

# Tidying Github DMCA Files

*Song Li*

*2019-04-24*

## GitHub DMCA Data Source

Assume the GitHub received DMCA mails repo is cloned in folder ../dmca.

```
dmca_src <- '../dmca'
```

## Tidying the Texts

List DMCA mails in each folder for a quick view. It's not necessary for the next steps, however.

```
dmca_files = list.files(dmca_src, recursive=TRUE, pattern='*/*.md')
head(dmca_files)
```

```
## [1] "2013/2013-03-06-LayerVault-counternotice.md"
## [2] "2013/2013-03-06-LayerVault.md"
## [3] "2013/2013-03-12-DxO-Labs.md"
## [4] "2013/2013-03-14-Electronic-Arts.md"
## [5] "2013/2013-03-21-DxO-Labs-counternotice.md"
## [6] "2013/2013-03-25-Apple-Inc.md"
```

Read content of each mail into a data frame.

```
library(readtext)
dmca_docs <- readtext(paste0(dmca_src, '*/*.md'))
head(dmca_docs)
```

```
## readtext object consisting of 6 documents and 0 docvars.
## # Description: data.frame [6 x 2]
##   doc_id      text
## * <chr>      <chr>
## 1 2013-03-06-LayerVault-counternotice.md "\"[private]\n\"..."
## 2 2013-03-06-LayerVault.md              "\"Attn: Copy\"..."
## 3 2013-03-12-DxO-Labs.md                 "\"Takedown N\"..."
## 4 2013-03-14-Electronic-Arts.md          "\"I write on\"..."
## 5 2013-03-21-DxO-Labs-counternotice.md  "\"March 21, \"..."
## 6 2013-03-25-Apple-Inc.md               "\"We represe\"..."
```

Tidy documents into sentences first.

```
library(dplyr)
library(tidytext)
library(stringr)

tidy_sentences <- dmca_docs %>%
  unnest_tokens(sentence, text, token='sentences') # by sentence
head(tidy_sentences)
```

```
## readtext object consisting of 6 documents and 0 docvars.
## # Description: data.frame [6 x 3]
##   doc_id      sentence      text
## * <chr>      <chr>      <chr>
```

```
## 1 20107-05-22-Pea~ 300 hudson street new york, ny 10013 05/18~ "\"\".~
## 2 20107-05-22-Pea~ pearson holds certain rights in the titles list~ "\"\".~
## 3 20107-05-22-Pea~ we have included below examples of locations on~ "\"\".~
## 4 20107-05-22-Pea~ you are not authorized to host, distribute, or ~ "\"\".~
## 5 20107-05-22-Pea~ pearson demands that you immediately and perman~ "\"\".~
## 6 20107-05-22-Pea~ this includes the copies available at the links~ "\"\".~
```

From the above, we can notice some obvious things to handle.

- Many lines are from the same submission template.
- Numbers are not so useful.
- "\_" in words are indication of italic that should be removed.
- In general, stop words are not interesting.
- Words like 'github.com' are too common for this specific corpus.

```
# Apply stop words from tidytext package then customize
data(stop_words)
custom_stop_words = c('github.com', 'www.github.com', 'https', 'http')
for (w in custom_stop_words){
  stop_words <- add_row(stop_words, word = w)
}

tidy_words <- tidy_sentences %>%
  filter(!str_detect(sentence, '^\\*\\*\\*')) %>% # remove lines from template
  mutate(sentence = str_replace_all(sentence, '_', '')) %>% # remove all "_"
  unnest_tokens(word, sentence) %>% # by word
  filter(!str_detect(word, '\\d')) %>% # remove digit only "words"
  anti_join(stop_words) # remove stop words
head(tidy_words)
```

```
##           doc_id      word
## 1 20107-05-22-Pearson.md  hudson
## 2 20107-05-22-Pearson.md  street
## 3 20107-05-22-Pearson.md   york
## 4 20107-05-22-Pearson.md    ny
## 5 20107-05-22-Pearson.md attention
## 6 20107-05-22-Pearson.md  writing
```

## Samples Using the Tidy Texts

Next we can play with the data. Group the data by year first.

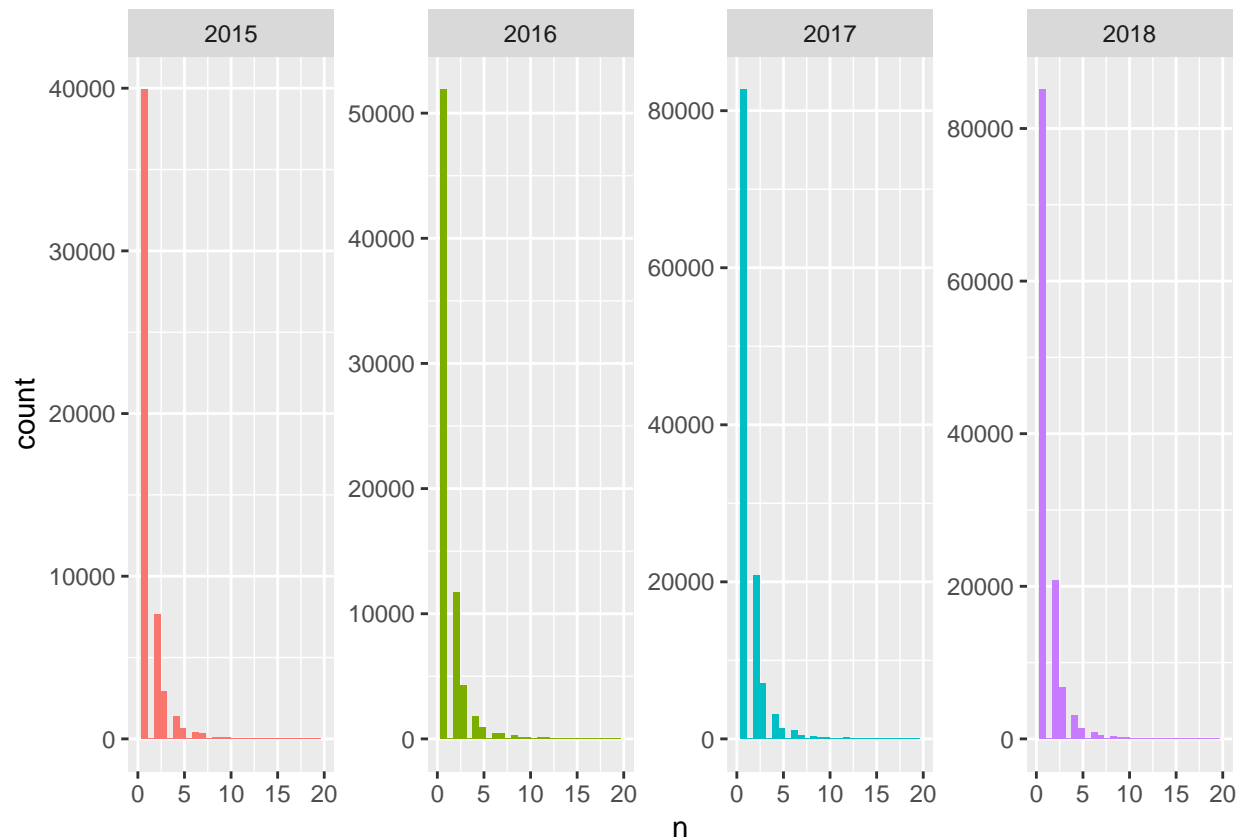
```
year_of_interest = c('2015', '2016', '2017', '2018')
year_words <- tidy_words %>%
  mutate(year = substr(doc_id, 0, 4)) %>%
  filter(is.element(year, year_of_interest)) %>%
  count(doc_id, word, year, sort=TRUE) %>%
  ungroup()
total_words <- year_words %>%
  group_by(year) %>%
  summarize(total = sum(n))
year_words <- left_join(year_words, total_words)
head(year_words)
```

```
## # A tibble: 6 x 5
##   doc_id      word      year      n total
##   <chr>      <chr>      <chr> <int> <int>
```

```
## 1 2016-06-08-Monotype.md fonts      2016    1023 165979
## 2 2016-06-08-Monotype.md helvetica 2016     979 165979
## 3 2018-04-05-HexRays.md  ida       2018     912 225686
## 4 2018-04-05-HexRays.md  dirtbags 2018     906 225686
## 5 2018-04-05-HexRays.md  vera     2018     906 225686
## 6 2018-04-05-HexRays.md  libs     2018     904 225686
```

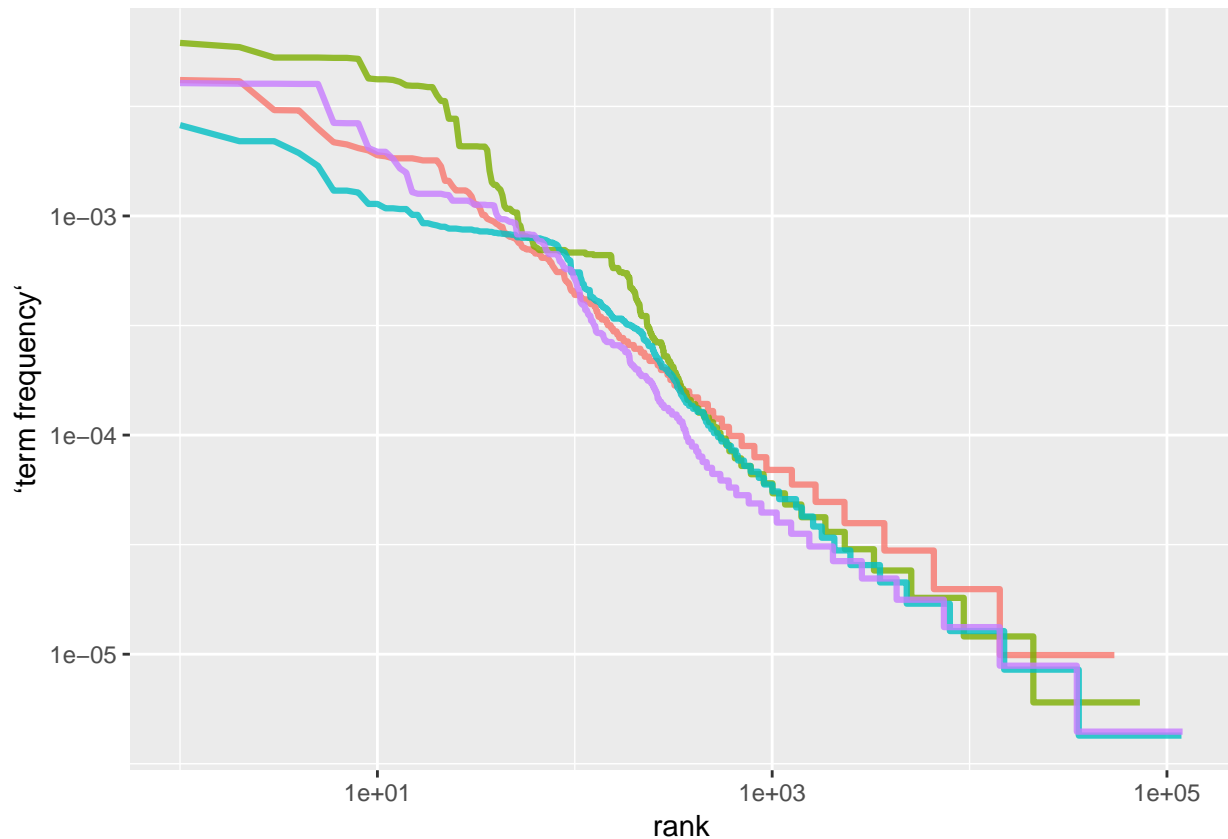
Plot the words count. Seems so many words appear just once or very few. Sum of words increases by year.

```
library(ggplot2)
ggplot(year_words, aes(n, fill = year)) +
  geom_histogram(show.legend = FALSE) +
  xlim(0, 20) +
  facet_wrap(~year, ncol = 4, scales = "free_y")
```



Plot term frequency to show Zipf's law.

```
freq_by_rank <- year_words %>%
  group_by(year) %>%
  mutate(rank = row_number(),
         `term frequency` = n/total)
ggplot(freq_by_rank, aes(rank, `term frequency`, color = year)) +
  geom_line(size = 1.1, alpha = 0.8, show.legend = FALSE) +
  scale_x_log10() +
  scale_y_log10()
```



TF-IDF can show what are the most specific requests in each year. These might be relatively lesser requests in fact. No jetbrains here in any year but it is one of the top frequent terms, probably because it's common every year.

```
year_words %>%
  filter(n > 10) %>%
  bind_tf_idf(word, year, n) %>%
  select(-total) %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  group_by(year) %>%
  top_n(10) %>%
  ungroup() %>%
  ggplot(aes(word, tf_idf, fill = year)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~year, ncol = 2, scales = "free") +
  coord_flip()
```

