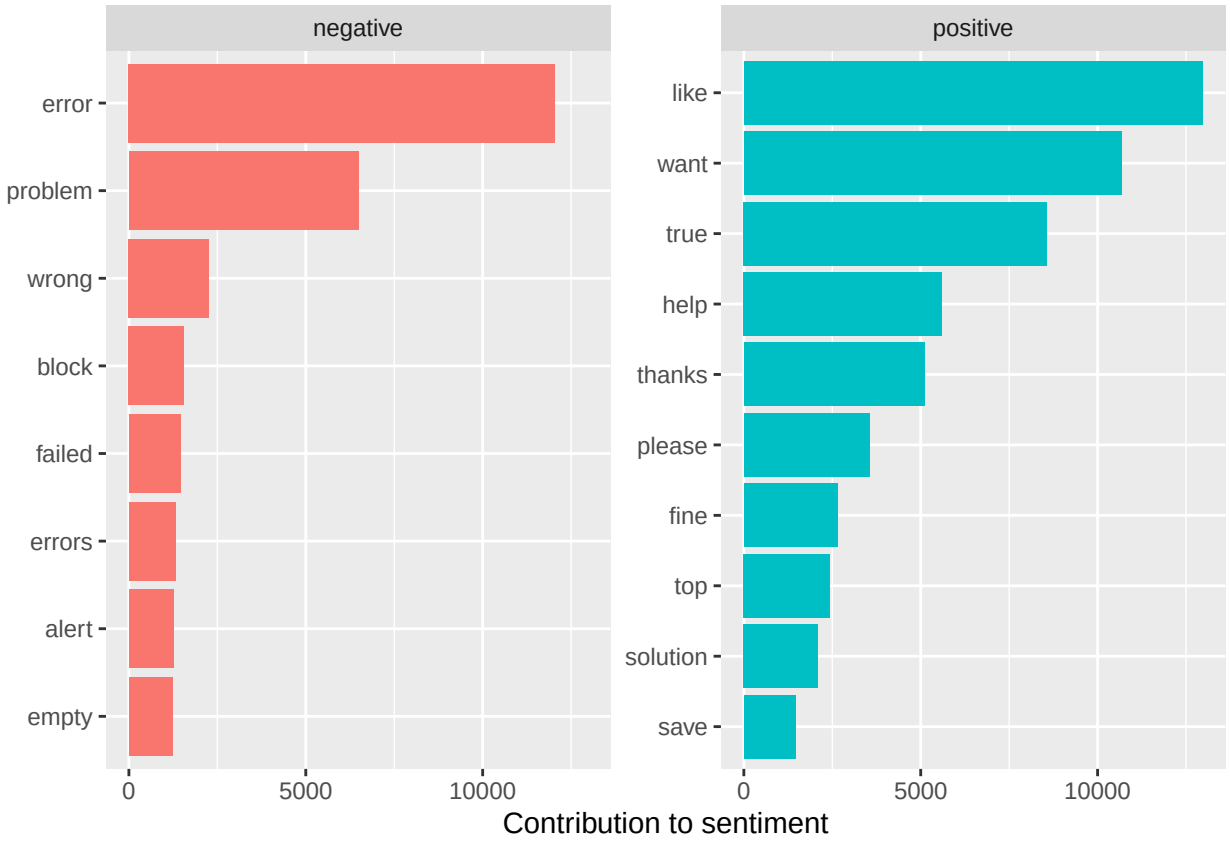
You can see that there are more unique negative words than positives while total number of positive words is more than negative ones.

```
q_small_afinn_count %>%
  group_by(sentiment) %>%
  summarize(total = sum(n), count = n(), .groups = 'drop')
```

```
## # A tibble: 2 x 3
##   sentiment total count
##   <chr>     <int> <int>
## 1 negative  27630     8
## 2 positive  62912    16
```

The graph shows the 10 most common negative and positive words contribution.

```
q_small_afinn_count %>%
  group_by(sentiment) %>%
  top_n(10, n) %>%
  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()
```



Term Frequency

Using term frequency and inverse document frequency allows us to find words that are characteristic for one document within a collection of documents.

```
q_words <- q_small %>%
  count(Id, word, sort = T) %>%
  ungroup()

total_words <- q_words %>%
  group_by(Id) %>%
  summarize(total = sum(n), .groups = 'drop')

q_words <- left_join(q_words, total_words, by = "Id")
```

The list arranged by descending tf_idf values and as you can see there are not meaningful words at all because it comes from short questions which includes related codes. As a result, it is hard to get a clear result when searching words that represent the selected tags. However, tf_idf is a strong tool to explore identifying words.

```
q_words %>%
  left_join(tags, by = "Id") %>%
  filter(tag %in% c( "ruby", "python", "php", "javascript", "java", "c")) %>%
  group_by(tag) %>%
  bind_tf_idf(word, tag, n) %>%
  arrange(desc(tf_idf))
```

```
## Warning: A value for tf_idf is negative:
```

```
##  Input should have exactly one row per document-term combination.
```

```
## # A tibble: 969,683 x 8
## # Groups:   tag [6]
##          Id word             n total tag          tf   idf   tf_idf
##       <int> <chr>        <int> <int> <chr>     <dbl> <dbl>    <dbl>
##  1 34518090 view.aspx       55  2615 python 0.000305  1.79 0.000547
##  2 25763270 volodymyr       32   445 ruby   0.000242  1.79 0.000434
##  3 34423850 gitlab          50   690 ruby   0.000379  1.10 0.000416
##  4 35423280 sampleitem     112   600 c      0.000222  1.79 0.000398
##  5 10807640 rich:column     64   435 java   0.000187  1.79 0.000335
##  6 24075890 mepager         24   702 ruby   0.000182  1.79 0.000326
##  7 27924020 local_mac       47   930 php    0.000182  1.79 0.000325
##  8 11572110 item_group      45   341 php    0.000174  1.79 0.000312
##  9 19782590 baseiar         42   482 php    0.000162  1.79 0.000291
## 10 38722680 billing_item    21   385 ruby   0.000159  1.79 0.000285
## # ... with 969,673 more rows
```

```
q_words %>%
  left_join(tags, by = "Id") %>%
  filter(tag %in% c( "ruby", "python", "php", "javascript", "java", "c")) %>%
  group_by(tag) %>%
  bind_tf_idf(word, tag, n) %>%
  select(-Id) %>%
  arrange(desc(tf_idf)) %>%
  top_n(6) %>%
  ggplot(aes(word, tf_idf, fill = tag)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~tag, scales = "free_y")+
  coord_flip() +
  theme(axis.text.x = element_text(angle = 30))
```

```
## Warning: A value for tf_idf is negative:
##  Input should have exactly one row per document-term combination.
```

```
## Selecting by tf_idf
```

Conclusion

As a result of this project,

- I learned the most common tags and words in the questions between 2008-2016.
- I observed the question count of programming languages by year and the trends about it.
- I approached to questions in terms of sentiments.
- I made tf_idf analysis of the dataset even the results were not encouraging.

It's hard to work on a machine learning algorithm because of convenience of the data. Topic modeling with Latent Dirichlet Allocation (LDA) seems to be applicable on this dataset and it is a further idea to follow for me.