16 - Word Cloud

Data and Information Engineering

SYS 2202 | Fall 2019

16-wordcloud.pdf

Contents

Required Packages and Data

```r
library(SnowballC)   # install.packages("SnowballC")
library(tidytext)    # for working with text in a tidy way
library(tidyverse)   # for data manipulation
```

1 Text Mining

We are going to be working with functions from `tidyverse` (e.g., `stringr`, `dplyr`) and `tidytext` to do some basic text mining and build a word cloud. A good introduction to the `tidytext` package is the free book Text Mining with R (by Silge and Robinson)

1.1 Goals

We are going to analyze a set of documents related to business analytics. Specifically, we are going to break a document down into a frequency distribution of its words and examine the most frequent (and potentially the most important words).

Like all topics we have covered this semester, we are only scratching the surface of what is possible in the field of text mining and text analytics. Document clustering, author attribution, sentiment analysis, natural language processing (NLP), entity extraction, word and document networks, etc. are just some examples of where you can go with this. Hopefully, we cover enough so you can start to imagine and think about what is possible with text data.

We have 26 plain text (.txt) documents. We are going to read these into R and create a character vector where each element is a document.

1.2 Read in Text Documents

Here we will do this manually with a loop and `read_file()`. The data files can be found here https://raw.githubusercontent.com/mdporter/ST597/master/data/BA_skills/ba-xx.txt, where `xx` is two digits between `01-26`.

```
#- read in all documents
base_url = "https://raw.githubusercontent.com/mdporter/ST597/master/data/BA_skills/ba-"
end_url = ".txt"

docs = character(26)        # create vector of 26 blank elements
for(i in 1:26){             # for loop to set the value of i
  file_num = str_pad(i, width=2, side="left", pad="0")  # make 2 digit number
  url = str_c(base_url, file_num, end_url)
  docs[i] = read_file(url)
}
```

```
#- example document
# docs[22]                          # raw form
cat(str_wrap(docs[22], width=75)) # displayed form
#> My very simple take: Programming in R/Python both for data analysis and
#> for visualization. Equally important more or less in my view. Beyond that,
#> hands-on data set analysis. Teach people to look at data and decide the
#> best approach themselves rather than telling them which approach to take
#> and grading on their ability to do so. Manager of Analytics
```

1.3 Make it tidy

A tidy text format is a table with one *token* per row.

- A token can be a: word, n-gram, sentence, line, paragraph, tweet
- This will let us use the power of our tidyverse functions

First, we will get the documents into a tibble:

```
text_df = tibble(document = 1:length(docs), text=docs)
```

Then, we can *unnest* the documents so there is one *word* per row:

```
library(tidytext)  # for unnest_tokens()
(word_df = text_df %>%
  unnest_tokens(output=word,     # name of new column
                input=text,      # the column to unnest
                token="words",   # what to use as the tokens
                to_lower=TRUE,   # convert all words to lowercase
                strip_punct=TRUE)) # remove punctuation
#> # A tibble: 5,706 x 2
#>   document word
#>      <int> <chr>
#> 1        1 obviously
#> 2        1 i
#> 3        1 know
#> 4        1 more
#> 5        1 about
#> 6        1 basketball
#> # ... with 5,700 more rows
```

1.4 Explore

Notice that there are many words are uninteresting: “to”, “and”, “of”, “the”. We also have lots of numbers

```
word_df %>%
  filter(str_detect(word, "[0-9]")) %>%
  distinct()
#> # A tibble: 45 x 2
#>   document word
#>      <int> <chr>
#> 1        1 2
#> 2        2 d3
#> 3        3 100
#> 4        4 8
#> 5        9 1
#> 6        9 2
#> # ... with 39 more rows
```

And, consider if any of these words should be considered together?

```
word_df %>%
  filter(str_detect(word, pattern="[Aa]naly")) %>%
  pull() %>% unique()
#> [1] "analytics"  "analysts"   "analyzing"  "analyze"    "analytical"
#> [6] "analysis"   "analyst"    "analytic"
```

2 Transformations

Before we start our data analysis and modelling, it is often necessary to modify the text in some ways. For example, the basic step of extracting the words is one task that is usually performed. To help understand the information in text, we can also:

- remove whitespace
- convert letters to same case (e.g., lowercase) [*already done in* `unnest_tokens()`]
- removing punctuation [*already done in* `unnest_tokens()`]

- removing *stop words*, common words that do not carry much meaning to the analysis (e.g., “an”, “a”, “the”)
- removing numbers or other non-text characters
- etc.

2.1 Cleaning Text

Most of these can be done with a combination of `mutate()`, `anti_join()`, and `filter()`:

```
#-- Stop Words
(data(stop_words, package="tidytext"))
#> [1] "stop_words"

#-- List of software (named list using desired case)
software_list = c(r='R', sql='SQL', python ='Python',
                  tableau='Tableau', d3='D3', mysql='MySQL',
                  sas='SAS', spss='SPSS', excel='Excel')

#-- Clean text
clean_df = word_df %>%
  mutate(word = recode(word, !!!software_list)) %>% # convert important words
  mutate(word = str_trim(word, side="both"),     # remove extra whitespace
         word = str_remove_all(word, "'"),        # remove apostrophes
         word = str_remove_all(word, "[0-9]")    # remove numbers
         #word = str_remove(word, "[[:punct:]]+") # remove punctuation
         #word = str_to_lower(word) # unnest_tokens() converted to lowercase
         ) %>%
  anti_join(stop_words, by="word") %>%    # remove rows with stopwords
  filter(word != "")                      # ignore blanks
```

2.2 Stemming (and Lemmatization)

We noticed a potential problem when multiple words correspond to the same concept or idea. For example, “analyzing”, “analyze”, and “analysis” could potentially be grouped together for frequency analysis (note: this could potentially be done after processing, but then we will be forced to deal with much larger data).

Stemming and *Lemmatization* refer to the process of reducing words to a base or root form so multiple words that carry similar meaning/information can be combined. *Stemming* uses letter patterns (think regex) while *lemmatization* finds the part of speech to help guide the stemming. Some more details can be found here http://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html.

Stemming can be achieved using Porter’s (not me!) stemming algorithm http://snowball.tartarus.org/algorithms/porter/stemmer.html. But this requires a function from the `SnowballC` package, which must be installed and loaded. Here is an example of how the stemming works

```
library(SnowballC)    # for wordStem() function
clean_df %>%
  filter(str_detect(word, pattern="[Aa]naly")) %>%
  mutate(stemmed=wordStem(word)) %>%
  distinct(word, stemmed) %>%
  knitr::kable()
```

word	stemmed
analytics	analyt
analysts	analyst
analyzing	analyz

word	stemmed
analyze	analyz
analytical	analyt
analysis	analysi
analyst	analyst
analytic	analyt

Stemming may not great for word cloud, because the stemmed version may not make much sense. One approach is to stem the words, then use one representative word in the word cloud. However, we will not go into this much detail here.

3 Word Counts

In the tidy format, getting word frequencies is easy:

- n is the total number of times a word appears in all the documents (so a word that appears more than once in a document will be counted more than once.)
- n_docs is the number of documents that contain the word (so a word that appears more than once in a document will only be counted once.)

```
(counts = clean_df %>%
  group_by(word) %>%
  summarize(n = n(),
            n_docs = n_distinct(document)) %>%
  arrange(-n))
#> # A tibble: 1,145 x 3
#>   word          n n_docs
#>   <chr>     <int>  <int>
#> 1 data         81     20
#> 2 analytics    69     22
#> 3 skills       28     12
#> 4 business     19     13
#> 5 program      18      7
#> 6 R            17      9
#> # ... with 1,139 more rows
```

We can check the frequence of software mentions

```
counts %>% filter(word %in% software_list) %>% knitr::kable()
```

word	n	n_docs
R	17	9
SQL	13	9
Python	12	9
Tableau	5	4
SAS	3	3
Excel	2	2
MySQL	1	1
SPSS	1	1

(But we don't know from the words alone if these are positive or negative metnions. e.g., document 14 “Move away from Excel”).

3.1 Word Clouds

A word cloud is a graphical representation of text that sizes and colors the words. Size is usually considered to be proportional to the frequency of the word's occurrence, but in general could be related to some other measure of *importance*.

The R package `ggwordcloud` implements a wordcloud geom for use with `ggplot2`. The package has a helpful webpage with examples: `ggwordcloud` R package help

```
library(ggwordcloud)
set.seed(2019)    # the text location is based on random initialization
counts %>% filter(n > 6 | word %in% software_list) %>%
  mutate(software = word %in% software_list) %>%
  ggplot() +
  geom_text_wordcloud(aes(label=word, size=n, color=software)) +
  scale_size_area(max_size = 15) +
  theme_minimal()
```