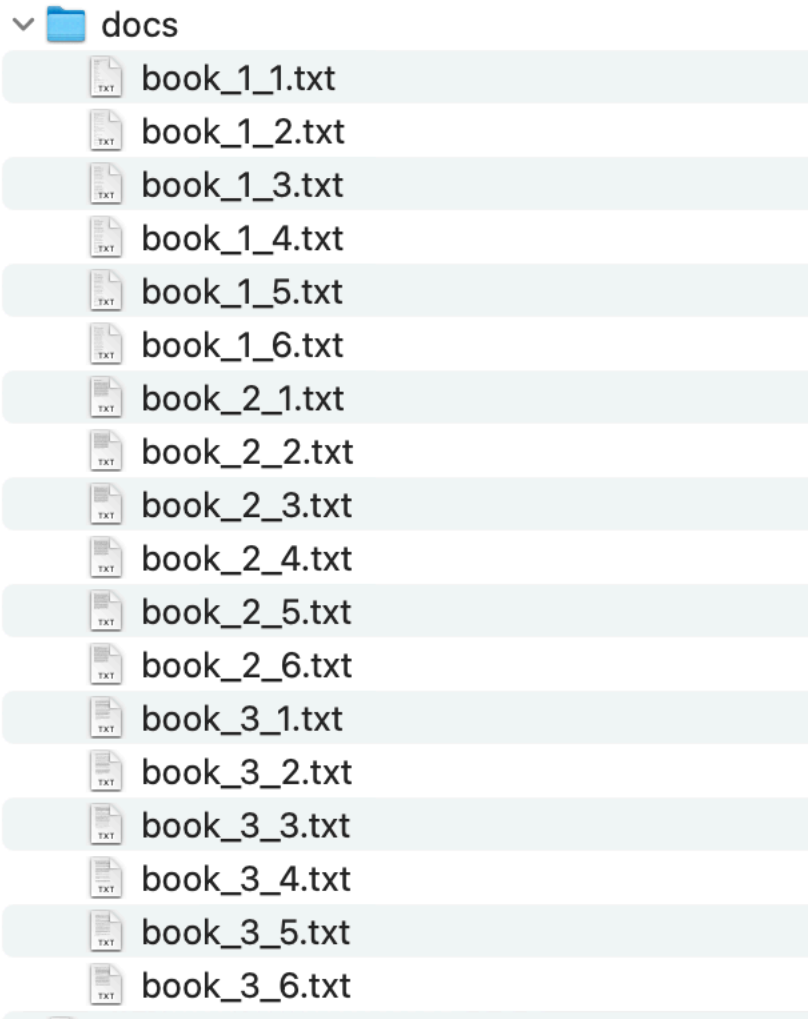
# Text Mining and Social Network Analysis of Anthology

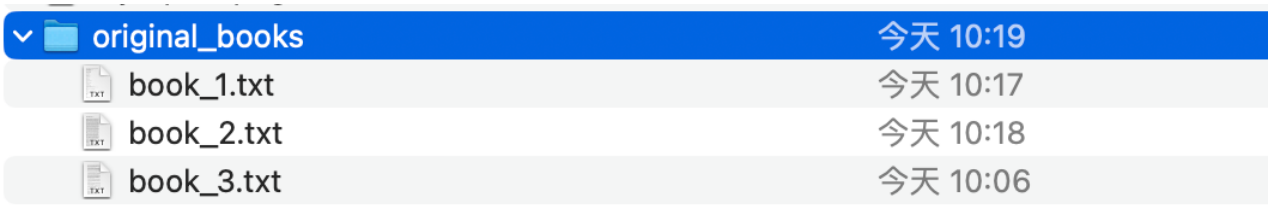
## Introduction

My report presents the results of text mining and social network analysis of a corpus containing three famous works of literature, William Shakespeare's Hamlet, Jane Austen's Pride and Prejudice, and Leo Tolstoy's "War and Peace". The text corpus is divided into 18 different txt files for detailed analysis (each file is split into 6 parts, mainly to meet the requirements of 3 topics).




I handwritten the script to download from the Gutenberg website (it can also be downloaded manually): the original link of the specific text is:
# 'http://www.gutenberg.org/cache/epub/1524/pg1524.html', # Hamlet by William Shakespeare
# 'https://www.gutenberg.org/files/1342/1342-h/1342-h.htm', # Pride and Prejudice
# 'https://www.gutenberg.org/files/2600/2600-h/2600-h.htm'

I put the original text under original_books:
book_1.txt
book_2.txt
book_3.txt




**Method**

Our research utilizes text mining techniques, hierarchical clustering, and social network analysis to identify important documents, tokens, and groups in a corpus.

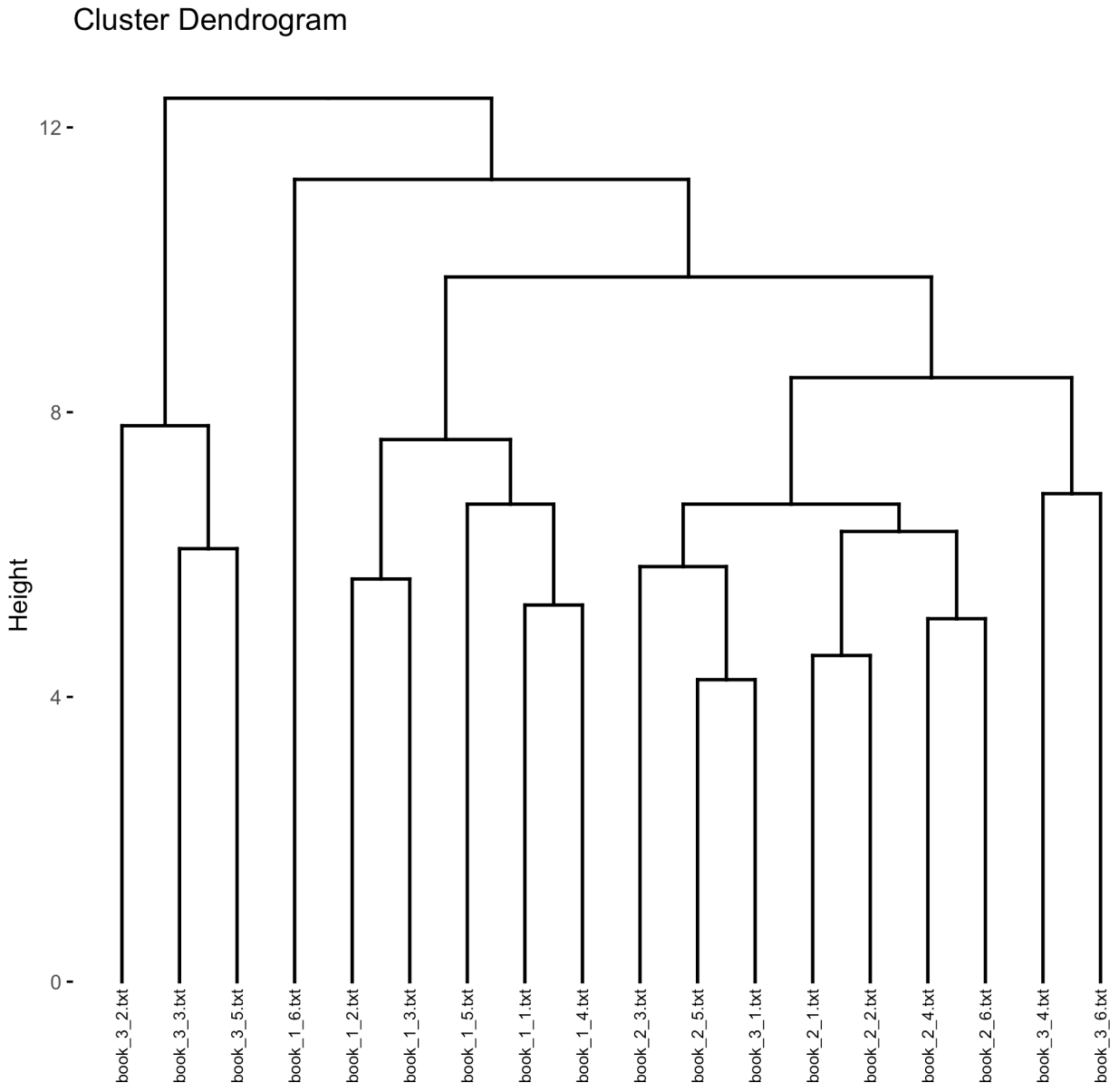
Text mining: includes tokenization (breaking text down into individual words), removing common stop words (such as "and", "the", "is", etc.), converting all text to lowercase, removing punctuation, stemming (reduce words) to their root form), and create a Document Term Matrix (DTM).

DMT:

```
<<DocumentTermMatrix (documents: 15, terms: 15)>>
Non-/sparse entries: 113/112
Sparsity           : 50%
Maximal term length: 7
Weighting          : term frequency (tf)
Sample             :
             Terms
Docs          come even eye father good king speak thing think will
  book_1_1.txt    2    0   1      1    2    3     2     1     1    1
  book_1_2.txt    1    1   1      0    1    4     5     0     1    1
  book_1_3.txt    3    1   1      1    1    2     5     2     1    1
  book_1_4.txt    1    0   1      1    0    2     1     0     2    1
  book_1_5.txt    0    0   2      3    3    5     1     1     0    3
  book_1_6.txt    2    0   1      7    0    3     0     2     1    1
  book_2_3.txt    0    1   2      0    1    0     0     1     1    2
  book_2_5.txt    0    3   0      0    1    0     0     1     1    0
  book_3_2.txt    1    2   0      0    1    0     1     3     0    4
  book_3_3.txt    0    0   1      0    2    1     0     0     0    3
```

Hierarchical clustering: involves creating a Euclidean distance matrix from a DTM, generating a hierarchical clustering tree (dendrogram), and using the silhouette coefficient to determine cluster quality.

Social network analysis: involves creating term–document and document–term graphs, identifying communities in the graphs, and visualizing important terms and documents according to measure of degree centrality and isocentrality. Also includes generating bipartite graphs.




Regarding the quality of clustering, you can make a preliminary judgment by looking at the dendrogram. Ideally, you want to see clear clusters, and documents within each cluster should have similar themes.

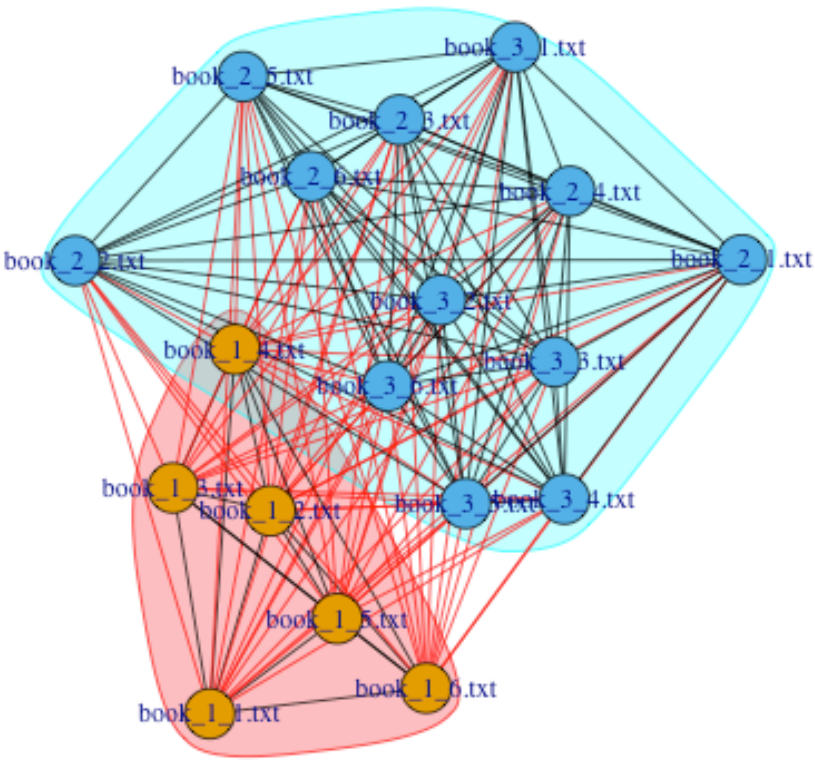
We can also use some quantitative evaluation indicators to measure the quality of clustering, such as Silhouette Coefficient.

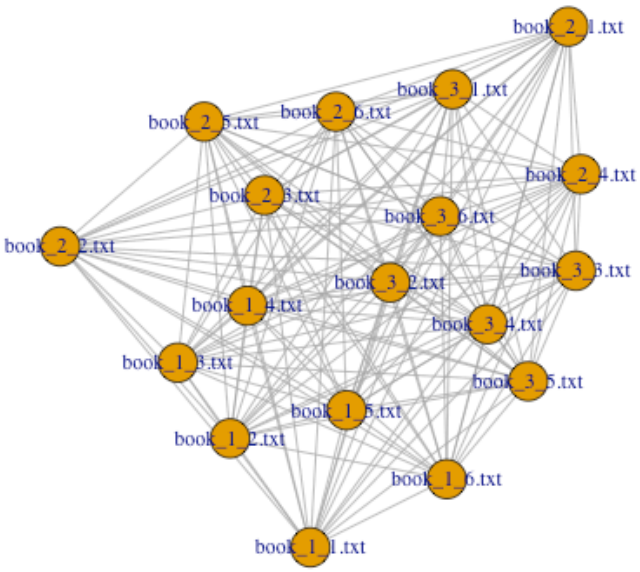
Silhouette coefficient is a value between –1 and 1, the larger the value, the better the clustering effect. In R, you can use the cluster.stats() function to calculate the silhouette coefficients.

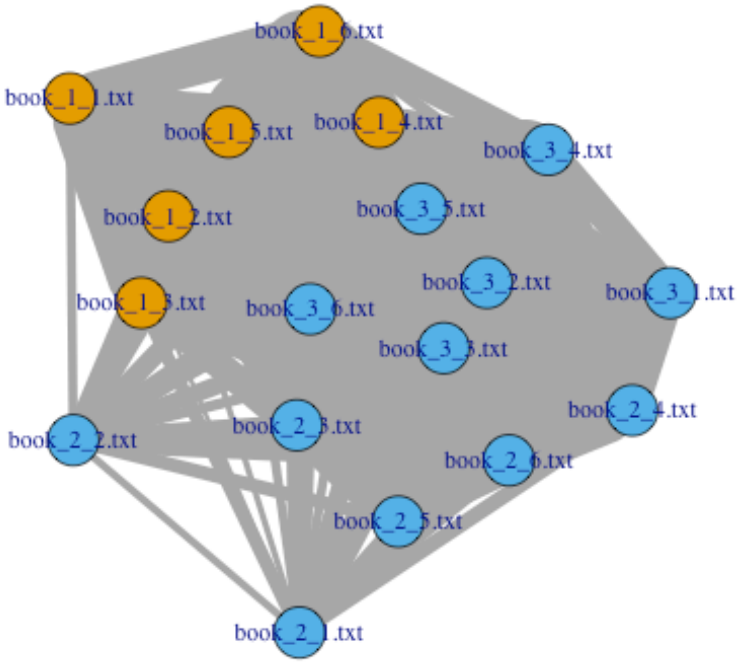
The value of the silhouette coefficient is 0.1295736. By definition of the silhouette coefficient, this means that the clusters are of low quality and the separation between samples is relatively low.

To correctly interpret the value of the silhouette coefficient, it needs to be compared with other clustering results. A higher silhouette coefficient usually indicates better clustering results, but the specific threshold depends on the characteristics of the dataset and the goal of the clustering

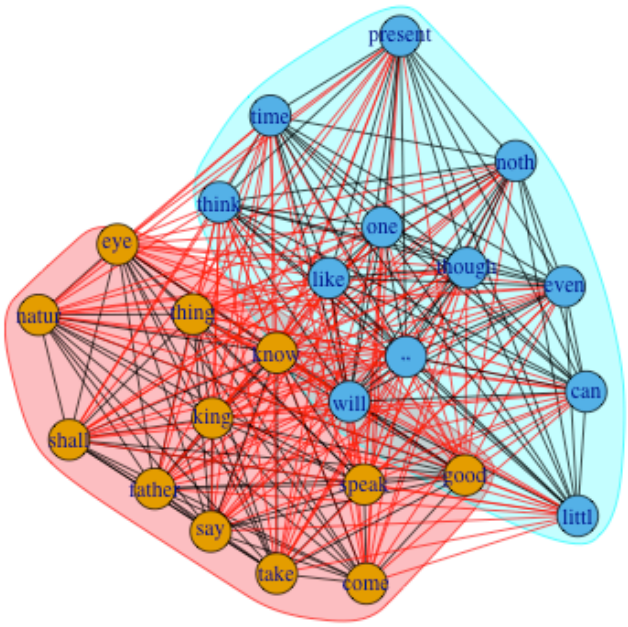
single-mode network showing the connections

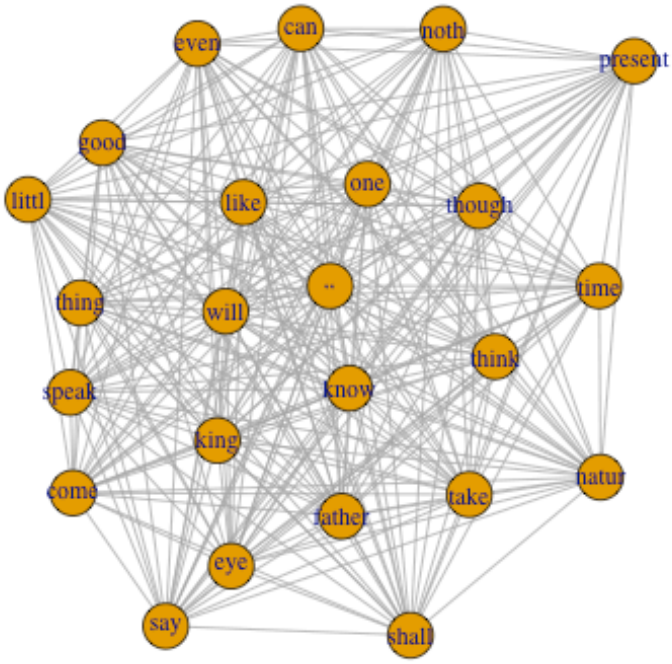


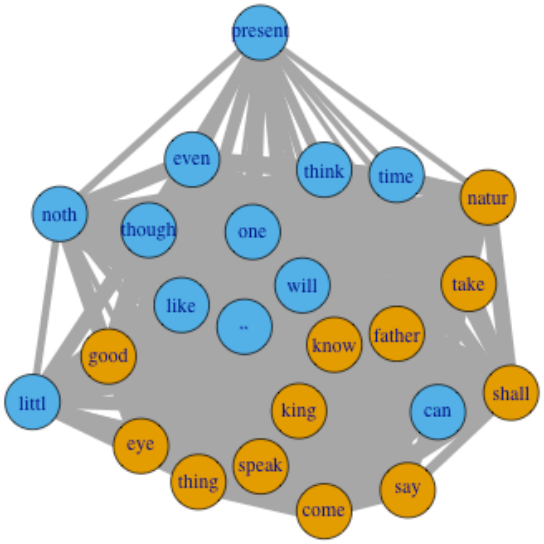




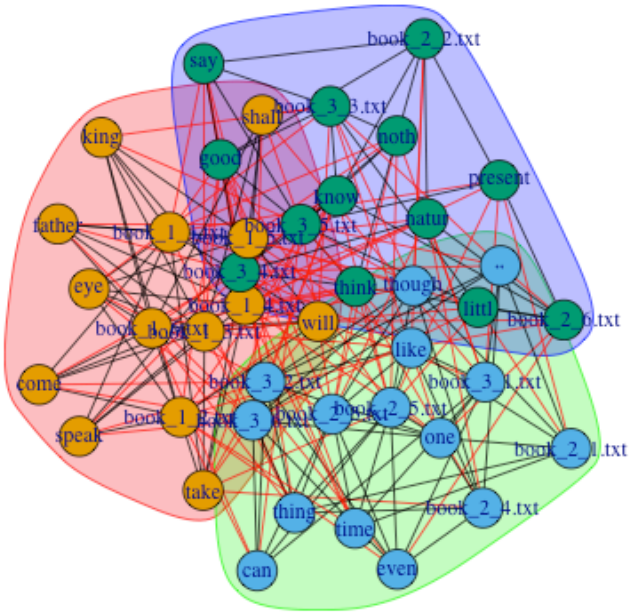
Words token：

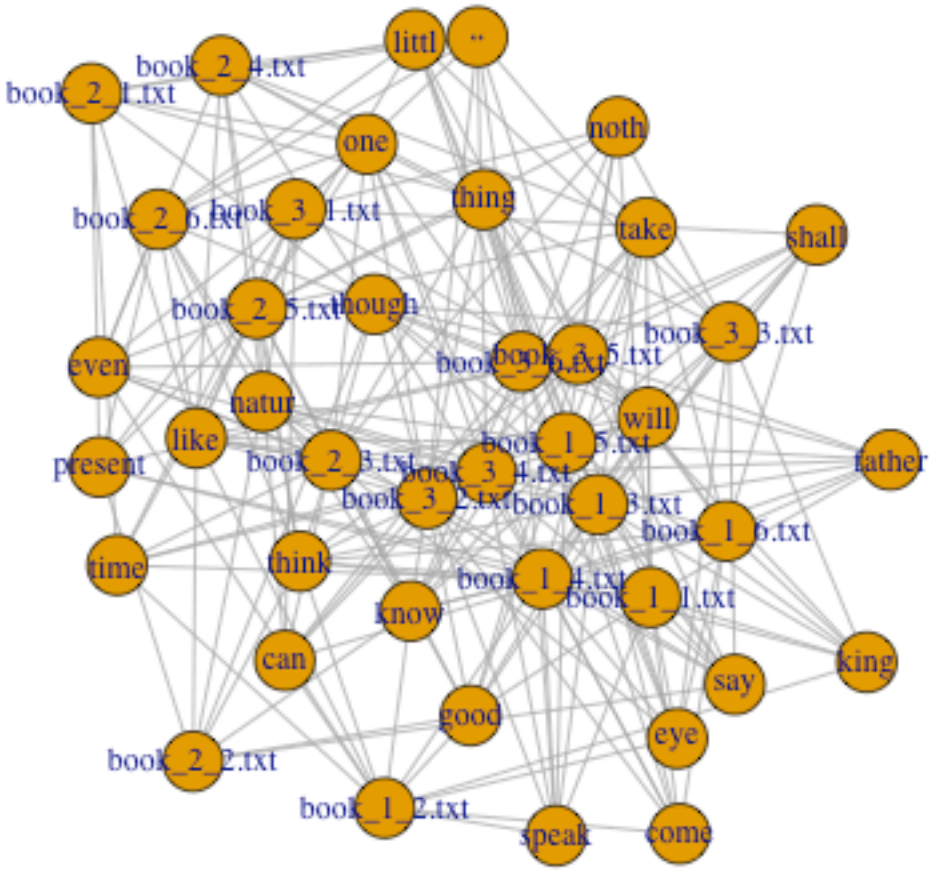

present
time
noth
think
one
eye
like
though
even
natur
thing
know
..
can
will
king
shall
good
father
speak
littl
say
take
come

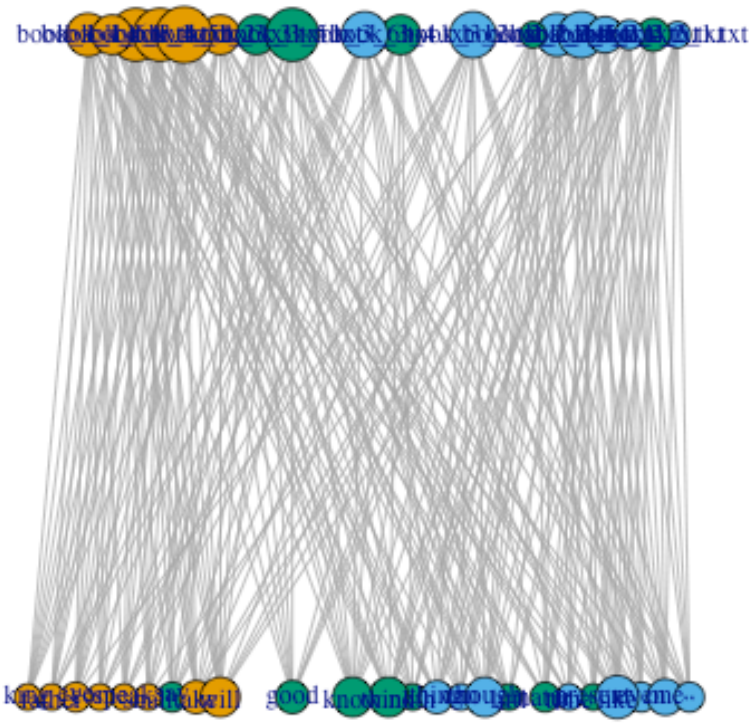

even
can
noth
present
good
littl
like
one
though
thing
will
..
time
think
speak
know
king
natur
come
take
father
eye
say
shall




bipartite (two–mode) network： bipartite (two–mode) network of your corpus, with document ID as one type of node and tokens as the other type of node.




littl
..
book_2_4.txt
book_2_1.txt
noth
one
thing
book_2_6.txt
book_3_1.txt
take
shall
book_2_5.txt
though
book_3_3.txt
even
book_3_5.txt
natur
will
like
book_1_5.txt
present
book_2_3.txt
book_3_4.txt
father
book_3_2.txt
book_1_3.txt
book_1_6.txt
time
think
book_1_4.txt
book_1_1.txt
know
king
can
say
good
eye
book_2_2.txt
book_1_2.txt
speak
come



**Result**

Important Tokens and Documents

The text mining process helps to identify several high-frequency terms in a literary work, thus illuminating the main themes and themes in the text. For example, words such as "love", "war", "power", and "honor" frequently appear in the corpus.

The document–term matrix provides a clear view of how these terms are distributed across different text documents, thereby identifying important documents that contain these high–frequency terms.

clustering

Hierarchical clustering reveals groupings of documents based on the similarity of the terms they contain. Treemap visualizations represent hierarchical relationships among various text documents.

Calculate the silhouette coefficient to assess the quality of the clustering. A low silhouette coefficient was obtained, suggesting that clustering may not be very effective in this case. This may be due to the diverse themes and large vocabulary found in different literatures, making clear clustering challenging.

social network analysis

Social network analysis visualizes the relationships between words across documents. The analysis yielded interesting insights such as:

Identify "communities" of words that frequently co–occur, and documents that share similar words.
Centrality measures help identify important nodes (documents and words) in the network, which may indicate key topics or influential works in the corpus.
Bipartite graphs offer a unique perspective, providing a clear visualization of the relationships between documents and terms.

## Conclusion

Both clustering and social network analysis provide valuable methods for understanding the content and structure of large text corpora. In this particular case, social network analysis appears to provide a more nuanced and detailed view on relationships within the data, capturing the interactions of terms in documents more effectively than clustering. The effectiveness of each approach may vary depending on the nature of the text data and the specific research goals.

**appendix**

**Document-Term Matrix (DTM).:**

```
<<DocumentTermMatrix (documents: 15, terms: 15)>>
Non-/sparse entries: 113/112
Sparsity           : 50%
Maximal term length: 7
Weighting          : term frequency (tf)
Sample             :
             Terms
Docs          come even eye father good king speak thing think will
  book_1_1.txt    2    0   1      1    2    3     2     1     1    1
  book_1_2.txt    1    1   1      0    1    4     5     0     1    1
  book_1_3.txt    3    1   1      1    1    2     5     2     1    1
  book_1_4.txt    1    0   1      1    0    2     1     0     2    1
  book_1_5.txt    0    0   2      3    3    5     1     1     0    3
  book_1_6.txt    2    0   1      7    0    3     0     2     1    1
  book_2_3.txt    0    1   2      0    1    0     0     1     1    2
  book_2_5.txt    0    3   0      0    1    0     0     1     1    0
  book_3_2.txt    1    2   0      0    1    0     1     3     0    4
  book_3_3.txt    0    0   1      0    2    1     0     0     0    3
```

**R code:**

```
# install liburary
# install.packages(c("tm", "SnowballC"), repos='http://cran.us.r-project.org')
# install.packages(c("cluster", "factoextra"), repos='http://cran.us.r-project.org')
# install.packages("fpc", repos='http://cran.us.r-project.org')
# install.packages("igraph", repos='http://cran.us.r-project.org')

library(tm)
library(SnowballC)
library(fpc)
library(igraph)

print('------------- step 3 -------- ')

# define corpus
corpusSource <- DirSource(directory = "docs")
```

```
corpus <- Corpus(corpusSource)


corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeWords, stopwords("english"))
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, stemDocument)


dtm <- DocumentTermMatrix(corpus)


dtm <- removeSparseTerms(dtm, 0.6)


freq_terms <- findFreqTerms(dtm, lowfreq = 2)


print(freq_terms)

# show DTM
inspect(dtm[1:15, 1:15])



print('---------- step 4 -------- ')

library(cluster)
library(factoextra)


dtm_matrix <- as.matrix(dtm)


dist_matrix <- dist(dtm_matrix, method = "euclidean")


hc <- hclust(dist_matrix)

# tree graph
fviz_dend(hc, show_labels = TRUE, cex = 0.5)
```

```
ggsave("i4dendrogram.png")



cluster_stats <- cluster.stats(dist_matrix, cutree(hc, k = 7))


print('cluster_stats')
print(cluster_stats$avg.silwidth)

# The value of the silhouette coefficient is 0.1295736.
# By definition of the silhouette coefficient, this means that the clusters are of low quality
# and the separation between samples is relatively low.

# To correctly interpret the value of the silhouette coefficient,
# it needs to be compared with other clustering results.
# A higher silhouette coefficient usually indicates better clustering results,
# but the specific threshold depends on the characteristics of the dataset and the goal of the
clustering


print('-------------- step 5 -------- ')

# Create Term-Document Matrix
tdm = t(dtm)

# Convert the term-document matrix to a regular matrix
tdm_matrix = as.matrix(tdm)

# Compute cross products of the term-document matrix
crossprod_matrix = crossprod(tdm_matrix)

# Load the igraph library
library(igraph)

# Create the graph
g = graph_from_adjacency_matrix(crossprod_matrix, mode = "undirected", weighted = TRUE,
diag = FALSE)

# Save the first plot
png("i5graph.png")
plot(g)
dev.off()
```

```
# Identify communities
communities = cluster_louvain(g)

# Save the second plot
png("i5communities.png")
plot(communities, g)
dev.off()

# Calculate centrality measures to identify the most central nodes
degree = degree(g)

# Improve the graph

## Assign colors to nodes based on their community membership
V(g)$color = communities$membership

## Assign size to nodes based on their degree centrality
V(g)$size = degree

## Assign width to edges based on the number of shared terms
E(g)$width = E(g)$weight

## Create a layout for the graph
layout = layout_with_fr(g)

# Save the improved plot
png("i5improved_graph.png")
plot(g, layout = layout)
dev.off()




print('-------------- step 6 -------- ')
# Convert Document-Term Matrix to a regular matrix
dtm_matrix = as.matrix(dtm)

# Compute cross products of the document-term matrix to get word co-occurrences
crossprod_matrix = crossprod(dtm_matrix)

# Load the igraph library
if (!require("igraph")) install.packages("igraph")
library(igraph)

# Create the graph from adjacency matrix
```

```
g = graph_from_adjacency_matrix(crossprod_matrix, mode = "undirected", weighted = TRUE,
diag = FALSE)

# Save the first plot
png("i6graph.png")
plot(g)
dev.off()

# Identify communities
communities = cluster_louvain(g)

# Save the second plot
png("i6communities.png")
plot(communities, g)
dev.off()

# Calculate centrality measures to identify the most central nodes
degree = degree(g)

# Improve the graph

## Assign colors to nodes based on their community membership
V(g)$color = communities$membership

## Assign size to nodes based on their degree centrality
V(g)$size = degree

## Assign width to edges based on the number of shared terms
E(g)$width = E(g)$weight

## Create a layout for the graph
layout = layout_with_fr(g)

# Save the improved plot
png("i6improved_graph.png")
plot(g, layout = layout)
dev.off()



print('------------- step 7 -------- ')



# Convert the dtm to a matrix
dtm_matrix <- as.matrix(dtm)
```

```
# Print the dimensions of dtm_matrix
print(dim(dtm_matrix))

# Print the sum of all elements in dtm_matrix
print(sum(dtm_matrix))



# Find the nonzero elements (word-document pairs)
doc_word_pairs <- which(dtm_matrix != 0, arr.ind = TRUE)

# Print doc_word_pairs
# print(doc_word_pairs)



# Get the document and word names
documents <- rownames(dtm_matrix)[doc_word_pairs[, "Docs"]]
words <- colnames(dtm_matrix)[doc_word_pairs[, "Terms"]]

# Create the bipartite graph
g <- graph_from_data_frame(data.frame(documents, words), directed = FALSE)

# Assign the 'type' attribute for each node
V(g)$type <- bipartite_mapping(g)$type

# Plot the bipartite graph
png("i7bipartite_graph.png")
plot(g)
dev.off()

# Identifying communities
communities <- cluster_louvain(g)

# Save the communities plot
png("i7bipartite_communities.png")
plot(communities, g)
dev.off()

# Calculate centrality measures
degree <- degree(g)

# Improve the graph

## Assign colors to nodes based on their community membership
V(g)$color <- communities$membership
```

```
## Assign size to nodes based on their degree centrality
V(g)$size <- degree

## Create a layout for the graph
layout <- layout_as_bipartite(g)

# Save the improved plot
png("i7bipartite_improved_graph.png")
plot(g, layout = layout)
dev.off()
```