Big Data and Analytics

Project 3

Matt, Eric, Braden, Nabi, Rahmana

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.

d.

For part D of the project, I found it best to use the openNLP library rather than wordnet. Wordnet works in terms of analyzing synsets, but it doesn’t function as well as openNLP for analyzing parts of speech within the context of the sentence. This makes NLP challenging because one word could have different POS depending on the context.

R functions used:

require(openNLP) <- library

Maxent\_Sent\_Token\_Annotator()

Maxent\_Word\_Token\_Annotator()

annotate(text, list(sent\_token\_annotator, word\_token\_annotator))

require(ngram) <- library

rcorpus(nwords=nchar(x), alphabet= x, minwordlen = 6)

ngram (x , n=3)

require(zipfR) <- library

paste(c("he wants to back", collapse= ""))

as.String(text)

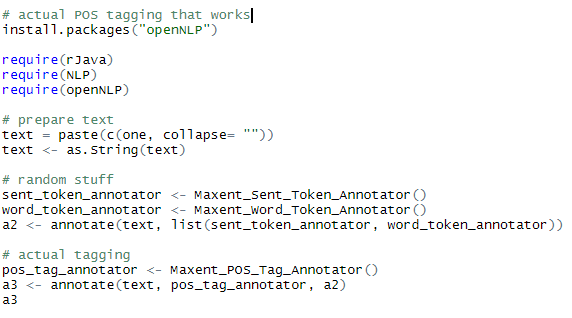
ItaRi.spc

summary(ItaRi.spc)

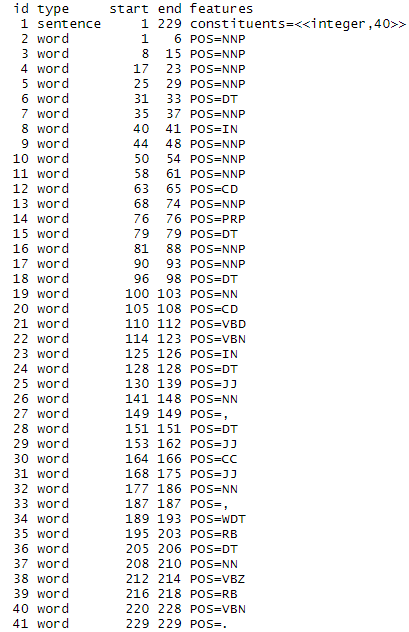
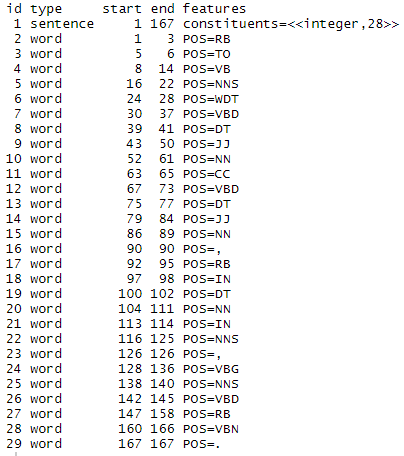
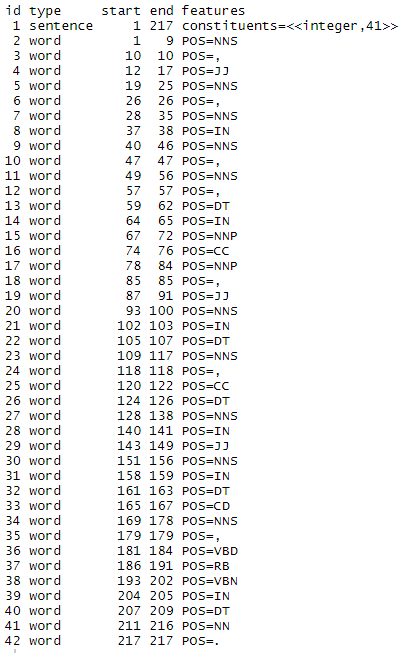
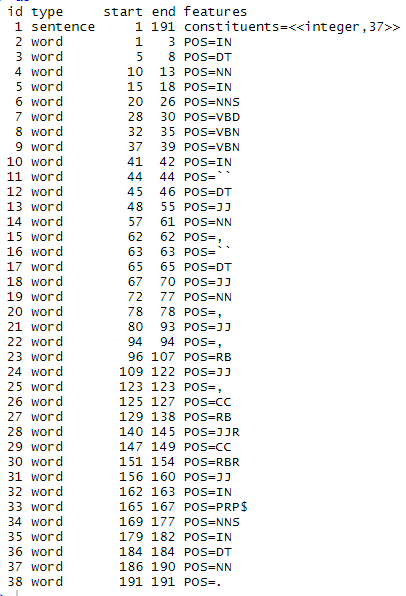
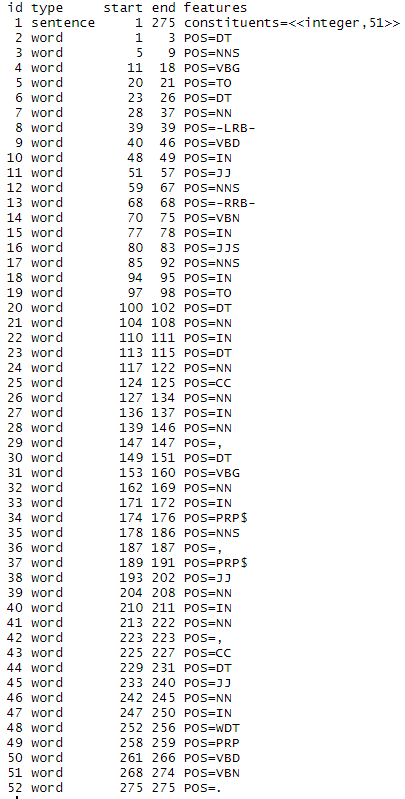
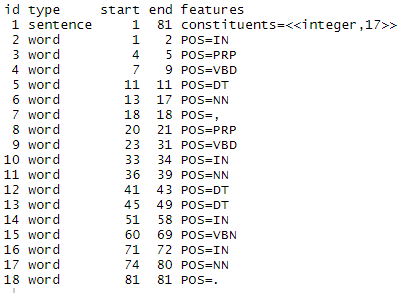
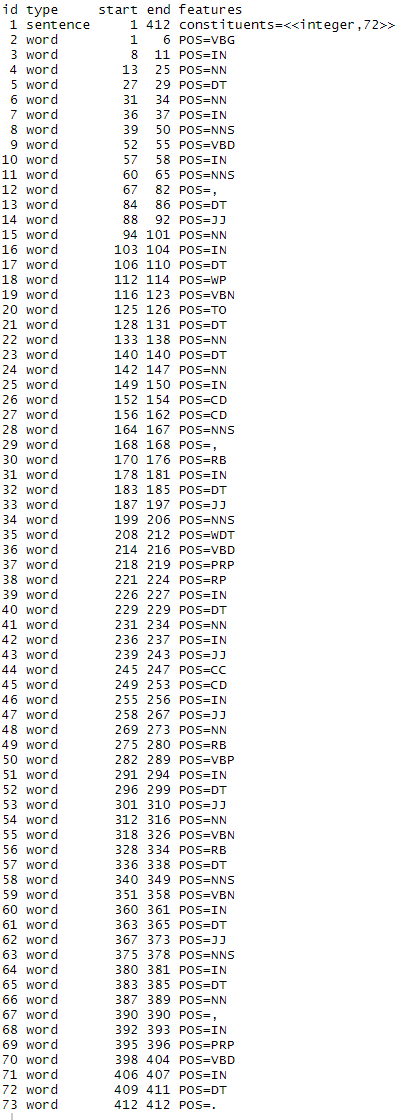
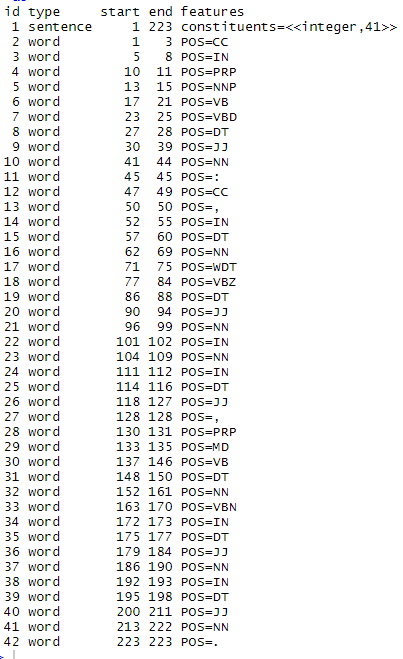
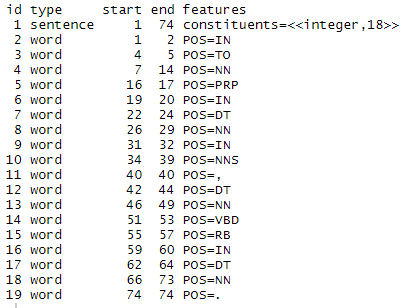
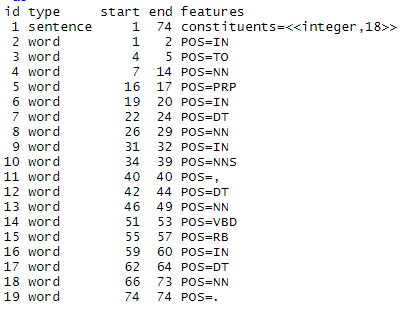
with(ItaRi.spc, plot(m, Vm, main="Frequency Spectrum"))

plot(BrownInform.spc)

Code:

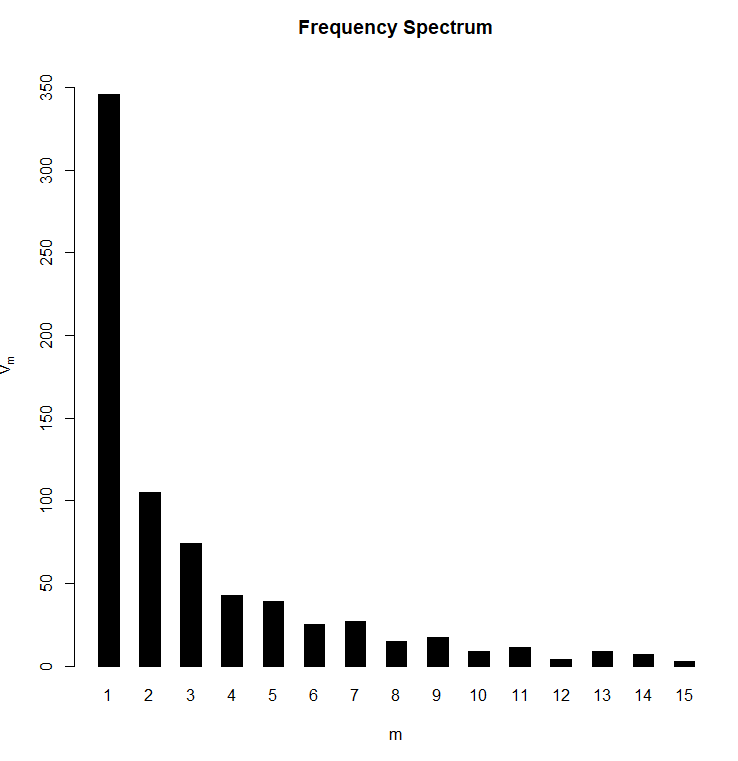


For the following examples, the ten longest sentences found above were run through the POS taggers using the code directly above. The sentences are ordered as they are above, with the POS in the last column of these print blocks

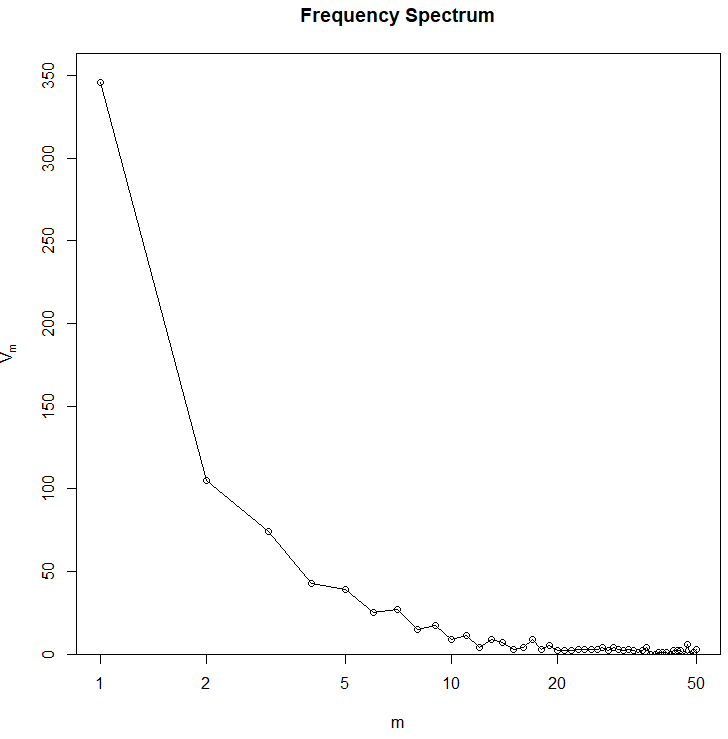
1. 
2. 
3. 
4. 
5. 
6. 
7. 
8. 
9. 
10. 

Here we find that the parts of speech could have been different if the words around it were structured differently. We find a healthy amount of verbs to nouns, and the analysis accurately reflected what we understand when naturally reading the sentences.

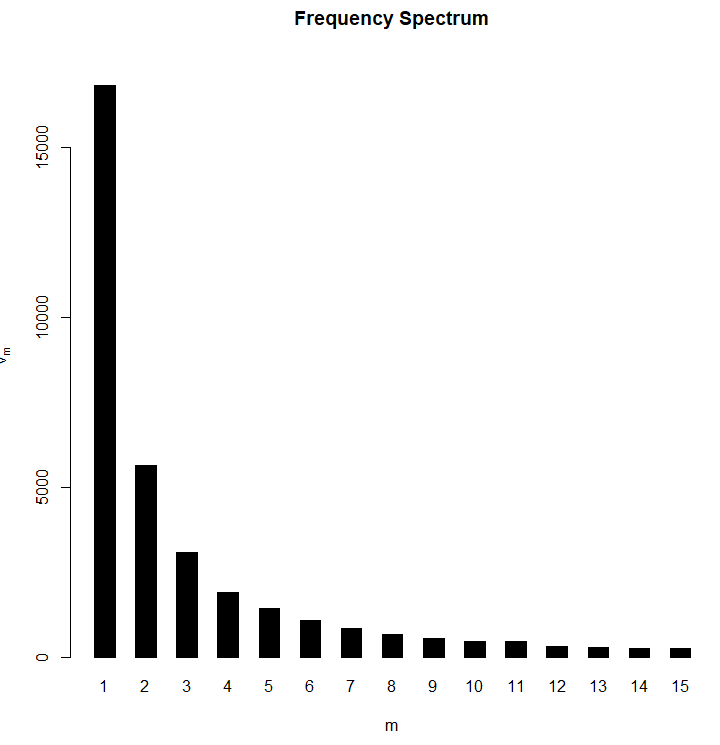
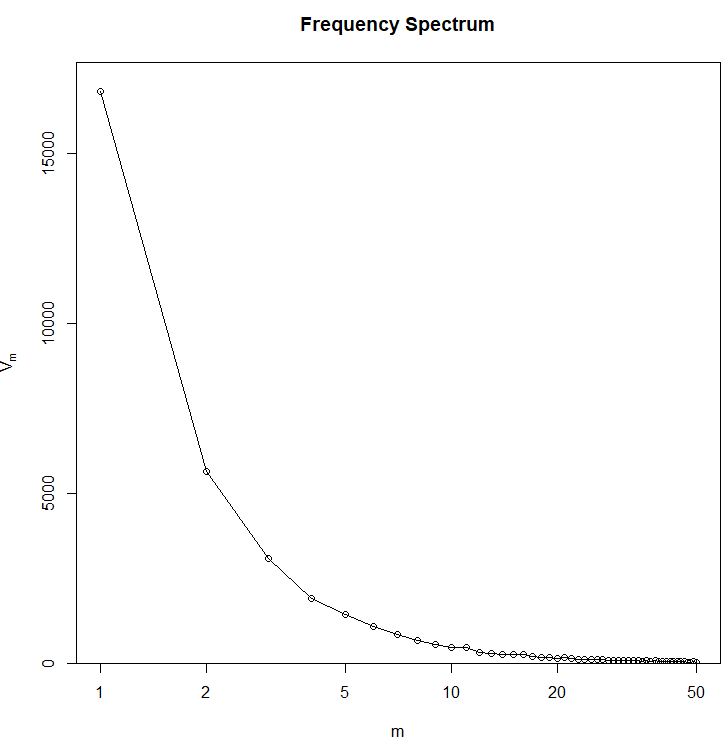
e. According to Zipf’s law, the frequency of any word used in any corpus is the frequency of the most used word divided by it’s rank. This means that the second most used word will be used ½ as many times as the most used word, the 3rd 1/3 as many times, and so on. Using plot with parameters from the zipfR library, such as ItaRi and the spc function, we were able to generate a zipf distribution that accurately displays this property.



This is made using the ItaRi data set. If we take the log on the x axis, we achieve this result:

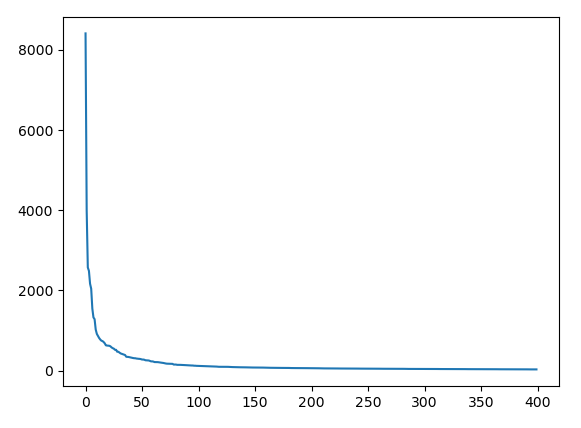


This can also be demonstrated with the BrownInform frequency set:

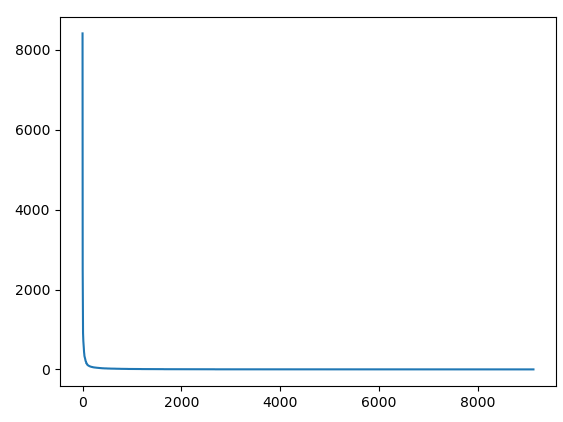
 

Notice that the zipf curves on the BrownInform set are much smoother and follow the law more closely. Not all data sets are this strictly adherent to the rule, but they can all be interpolated close enough to accurately demonstrate the same properties.

When running this through our own program, we wanted to see if TwentyThousandLeagues.txt would also follow this distribution. As shown below, this is clearly indicated:

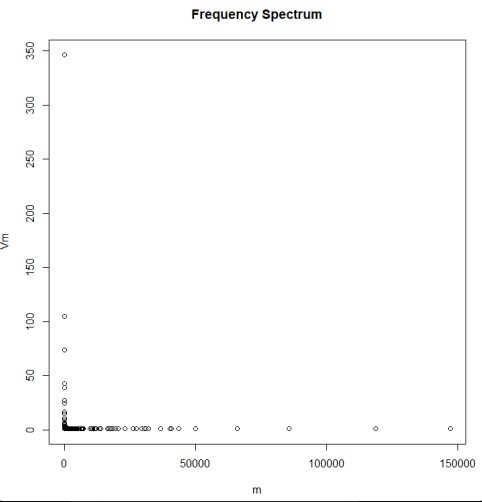


This sample was used on a slice of the data set to make the visualization nicer, but if the entire set it used, we get this distribution:

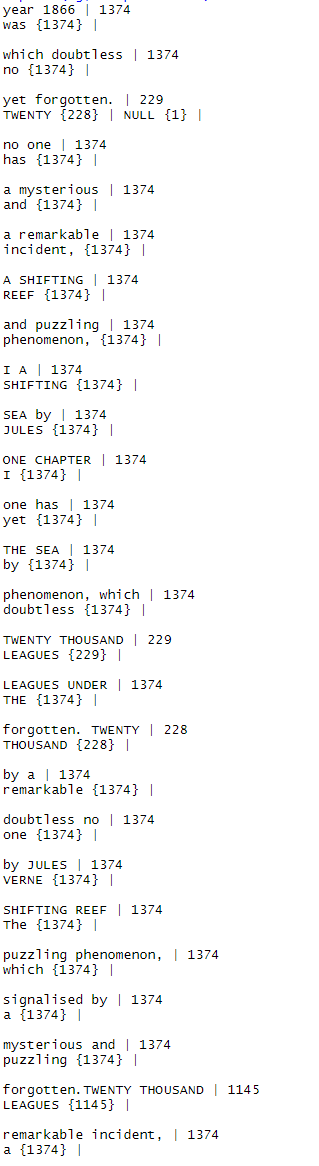


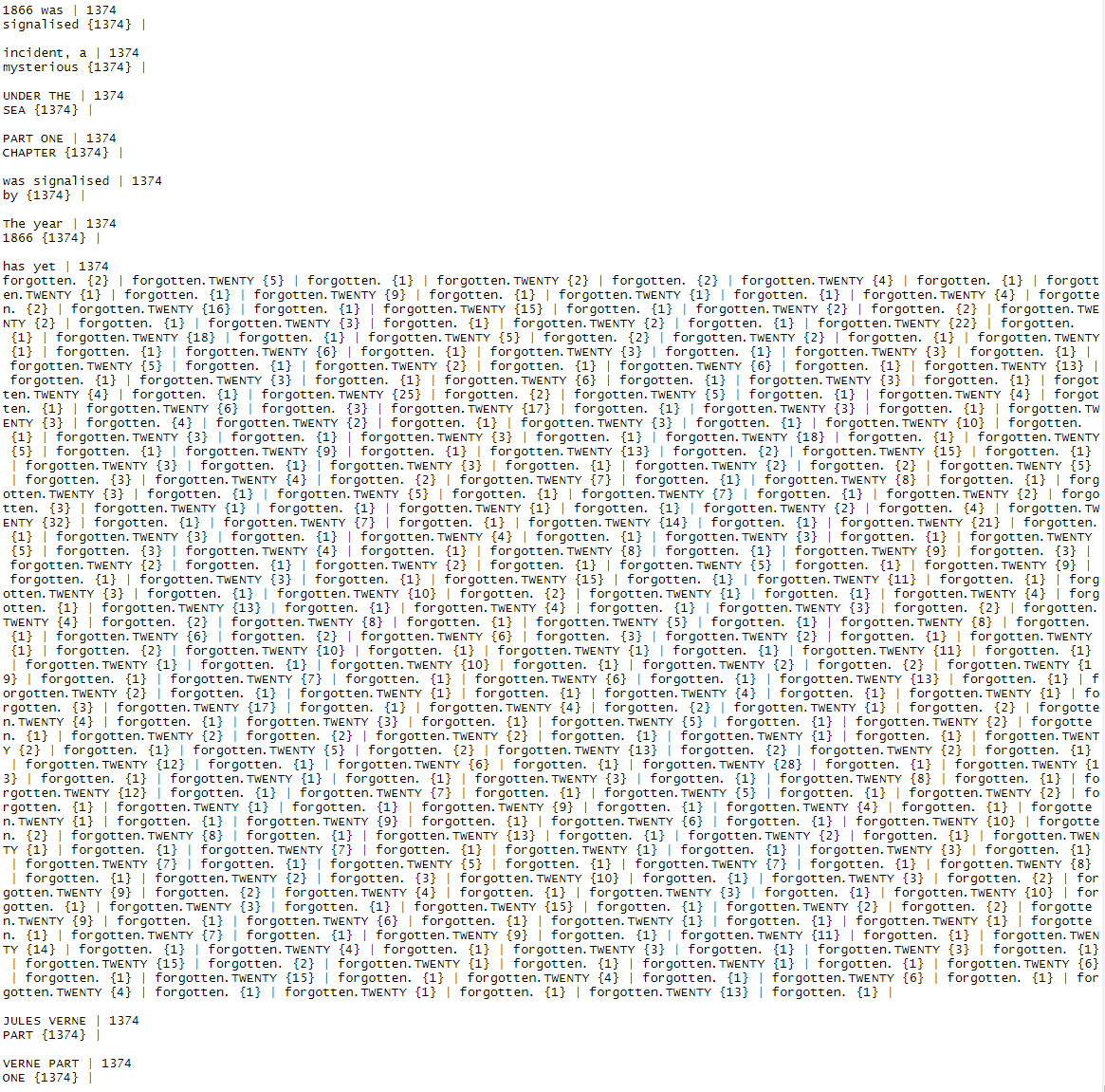
This is the full text, and we find that as we approach lesser used words, we find and increasing number of words that are only used a single time.

Using the zipfR library, we find that this is also true for other texts:

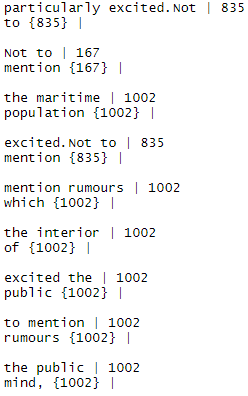


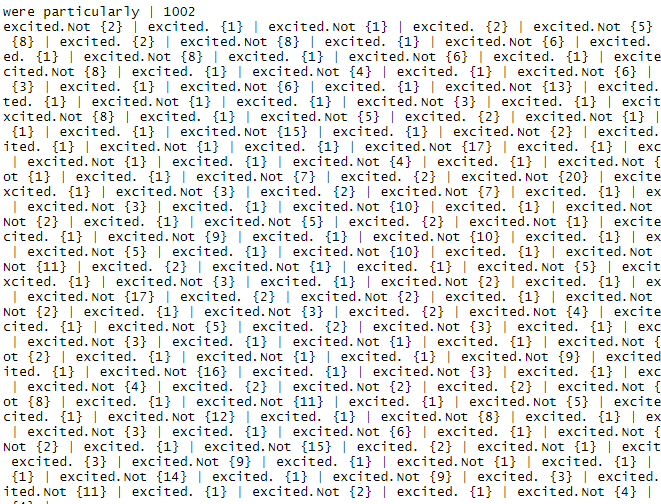
This result is fascinating, and universal through all text and all languages.

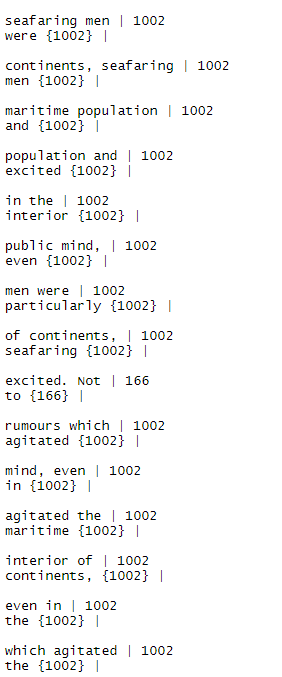
f. sentence 1 



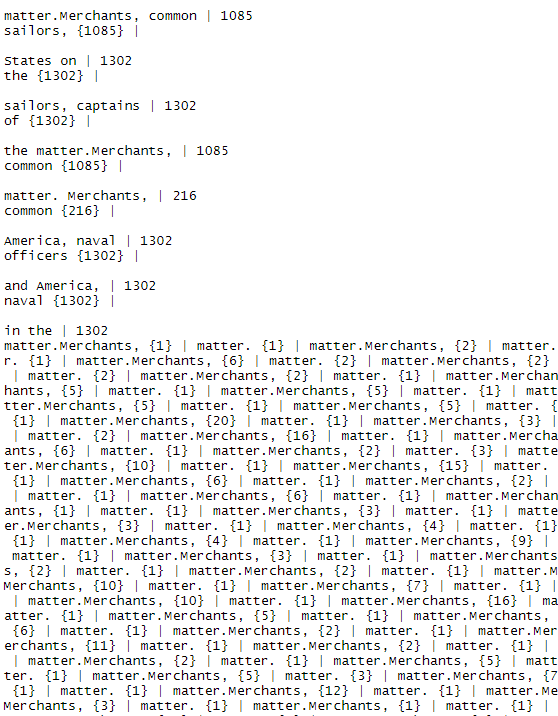
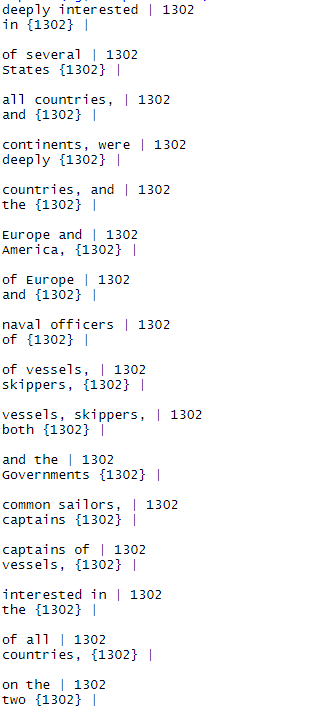
Sentence 2

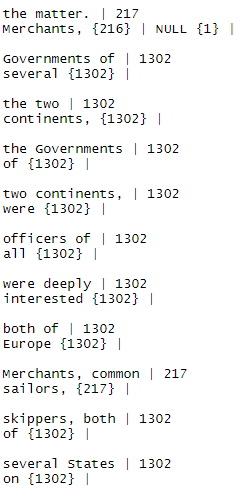




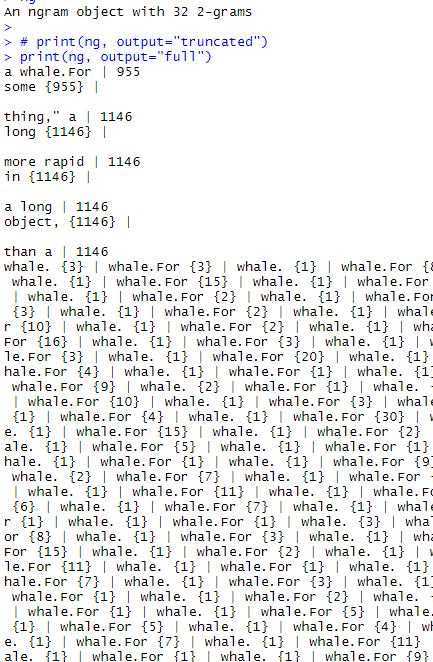


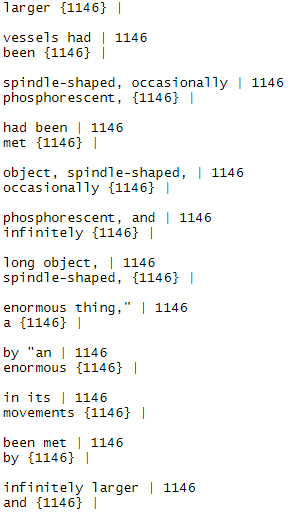
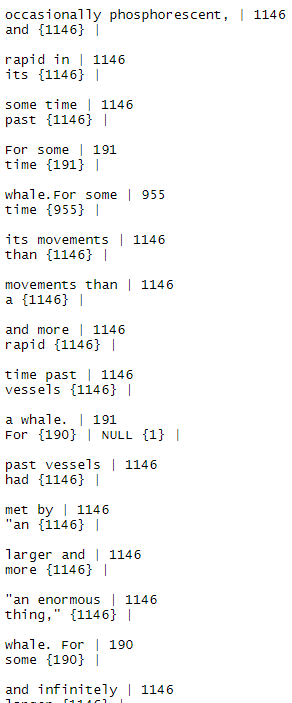
Sentence 3



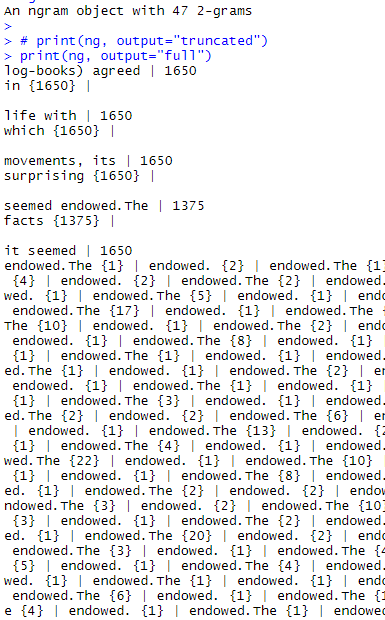
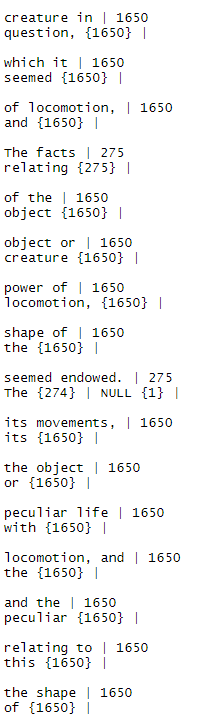


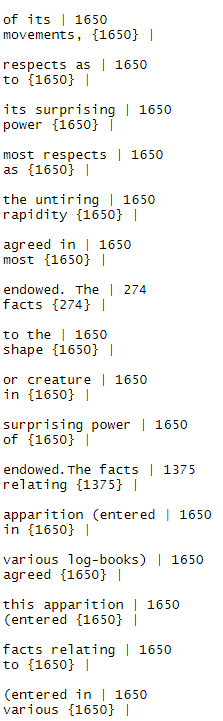
Sentence 4



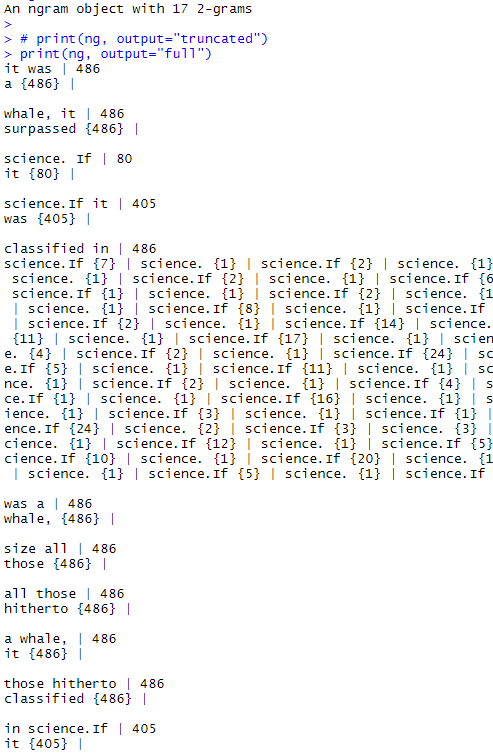
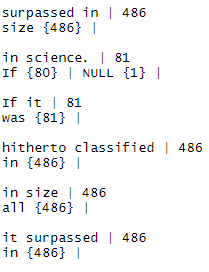


Sentence 5

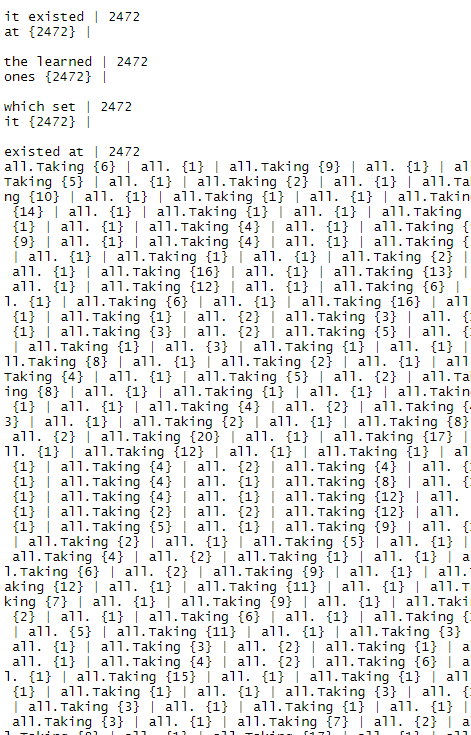
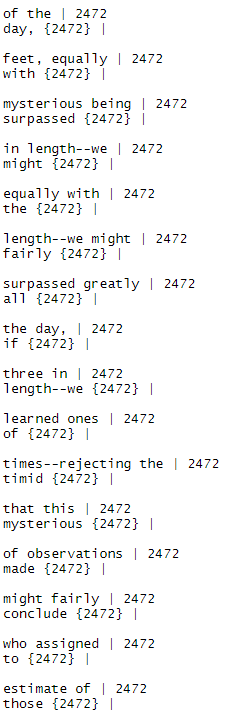
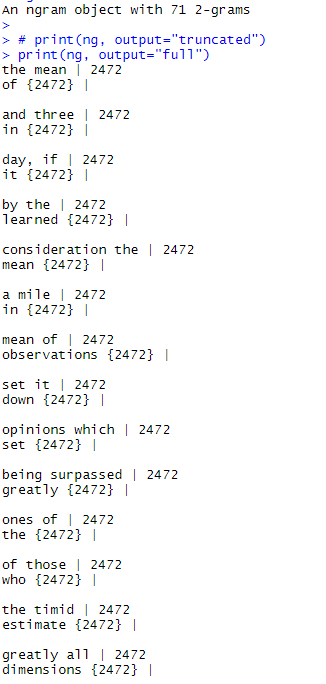
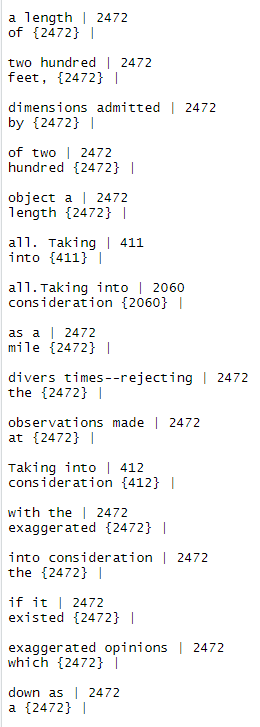
 

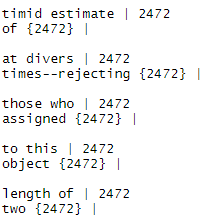
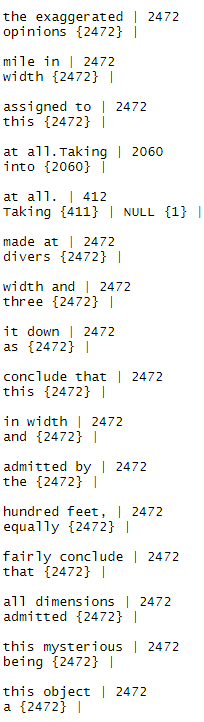


Sentence 6

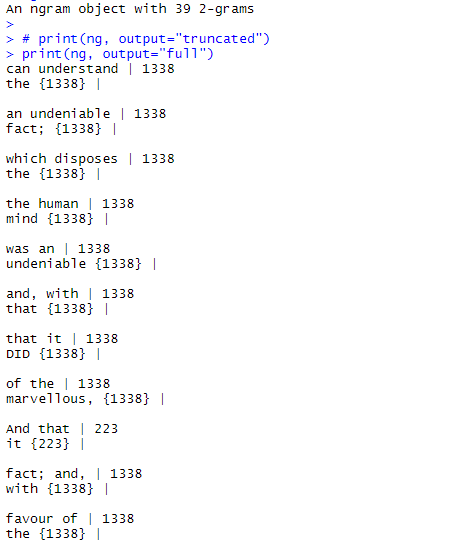
 

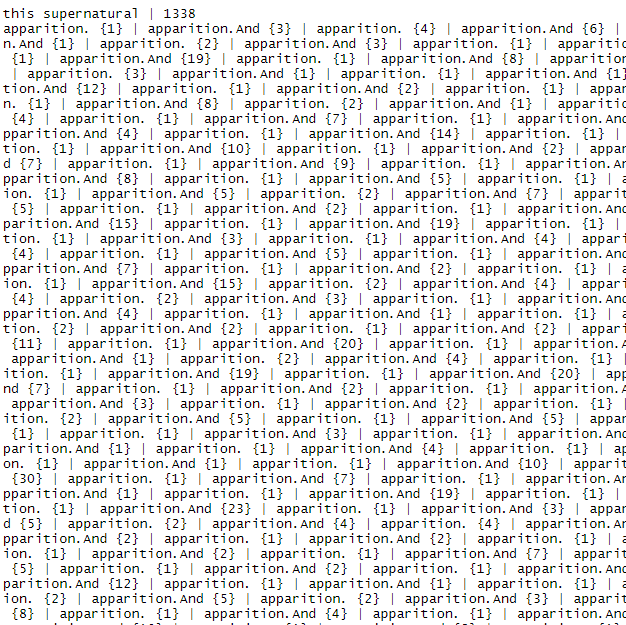
Sentence seven

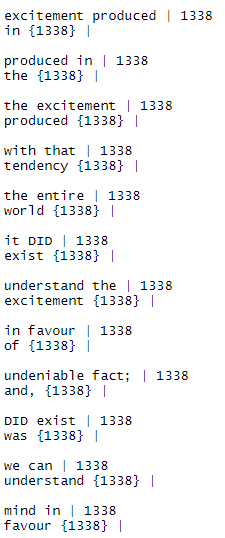
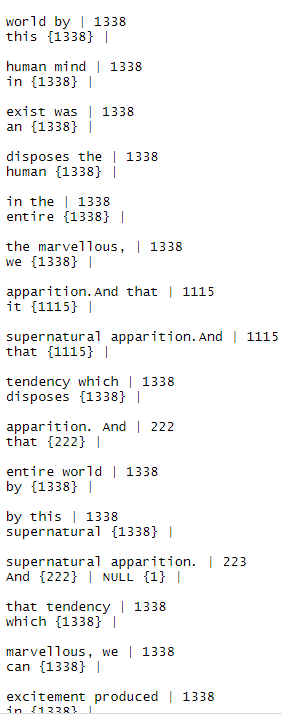
 



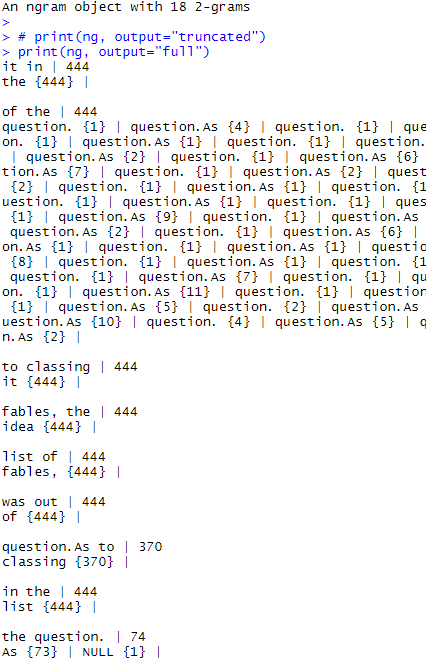
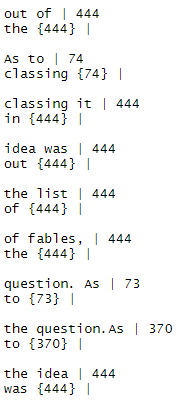
Sentence 8



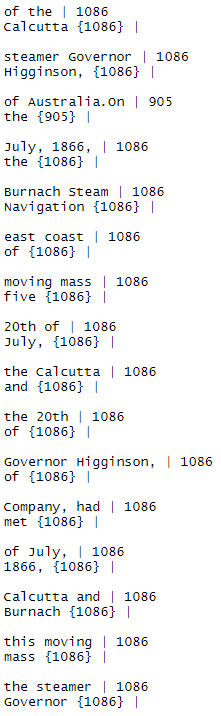


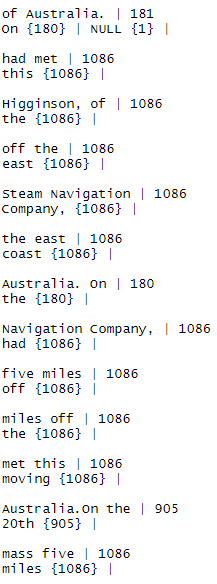


Sentence 9

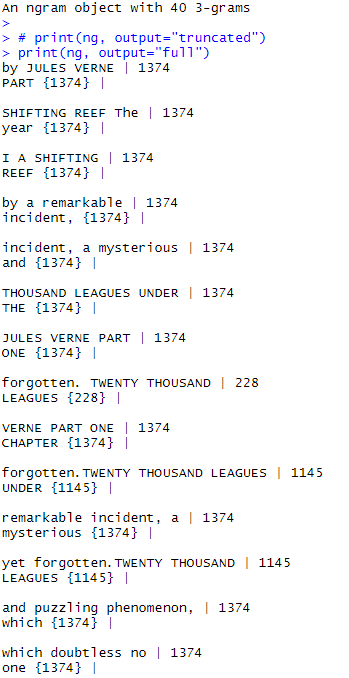
Sentence 10

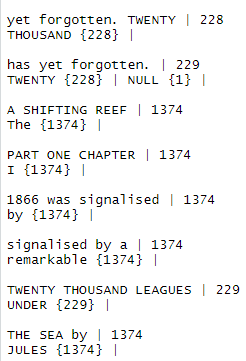
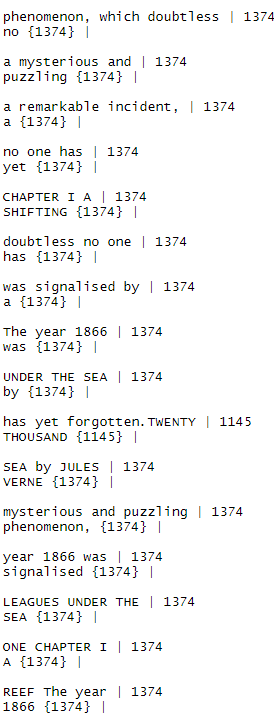
 



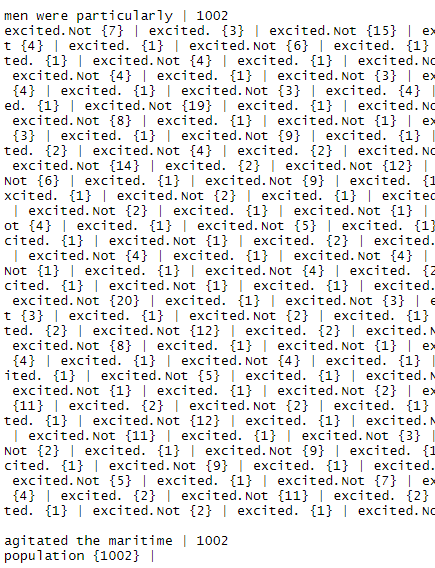
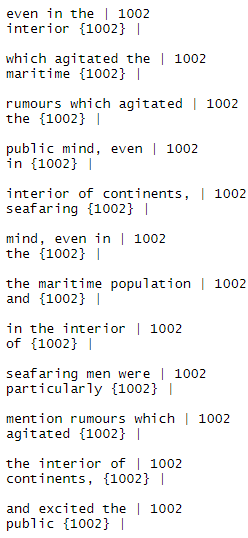
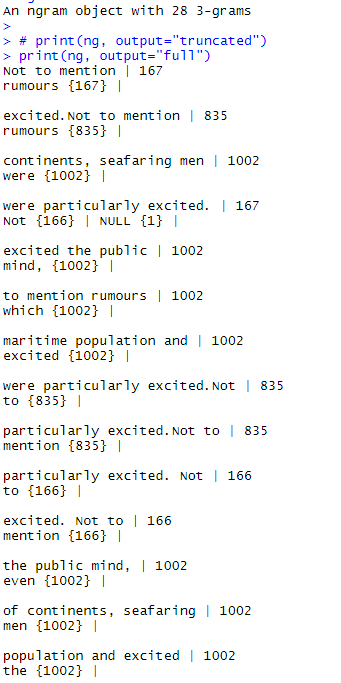
Trigrams:

Sentence 1

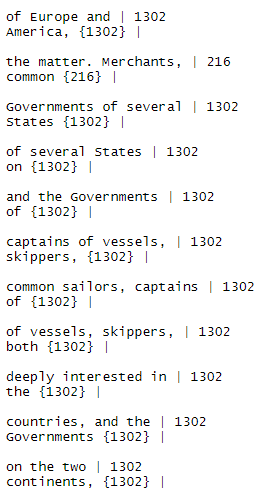
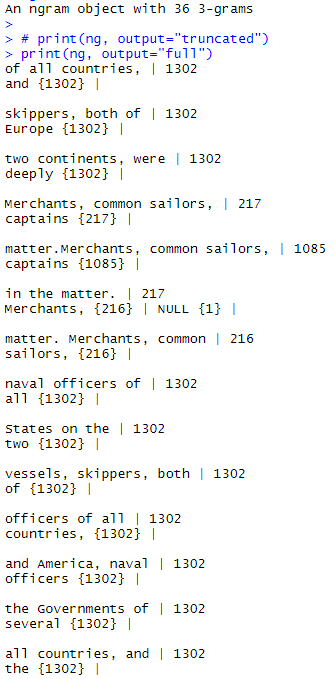


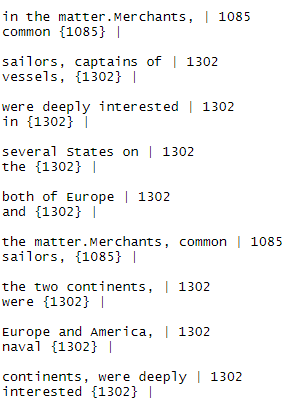

Sentence 2

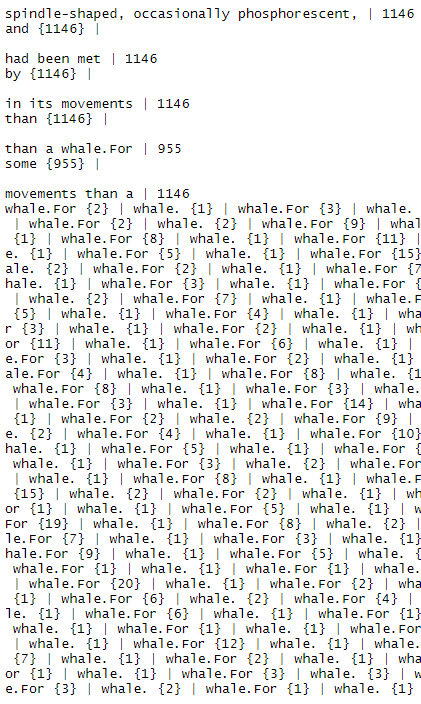
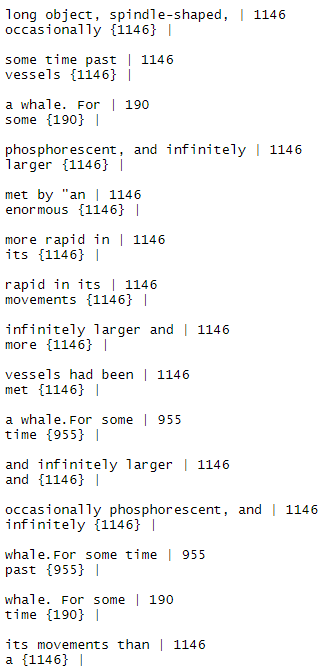
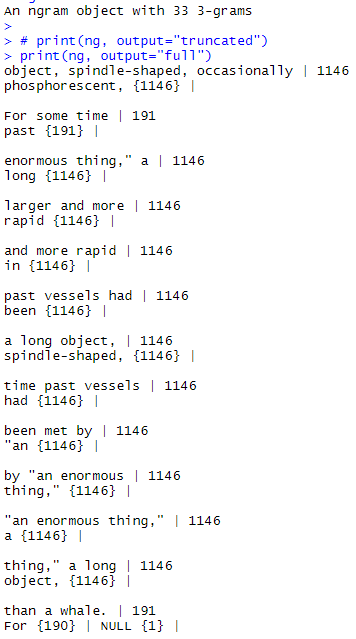


Sentences 3

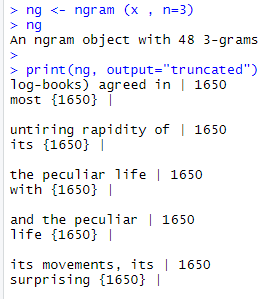




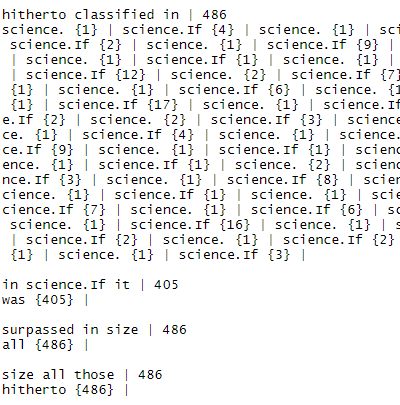
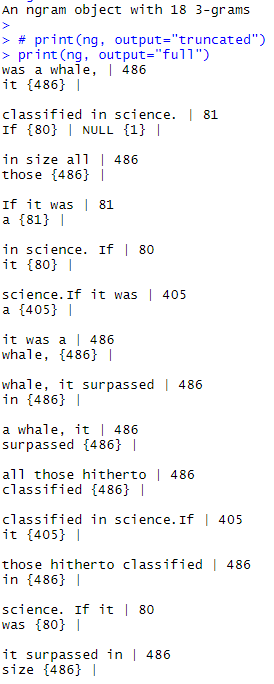


Sentence 4 

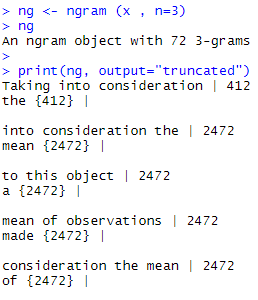
Sentence 5



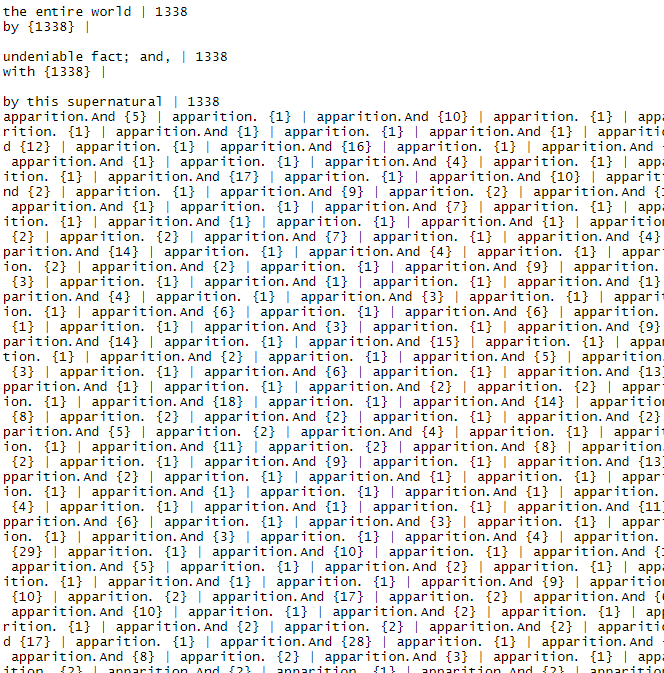
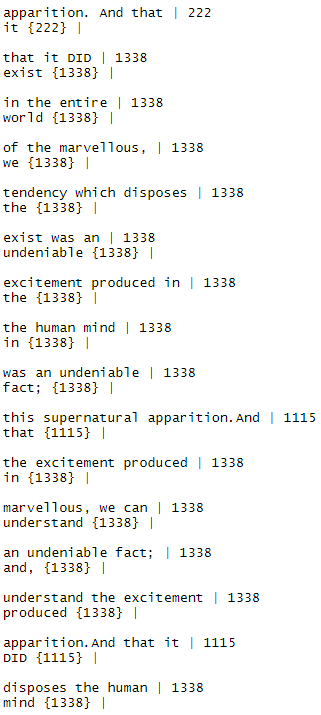
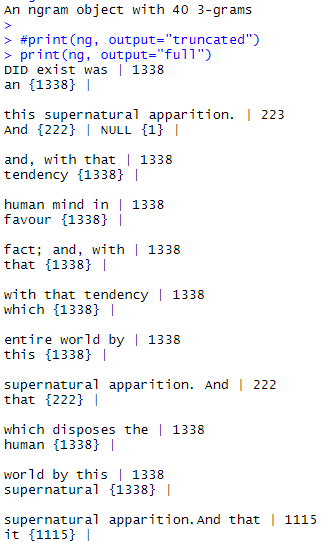
Sentence 6

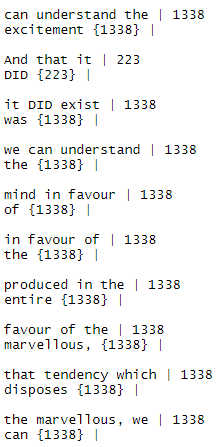


Sentence 7

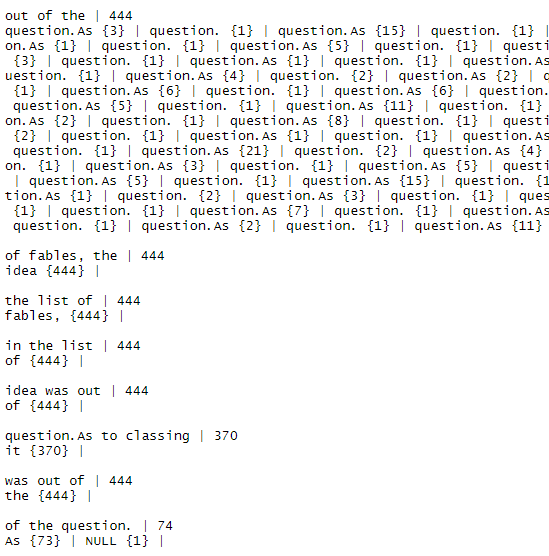
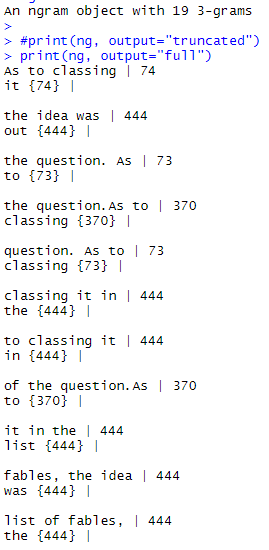


Sentence 8

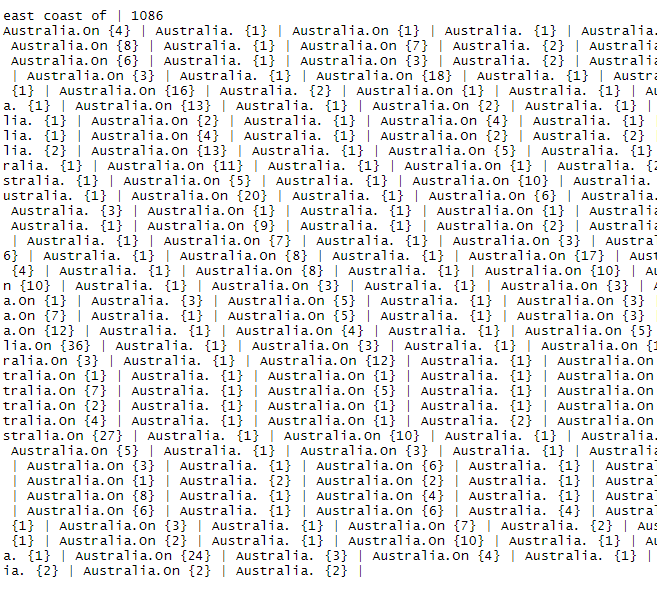
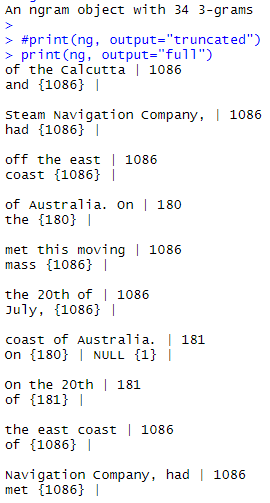


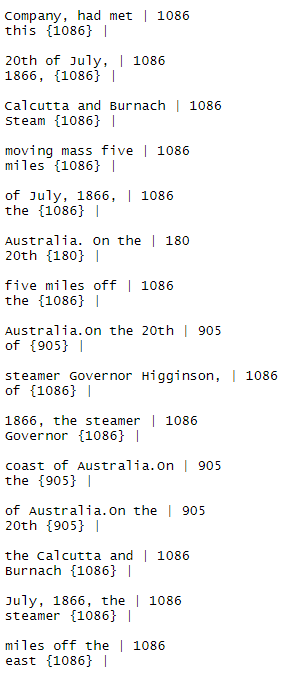
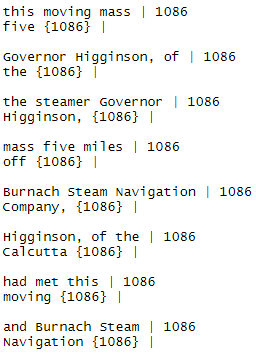


Sentence 9



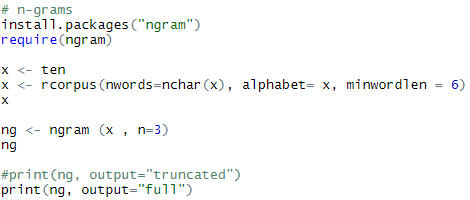
Sentence 10



Analysis:

The bigram/trigram analysis was done using the ngram library with the code below:



A minimum word length of 6 was established, but after analyzing the results, it doesn’t make much sense to exclude words less than this because there are key important pairs of words that appear in the document more than once that should be noted. To exclude any words from these sentences would create bigrams and trigrams that wouldn’t make contextual sense and therefore wouldn’t be useful in any way. Each record in the bigrams and trigrams above have 2 or 3 words that are found together, as well as the word that came after these pairings.

g.

Library(readtext):

readtext

library(stringi):

stri\_replace\_all, stri\_trim, stri\_trans\_tolower

library(tm):

removePunctuation

library(quanteda):

dfm, corpus, topfeatures, dfm\_weight, stopwords

library(corpustools):

create\_tcorpus, search\_features, kwic

library(tidytext):

tidy, rename, get\_sentiments, filter, semi\_join, arrange

After reading through the provided PDF (text\_analysis\_in\_R.pdf), we began our exploration of *Twenty Thousand Leagues Under the Sea* using the libraries requested: corpustools, stringi, quanteda, and tidytext. In the following section we will go through the methods we used from each library, starting with stringi to clean the text, quanteda to do some basic analysis, corpustools to learn about the content of the book based on words we discovered were important, and tidytext to do some sentiment analysis. All code is in braden\_script\_.R.

Beginning with stringi, we simply followed the document and used the methods they did to clean the text. Specifically, we used readtext to read in the document, separated out the text by accessing that specific part of the returned object, and then used the three methods to remove html, trim whitespace, and make everything lowercase. Here is a taste of what the processed text looks like:

*no doubt the presence of the nautilus, more formidable than\nitself, and on which its suckers or its jaws had no hold. yet, what\nmonsters these poulps are!*

Continuing on to the quanteda library, we first created a document term matrix using two methods, straight from our processed text and by first creating a quanteda corpus. Both resulted in the same matrix, but using the quanteda corpus would have allowed multiple documents to be stored as well. Here are the results for the dtm from the text:

*Document-feature matrix of: 1 document, 6,070 features (0.0% sparse).*

*features*

*docs twenti thousand leagu sea jule vern part one chapter shift*

*text1 41 46 32 386 1 1 105 359 46 3*

*[ reached max\_nfeat ... 6,060 more features ]*

Notice that we removed stopwords, removed numbers, and used stemming. Much of the processing done using stringi can be done in line with the dtm method as parameters, making it slightly easier as we would need to do less preprocessing. Here are the results using the corpus instead of the processed text:

*Document-feature matrix of: 1 document, 5,942 features (0.0% sparse).*

*features*

*docs twenti thousand leagu sea jule vern part one chapter shift*

*TwentyThousandLeagues.txt 44 46 32 387 1 1 105 365 46 3*

*[ reached max\_nfeat ... 5,932 more features ]*

The counts are slightly different, but essentially the same. Notice also that we have the name of our document associated with the matrix. If we were processing multiple documents at once, this would be a better way of creating the dtm as we could track the various documents more easily. Our exploration of the text then used the document term matrix to print the 20 most common words, shown here:

*captain nautilus sea nemo one said ned water us conseil land will like*

*638 519 386 378 359 333 321 311 297 282 271 264 218*

*two day sir time long see go*

*211 194 194 179 177 173 161*

Some of these words are pretty generic, but some like ‘captain’, ‘nautilus’, ‘nemo’, and ‘ned’ provide us with interesting starting points for further exploration. A lot of these words, combined with the title of the book, also let us know that the story takes place at sea. The word conseil is also interesting, I believe it is a French word for advice but could also be a title. We finally used the quanteda package to do tfidf analysis on the text. This would normally weight the importance of words based on how often they show up in various documents, but as we only have one the results were all 0 and not useful (we used it just to see what would happen, it went as we predicted). There are also other cool analysis features in quanteda, like some machine learning models both supervised and unsupervised, that require multiple documents so we were unable to test them. In the future if this project provided two different books, it would be useful to explore some of these libraries more thoroughly.

Now for sentiment analysis using tidytext. Tidytext is a different way of storing text data using dataframes instead of class objects like the corpus from quanteda. Tidytext provides a method for going from document term matrices to the tidy format, which requires one token (word) per row. Since we’re doing sentiment analysis, we can’t use our prior dtm’s because they have stemming and the sentiment datasets do not. So we first create another dtm without stemming, then use the tidy method to put our text in the tidy format. The one thing worth noting here, is that our tidy object does not contain the original text, it contains the text as if we had already used a count method on it (all words show up once, associated with their count). Tidytext provides dictionary-based sentiment analysis corpus’s, and we used the “bing” one for our analysis. We pulled all of the positive words from that corpus, and semi-joined it with our tidy data to find the positive words used the most in the text. We did the same thing with negative words. This provided us with further points of exploration when we finally get to investigating content with corpustools. Here are the results for both:

*Positive:*

*document word count*

*<chr> <chr> <dbl>*

*1 text1 like 206*

*2 text1 well 137*

*3 text1 great 119*

*4 text1 master 67*

*5 text1 good 66*

*6 text1 work 50*

*7 text1 right 47*

*8 text1 enough 44*

*9 text1 strong 36*

*10 text1 better 36*

*Negative:*

*document word count*

*<chr> <chr> <dbl>*

*1 text1 struck 53*

*2 text1 lost 39*

*3 text1 monster 37*

*4 text1 noise 36*

*5 text1 darkness 36*

*6 text1 fell 32*

*7 text1 shock 31*

*8 text1 terrible 30*

*9 text1 dark 26*

*10 text1 attack 26*

Now we get to investigating the text with corpustools. None of us in this group have read the book, so the majority of insight comes purely from what we found in this part. First to get the data ready, we created a tcorpus from our readtext object. Now with that tcorpus, we use search\_features and kwic to find occurrences of queries within so many words of each other. These queries are informed by our previous analysis of word frequency and sentiment. Our first thought was to determine who the captain is, since that is the most common word. Nautilus is the second most common, so we searched to see if those words occurred within 2 words of each other and then examined the times that it did. Those results actually revealed to us that Nemo, the fourth most common word, is the captain (we did the search with nemo and captain to make sure). Below are the results for both.

*Nautilus captain ~2*

*[1] "...returning to the <Nautilus>, <Captain> Nemo continued his..."*

*[2] "...interior of the <Nautilus>. <Captain> Nemo came down..."*

*Nemo captain ~2*

*[1] "...to you but <Captain> <Nemo>; and you..." "...? \" replied <Captain> <Nemo>. \" Did..."*

*[3] "...? \" answered <Captain> <Nemo>, slightly shrugging..." "...\" quietly answered <Captain> <Nemo>, \" they..."*

*[5] "...been mortal if <Captain> <Nemo> had discharged into..."*

The rest of our exploration continued in a similar way, that is a long example for the others the queries can be seen in our script. Here are some of the topics we thought about however, followed by a description of what we gathered about the story through these searches (once again, none of us have read the book). Topics we explored: who is the captain, is the nautilus a boat, does the nautilus sink, is the nautilus a sub, is the nautilus destroyed, does nemo have an enemy, who is ned, what is the monster (a common negative word), do we find the monster, does nemo die.

Provided at the very end of this document is a dump of our output from these searches if the reader is interested, but here is what we learned for each question we thought of.

*Who is the captain:* As stated before, we figured out the captain is Nemo, who also appears to be the main character.

*Is the nautilus a boat, is it a sub, does it sink, is it destroyed:* We did figure out that the nautilus is the name of the boat, and it is in fact a submarine. As for whether or not it is destroyed we are not sure. There are many references to it sinking, but because it is a sub that is to be expected. There are some lines about hull breaches, but context seems to imply they could be referring to how the sub is designed, not that they happen. We also know that there is an enemy, and potentially some battles, but the context we found did not seem to reveal the result of those fights.

*Does Nemo have an enemy:* As stated in the last section, yes it appears Nemo has some people hunting him, with some difficulty because of his submarine. Potentially those people are the governments of the world.

*Who is Ned:* This seemed a less important question, but Ned in one of the top 20 words so he is probably an important character. It seems that Ned is part of the crew, potentially a high ranking one, maybe one who leaves at some point in the book?

*What is the monster and do they find it:* The monster appears to be a narwhale, that the entire world is looking for. Based on our exploration, I would guess that the narwhale is never found.

*Does Nemo die:* there is never a direct reference to Nemo dying, rather a question about whether he perishes exactly like what we ourselves were asking. It seems that Nemo and some companions disappear and people aren’t really sure what happens to them.

In terms of the overall theme of the book, here is my prediction based on our exploration using these libraries. There is this creature called the narwhale and it is somewhat of a legend. The nations of the world want to capture/kill it, and there are sightings but it is very elusive. Captain Nemo and the Nautilus are either hired/created by the governments to find it, or he is an independent party who is also interested in finding it. At some point, his intentions are not aligned with the world governments; maybe they want to kill it and he wants to save it, something like that. Either way it turns into a race with him and the governments, maybe some battles involved. The book ends open-ended, Nemo and some of his crew disappear and are not heard from, probably after going off on some sort of last-ditch effort for the narwhale. The story probably wants the reader to decide for themselves: did Nemo find and save the narwhale and disappear to enjoy the rest of their lives, or did they all die, or maybe they did save the narwhale but died in the process.

That was the most interesting part of this project, just how much I was able to learn about the story through analyzing the words and reading a couple sentences. Below is a dump of all the queries ran, with output. It’s unstructured, doesn’t provide anything that I didn’t already describe in the analysis, just provided in case the reader is interested what passages I found and used to make these predictions about the story. To reiterate, the following results are an unstructured dump of the searches and resulting passages I found that allowed me to intuit what I could about the questions I was exploring and the story in general.

*> #see how often captain and nemo are within 2 words of each other*

*> hits <- tc$search\_features('"captain nemo"~2')*

*created index for "token" column*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] "...to you but <Captain> <Nemo>; and you..." "...? \" replied <Captain> <Nemo>. \" Did..."*

*[3] "...? \" answered <Captain> <Nemo>, slightly shrugging..." "...\" quietly answered <Captain> <Nemo>, \" they..."*

*[5] "...been mortal if <Captain> <Nemo> had discharged into..."*

*> #same but for nautilus and ship within 5 words*

*> hits <- tc$search\_features('"nautilus ship"~5')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] "...idea, the <Nautilus> would attack the <ship> at her waterline..."*

*> #same but for nautilus and ship within 5 words*

*> hits <- tc$search\_features('"nautilus boat"~5')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] "...Commander! your <Nautilus> is certainly a marvellous <boat>. \" \"..."*

*[2] "...not suit a <boat> like the <Nautilus>. \" \"..."*

*[3] "...past eight the <Nautilus> <boat> ran softly aground..."*

*[4] "...placed in the <boat> before leaving the <Nautilus>. Then,..."*

*[5] "...in that light <boat> of the <Nautilus> does not frighten..."*

*> #can we figure out if nemo dies*

*> hits <- tc$search\_features('"nemo dead"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] "...Where was Captain <Nemo>? Had he [...] Were his companions <dead> with him?..."*

*> #can we figure out if nemo dies*

*> hits <- tc$search\_features('"nemo dead"~10')*

*> kwic <- tc$kwic(hits, ntokens = 7)*

*> head(kwic$kwic, 5)*

*[1] "...through the water. Where was Captain <Nemo>? Had he succumbed? Were his companions <dead> with him? At the moment the..."*

*> hits <- tc$search\_features('"nemo alive"~10')*

*> kwic <- tc$kwic(hits, ntokens = 7)*

*Error in `$<-.data.frame`(`\*tmp\*`, "is\_kw", value = logical(0)) :*

*replacement has 0 rows, data has 15*

*> hits <- tc$search\_features('"nemo surviv\*"~10')*

*> kwic <- tc$kwic(hits, ntokens = 7)*

*Error in `$<-.data.frame`(`\*tmp\*`, "is\_kw", value = logical(0)) :*

*replacement has 0 rows, data has 15*

*> head(kwic$kwic, 5)*

*[1] "...through the water. Where was Captain <Nemo>? Had he succumbed? Were his companions <dead> with him? At the moment the..."*

*> hits <- tc$search\_features('"nemo \*liv\*"~10')*

*> kwic <- tc$kwic(hits, ntokens = 7)*

*> head(kwic$kwic, 5)*

*[1] "...pressure of the maelstrom? Does Captain <Nemo> still <live>? And does he still follow under..."> hits <- tc$search\_features('"nautilus s?nk\*"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] "...weight required to <sink> the <Nautilus>, I had..."*

*[2] "...horizon, the <Nautilus> <sank> to between twenty..."*

*[3] "..., and the <Nautilus> <sank> slowly beneath the..."*

*[4] "...reservoirs. The <Nautilus> began to <sink>, following a..."*

*[5] "...if necessary, <sink> this cursed <Nautilus>. \" \"..."*

*> #does the nemo die*

*> hits <- tc$search\_features('"nemo die"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] "...an invention should <die> with its inventor! \" Captain <Nemo> did not reply..."*

*> #does the nemo die*

*> hits <- tc$search\_features('"nemo kill\*"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*Error in `$<-.data.frame`(`\*tmp\*`, "is\_kw", value = logical(0)) :*

*replacement has 0 rows, data has 7*

*> head(kwic$kwic, 5)*

*[1] "...an invention should <die> with its inventor! \" Captain <Nemo> did not reply..."*

*> #does the nemo die*

*> hits <- tc$search\_features('"nemo kill\*"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*Error in `$<-.data.frame`(`\*tmp\*`, "is\_kw", value = logical(0)) :*

*replacement has 0 rows, data has 7*

*> #does the nemo die*

*> hits <- tc$search\_features('"nemo gone"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] "...steps. Captain <Nemo> was there. He had not <gone> to rest...."*

*> #does nemo die*

*> hits <- tc$search\_features('"nemo dead"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] "...Where was Captain <Nemo>? Had he [...] Were his companions <dead> with him?..."*

*> #does the nemo die*

*> hits <- tc$search\_features('"nemo s?nk\*"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] "...the sun seemed <sinking> towards the horizon. Captain <Nemo> had already risen..."*

*[2] "...Spanish Government had <sunk>. Here Captain <Nemo> came, according..."*

*> #is the nautilus a sub*

*> hits <- tc$search\_features('"nautilus sub\*"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] "...thought that the <Nautilus> was about to return to its <submarine> navigation. I..."*

*[2] "...light. The <Nautilus> remained motionless, [...] of its screw <subdued> by the inclination..."*

*[3] "...never so deeply <submerged> as to cause [...] break. The <Nautilus> followed it to..."*

*[4] "...companions within the <Nautilus>, but a [...] either monstrous or <sublime>, which time..."*

*[5] "...thought that the <Nautilus> was about to continue its <submarine> excursion, and..."*

*> #try to see if nautilus actually sinks*

*> hits <- tc$search\_features('"nautilus destroy\*"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*Error in `$<-.data.frame`(`\*tmp\*`, "is\_kw", value = logical(0)) :*

*replacement has 0 rows, data has 7*

*> head(kwic$kwic, 5)*

*[1] "...thought that the <Nautilus> was about to return to its <submarine> navigation. I..."*

*[2] "...light. The <Nautilus> remained motionless, [...] of its screw <subdued> by the inclination..."*

*[3] "...never so deeply <submerged> as to cause [...] break. The <Nautilus> followed it to..."*

*[4] "...companions within the <Nautilus>, but a [...] either monstrous or <sublime>, which time..."*

*[5] "...thought that the <Nautilus> was about to continue its <submarine> excursion, and..."*

*> #try to see if nautilus actually sinks*

*> hits <- tc$search\_features('"nautilus wreck\*"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*Error in `$<-.data.frame`(`\*tmp\*`, "is\_kw", value = logical(0)) :*

*replacement has 0 rows, data has 7*

*> #try to see if nautilus actually sinks*

*> hits <- tc$search\_features('"nautilus gone"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] ".... Had the <Nautilus> quitted the surface [...]? Had it <gone> back to the..."*

*[2] "...future? The <Nautilus>, leaving the [...] Gibraltar, had <gone> far out...."*

*[3] "...Now if the <Nautilus>, on leaving [...] Gibraltar, had <gone> to the south..."*

*[4] "...sea, the <Nautilus> could not have <gone> through it...."*

*[5] "...was when the <Nautilus> had <gone> under the waves..."*

*> #try to see if nautilus actually sinks*

*> hits <- tc$search\_features('"nautilus attack\*"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] "...shore, the <Nautilus> will have nothing [...] fear from their <attacks>. \" The..."*

*[2] "...light to the <Nautilus>, still protected her from outward <attack>, and transformed..."*

*[3] "...do to the <Nautilus>? Can it <attack> it beneath the..."*

*[4] "...I thought the <Nautilus> was preparing for <attack>; but Captain..."*

*[5] "...idea, the <Nautilus> would <attack> the ship at..."*

*> #try to see if nautilus actually sinks*

*> hits <- tc$search\_features('"nautilus breach\*"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*Error in `$<-.data.frame`(`\*tmp\*`, "is\_kw", value = logical(0)) :*

*replacement has 0 rows, data has 7*

*> #try to see if nautilus actually sinks*

*> hits <- tc$search\_features('"nautilus hull"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] "...part of the <hull> of the <Nautilus>, and occupies..."*

*[2] "...made in the <hull> of the <Nautilus>, that corresponds..."*

*[3] "...raised about the <hull> of the <Nautilus>, and furnished..."*

*[4] "...shells. The <hull> of the <Nautilus>, resembling a..."*

*[5] "...pressure, the <hull> of the <Nautilus> quivered like a..."*

*> #try to see if nautilus actually sinks*

*> hits <- tc$search\_features('"nautilus hole"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] "...a man - <hole> made in the hull of the <Nautilus>, that corresponds..."*

*> #try to see if nautilus actually sinks*

*> hits <- tc$search\_features('"hull hole"~10')*

*> kwic <- tc$kwic(hits, ntokens = 3)*

*> head(kwic$kwic, 5)*

*[1] "...a man - <hole> made in the <hull> of the Nautilus..."*

*> #what monster (from negative words)*

*> hits <- tc$search\_features('"monster"')*

*> kwic <- tc$kwic(hits, ntokens = 10)*

*> head(kwic$kwic, 5)*

*[1] "...between the United States and Europe, respectively signalled the <monster> to each other in 42 ° 15 ' N...."*

*[2] "...moment it was decided to pursue the monster, the <monster> did not appear. For two months no one heard..."*

*[3] "...end of my life, was to chase this disturbing <monster> and purge it from the world. But I had..."*

*[4] "..., my friend, it has to do with the <monster> - - the famous narwhal. We are going to..."*

*[5] "...the soul of it. On the question of the <monster> there was no doubt in his mind, and he..."*

*> #do we find the monster*

*> hits <- tc$search\_features('"find monster"~10')*

*> kwic <- tc$kwic(hits, ntokens = 10)*

*Error in `$<-.data.frame`(`\*tmp\*`, "is\_kw", value = logical(0)) :*

*replacement has 0 rows, data has 21*

*> #who is ned*

*> hits <- tc$search\_features('"ned"')*

*> kwic <- tc$kwic(hits, ntokens = 10)*

*> head(kwic$kwic, 5)*

*[1] "...Go ahead, \" cried Commander Farragut. CHAPTER IV <NED> LAND Captain Farragut was a good seaman, worthy of..."*

*[2] "...or failure of the expedition. But, seeing that <Ned> Land let me speak without saying too much himself,..."*

*[3] "...and I rowing, we sped along quickly, and <Ned> steered in the straight passage that the breakers left between..."*

*[4] "...The boat was well handled, and moved rapidly. <Ned> Land could not restrain his joy. He was like..."*

*[5] "...no other quadruped on this island. \" \" Friend <Ned> is uneasy about it, \" said Conseil. \"..."*

*> hits <- tc$search\_features('"ned nemo"~10')*

*> kwic <- tc$kwic(hits, ntokens = 10)*

*> head(kwic$kwic, 5)*

*[1] ".... \" \" No one will force you, Master <Ned>, \" said Captain <Nemo>. \" Is Conseil going to risk it? \"..."*

*[2] "..., seeing I did not reply. \" Well, <Ned>, do you wish me to ask Captain <Nemo> his intentions concerning us? \" \" Yes, sir..."*

*[3] "...- - - - \" I was silenced; Captain <Nemo> rose. \" Whatever <Ned> Land thinks of, attempts, or tries, what..."*

*[4] "...or St. Lawrence had faded away; and poor <Ned>, in despair, had isolated himself like Captain <Nemo>. Conseil and I, however, never left each..."*

*[5] "...that the time has come to part company with Captain <Nemo>. \" \" Friend <Ned>, I do not despair of this stout Nautilus,..."*

What I learned from pard a-c:

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.

What I learned from d-f:

One of the most interesting things I learned about this section was the zipf’s law analysis. It’s very fascinating to see that this simple property can help us analyze other languages that haven’t been decoded yet, and give us small insights into how their alphabets look. This mathematical property allows us to look into the past to uncover universal similarities between cultures. Another interesting piece here is the parts of speech tagging. One of the key components to analyzing speech and text is tokenization, which is what this strategy does. This allows for easier analysis of language and intention, and I had not previously worked with libraries that did this in the past other than programming language compilers.

Part g of this project helped me to learn just how much insight into the meaning of text can be gleaned through data analysis. I didn’t read the book, but I feel as if I have a relatively good idea about what goes on in the story. I took NLP last semester, so all the text processing and the sentiment analysis and frequency analysis was not new to me. Even the idea of gleaning information through NLP wasn’t new, my final project in NLP was a summarization project. But I never gained insight in that way, through word queries based on the most important words and making intelligent guesses about what may happen in the text. Like at first, I thought the Nautilus was a ship and when I saw it sunk I thought it was destroyed. But then I thought what if it’s a submarine, and turned out it was. That was the most interesting takeaway for me, as someone who’s done NLP, the process I used here to gain insight into the text.