Text Mining Pipeline

Code

A simple R project adapted for self-study purpose.

Overview

- Section 1: Create R Project
- Section 2: Loading Data into R
    - 2-1: Text File
- Section 3: Text Pre-processing
- Section 4: Text Normalization
    - 4-1: Stemming
    - 4-2: Lemmatization
- Section 5: Text Representation
    - 5-1: Document Term Matrix (DTM)
- Section 6: Text Mining
    - 6-1: Word Frequency
    - 6-2: Term Correlation
- Section 7: Simple Graphics
    - 7-1: Histogram
    - 7-2: Word Cloud

Section 1: Create R Project

Create a New Project in RStudio

The steps to create a new project in R is specified as follows:

1. Click the **File > New Project** from the top menu.
2. Click **New Directory**.
3. Click **New Project**.
4. Enter the directory name to store your project, e.g. “simpleTextMining”.
5. *(optional)* Place the project under your selected subdirectory.
6. Click **Create Project** button.

Create Subdirectories for Project

The subfolders created for this project, together with the files in these subfolders are as listed below:

1. **doc**: text documents associated with the project.
2. **data**: raw data and metadata.
3. **output**: files generated during cleanup and analysis.
4. **src**: source for the project’s scripts and programs
5. **bin**: programs brought in from elsewhere or compiled locally

Finally, all files are named to reflect their content or function.

Create R Script File

Create an R Script file to save your codes.

Write the code you want to run directly in an .R script file, and then running the selected lines (keyboard shortcut: **Ctrl + Enter**) in the interactive R console.

Save your R Script file in the **src** folder.

Check Working Directory

When you are working with R using a Project environment, the workspace will be automatically loaded when you open the project. To change the working directory, call `setwd()`.

Hide

```
# Get current working directory
getwd()
```

```
[1] "C:/Users/Darren Lee/Jupyter_Notebook/MS_DSA/CDS522_2021/CDS522_R/src"
```

Load R Packages

Call `library({package_name})` to load the package.

Hide

```
# Load package
library(tm)
```

```
package 愀牠tm愀牛 was built under R version 3.6.3Loading required package: NLP
package 愀牠NLP愀牛 was built under R version 3.6.3
```

Section 2: Loading Data into R

Load and View Data in R

Hide

```
# Create Corpus from .txt
docs <- Corpus(DirSource("../data"))

# View Corpus Information
print(docs)
```

```
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 10
```

Hide

```
print(summary(docs))
```

```
                      Length Class             Mode
childrenstories_01.txt 2      PlainTextDocument list
childrenstories_02.txt 2      PlainTextDocument list
childrenstories_03.txt 2      PlainTextDocument list
childrenstories_04.txt 2      PlainTextDocument list
childrenstories_05.txt 2      PlainTextDocument list
childrenstories_06.txt 2      PlainTextDocument list
childrenstories_07.txt 2      PlainTextDocument list
childrenstories_08.txt 2      PlainTextDocument list
childrenstories_09.txt 2      PlainTextDocument list
childrenstories_10.txt 2      PlainTextDocument list
```

Hide

```
cat("Welcome to this simple text mining project")
```

```
Welcome to this simple text mining project
```

Hide

```
require(tm)
```

Inspect Document Contents

You can examine the contents of a particular document (e.g. the first document)

Hide

```
# Inspect a particular document (e.g. the 1st doc)
writeLines(as.character(docs[[1]]))
```

```
Was it just another game of hide and seek? No. It was not. First she fell into a deep, dark h
ole in the ground and then they found a treasure. Did it end there? No! It did not. Read more
about this thrilling adventure of Sally and friends in this free illustrated kidsâ€™ book. Th
e fun never ends when Sallyâ€™s around!
```

Section 3: Text Pre-processing

Data Cleaning using `tm` Package

Data cleaning is an important step in text analysis.

It could take up to few cycles to achieve a mature cleaning pipeline as new issues are often found during the process of cleaning.

`tm` package offers a number of text transformation functions. Call `getTransformation()` to list these transformation functions.

Hide

```
# checkout tm package transformation functions
getTransformations()
```

```
[1] "removeNumbers"     "removePunctuation" "removeWords"       "stemDocument"      "stripWhi
tespace"
```

Create A New Function: `toSpace`

Hide

```
# Create toSpace content transformer
toSpace <- content_transformer(
  function(x, pattern) {
    return (gsub(pattern, " ", x))
  }
)
```

The description of the parameters and variables in the new function: `toSpace` :

- `gsub()` : replace all occurrences of a pattern.
- `pattern` : a pattern to search for (assumed to be a regex)
  - an additional argument `fixed = TRUE` can be specified to look for a pattern without using regex.
- `replacement` : a character string to replace the occurrence (or occurrences for `gsub`) of pattern.
  - Here, `replacement = " "`
- `x` : a character vector to search for pattern. Each element will be searched separately.

Before Transformation

Hide

```
# select a doc
docIndex <- 3

# before transformation
writeLines(as.character(docs[[docIndex]]))
```

```
Love shines through this great illustrated kidsâ€™ book . Read how a little girl makes chores
fun and easy to do. A  fantastic addition to your little oneâ€™s free bed time story collecti
on.
```

Replace Special Punctuation with Space

Call the helper function `toSpace()` as previously defined.

Hide

```
# eliminate hyphen using toSpace content transformation
docs <- tm_map(docs, toSpace, "-")
writeLines(as.character(docs[[docIndex]]))
```

```
Love shines through this great illustrated kidsâ€™ book . Read how a little girl makes chores
fun and easy to do. A  fantastic addition to your little oneâ€™s free bed time story collecti
on.
```

Remove Punctuation

Hide

```
# apply removePunctuation
docs <- tm_map(docs, removePunctuation)
writeLines(as.character(docs[[docIndex]]))
```

```
Love shines through this great illustrated kidsâ€™ book  Read how a little girl makes chores
fun and easy to do A  fantastic addition to your little oneâ€™s free bed time story collectio
n
```

Convert to Lower Case

Hide

```
# convert corpus to lower case
docs <- tm_map(docs, content_transformer(tolower))
writeLines(as.character(docs[[docIndex]]))
```

```
love shines through this great illustrated kidsâ€™ book  read how a little girl makes chores
fun and easy to do a  fantastic addition to your little oneâ€™s free bed time story collectio
n
```

Remove Numbers

Hide

```
# remove digits in corpus
docs <- tm_map(docs, removeNumbers)
writeLines(as.character(docs[[docIndex]]))
```

```
love shines through this great illustrated kidsâ€™ book  read how a little girl makes chores
fun and easy to do a  fantastic addition to your little oneâ€™s free bed time story collectio
n
```

Remove Stopwords

Example of stopwords recognized by the `tm` package includes:

- articles: a, an, the
- conjuctions: and, or, but
- common verbs: is
- qualifiers: yet, however

Hide

```
# remove stopwords using the standard list in tm
docs <- tm_map(docs, removeWords, stopwords("english"))
writeLines(as.character(docs[[docIndex]]))
```

```
love shines   great illustrated kidsâ€™ book  read   little girl makes chores fun  easy     f
antastic addition   little oneâ€™s free bed time story collection
```

Strip Extra Whitespace

Hide

```
# remove whitespace optional to remove extra whitespace
docs <- tm_map(docs, stripWhitespace)
writeLines(as.character(docs[[docIndex]]))
```

```
love shines great illustrated kidsâ€™ book read little girl makes chores fun easy fantastic a
ddition little oneâ€™s free bed time story collection
```

Section 4: Text Normalization

4-1: Stemming

Stemming is the process of reducing related words to their common root. For example:

> offer, offered, offering $\rightarrow$ offer

Simple stemming algorithms (in `tm` package) simply chop off the ends of the words.

To perform stemming, pass function `stemDocument()` (from `SnowballC` package) to `tm_map()` of `tm` package.

Hide

```
library(SnowballC)
```

```
package 'SnowballC' was built under R version 3.6.3
```

Hide

```
# duplicate object for testing
docs.stem <- docs

# stem the corpus
docs.stem <- tm_map(docs.stem, stemDocument)
writeLines(as.character(docs.stem[[2]]))
```

```
read warm tale camaraderi affect set wild beauti savannah free illustr kid book ginger giraff
use long neck save anim blaze forest fire follow jungl path meet yet anoth adventur
```

4-2: Lemmatization

Lemmatization is the process of grouping together the inflected forms of a word. It is much more sophisticated as compared to stemming.

The resulting *lemma* can be analyzed as a single item.

To perform lemmatization, pass function `lemmatize_string()` (from `textstem` package) to `tm_map()` of `tm` package.

Hide

```
# Lemmatization
# load textstem package
library(textstem)
```

```
package 勘笸textstem勘牷 was built under R version 3.6.3Loading required package: koRpus.lan
g.en
package 勘笸koRpus.lang.en勘牷 was built under R version 3.6.3Loading required package: koRpu
s
package 勘笸koRpus勘牷 was built under R version 3.6.3Loading required package: sylly
package 勘笸sylly勘牷 was built under R version 3.6.3Registered S3 method overwritten by 'dat
a.table':
  method           from
  print.data.table
For information on available language packages for 'koRpus', run

  available.koRpus.lang()

and see ?install.koRpus.lang()


Attaching package: 勘笸koRpus勘牷

The following object is masked from 勘笸package:tm勘牷:

    readTagged
```

Hide

```
# duplicate object for testing
docs.lemma <- docs

# lemmatize the corpus (require textstem)
docs.lemma <- tm_map(docs.lemma, lemmatize_strings)
writeLines(as.character(docs.lemma[[2]]))
```

```
read warm tale camaraderie affection set wild beautiful savannah free illustrate kid book gin
ger giraffe use long neck save animal blaze forest fire follow jungle path meet yet another a
dventure
```

Section 5: Text Representation

5-1: Document Term Matrix (DTM)

Document Term Matrix (or **DTM** for short) is a matrix that lists all occurrences of words (column) in the corpus, by document (row).

- A word that appears in a particular document will have its respective matrix entry in the corresponding row and column assign to 1, else 0.

- A word that appears *n* times in a document will be recorded as `n` in the respective matrix entry.

Example

We have the two documents, *Doc1* and *Doc2*, with the following content:

- *Doc1*: goats are happy
- *Doc2*: goats are fat

The corresponding DTM will look like:

	goats	**are**	**happy**	**fat**
Doc1	1	1	1	0
Doc2	1	1	0	1

DTM can become very huge, depending on the corpus. The dimension of the DTM is the **# of documents** multiplied by the **# of words in the corpus**. **Sparsity** often happens since majority of words only appear in few documents.

Hide

```
# Document Term Matrix: DTM

# Create Document Term Matrix
dtm <- DocumentTermMatrix(docs.lemma)

# View summary of document term matrix
dtm
```

```
<<DocumentTermMatrix (documents: 10, terms: 163)>>
Non-/sparse entries: 219/1411
Sparsity           : 87%
Maximal term length: 12
Weighting          : term frequency (tf)
```

Hide

```
# Inspect document term matrix
inspect(dtm)
```

```
<<DocumentTermMatrix (documents: 10, terms: 163)>>
Non-/sparse entries: 219/1411
Sparsity           : 87%
Maximal term length: 12
Weighting          : term frequency (tf)
Sample             :
                     Terms
Docs                  adventure book find free fun illustrate little read story time
  childrenstories_01.txt         1    1    1    1   1          1      0    1     0    0
  childrenstories_02.txt         1    1    0    1   0          1      0    1     0    0
  childrenstories_03.txt         0    1    0    1   1          1      2    1     1    1
  childrenstories_04.txt         0    0    0    0   0          0      0    0     0    0
  childrenstories_05.txt         0    1    0    0   0          0      0    0     0    0
  childrenstories_06.txt         1    1    0    1   0          0      1    0     1    1
  childrenstories_07.txt         0    0    0    0   1          0      0    1     2    1
  childrenstories_08.txt         0    1    0    2   0          0      0    2     1    0
  childrenstories_09.txt         0    0    2    1   0          0      1    1     2    0
  childrenstories_10.txt         0    1    0    2   0          0      2    2     0    0
```

Hide

```
# Inspect document term matrix by specifying rows and columns
inspect(dtm[1:5, 11:20])
```

```
<<DocumentTermMatrix (documents: 5, terms: 10)>>
Non-/sparse entries: 17/33
Sparsity           : 66%
Maximal term length: 10
Weighting          : term frequency (tf)
Sample             :
                     Terms
Docs                  free friend fun game grind hide hole illustrate just kidsâ€™
  childrenstories_01.txt    1      1   1    1     1    1    1          1    1       1
  childrenstories_02.txt    1      0   0    0     0    0    0          1    0       0
  childrenstories_03.txt    1      0   1    0     0    0    0          1    0       1
  childrenstories_04.txt    0      0   0    0     0    0    0          0    0       0
  childrenstories_05.txt    0      1   0    0     0    0    0          0    0       0
```

Section 6: Text Mining

6-1: Words Frequency

When constructing the DTM, the corpus of text is converted into a **mathematical object** that can be analyzed and manipulated using quantitative techniques of matrix algebra.

To get the frequency of each word in the corpus, we can sum over all rows based on the columns.

Hide

```
# get frequency of each word
freq <- colSums(as.matrix(dtm))

# check dimension of frequency (number of words/columns)
length(freq)
```

```
[1] 163
```

Check Frequent vs. Infrequent Words

We can also sort the words (`freq`) in descending order based on term count.

Hide

```
# Create sort order
ord <- order(freq, decreasing = TRUE)

# Inspect most frequently occurring terms
freq[head(ord)]
```

```
     free      read      book     story    little adventure
        9         9         7         7         6         3
```

Hide

```
# Inspect least frequently occurring terms
freq[tail(ord)]
```

```
    much     open    stand    truly validate      way
       1        1        1        1        1        1
```

Terms Reduction

We can reduce the term in the DTM by specifying the following parameters:

- number of documents the word appears in the corpus: 2 to 8 documents
- length of words: 4 to 20 characters

Hide

```
# Create document term matrix with term reduction
# - Include only words that occur in 2-8 documents.
# - enforce lower & upper limit to the length of words (4-20 characters)
dtm.tr <- DocumentTermMatrix(
  docs.lemma, control = list(
    wordLengths = c(4,20), bounds = list(global = c(2,8))
  )
)

dtm.tr
```

```
<<DocumentTermMatrix (documents: 10, terms: 29)>>
Non-/sparse entries: 81/209
Sparsity           : 72%
Maximal term length: 12
Weighting          : term frequency (tf)
```

Hide

```
# Find frequent terms
findFreqTerms(dtm.tr, lowfreq = 5)
```

```
[1] "book"   "free"   "read"   "little" "story"
```

Find Frequent Terms

Call function `findFreqTerm()`, then specify the DTM and filter by `lowfreq = 5` (the output shows only words with 5 or more occurrences in the corpus).

Note that the results is sorted alphabetically, not by frequency.

Hide

```
# Find frequent terms
findFreqTerms(dtm.tr, lowfreq = 5)
```

```
[1] "book"   "free"   "read"   "little" "story"
```

6-2: Terms Correlation

Correlation is a quantitative measure of the co-occurrence of words in the corpus. The correlated terms can be identified by calling `findAssocs()` in `tm` package, then specify the term of interest and correlation limit.

Hide

```
# Find terms correlation
findAssocs(dtm.tr, "read", 0.5)
```

```
$read
        free        aloud         next childrenâ€™s
        0.80         0.79         0.79         0.51
```

Example calculation of correlation score

Pearson Correlation Coefficient is defined as follows:

$$r = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{\left(n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right)\left(n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right)}}$$

Hide

```
trydata <- c(
  "", "word1", "word1 word2", "word1 word2 word3",
  "word1 word2 word3 word4", "word1 word2 word3 word4 word5"
)

trydtm <- DocumentTermMatrix(VCorpus(VectorSource(trydata)))
trydtm
```

```
<<DocumentTermMatrix (documents: 6, terms: 5)>>
Non-/sparse entries: 15/15
Sparsity           : 50%
Maximal term length: 5
Weighting          : term frequency (tf)
```

Hide

```
as.matrix(trydtm)
```

```
    Terms
Docs word1 word2 word3 word4 word5
   1     0     0     0     0     0
   2     1     0     0     0     0
   3     1     1     0     0     0
   4     1     1     1     0     0
   5     1     1     1     1     0
   6     1     1     1     1     1
```

Hide

```
findAssocs(trydtm, "word1", 0.0)
```

```
$word1
word2 word3 word4 word5
 0.63  0.45  0.32  0.20
```

Section 7: Simple Graphics

7-1: Histogram

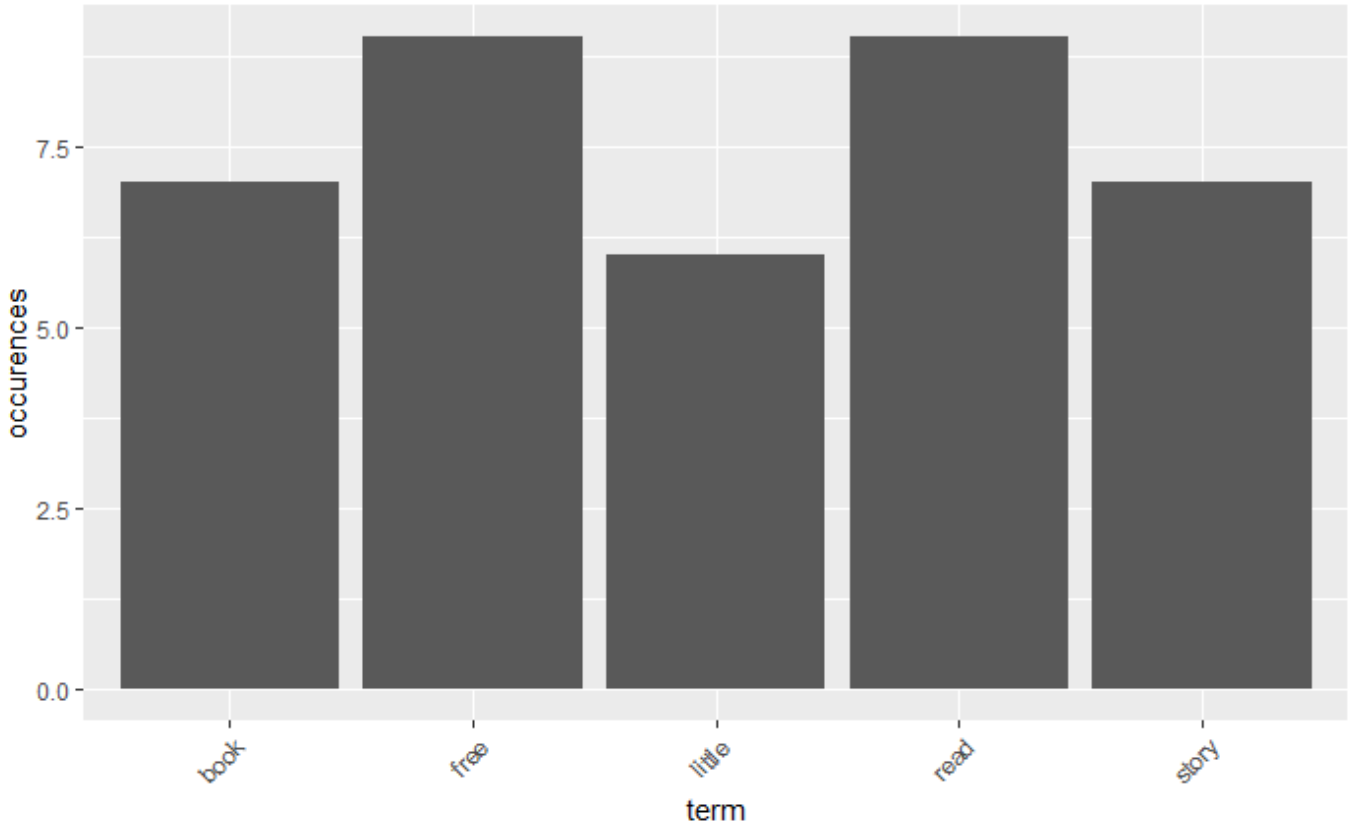
Hide

```
freq.tr <- colSums(as.matrix(dtm.tr))

# Plot simple frequency histogram
# Create a data frame (consists of name of the column)
wordfreq <- data.frame(
  term = names(freq.tr), occurences = freq.tr
)

# load ggplot2 package
library(ggplot2)

#invoke ggplot(plot only terms more than 3 times, label x and y-axis using aes)
phisto<-ggplot(subset(wordfreq, freq.tr>3), aes(term, occurences))
#set the height of the bar using stat="bin" or "identity" ("identify" means the height is bas
ed on the data value mapped to y-axis)
phisto<-phisto + geom_bar(stat="identity")
#specify that the x-axis text is at 45 degree angle and horizontally justified
phisto<-phisto + theme(axis.text.x=element_text(angle=45, hjust=1))
#display histogram
phisto
```



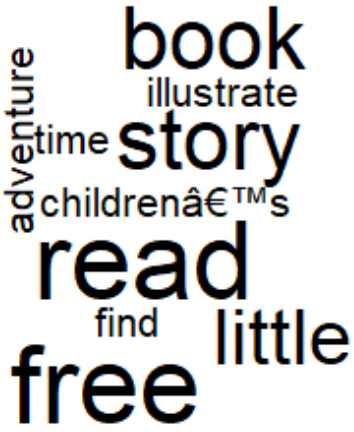
7-2: Word Cloud

Hide

```
# load wordcloud package
library(wordcloud)

# setting the seed before each plot to ensure consistent look for clouds
set.seed(32)

# limit words by specifying min frequency
wordcloud(names(freq.tr), freq.tr, min.freq = 3, scale = c(3.5, 0.25))
```



Hide

```
# limit words by specifying min frequency (with color)
wordcloud(names(freq.tr), freq.tr, min.freq = 3, scale = c(3.5, 0.5), colors = brewer.pal(6,
"Dark2"))
```


story
time
find
adventure
illustrate
childrenâ€™s
read
little
free
book