A.

First, in order to create and understand the text, the following packages were installed.

install.packages("tm")

library(tm)

install.packages("quanteda")

library(quanteda)

Then, the corpus was created and inspected.

doc = VCorpus(DirSource(path, ignore.case=TRUE, mode="text"))

inspect(doc)

<<VCorpus>>

Metadata: corpus specific: 0, document level (indexed): 0

Content: documents: 2

[[1]]

<<PlainTextDocument>>

Metadata: 7

Content: chars: 21

[[2]]

<<PlainTextDocument>>

Metadata: 7

Content: chars: 577411

str(doc)

List of 2

$ dummy.txt :List of 2

..$ content: chr "dummy data goes here."

..$ meta :List of 7

.. ..$ author : chr(0)

.. ..$ datetimestamp: POSIXlt[1:1], format: ...

.. ..$ description : chr(0)

.. ..$ heading : chr(0)

.. ..$ id : chr "dummy.txt"

.. ..$ language : chr "en"

.. ..$ origin : chr(0)

.. ..- attr(\*, "class")= chr "TextDocumentMeta"

..- attr(\*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"

$ TwentyThousandLeagues.txt:List of 2

..$ content: chr [1:12131] "" "TWENTY THOUSAND LEAGUES UNDER THE SEA" "" "by" ...

..$ meta :List of 7

.. ..$ author : chr(0)

.. ..$ datetimestamp: POSIXlt[1:1], format: ...

.. ..$ description : chr(0)

.. ..$ heading : chr(0)

.. ..$ id : chr "TwentyThousandLeagues.txt"

.. ..$ language : chr "en"

.. ..$ origin : chr(0)

.. ..- attr(\*, "class")= chr "TextDocumentMeta"

..- attr(\*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"

- attr(\*, "class")= chr [1:2] "VCorpus" "Corpus"

These functions show basic meta data about the documents. I added an extra dummy.txt file for the later matrices to look better and to work as designed. Next, the target document from *Twenty Thousand Leagues* was inspected.

indexed\_doc = doc[[2]]

str(indexed\_doc)

List of 2

$ content: chr [1:12131] "" "TWENTY THOUSAND LEAGUES UNDER THE SEA" "" "by" ...

$ meta :List of 7

..$ author : chr(0)

..$ datetimestamp: POSIXlt[1:1], format: ...

..$ description : chr(0)

..$ heading : chr(0)

..$ id : chr "TwentyThousandLeagues.txt"

..$ language : chr "en"

..$ origin : chr(0)

..- attr(\*, "class")= chr "TextDocumentMeta"

- attr(\*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"

This reveals only simple meta data, such as the fact that the text file produces a character array of length 12131 based on the number of lines in the story. Next, I made a document term matrix.

docdtm = DocumentTermMatrix(doc)

inspect(docdtm)

<<DocumentTermMatrix (documents: 2, terms: 14909)>>

Non-/sparse entries: 14911/14907

Sparsity : 50%

Maximal term length: 26

Weighting : term frequency (tf)

Sample :

Terms

Docs and for had not that the

dummy.txt 0 0 0 0 0 0

TwentyThousandLeagues.txt 2366 559 620 881 926 8355

Terms

Docs this was which with

dummy.txt 0 0 0 0

TwentyThousandLeagues.txt 709 1307 730 853

str(docdtm)

List of 6

$ i : int [1:14911] 1 1 1 1 2 2 2 2 2 2 ...

$ j : int [1:14911] 3691 4403 6049 6511 1 2 3 4 5 6 ...

$ v : num [1:14911] 1 1 1 1 1 1 1 1 1 1 ...

$ nrow : int 2

$ ncol : int 14909

$ dimnames:List of 2

..$ Docs : chr [1:2] "dummy.txt" "TwentyThousandLeagues.txt"

..$ Terms: chr [1:14909] "'artocarpus'" "'bread-fruit'" "'seafrog,'" "'these" ...

- attr(\*, "class")= chr [1:2] "DocumentTermMatrix" "simple\_triplet\_matrix"

- attr(\*, "weighting")= chr [1:2] "term frequency" "tf"

This data shows common terms in the story such as and, for, had, not, etc. The inverse of a term document matrix is now shown.

doctdm = TermDocumentMatrix(doc)

inspect(doctdm)

<<TermDocumentMatrix (terms: 14909, documents: 2)>>

Non-/sparse entries: 14911/14907

Sparsity : 50%

Maximal term length: 26

Weighting : term frequency (tf)

Sample :

Docs

Terms dummy.txt TwentyThousandLeagues.txt

and 0 2366

for 0 559

had 0 620

not 0 881

that 0 926

the 0 8355

this 0 709

was 0 1307

which 0 730

with 0 853

str(doctdm)

List of 6

$ i : int [1:14911] 3691 4403 6049 6511 1 2 3 4 5 6 ...

$ j : int [1:14911] 1 1 1 1 2 2 2 2 2 2 ...

$ v : num [1:14911] 1 1 1 1 1 1 1 1 1 1 ...

$ nrow : int 14909

$ ncol : int 2

$ dimnames:List of 2

..$ Terms: chr [1:14909] "'artocarpus'" "'bread-fruit'" "'seafrog,'" "'these" ...

..$ Docs : chr [1:2] "dummy.txt" "TwentyThousandLeagues.txt"

- attr(\*, "class")= chr [1:2] "TermDocumentMatrix" "simple\_triplet\_matrix"

- attr(\*, "weighting")= chr [1:2] "term frequency" "tf"

Next, term frequency is visualized below.

freq = termFreq(indexed\_doc)

freq

'artocarpus' 'bread-fruit' 'seafrog,' 'these

1 1 1 1

"'pon "about

1 1

Along with many more rows, this information shows the term frequency before the data is cleaned up.

B.

Now, the ten longest terms are found using tokenization. The document was first tokenized, converted to a character vector, and then the vector was sorted by length. The results follow.

tokens = tokens(indexed\_doc$content)

tokenlist = as.character(tokens)

tokenlist = tokenlist[order(nchar(tokenlist), decreasing=TRUE)]

tokenlist[1:10]

[1] "Mohammed-ben-Abdallah" "Compagnie-Nationale"

[3] "emperor-holocanthus" "petromyzons-pricka"

[5] "carefully-studied" "fellow-countryman"

[7] "Clermont-Tonnerre" "pectinibranchidae"

[9] "seven-thousandths" "frightful-looking"

This shows that the longest ten terms all tend to be proper nouns. A few are also hyphenated adjectives or numbers. Next, the same follows for sentences. For this part, a new tokenizer library was needed to stop sentences from being broken up between lines.

sentences = as.String(indexed\_doc$content)

sentences = tokenize\_sentences(sentences)

sentences = sentences[order(nchar(sentences), decreasing=TRUE)]

[1] "TWENTY THOUSAND LEAGUES UNDER THE SEA by JULES VERNE PART ONE CHAPTER I A SHIFTING REEF The year 1866 was signalised by a remarkable incident, a mysterious and puzzling phenomenon, which doubtless no one has yet forgotten."

[2] "Not to mention rumours which agitated the maritime population and excited the public mind, even in the interior of continents, seafaring men were particularly excited."

[3] "Merchants, common sailors, captains of vessels, skippers, both of Europe and America, naval officers of all countries, and the Governments of several States on the two continents, were deeply interested in the matter."

[4] "For some time past vessels had been met by \"an enormous thing,\" a long object, spindle-shaped, occasionally phosphorescent, and infinitely larger and more rapid in its movements than a whale."

[5] "The facts relating to this apparition (entered in various log-books) agreed in most respects as to the shape of the object or creature in question, the untiring rapidity of its movements, its surprising power of locomotion, and the peculiar life with which it seemed endowed."

[6] "If it was a whale, it surpassed in size all those hitherto classified in science."

[7] "Taking into consideration the mean of observations made at divers times--rejecting the timid estimate of those who assigned to this object a length of two hundred feet, equally with the exaggerated opinions which set it down as a mile in width and three in length--we might fairly conclude that this mysterious being surpassed greatly all dimensions admitted by the learned ones of the day, if it existed at all."

[8] "And that it DID exist was an undeniable fact; and, with that tendency which disposes the human mind in favour of the marvellous, we can understand the excitement produced in the entire world by this supernatural apparition."

[9] "As to classing it in the list of fables, the idea was out of the question."

[10] "On the 20th of July, 1866, the steamer Governor Higginson, of the Calcutta and Burnach Steam Navigation Company, had met this moving mass five miles off the east coast of Australia."

C.

In order to create the word cloud, first the “wordcloud” package was needed.

install.packages("wordcloud")

library(wordcloud)

Next, punctuation and numbers had to be removed from the text as well as any stop words. This was done with the following functions.

removeNumPunc = function(x) gsub("[^[:alpha:][:space:]]\*", "", x)

doc = tm\_map(doc, content\_transformer(removeNumPunc))

myStopWords = c(stopwords("english"))

doc = tm\_map(doc, removeWords, myStopWords)

Now, doc was a cleaned Vcorpus object. This object was then transformed into a dataframe that stored the frequency of each term.

doctf = termFreq(doc[[2]])

docdf = as.data.frame(doctf)

Visually, the data frame looks like this:

> head(docdf)

doctf

abandon 4

abandoned 8

abandoning 3

abatebut 1

abated 1

abduction 1

Now, the functions used to turn this into a word cloud follow, along with the visualization.

pal = brewer.pal(9, "BuGn")

pal = pal[-(1:4)]

wordcloud(row.names(docdf), docdf[[1]], colors=pal, max.words=50, scale=c(3,.1))

A screenshot of a cell phone

Description automatically generated

This word cloud shows that the is the most common word, an obvious oversight of the stop words list. However, the rest of the words make sense in the context of the story. Nautical themed words such as captain, sea, water, ocean stand out. As well as names of characters or boats in the story, such as nautilus, nemo, or ned. I have not read the book, so I would need to research more to understand why conseil or one are such prominent words in the story.

What I learned.

In this section I learned how to compare many documents against each other. In both the document term matrix and its inverse, I could see the variation in word frequency for each document. For each individual document, I discovered how to view the term frequency of each document. This data then had to be manipulated in R to perform several functions on it, such as ranking words by frequency or visualizing them in a word cloud. This also required cleaning up a dataset, which required a stop words library as well as a punctuation remover using regular expressions. This showed that text analytics seeks out meaningful words that are unique to a document, not just repeated words or punctuation.