Text mining in practice - Bag of Words - TDM and DTM

Digital Humanities DSC 105 Spring 2023

Marcus Birkenkrahe

March 24, 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 `6_tdm_dtm_practice.org` from GitHub to code along.

Create clean corpus

- Load coffee tweet data

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

- Define function to clean corpus:

```
clean_corpus <- function(corpus) {
  corpus <- tm_map(corpus,
                   removePunctuation)
  corpus <- tm_map(corpus,
                   content_transformer(tolower))
```

```
  corpus <- tm_map(corpus,
                   removeWords,
                   words = c(stopwords("en"), "coffee"))
  corpus <- tm_map(corpus,
                   stripWhitespace)
  return(corpus)
}
```

- Run function to create `clean_corp`:

```
clean_corpus <- function(corpus) {
  corpus <- tm_map(corpus,
                   removePunctuation)
  corpus <- tm_map(corpus,
                   content_transformer(tolower))
  corpus <- tm_map(corpus,
                   removeWords,
                   words = c(stopwords("en"), "coffee"))
  corpus <- tm_map(corpus,
                   stripWhitespace)
  return(corpus)
}
clean_corp <- clean_corpus(coffee_corpus)
```

- Check the results: R objects and print original and cleaned tweet:

```
ls()
## original tweet
content(coffee_corpus[[999]])
## lower case, no punctuation, no stopwords, no "coffee"
content(clean_corp[[999]])
```

```
 [1] "clean_corp"    "clean_corpus"  "coffee_corpus" "coffee_df"
 [5] "coffee_dtm"    "coffee_m"      "coffee_src"    "coffee_tdm"
 [9] "coffee_vec"    "coffee_wfm"    "i"
[1] "First morning coffee after Ramadan http://t.co/ZEu6cl9qGY"
[1] "first morning ramadan httptcozeu6cl9qgy"
```

TDM vs DTM

	Tweet 1	Tweet 2	Tweet 3	…	Tweet N
Term 1	0	0	0	0	0
Term 2	1	1	0	0	0
Term 3	1	0	0	0	0
…	0	0	3	1	1
Term M	0	0	0	1	0

Term Document Matrix (TDM)

	Term 1	Term 2	Term 3	…	Term M
Tweet 1	0	1	1	0	0
Tweet 2	0	1	0	0	0
Tweet 3	0	0	0	3	0
…	0	0	0	1	1
Tweet N	0	0	0	1	0

Document Term Matrix (DTM)

- Create TDM with `tm::TermDocumentMatrix` and print the structure

```
coffee_tdm <- TermDocumentMatrix(clean_corp)
str(coffee_tdm)
```

```
List of 6
 $ i       : int [1:7391] 188 765 1759 2829 30 347 518 649 792 1028 ...
 $ j       : int [1:7391] 1 1 1 1 2 2 2 2 2 2 ...
 $ v       : num [1:7391] 1 1 1 1 1 1 1 1 1 1 ...
 $ nrow    : int 3076
 $ ncol    : int 1000
 $ dimnames:List of 2
  ..$ Terms: chr [1:3076] "0630" "1000" "1026" "1030" ...
  ..$ Docs : chr [1:1000] "1" "2" "3" "4" ...
 - attr(*, "class")= chr [1:2] "TermDocumentMatrix" "simple_triplet_matrix"
 - attr(*, "weighting")= chr [1:2] "term frequency" "tf"
```

- Transpose it with the `base::t` function:

```
coffee_dtm <- t(coffee_tdm)
str(coffee_dtm)
```

```
List of 6
 $ i       : int [1:7391] 1 1 1 1 2 2 2 2 2 2 ...
 $ j       : int [1:7391] 188 765 1759 2829 30 347 518 649 792 1028 ...
 $ v       : num [1:7391] 1 1 1 1 1 1 1 1 1 1 ...
 $ nrow    : int 1000
```

```
 $ ncol    : int 3076
 $ dimnames:List of 2
  ..$ Docs : chr [1:1000] "1" "2" "3" "4" ...
  ..$ Terms: chr [1:3076] "0630" "1000" "1026" "1030" ...
 - attr(*, "class")= chr [1:2] "DocumentTermMatrix" "simple_triplet_matrix"
 - attr(*, "weighting")= chr [1:2] "term frequency" "tf"
```

- `t` does the same thing as `DocumentTermMatrix`:

```
identical(coffee_dtm,DocumentTermMatrix(clean_corp))
```

```
[1] TRUE
```

- The `qdap` package relies on a Word Frequency Matrix (WFM):

	Tweet 1
Term 1	0
Term 2	1
Term 3	1
...	0
Term M	0

Word Frequency Matrix (WFM)

```
library(qdap)
coffee_wfm <- wfm(coffee_df$text)
str(coffee_wfm)
head(coffee_wfm)
```

```
 'wfm' num [1:3259, 1] 1 1 1 1 1 2 388 1 1 1 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:3259] "'" "'am" "'decaf'" "'di" ...
  ..$ : chr "all"
all
'         1
'am       1
```

```
'decaf'   1
'di'      1
'never'   1
'roya'    2
```

- When should you use a TDM instead of DTM?

  Answer: when you want the terms (words) as rows and documents as columns.

Analyze the document-term matrix (DTM)

	Term 1	Term 2	Term 3	…	Term M
Tweet 1	0	1	1	0	0
Tweet 2	0	1	0	0	0
Tweet 3	0	0	0	3	0
…	0	0	0	1	1
Tweet N	0	0	0	1	0

Document Term Matrix (DTM)

- The DTM is useful when you are comparing authors within rows, or when the data is arranged chronologically and you want to preserve the time series (of records or rows).
- Let's look at these matrices:

```
class(coffee_dtm)
class(coffee_tdm)
```

```
[1] "DocumentTermMatrix"    "simple_triplet_matrix"
[1] "TermDocumentMatrix"    "simple_triplet_matrix"
```

- We want to reclassify the object `as.matrix` to examine it more closely.
- Print the `coffee_dtm` object for `clean_corp`

```
coffee_dtm
```

```
<<DocumentTermMatrix (documents: 1000, terms: 3076)>>
Non-/sparse entries: 7391/3068609
Sparsity           : 100%
Maximal term length: 27
Weighting          : term frequency (tf)
```

- Convert the object to a `matrix` and print the dimension- how many tweets and how many terms does the matrix contain?

```
coffee_m <- as.matrix(coffee_dtm)
dim(coffee_m) # rows x columns
```

```
[1] 1000 3076
```

- Have a look at the upper left and lower right corner of the matrix:

```
coffee_m[1:5,1:10]
coffee_m[995:1000,3071:3076]
```

```
    Terms
Docs 0630 1000 1026 1030 110 1100 11am 1214 1230 1239
   1    0    0    0    0   0    0    0    0    0    0
   2    0    0    0    0   0    0    0    0    0    0
   3    0    0    0    0   0    0    0    0    0    0
   4    0    0    0    0   0    0    0    0    0    0
   5    0    0    0    0   0    0    0    0    0    0
      Terms
Docs   zaykennedy69 zeledmalegisele ziggy zokuhq zombie zzzquil
  995             0               0     0      0      0       0
  996             0               0     0      0      0       0
```

```
997             0                 0     0     0     0     0
998             0                 0     0     0     0     0
999             0                 0     0     0     0     0
1000            0                 0     0     0     0     0
```

- Print the subset of `coffee_m` containing documents 25 through 35 and the terms "hot" and "starbucks":

```
coffee_m[25:35,c("hot","starbucks")]
```

```
    Terms
Docs hot starbucks
  25   0         0
  26   0         1
  27   0         1
  28   1         0
  29   0         0
  30   0         0
  31   1         0
  32   0         0
  33   0         0
  34   0         1
  35   0         0
```

☐ How would you phrase this result?

- Print the tweets 25 through 35 from `clean_corp`:

```
for (i in 25:35) print(content(clean_corp[[i]]))
```

```
[1] "sometimes start dancing table can"
[1] "rt leslieks starbucks morning free tomorrow amp going wfriends get gunsense r
[1] "starbucks best confessyourunpopularopinion"
[1] "rt themindblowing fat burning foods grapefruit watermelon berries hot peppers
[1] "witnessed girl pet bird pooping shoulder amp girl sharing ice cream dog shop
[1] " biological watch thinks 543 pm want arabic cake"
[1] " wanna lay couch blankets hot watch old movies"
[1] "rt dreyess1 rehab addicts"
[1] "finally home food beer fridge also tomorrow will rough ordered pizza rat time
[1] "just drank entire venti starbucks amp still think im gonna fall asleep wtf"
[1] "rt uberfacts 1000 chemicals single cup 26 tested half caused cancer "
```

- You can also loop over these with `while`:

```
i = 25
while (i <= 35) {
  print(content(clean_corp[[i]]))
  i <- i + 1
}
```

```
[1] "sometimes start dancing table can"
[1] "rt leslieks starbucks morning free tomorrow amp going wfriends get gunsense r
[1] "starbucks best confessyourunpopularopinion"
[1] "rt themindblowing fat burning foods grapefruit watermelon berries hot peppers
[1] "witnessed girl pet bird pooping shoulder amp girl sharing ice cream dog shop
[1] " biological watch thinks 543 pm want arabic cake"
[1] " wanna lay couch blankets hot watch old movies"
[1] "rt dreyess1 rehab addicts"
[1] "finally home food beer fridge also tomorrow will rough ordered pizza rat time
[1] "just drank entire venti starbucks amp still think im gonna fall asleep wtf"
[1] "rt uberfacts 1000 chemicals single cup 26 tested half caused cancer "
```

- Or like this:

```
i = 25
while (i %in% 25:35) {
  print(content(clean_corp[[i]]))
  i <- i + 1
}
```

```
[1] "sometimes start dancing table can"
[1] "rt leslieks starbucks morning free tomorrow amp going wfriends get gunsense r
[1] "starbucks best confessyourunpopularopinion"
[1] "rt themindblowing fat burning foods grapefruit watermelon berries hot peppers
[1] "witnessed girl pet bird pooping shoulder amp girl sharing ice cream dog shop
[1] " biological watch thinks 543 pm want arabic cake"
[1] " wanna lay couch blankets hot watch old movies"
[1] "rt dreyess1 rehab addicts"
[1] "finally home food beer fridge also tomorrow will rough ordered pizza rat time
[1] "just drank entire venti starbucks amp still think im gonna fall asleep wtf"
[1] "rt uberfacts 1000 chemicals single cup 26 tested half caused cancer "
```

Analyze the term-document matrix (TDM)

	Tweet 1	Tweet 2	Tweet 3	…	Tweet N
Term 1	0	0	0	0	0
Term 2	1	1	0	0	0
Term 3	1	0	0	0	0
…	0	0	3	1	1
Term M	0	0	0	1	0

Term Document Matrix (TDM)

- The TDM (term-document matrix) has terms in the first column and documents (e.g. tweets) across the top as column or feature names.
- TDM is used for language analysis: you likely have many more terms than authors or documents, and it is easier to analyze tables with many records than tables with many columns.
- Print the TDM:

```
coffee_tdm
```

```
<<TermDocumentMatrix (terms: 3076, documents: 1000)>>
Non-/sparse entries: 7391/3068609
Sparsity           : 100%
Maximal term length: 27
Weighting          : term frequency (tf)
```

- To analyse the information, we change the TDM into a simple matrix and print the dimensions:

```
coffee_m <- as.matrix(coffee_tdm)
dim(coffee_m)  # rows x columns
```

```
[1] 3076 1000
```

- Have a look at the upper left and lower right corner of the matrix:

```
coffee_m[1:5,1:10]
coffee_m[3071:3076,995:1000]
```

```
      Docs
Terms  1 2 3 4 5 6 7 8 9 10
  0630 0 0 0 0 0 0 0 0 0  0
  1000 0 0 0 0 0 0 0 0 0  0
  1026 0 0 0 0 0 0 0 0 0  0
  1030 0 0 0 0 0 0 0 0 0  0
  110  0 0 0 0 0 0 0 0 0  0
 Docs
Terms             995 996 997 998 999 1000
  zaykennedy69      0   0   0   0   0    0
  zeledmalegisele   0   0   0   0   0    0
  ziggy             0   0   0   0   0    0
  zokuhq            0   0   0   0   0    0
  zombie            0   0   0   0   0    0
  zzzquil           0   0   0   0   0    0
```

- Print the subset of `coffee_m` containing the terms (in rows) "hot" and "starbucks" and documents (in columns) 25 through 35:

```
coffee_m[c("hot","starbucks"), 25:35]
```

```
           Docs
Terms       25 26 27 28 29 30 31 32 33 34 35
  hot        0  0  0  1  0  0  1  0  0  0  0
  starbucks  0  1  1  0  0  0  0  0  0  1  0
```