Text mining in practice - Bag of Words - data cleaning

Digital Humanities DSC 105 Spring 2023

Marcus Birkenkrahe

February 23, 2023

README

- This lecture closely follows the 3rd part of the DataCamp lesson "Jumping into Text Minin with Bag-of-Words" by Ted Kwartler, part of his course on "Text Mining with Bag-of-Words in R".

- Download and open the practice file `4_cleaning_practice.org` from GitHub to code along.

Cleaning and preprocessing text

- Base R cleaning functions:

TM Function	Description	Before	After
`tolower()`	Makes all text lowercase	`Starbucks is from Seattle.`	`starbucks is from seattle.`
`removePunctuation()`	Removes punctuation like periods and exclamation points	`Watch out! That coffee is going to spill!`	`Watch out That coffee is going to spill`
`removeNumbers()`	Removes numbers	`I drank 4 cups of coffee 2 days`	`I drank cups of coffee days ago.`
`stripWhiteSpace()`	Removes tabs and extra spaces	`I like coffee.`	`I like coffee.`
`removeWords()`	Removes specific words (e.g. "the", "of") defined by the data scientist	`The coffee house and barista he visited were nice, she said hello.`	`The coffee house barista visited nice, said hello.`

Figure 1: Text mining functions

```
tolower("WHATSHAPPENING")
```

```
[1] "whatshappening"
```

Is `tm` loaded?

```
library(tm)
search()
```

```
 [1] ".GlobalEnv"               "package:SnowballC"
 [3] "package:tm"               "package:NLP"
 [5] "package:qdap"             "package:RColorBrewer"
 [7] "package:qdapTools"        "package:qdapRegex"
 [9] "package:qdapDictionaries" "package:stringr"
```

```
[11] "package:stringi"          "ESSR"
[13] "package:stats"            "package:graphics"
[15] "package:grDevices"        "package:utils"
[17] "package:datasets"         "package:methods"
[19] "Autoloads"                "package:base"
```

Create sample corpus

- Create corpus so we have something to work with:

```
library(tm)
coffee_df <- read.csv("../data/coffee.csv") # dataframe
coffee_vec <- coffee_df$text # vector
coffee_src <- VectorSource(coffee_vec) # source
coffee_corpus <- VCorpus(coffee_src)
```

Invoke functions separately

- Each function separately:

```
t <- coffee_corpus[[2]]
content(t)
tolower(t)
content(removePunctuation(t))
content(removeNumbers(t))
stripWhitespace("There      is    a     lot    of space    .")
content(removeWords(t, c("in", "the", "at")))
```

```
Error: object 'coffee_corpus' not found
Error in UseMethod("content", x) :
  no applicable method for 'content' applied to an object of class "function"
Error in as.character(x) :
  cannot coerce type 'closure' to vector of type 'character'
Error in UseMethod("removePunctuation") :
  no applicable method for 'removePunctuation' applied to an object of class "func
Error in UseMethod("removeNumbers") :
  no applicable method for 'removeNumbers' applied to an object of class "functio
[1] "There is a lot of space ."
Error in UseMethod("removeWords", x) :
  no applicable method for 'removeWords' applied to an object of class "function"
```

Nest functions

- Nested functions:

```
content(t)
tc <-
  tolower(
    content(
      removePunctuation(
        removeNumbers(
          removeWords(t, c("in", "the", "at", "will", "only", "this"))))))
tc
```

```
Error in UseMethod("content", x) :
  no applicable method for 'content' applied to an object of class "function"
Error in UseMethod("removeWords", x) :
  no applicable method for 'removeWords' applied to an object of class "function"
Error: object 'tc' not found
```

Create a pipeline of functions with `|>`

- Function pipeline

```
content(t)
t |>
  removeWords(c("in", "the", "at", "this", "will", "only"))  |>
  removeNumbers() |>
  removePunctuation() |>
  content() |>
  tolower()
```

```
Error in UseMethod("content", x) :
  no applicable method for 'content' applied to an object of class "function"
Error in UseMethod("removeWords", x) :
  no applicable method for 'removeWords' applied to an object of class "function"
```

Apply function with the `tm_map` wrapper

- Apply functions to the whole corpus with the `tm_map` wrapper:

```
## run functions as an argument to tm_map
tm_map(coffee_corpus, removeNumbers) -> cc_no_numbers
tm_map(coffee_corpus, removePunctuation) -> cc_no_punctuation
## check content
content(cc_no_numbers[[2]])
content(cc_no_punctuation[[2]])

Error in tm_map(coffee_corpus, removeNumbers) :
  object 'coffee_corpus' not found
Error in tm_map(coffee_corpus, removePunctuation) :
  object 'coffee_corpus' not found
Error in content(cc_no_numbers[[2]]) : object 'cc_no_numbers' not found
Error in content(cc_no_punctuation[[2]]) :
  object 'cc_no_punctuation' not found
```

- These functions live in different environments:

```
library(tm)
library(qdap)
environment(tolower)
environment(removePunctuation)
environment(removeNumbers)
environment(removeWords)
environment(stripWhitespace)
environment(replace_abbreviation)

<environment: namespace:base>
<environment: namespace:tm>
<environment: namespace:tm>
<environment: namespace:tm>
<environment: namespace:tm>
<environment: namespace:qdap>
```

- To work, `tm_map` must transform a function from another package with `content_transformer` (this also takes a lot longer):

```
library(tm)
library(qdap)
## where is replace_abbreviation?
```

```
environment(replace_abbreviation)
## run this function with tm_map - store result in repl
tm_map(coffee_corpus, content_transformer(replace_abbreviation)) -> repl
## print content with and without abbrevs replaced
content(coffee_corpus[[2]])
content(repl[[2]])
```

```
<environment: namespace:qdap>
Error in tm_map(coffee_corpus, content_transformer(replace_abbreviation)) :
  object 'coffee_corpus' not found
Error in content(coffee_corpus[[2]]) : object 'coffee_corpus' not found
Error in content(repl[[2]]) : object 'repl' not found
```

Word stemming with `stemDocument`

- Word stemming with `tm::stemDocument`: requires installing `SnowballC`:

```
library(qdap)
library(SnowballC)
stem_words <- stemDocument(c("complicatedly",
                             "complicated",
                             "complication",
                             "complicate"))
stem_words
```

```
[1] "complic" "complic" "complic" "complic"
```

- Interestingly, the stem of `Complicate` is recognized, but not the stem of `ComplicatE` or `COMPLICATE`.

Completing word stems with `stemCompletion`

- You can complete the words using a single word dictionary (i.e. all stems are mapped onto a single word):

```
stemCompletion(stem_words, c("complicate"))
```

```
     complic      complic      complic      complic
"complicate" "complicate" "complicate" "complicate"
```

- You can use a corpus as completion dictionary:

```
stemCompletion(stem_words, coffee_corpus)
```

```
Error in stemCompletion(stem_words, coffee_corpus) :
  object 'coffee_corpus' not found
```

- `coffee_corpus` does not contain a matching word!
- Create a new corpus just for `stem_words` to test the function `stemCompletion`, starting with the vector `c("complicate")`:

```
my_vec <- c("complicate")
my_src <- VectorSource(my_vec)
my_corpus <- VCorpus(my_src)
stemCompletion(stem_words, my_corpus)
```

```
     complic      complic      complic      complic
"complicate" "complicate" "complicate" "complicate"
```

Full-text corpus data online

- One can look for "full-text corpus data" online (link) - it's fast but you only have a limited number of (free) searches per day.
- What's interesting about this: "Marxism" relates to Karl Marx, who came up with his theories in the 1840s. How then could "marxism" be mentioned in books published before that date?

NEXT Cleaning with `qdap`

- To see the full range of arguments of a function, pass the function name as an argument to `args()` - e.g. for `qdap::bracketX`:

```
library(qdap)
args(bracketX)
```

```
function (text.var, bracket = "all", missing = NULL, names = FALSE,
    fix.space = TRUE, scrub = fix.space)
NULL
```

These are the most widely used online corpora, and they are used for many different purposes by teachers and researchers at universities throughout the world. In addition, the corpus data (e.g. full-text, word frequency) has been used by a wide range of companies in many different fields, especially technology and language learning.

The links below are for the free online interface. You can also purchase and download the corpora for use on your own computer.

Corpus (see tour)	Download	# words	Dialect	Time period	Genre(s)
News on the Web (NOW)		16.8 billion+	20 countries	2010-yesterday	Web: News
iWeb: The Intelligent Web-based Corpus		14 billion	6 countries	2017	Web
Global Web-Based English (GloWbE)		1.9 billion	20 countries	2012-13	Web (incl blogs)
Wikipedia Corpus		1.9 billion	(Various)	2014	Wikipedia
Coronavirus Corpus		1.5 billion	20 countries	Jan 2020-Dec 2022	Web: News
Corpus of Contemporary American English (COCA)		1.0 billion	American	1990-2019	Balanced
Corpus of Historical American English (COHA)		475 million	American	1820-2019	Balanced
The TV Corpus		325 million	6 countries	1950-2018	TV shows
The Movie Corpus		200 million	6 countries	1930-2018	Movies
Corpus of American Soap Operas		100 million	American	2001-2012	TV shows
Hansard Corpus		1.6 billion	British	1803-2005	Parliament
Early English Books Online		755 million	British	1470s-1690s	(Various)
Corpus of US Supreme Court Opinions		130 million	American	1790s-present	Legal opinions
TIME Magazine Corpus		100 million	American	1923-2006	Magazine
British National Corpus (BNC) *		100 million	British	1980s-1993	Balanced
Strathy Corpus (Canada)		50 million	Canadian	1920s-2000s	Balanced
CORE Corpus		50 million	6 countries	2014	Web
From Google Books n-grams (compare)					
American English		155 billion	American	1500s-2000s	(Various)
British English		34 billion	British	1500s-2000	(Various)

Figure 2: English language corpora (english-corpora.org)

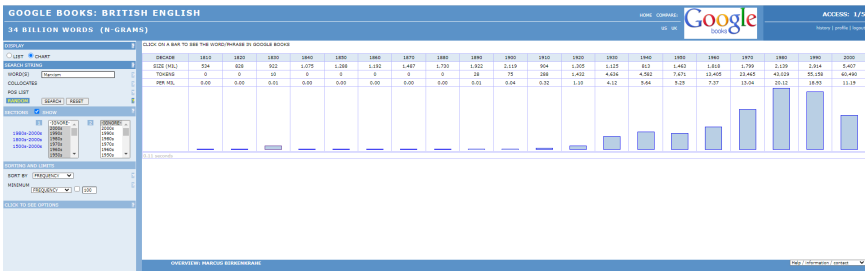

Figure 3: Google Books corpora - search example "Marxism"

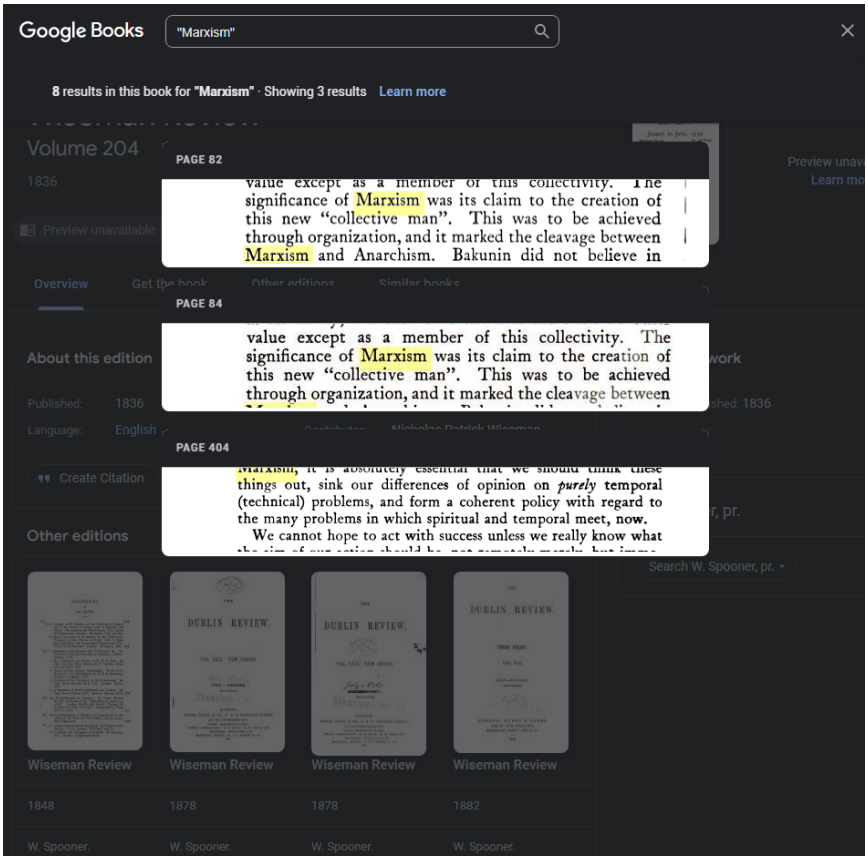

Figure 4: Google Books corpora - search example "Marxism" - results

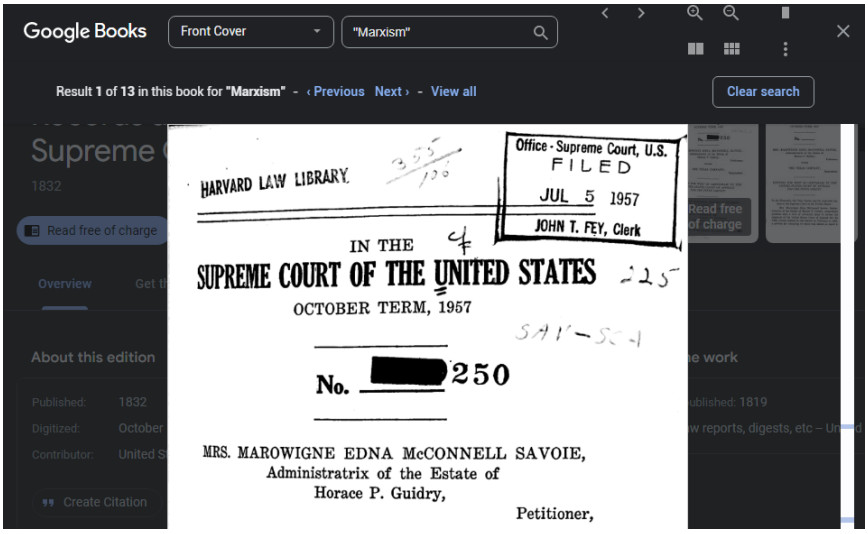

Figure 5: Google Books corpora - search example "Marxism" - results

- To find out more, e.g. about the options for the parameter `bracket`, look at the `help` page (when you do this in an Emacs Org-mode code block, interrupt the process manually with `C-g` to go on):

  ```
  help(bracketX)
  ```

Text cleaning functions in `qdap`

- The `qdap` package offers other text cleaning functions:
  - `bracketX()`: Remove all text within brackets (e.g. "It's (so) cool" becomes "It's cool", "Yes" becomes "Yes")
  - `replace_number()`: Replace numbers with their word equivalents (e.g. "2" becomes "two")
  - `replace_abbreviation()`: Replace abbreviations with their full text equivalents (e.g. "Sr" becomes "Senior")
  - `replace_contraction()`: Convert contractions back to their base words (e.g. "shouldn't" becomes "should not")
  - `replace_symbol()`: Replace common symbols with their word equivalents (e.g. "$" becomes "dollar")

Test text cleaning functions in qdap

- Define a sample text vector:

```
## define text vector
text <-
  "<b>She</b> woke up at       6 A.M. It\'s so
   early!  She was only 10% awake and began drinking
   coffee in front of her computer."
text
```

```
[1] "<b>She</b> woke up at       6 A.M. It's so\n   early!  She was only 10% awake
```

- Remove text within brackets:

```
text
bracketX(text)
```

```
[1] "<b>She</b> woke up at       6 A.M. It's so\n   early!  She was only 10% awake
[1] "She woke up at 6 A.M. It's so early! She was only 10% awake and began drinkin
```

- Replace all numbers with words:

```
text
replace_number(text)
```

```
[1] "<b>She</b> woke up at       6 A.M. It's so\n   early!  She was only 10% awake
[1] "<b>She</b> woke up at six A.M. It's so early! She was only ten% awake and beg
```

- Replace abbreviations:

```
text
replace_abbreviation(text)
```

```
[1] "<b>She</b> woke up at       6 A.M. It's so\n   early!  She was only 10% awake
[1] "<b>She</b> woke up at 6 AM It's so early! She was only 10% awake and began dr
```

- Replace contractions:

```
text
replace_contraction(text)
```

```
[1] "<b>She</b> woke up at       6 A.M. It's so\n   early!  She was only 10% awake
[1] "<b>She</b> woke up at 6 A.M. it is so early! She was only 10% awake and begar
```

- Replace symbols with words:

```
text
replace_symbol(text)
```

```
[1] "<b>She</b> woke up at       6 A.M. It's so\n   early!  She was only 10% awake
[1] "<b>She</b> woke up at 6 A.M. It's so early! She was only 10 percent awake and
```

- Run all of these on `text` together using a pipeline `|>`:

```
text |>
  bracketX() |>
  replace_number() |>
  replace_abbreviation() |>
  replace_contraction() |>
  replace_symbol()
```

```
[1] "She woke up at six AM it is so early! She was only ten percent awake and bega
```

- Run all of these on `text` together as *nested* functions:

```
bracketX(
  replace_number(
    replace_abbreviation(
      replace_contraction(
        replace_symbol(text)))))
```

```
[1] "She woke up at six AM it is so early! She was only ten percent awake and bega
```

NEXT stop: `stopwords`!