Lecture 20: Working with Tokens

Taylor Arnold

NLP

We have been using basic string processing functions in **stringi** to perform basic web scraping and data manipulation tasks.

Today, we extend these ideas by using the **tidytext** package to parse the actual text and extract meaningful information from it.

tidytext

The basic idea of cleanNLP is to turn text into a data frame with one row per word. The basic usage is as follows:

```
library(tidytext)

input <- data_frame(text = c("Penguins are awesome.",
                             "Birds that can swim!"),
                    id = c("p1", "p2"))
data <- unnest_tokens(input, word, text)
```

```
data
```

```
## # A tibble: 7 x 2
##      id     word
##   <chr>    <chr>
## 1    p1 penguins
## 2    p1      are
## 3    p1  awesome
## 4    p2    birds
## 5    p2     that
## 6    p2      can
## # ... with 1 more rows
```

website text

Previously, when we scrapped data from Wikipedia we did not do anything with the raw text (though you should have in one of the associated labs). Here is how we could have grabbed the text from the Lagos page:

```
url <- "https://en.wikipedia.org/wiki/Lagos"
wpage <- data_frame(line = readLines(url))
wpage <- filter(wpage, stri_detect(line, fixed = "<p>"))
wpage <- mutate(wpage,
        line = stri_replace_all(line, "", regex = "<[^>]+>"))
```

Here is the result (using `stri_wrap` just for display purposes):

```
text <- stri_flatten(wpage$line, collapse = " ")
stri_wrap(text)[1:10]
```

```
##  [1] "Lagos / le  s/[11] (Yoruba: Èkó) is a city in the"
##  [2] "Nigerian state of Lagos. The city, with its adjoining"
##  [3] "conurbation, is the largest in Nigeria, as well as"
##  [4] "on the African continent. It is one of the fastest"
##  [5] "growing in the world,[12][13][14][15][16][17][18] and"
##  [6] "also one of the most populous urban agglomerations."
##  [7] "[19][20] Lagos is a major financial centre in Africa;"
##  [8] "the megacity has the highest GDP,[4] and also houses"
##  [9] "one of the largest and busiest ports on the continent."
## [10] "[21][22][23] Lagos initially emerged as a port city"
```

website text

Let's build a small example with just three cities:

```
urls <- c("https://en.wikipedia.org/wiki/Lagos",
          "https://en.wikipedia.org/wiki/London",
          "https://en.wikipedia.org/wiki/Saint_Petersburg")

df <- data_frame(city = c("Lagos", "London",
                          "Saint Petersburg"))
df$text <- NA
```

And cycle over these to extract a text column in our dataset:

```
for (i in 1:3) {
    wpage <- data_frame(line = readLines(urls[i]))
    wpage <- filter(wpage, stri_detect(line, fixed = "<p>"))
    wpage <- mutate(wpage,
        line = stri_replace_all(line, "", regex = "<[^>]+>"))

    df$text[i] <- stri_flatten(wpage$line, collapse = " ")
}
stri_length(df$text)
```

```
## [1] 31946 77565 72053
```

With **tidytext**, we can extract the tokens from these three pages, keeping the city name intact:

```
tokens <- unnest_tokens(df, word, text)
tokens
```

```
## # A tibble: 29,519 x 2
##    city    word
##   <chr>   <chr>
## 1 Lagos   lagos
## 2 Lagos  le  s
## 3 Lagos      11
## 4 Lagos  yoruba
## 5 Lagos     èkó
## 6 Lagos      is
## # ... with 2.951e+04 more rows
```

top tokens

Let's use our new grouping function to find the top words in each city page:

```
temp <- group_by(tokens, city, word)
temp <- count(temp, city)
temp <- group_by(temp, city)
top_n(temp, n = 3)
```

```
## # A tibble: 9 x 3
## # Groups:   city [3]
##     city   word     n
##    <chr>  <chr> <int>
## 1  Lagos  lagos   199
## 2  Lagos     of   198
## 3  Lagos    the   417
## 4 London london   409
## 5 London     of   520
## 6 London    the  1040
## # ... with 3 more rows
```

stop words

Two of the city names pop up, but the other words are just common, boring English terms. We can use a stopword list to remove these.

stop words

```
stop_words
```

```
## # A tibble: 1,149 x 2
##         word lexicon
##        <chr>   <chr>
## 1          a   SMART
## 2        a's   SMART
## 3       able   SMART
## 4      about   SMART
## 5      above   SMART
## 6 according   SMART
## # ... with 1,143 more rows
```

top tokens

The anti_join function returns the first dataset with all rows matching the second removed.

```
temp <- anti_join(tokens, stop_words)
temp <- group_by(temp, city, word)
temp <- count(temp, city)
temp <- group_by(temp, city)
top_n(temp, n = 3)
```

```
## # A tibble: 9 x 3
## # Groups:   city [3]
##     city   word     n
##    <chr>  <chr> <int>
## 1  Lagos   city    33
## 2  Lagos island    35
## 3  Lagos  lagos   199
## 4 London    160    71
## 5 London   city   116
## 6 London london   409
## # ... with 3 more rows
```

custom stop words

We can build a better list using our data itself:

```
temp <- group_by(tokens, word)
temp <- ungroup(count(temp))
custom_stop_words <- top_n(temp, n = 300)
custom_stop_words
```

```
## # A tibble: 320 x 2
##    word      n
##   <chr> <int>
## 1     1    16
## 2   100    12
## 3    12    16
## 4    16    13
## 5   160   156
## 6  18th    12
## # ... with 314 more rows
```

top tokens

Which yields these much improved results:

```
temp <- anti_join(tokens, custom_stop_words)
temp <- group_by(temp, city, word)
temp <- count(temp, city)
temp <- group_by(temp, city)
```

top tokens

```
top_n(temp, n = 3)
```

```
## # A tibble: 13 x 3
## # Groups:   city [3]
##     city     word     n
##    <chr>    <chr> <int>
## 1  Lagos   africa     9
## 2  Lagos    ikoyi     9
## 3  Lagos nigerian    10
## 4 London     born    10
## 5 London    found    10
## 6 London    roman     9
## # ... with 7 more rows
```

idea for using this

Here are some ways that you can use this in your data analysis projects:

- find top word or words for each location and plot on a map
- count number of words in each page and use this as metadata
- create a list of interesting words and use `semi_join` (the opposite of `anti_join`) to filter only those words that are on this list