Homework 1

Victoria Xiu Xie

2/18/2022

Question 1

```
# load data
speeches <- corpus_subset(data_corpus_inaugural, President == "Reagan")
#meta(speeches)
#ndoc(speeches)

# function to calculate ttr
q1_fn <- function(dfm){
  ttr_ls <- vector(mode = "list", length = ndoc(speeches))

  for(i in 1:ndoc(dfm)){
    ttr = ntype(dfm)[i]/ntoken(dfm)[i]
    ttr_ls <- append(ttr_ls, ttr)
  }
  return(ttr_ls)}
q1_fn(speeches)
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## $`1981-Reagan`
## [1] 0.3244604
##
## $`1985-Reagan`
## [1] 0.3179787
```

```
# dfm (pre-processing - remove punctuation)
speeches_dfm <- dfm(speeches, remove_punct = TRUE, tolower= FALSE)
#topfeatures(speeches_dfm)

# cosine similarity
cos_similarity <- textstat_simil(speeches_dfm, speeches_dfm,
                                 margin = "documents",
                                 method = "cosine")
cos_similarity
```

```
## textstat_simil object; method = "cosine"
##             1981-Reagan 1985-Reagan
## 1981-Reagan       1.000       0.956
```

```
## 1985-Reagan       0.956       1.000
```

(a)

The raw type-token ratio for '1981-Reagan' is 0.324, and '1985-Reagan' 0.318.

(b)

The cosine similarity between the two speeches is 0.956. The top features from the document feature matrix include 'the', 'and', 'of', etc., which are predominantly stop words.

Question 2

```r
###TYPE: unique sequence of characters grouped together in some meaningful way, might plus punctuation
###TOKEN: instance of a type (dog eat dog world has 3 types and 4 tokens)
###TERM: a type that is part of system's dict such as short forms, bigrams, etc.

# stemming
speeches_dfm_2a <- dfm(speeches, remove_punct = TRUE, tolower= FALSE, stem = TRUE)
q1_fn(speeches_dfm_2a)
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## $`1981-Reagan`
## [1] 0.3322368
##
## $`1985-Reagan`
## [1] 0.3178627
```

```r
textstat_simil(speeches_dfm_2a, speeches_dfm_2a,
               margin = "documents",
               method = "cosine")
```

```
## textstat_simil object; method = "cosine"
##             1981-Reagan 1985-Reagan
## 1981-Reagan       1.000       0.957
## 1985-Reagan       0.957       1.000
```

```r
# remove stop words
speeches_dfm_2b <- dfm(speeches, remove_punct = TRUE, tolower= FALSE, remove = stopwords("english"))
q1_fn(speeches_dfm_2b)
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## $`1981-Reagan`
```

```
## [1] 0.6608544
##
## $`1985-Reagan`
## [1] 0.6059908
```

```r
textstat_simil(speeches_dfm_2b, speeches_dfm_2b,
               margin = "documents",
               method = "cosine")
```

```
## textstat_simil object; method = "cosine"
##             1981-Reagan 1985-Reagan
## 1981-Reagan       1.000       0.668
## 1985-Reagan       0.668       1.000
```

```r
# convert to lower case
speeches_dfm_2c <- dfm(speeches, remove_punct = TRUE, tolower= TRUE)
q1_fn(speeches_dfm_2c)
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## $`1981-Reagan`
## [1] 0.3466283
##
## $`1985-Reagan`
## [1] 0.3377535
```

```r
textstat_simil(speeches_dfm_2c, speeches_dfm_2c,
               margin = "documents",
               method = "cosine")
```

```
## textstat_simil object; method = "cosine"
##             1981-Reagan 1985-Reagan
## 1981-Reagan       1.000       0.959
## 1985-Reagan       0.959       1.000
```

```r
# tf-idf weighting
weighted_speeches_dfm <- dfm_tfidf(speeches_dfm)
q1_fn(weighted_speeches_dfm)
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## $`1981-Reagan`
## [1] 2.772021
##
## $`1985-Reagan`
## [1] 2.681159
```

```r
textstat_simil(weighted_speeches_dfm, weighted_speeches_dfm,
               margin = "documents",
               method = "cosine")
```

```
## textstat_simil object; method = "cosine"
##             1981-Reagan 1985-Reagan
## 1981-Reagan           1           0
## 1985-Reagan           0           1
```

(a)

Theoretical argument: Stemming should not affect either the type-token ratio or the cosine similarity by a large scale because it decreases the diversity in both types and tokens.

TTR: the new type-token ratio for '1981-Reagan' is 0.332, and '1985-Reagan' 0.318. The former has slightly increased, while the latter decreased.

Cosine similarity: the cosine similarity slightly increased to 0.957.

(b)

Theoretical argument: Getting rid of stop words should higher the TTR and lower the cosine similarity, because we are getting rid of a lot of the tokens that appear frequently, which make up a big part of the two documents' original similarity.

TTR: the new type-token ratio for '1981-Reagan' is 0.661, and '1985-Reagan' 0.606. Both have doubled compared to 1(a).

Cosine similarity: the cosine similarity has significantly decreased to 0.668.

(c)

Theoretical argument: Converting to lower case might not affect the TTR much, but increase the cosine similarity, because it affects tokens and types by approximately the same amount, and identifies more shared tokens that originally differ by capitalization.

TTR: the new type-token ratio for '1981-Reagan' is 0.347, and '1985-Reagan' 0.338. Both have increased slightly compared to 1(a).

Cosine similarity: the cosine similarity slightly increased to 0.959.

(d)

Theoretical argument: tf-idf weighting does not make sense here, because in the cause of a word appearing in both documents, the inverse document frequency is equal to 0. The corpus size is too small for tf-idf to be effective.

TTR: the new type-token ratio for '1981-Reagan' is 2.772, and '1985-Reagan' 2.681. Both have increased significantly.

Cosine similarity: the cosine similarity sis 0.

Question 3

```r
hdl1 <- "Nasa Mars rover: Perseverance robot all set for big test."
hdl2 <- "NASA Lands Its Perseverance Rover on Mars."

dfm_q3 <- dfm(c(hdl1, hdl2), remove_punct = TRUE, tolower= TRUE)
```

```r
mat_q3 <- as.matrix(dfm_q3)
doc1 = mat_q3[1, ]
doc2 = mat_q3[2, ]

# for loop to calculate euclidean dist
euc_sum = 0
for(i in 1:dim(mat_q3)[2]){
  euc_sum = euc_sum + (doc1[i] - doc2[i])^2
}
euc_dist <- sqrt(euc_sum)
```

(a)

The pre-processing of my choice includes the removal of punctuation and capitalization. This is because the 2 given document have relatively simple and similar structures and are short in length. The Euclidean distance I found is 3.

(b)

```r
man_dist = 0
for(i in 1:dim(mat_q3)[2]){
  man_dist = man_dist + abs(doc1[i] - doc2[i])
}
man_dist
```

```
## nasa
##    9
```

The Manhattan distance I found is 9.

(c)

```r
cos_num = 0
doc1_norm_sq = 0
doc2_norm_sq = 0

for(i in 1:dim(mat_q3)[2]){
  man_dist = man_dist + abs(doc1[i] - doc2[i])
  cos_num = cos_num + doc1[i] * doc2[i]
  doc1_norm_sq = doc1_norm_sq + doc1[i]^2
  doc2_norm_sq = doc2_norm_sq + doc2[i]^2
}
cos_num / (sqrt(doc1_norm_sq) * sqrt(doc2_norm_sq))
```

```
##      nasa
## 0.4780914
```

The cosine similarity I found is 0.478.

(d)

"robot" -> "rover": replace b, o, and t with v, e, and r –> Levenshtein distance is 3.

Question 4

(a)

```
n<-gutenberg_authors[,]

# list of authors
author_list <- c("Poe, Edgar Allan", "Twain, Mark", "Shelley, Mary Wollstonecraft","Doyle, Arthur Conan

#Here a list of the gutenberg_id associated with the books is given below
book_list<-c(932,1064,1065,32037,74,76,86,91,84,6447,15238,18247,108,126,139,244)

#Using the following command you can check the information associated with the first four novels for ea

#The gutenberg_id above were obtained with the following command
#meta <- gutenberg_works(author == "Doyle, Arthur Conan") %>% slice(1:4)

# Prepare data function
# @param author_name: author's name as it would appear in gutenberg
# @param num_texts: numeric specifying number of texts to select
# @param num_lines: num_lines specifying number of sentences to sample
meta <- gutenberg_works(gutenberg_id == book_list)
meta <- meta %>% mutate(author = unlist(str_split(author, ","))[1] %>% tolower(.))

prepare_dt <- function(book_list, num_lines, removePunct = TRUE){
  meta <- gutenberg_works(gutenberg_id == book_list)
  meta <- meta %>% mutate(author = unlist(str_split(author, ","))[1] %>% tolower(.))

  texts <- lapply(book_list, function(x) gutenberg_download(x,    mirror="http://mirrors.xmission.com/gu
                        #select(text) %>%
                        sample_n(500, replace=TRUE) %>%
                        unlist() %>%
                        paste(., collapse = " ") %>%
                        str_replace_all(., "^ +| +$|( ) +", "\\1"))

  # remove apostrophes
  texts <- lapply(texts, function(x) gsub("'|'", "", x))
  if(removePunct) texts <- lapply(texts, function(x)
  gsub("[^[:alpha:]]", " ", x))
  # remove all non-alpha characters
  output <- tibble(title = meta$title, author = meta$author, text =
  unlist(texts, recursive = FALSE))
}

# run function
set.seed(1984L)
texts_dt <- lapply(book_list, prepare_dt, num_lines = 500, removePunct = TRUE)
texts_dt <- do.call(rbind, texts_dt)
```

```
print(texts_dt$title)
```

```
##  [1] "The Fall of the House of Usher"
##  [2] "The Masque of the Red Death"
##  [3] "The Raven"
##  [4] "Eureka: A Prose Poem"
##  [5] "The Adventures of Tom Sawyer"
##  [6] "Adventures of Huckleberry Finn"
##  [7] "A Connecticut Yankee in King Arthur's Court"
##  [8] "Tom Sawyer Abroad"
##  [9] "Frankenstein; Or, The Modern Prometheus"
## [10] "Proserpine and Midas"
## [11] "Mathilda"
## [12] "The Last Man"
## [13] "The Return of Sherlock Holmes"
## [14] "The Poison Belt"
## [15] "The Lost World"
## [16] "A Study in Scarlet"
```

```
print(texts_dt$author)
```

```
##  [1] "poe"     "poe"     "poe"     "poe"     "twain"   "twain"   "twain"
##  [8] "twain"   "shelley" "shelley" "shelley" "shelley" "doyle"   "doyle"
## [15] "doyle"   "doyle"
```

(b)

```
df_q4 <- data.frame(texts_dt)
str(df_q4)
```

```
## 'data.frame':    16 obs. of  3 variables:
##  $ title : chr  "The Fall of the House of Usher" "The Masque of the Red Death" "The Raven" "Eureka: A
##  $ author: chr  "poe" "poe" "poe" "poe" ...
##  $ text  : chr  "
```

(c)

```
stopwords_en <- stopwords("en")

# Tokenization selections can optionally be passed as the filter argument
filter <- corpus::text_filter(drop_punct = TRUE, drop_number = TRUE, map_case = TRUE, drop = stopwords_e

# fits n-fold cross-validation
vocab_custom <- stylest_select_vocab(df_q4$text, df_q4$author,
                                     filter = filter, smooth = 1, nfold = 5,
                                     cutoff_pcts = c(25, 50, 75, 99))
print(vocab_custom$cutoff_pct_best)
```

```
## [1] 75
```

```
print(vocab_custom$miss_pct)
```

```
##          [,1]     [,2]     [,3]     [,4]
```

```
## [1,] 33.33333 33.33333 33.33333 33.33333
## [2,] 33.33333 33.33333  0.00000 33.33333
## [3,] 25.00000 25.00000 25.00000 25.00000
## [4,] 50.00000 50.00000 50.00000  0.00000
## [5,] 25.00000 25.00000 25.00000 50.00000
```

For pre-processing options, I chose to drop punctuation, numbers, and capitalization in order to have consistent characters and formatting. The 75 percentile has the best prediction rate. The mean rate of incorrectly predicted speakers of held-out texts is printed above.

(d)

```r
# subset features
vocab_subset <- stylest_terms(df_q4$text, df_q4$author, vocab_custom$cutoff_pct_best , filter = filter)

# fit model with "optimal" percentile threshold (i.e. feature subset)
style_model <- stylest_fit(df_q4$text, df_q4$author, terms = vocab_subset, filter = filter)

# report top 5 terms
authors <- unique(df_q4$author)
term_usage <- style_model$rate
lapply(authors, function(x) head(term_usage[x,][order(-term_usage[x,])])) %>% setNames(authors)
```

```
## $poe
##        upon        door         one     chamber         now     nothing
## 0.010938874 0.006981091 0.006871152 0.006761214 0.006431398 0.005112137
##
## $twain
##           t           s         tom         got        said         see
## 0.024639678 0.013830013 0.008213226 0.008001272 0.007895295 0.007789318
##
## $shelley
##         one           s         now         may        love        life
## 0.006079845 0.005326590 0.004788551 0.004465727 0.004142903 0.003927688
##
## $doyle
##        said        upon         one           s          us         man
## 0.010603680 0.010267056 0.008920557 0.006564183 0.005778725 0.005778725
```

Some of these terms makes a lot of sense because they are commonly used in the English language, whereas the term "s" does not make exact sense. My guess is that Twain, Shelley, and Doyle used a lot of "'s" in their writing to indicate possession, and after removing punctuation the "s" term was left out.

(e)

```r
# convert into data.frame
rate_mat <- data.frame(style_model[['rate']])

# select two authors
vec_poe <- rate_mat[2, ]
vec_twain <- rate_mat[4, ]

# create ratio vector
```

```r
vec_poe_to_twian <- vec_poe/vec_twain

# arrange and extract top 5
new_authors <- c("poe", "twain")
sorted_ratios <- lapply(new_authors, function(x) head(vec_poe_to_twian[x,][order(-vec_poe_to_twian[x,])]
sorted_ratios$poe[1:5]
```

```
##       raven     soul      thy   prince   velvet
## poe 84.0277 77.80343 65.35488 52.90633 46.68206
```

The top 5 terms are "raven", "soul", "thy", "prince", and "velvet." These are likely the words that Poe liked to use a lot that fits the theme of his writing, and Twain used very rarely.

(f)

```r
# load mystery
mystery <- readRDS('mystery_excerpt.rds')
pred <- stylest_predict(style_model, mystery)
pred$predicted
```

```
## [1] twain
## Levels: doyle poe shelley twain
```

```r
pred$log_probs
```

```
## 1 x 4 Matrix of class "dgeMatrix"
##          doyle       poe   shelley         twain
## [1,] -31.42456 -72.91229 -49.92632 -2.242651e-14
```

According to the fitted model, Twain is the most likely author.

(g)

```r
# create dfm of text excerpts
dfm_q4 <- dfm(df_q4$text)

# bigrams
bigram <- textstat_collocations(df_q4$text, min_count = 5)

# top 10 lambda
print(head(bigram[order(-bigram$lambda), ]$collocation, 10))
```

```
##  [1] "edgar allan"         "denser perfumed"     "whispering vows"
##  [4] "syllable expressing" "candelabrum amid"    "unseen censer"
##  [7] "allan poe"           "arabesque figures"   "densely crowded"
## [10] "unsuited limbs"
```

```r
# top 10 count
print(head(bigram[order(-bigram$count), ]$collocation, 10))
```

```
##  [1] "of the"   "in the"   "and the"  "to the"   "it was"   "on the"
##  [7] "of a"     "from the" "to be"    "that the"
```

The set of n-grams with top counts is more likely to be multi-word expressions, because the bigrams are all combinations of stop words.

Question 5

(a)

```
# Load data
q5_data <- corpus(data_corpus_ungd2017)

# Make snippets of 1 sentence each, then clean them
snippetData <- snippets_make(q5_data, minchar = 150, maxchar = 350)
snippetData <- snippets_clean(snippetData)
```

```
## Cleaning 6,751 snippets...

##    removed 190 snippets containing numbers of at least 1,000

##    ...finished.
```

```
head(snippetData, 10)
```

```
##          docID snippetID
## 1  Afghanistan    100001
## 2  Afghanistan    100002
## 3  Afghanistan    100003
## 4  Afghanistan    100009
## 5  Afghanistan    100011
## 6  Afghanistan    100012
## 7  Afghanistan    100015
## 8  Afghanistan    100016
## 9  Afghanistan    100017
## 10 Afghanistan    100020
##
## 1                                                     As I stand here before the General Assembly to
## 2                                Shaped by the Great Depression and tempered by the carnage of the Seco
## 3                   The United Nations, the International Monetary Fund, the World Bank and other orga
## 4                                                             There is an emerging consensus that ad
## 5                                                              Sixteen years after the tragedy of 11
## 6      Driven by transnational terrorist networks, criminal organizations, cybercrime and State spon
## 7  Terrorism is not only an attack on human life and basic freedoms, but an attack on the compact of
## 8                                                                      We must confront the threat o
## 9                             Lastly, despite the incorporation of tenets of the Universal Declaration
## 10                                                                       I welcome the chance for Af
```

(b)

```
# Sample the snippets
testData <- sample_n(snippetData, 1000)

# generate n-1 pairs from n test snippets for a minimum spanning tree
snippetPairsMST <- pairs_regular_make(testData)

# gold questions
gold_questions <- pairs_gold_make(snippetPairsMST, n.pairs = 10)
```

```
## Starting the creation of gold questions...
```

```
##    computing Flesch readability measure
##    selecting top different 10 pairs
##    applying min.diff.quantile thresholds of 3.54, 40.87
##    creating gold_reason text
##    ...finished.
```

```
print(gold_questions[, c("text1", "text2")])
```

```
##
## 1
## 2
## 3                                                                                    Bulgaria categori
## 4
## 5                                                         Ghana will also continue to be active in the mu
## 6
## 7
## 8                     Eliminating radicalism and religious fundamentalism should also be a major prio
## 9                        Practical approaches could allow us to work through existing controversies i
## 10 On behalf of the people and the Government of the Republic of Paraguay, I wish to express to the 
##
## 1
## 2
## 3
## 4  Accordingly, this year has witnessed numerous initiatives for fruitful cooperation, notably the l
## 5
## 6                                                                                             The Holy
## 7
## 8
## 9
## 10
```

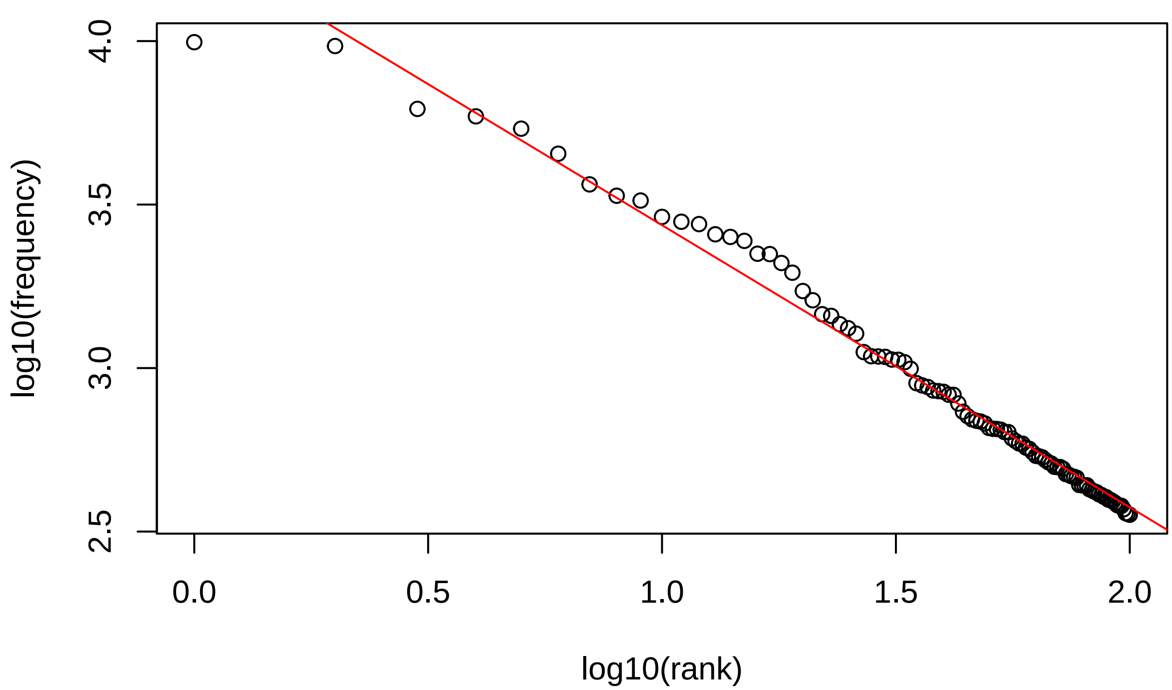
My classification: - 1. Text 2 is easier to read; - 2. Text 2 is easier to read; - 3. Text 1 is easier to read; - 4. Text 1 is easier to read; - 5. Text 1 is easier to read; - 6. Text 1 is easier to read; - 7. Text 2 is easier to read; - 8. Text 1 is easier to read; - 9. Text 2 is easier to read; - 10. Text 2 is easier to read;

For 9 of the 10 gold pairs was I in agreement with the automated classification - we differed on the 7th pair, which I think was hard to comprehend given the abundant statistics referenced, but the machine found easier to read because of its combination of shorter sentences.

Question 6

```
# Prepare data function
prepare_dt <- function(book_id, removePunct = TRUE){
  #meta <- gutenberg_works(gutenberg_id  == book_id)
  #meta <- meta %>% mutate(author = unlist(str_split(author, ","))[1] %>% tolower(.))
  text <- gutenberg_download(book_id, mirror="http://mirrors.xmission.com/gutenberg/") %>%
    select(text) %>%
    filter(text!="") %>%
    unlist() %>%
    paste(., collapse = " ") %>%
    str_replace_all(., "^ +| +$|( ) +", "\\1")
  text <- gsub("`|'", "", text) # remove apostrophes
```

```
  text <- gsub("[^[:alpha:]]", " ", text) # remove all non-alpha characters
  output <- tibble(text = text)
}

#title = meta$title, author = meta$author,
title <- c("Little Women", "The Great Gatsby")
author <- c("alcott", "fitzgerald")

# run function
pair_texts <- lapply(c(514, 64317), prepare_dt, removePunct = TRUE)
pair_texts <- do.call(rbind, pair_texts)
pair_texts$title <- title
pair_texts$author <- author

# create dfm
pair_texts_dfm <- dfm(pair_texts$text, remove_punct = TRUE, tolower = TRUE)
#print(pair_texts_dfm)

plot(log10(1:100), log10(topfeatures(pair_texts_dfm, 100)),
     xlab = "log10(rank)", ylab = "log10(frequency)", main = " ")

# Fits a linear regression to check if slope is approx -1.0
regression_q6 <- lm(log10(topfeatures(pair_texts_dfm, 100)) ~ log10(1:100))

# Adds the fitted line from regression to the plot
abline(regression_q6, col = "red")
```



Pre-processing includes the removal of punctuation and capitalization in order to have a more consistent vocabulary. In this graph we can see that the slope is approximately -1, which indicates that the relationship between log(collection size) and log(vocab size) is linear, demonstrating Zipf's Law.

Question 7

```r
# M = kT^b
M <- nfeat(pair_texts_dfm)
k <- 44
tokens_q7 <- tokens(pair_texts$text, remove_punct = TRUE, tolower = TRUE)
num_tokens_q7 <- sum(lengths(tokens_q7))
# t^b = M/k -> b = logt(M/k)
logb((M/k), base = num_tokens_q7)
```

```
## [1] 0.4592626
```

b is equal to 0.459 in this case. I removed punctuation and capitalization because we don't want to count in those as tokens/different tokens.

Questions 8

```r
kwic(pair_texts$text, pattern = 'class', valuetype = 'regex')
```

```
## Keyword-in-context with 11 matches.
##   [text1, 35414]             clothes which attracts a certain |   class   |
##   [text1, 41616] contributions were excellent being patriotic | classical |
##   [text1, 92058]                        enemies The men of my |   class   |
##   [text1, 99020]                  remained the baby Our drawing |   class   |
##   [text1, 99308]                    Twelve or fourteen in the |   class   |
##  [text1, 102656]                     the story belonged to that |   class   |
##  [text1, 148484]   antique coiffures statuesque attitudes and |  classic  |
##  [text1, 148725]                              If I only had a | classical |
##  [text1, 179267]                  indeed and there s another |   class   |
##  [text1, 180659]                      s laugh and dismiss the |   class   |
##   [text2, 34954]                   he was president of your |   class   |
##
##  of people and secures their
##  comical or dramatic but never
##  were heroes in the eyes
##  breaks up next week and
##  but I dare say they
##  of light literature in which
##  draperies But dear heart we
##  nose and mouth I should
##  who can t ask and
##  in metaphysics There might have
##  at Yale Tom and I
```

```r
#kwic(pair_texts$text, pattern = 'money', valuetype = 'regex')
#kwic(pair_texts$text, pattern = 'poor', valuetype = 'regex')
```

I experimented with keywords such as "class", "money", and "poor". The first two keywords usually appear in the context where the author is trying to describe a person/situation with wealth and status, while the last one usually appears in places associated with misfortune and suffering. In my opinion, Little Women focuses more on the unfairness that social hierarchy brings to people, while The Great Gatsby uses a lot of satire to illustrate the clout chasers.

Question 9

(a)

```r
# load data
data("data_corpus_ukmanifestos")
manifestos <- corpus_subset(data_corpus_ukmanifestos, Party == "Con")

# tokenize by sentences
sent_tokens <- unlist(tokens(manifestos, what = "sentence", include_docvars = TRUE))

# extract year metadata
yearnames <- list(unlist(names(sent_tokens)))
yearnames <- lapply(yearnames[[1]], function(x){strsplit(x, "_")[[1]][3]})
yearslist <- unlist(yearnames)

# create tibble
sentences_df <- tibble(text = sent_tokens, year = yearslist)

# filter out non-sentences (only sentences that end in sentence punctuation
sentences_df <- sentences_df[grepl( ("[\\.\\?\\!]$"), sentences_df$text), ]

# create quanteda corpus object
sent_corp <- corpus(sentences_df$text)
docvars(sent_corp, field = "Year") <- sentences_df$year

# Let's filter out any NAs
sentences_df <- na.omit(sentences_df)

# mean Flesch statistic per year
flesch_point <- sentences_df$text %>% textstat_readability(measure = "Flesch") %>%
  group_by(sentences_df$year) %>%
  summarise(mean_flesch = mean(Flesch)) %>%
  setNames(c("year", "mean")) %>% arrange(year)
print(flesch_point)
```

```
## # A tibble: 16 x 2
##    year   mean
##    <chr> <dbl>
##  1 1945   49.0
##  2 1950   43.9
##  3 1951   52.0
##  4 1955   49.1
##  5 1959   48.4
##  6 1964   45.8
##  7 1966   46.3
##  8 1970   46.1
##  9 1974   42.3
## 10 1979   47.5
## 11 1983   47.7
## 12 1987   46.7
## 13 1992   46.4
## 14 1997   49.9
## 15 2001   48.1
```

```
## 16 2005   49.5
```

```r
# We will use a loop to bootstrap a sample of texts and subsequently calculate standard errors
iters <- 10

# build function to be used in bootstrapping
boot_flesch <- function(party_data){
  N <- nrow(party_data)
  bootstrap_sample <- corpus_sample(corpus(c(party_data$text)), size = N, replace = TRUE)
  bootstrap_sample<- as.data.frame(as.matrix(bootstrap_sample))
  readability_results <- textstat_readability(bootstrap_sample$V1, measure = "Flesch")
  return(mean(readability_results$Flesch))
}

# apply function to each year
boot_flesch_by_year <- pblapply(unique(yearslist), function(x){
  sub_data <- sentences_df %>% filter(year == x)
  output_flesch <- lapply(1:iters, function(i) boot_flesch(sub_data))
  return(unlist(output_flesch))
})
names(boot_flesch_by_year) <- unique(yearslist)

# compute mean and std.errors
year_means <- lapply(boot_flesch_by_year, mean) %>% unname() %>% unlist()
year_ses   <- lapply(boot_flesch_by_year, sd) %>% unname() %>% unlist() # bootstrap standard error = sa

# Plot results--party
plot_dt <- tibble(year = unique(yearslist), mean = year_means, ses = year_ses)
plot_dt
```

```
## # A tibble: 16 x 3
##    year   mean   ses
##    <chr> <dbl> <dbl>
##  1 1945   49.2 1.61
##  2 1950   43.8 1.19
##  3 1951   52.4 2.28
##  4 1955   49.2 1.17
##  5 1959   49.3 0.979
##  6 1964   45.8 1.67
##  7 1966   46.1 1.47
##  8 1970   45.7 1.12
##  9 1974   42.1 0.429
## 10 1979   47.4 0.401
## 11 1983   47.2 0.852
## 12 1987   46.9 0.618
## 13 1992   46.1 0.602
## 14 1997   50.1 0.686
## 15 2001   48.9 0.936
## 16 2005   49.6 1.17
```

(b)

```r
colnames(plot_dt) <- c('year', 'bootstrap_mean', 'bootstrap_se')
merge(flesch_point, plot_dt)[, 1:3]
```

```
##    year     mean bootstrap_mean
## 1  1945 48.96587       49.21921
## 2  1950 43.89979       43.76124
## 3  1951 52.00880       52.35713
## 4  1955 49.08984       49.15044
## 5  1959 48.43225       49.32884
## 6  1964 45.78184       45.79685
## 7  1966 46.26760       46.06425
## 8  1970 46.09232       45.72158
## 9  1974 42.31458       42.12343
## 10 1979 47.47612       47.44255
## 11 1983 47.68076       47.22979
## 12 1987 46.66616       46.91250
## 13 1992 46.39530       46.10310
## 14 1997 49.90513       50.10019
## 15 2001 48.08872       48.90886
## 16 2005 49.48048       49.56690
```

The bootstrapped means are very close to the non-bootstrapped means. This is because the bootstrap samples are all derived from the original data, and so in calculating the mean, the bootstrap sample is just smoothing over the original data.