Assignment 3

SHIN MINSEO 35685377

2025-06-13

Task 1. Collect 20 documents.

20 documents were collected across four different genres including news, fiction, film reviews and tech articles.

Genre: News articles

1. Israel targets Iran's nuclear sites and military commanders in major attack - follow live. (n.d.). *BBC News*. https://www.bbc.com/news/live/c93ydeqyq71t
2. *Who are the victims of the Air India plane crash?* (2025, June 12). https://www.bbc.com/news/articles/cdd28legnzvo
3. Epstein, K. (2025, June 12). *Los Angeles is latest in Trump's calls to use military at protests*. https://www.bbc.com/news/articles/c626y7gjdzeo
4. Mackenzie, J. (2025, June 13). *North Korea claims warship launch successful on second try*. https://www.bbc.com/news/articles/c1mgd0252kpo
5. Fox-Skelly, J. (2025, June 10). *The everyday activity that can reveal your brain's age*. Bbc.com; BBC. https://www.bbc.com/future/article/20250609-can-your-walking-speed-reveal-your-brains-rate-of-ageing

Genre: Fiction books

1. Orwell, G. (1949). *1984*. Secker & Warburg.
2. Lee, H. (1960). *To Kill a Mockingbird*. Chelsea House Publishers. (Original work published 1960)
3. Fitzgerald, F. S. (1925). *The Great Gatsby*. Scribner.
4. Backman, F. (2014). *A Man Called Ove*. Thorndike Press.
5. Rowling, J. K. (1999). *Harry Potter and the Prisoner of Azkaban* (Vol. 3). Scholastic Inc.

Genre: Film reviews

1. Bradshaw, P. (2025, January 30). *Before Sunrise review – Richard Linklater’s brief encounter defies romantic convention*. The Guardian; The Guardian. https://www.theguardian.com/film/2025/jan/30/before-sunrise-review-richard-linklater-ethan-hawke-julie-delpy
2. Ide, W. (2023, November 12). Anatomy of a Fall review – electric Palme d’Or-winning courtroom thriller. *The Guardian*. https://www.theguardian.com/film/2023/nov/12/anatomy-of-a-fall-review-justine-triet-cannes-palme-dor-winner-sandra-huller
3. Kermode, M. (2020, February 10). Parasite review – a gasp-inducing masterpiece. *The Guardian*. https://www.theguardian.com/film/2020/feb/09/parasite-review-bong-joon-ho-tragicomic-masterpiece
4. *Her – review | Mark Kermode*. (2014, February 16). The Guardian. https://www.theguardian.com/film/2014/feb/16/her-spike-jonze-joaquin-phoenix-review
5. Bradshaw, P. (2019, October 3). *Joker review – the most disappointing film of the year*. The Guardian; The Guardian. https://www.theguardian.com/film/2019/oct/03/joker-review-joaquin-phoenix-todd-phillips

Genre: Tech articles

1. Peters, J. (2025, June 11). *Apple’s updated parental controls will require kids to get permission to text new numbers*. The Verge. https://www.theverge.com/news/685582/apple-parental-controls-child-safety-features-permission-text
2. Heath, A., & Field, H. (2025, June 13). *Meta is paying $14 billion to catch up in the AI race*. The Verge. https://www.theverge.com/meta/685711/meta-scale-ai-ceo-alexandr-wang
3. Peters, J. (2025, June 12). *Apple’s upgraded Siri might not arrive until next spring*. The Verge. https://www.theverge.com/news/686498/apple-upgraded-siri-ios-26-4
4. Lawler, R. (2025, June 12). *A massive Google Cloud outage messed up Google Home, Spotify, and other services*. The Verge. https://www.theverge.com/news/686365/cloudflare-spotify-google-home-is-down-outage-offline

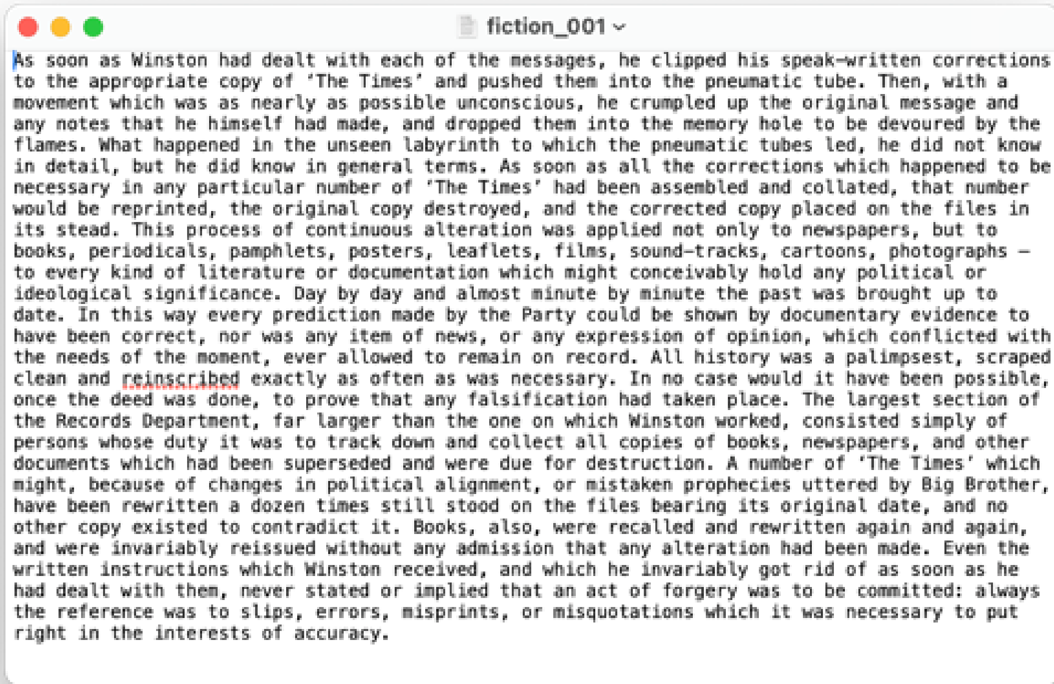
5. Roth, E. (2025, June 12). *Here's the $2,000 fully AI-generated ad that aired during the NBA Finals*. The Verge. https://www.theverge.com/news/686474/kalshi-ai-generated-ad-nba-finals-google-veo-3

Task 2. Creating corpus by converting into text format

```
cname = file.path(".","CorpusTexts")
dir(cname)

##  [1] "fiction_001.txt" "fiction_002.txt" "fiction_003.txt"
"fiction_004.txt"
##  [5] "fiction_005.txt" "news_001.txt"    "news_002.txt"    "news_003.txt"
##  [9] "news_004.txt"    "news_005.txt"    "review_001.txt"
"review_002.txt"
## [13] "review_003.txt"  "review_004.txt"  "review_005.txt"  "tech_001.txt"
## [17] "tech_002.txt"    "tech_003.txt"    "tech_004.txt"    "tech_005.txt"

docs <- Corpus(DirSource(cname))
summary(docs)

##                 Length Class             Mode
## fiction_001.txt 2      PlainTextDocument list
## fiction_002.txt 2      PlainTextDocument list
## fiction_003.txt 2      PlainTextDocument list
## fiction_004.txt 2      PlainTextDocument list
## fiction_005.txt 2      PlainTextDocument list
## news_001.txt    2      PlainTextDocument list
## news_002.txt    2      PlainTextDocument list
## news_003.txt    2      PlainTextDocument list
## news_004.txt    2      PlainTextDocument list
## news_005.txt    2      PlainTextDocument list
## review_001.txt  2      PlainTextDocument list
## review_002.txt  2      PlainTextDocument list
## review_003.txt  2      PlainTextDocument list
## review_004.txt  2      PlainTextDocument list
## review_005.txt  2      PlainTextDocument list
## tech_001.txt    2      PlainTextDocument list
## tech_002.txt    2      PlainTextDocument list
## tech_003.txt    2      PlainTextDocument list
## tech_004.txt    2      PlainTextDocument list
## tech_005.txt    2      PlainTextDocument list
```

I organized the 20 .txt files into a subfolder called "CorpusTexts" in my working directory. Each file was labeled according to its genre and numbering (e.g., news_001.txt, fiction_002.txt, etc.) Following the approach covered in lecture, tm package is used to create a text corpus. First, I defined the path to the folder using file.path() and then passed it to DirSource() and Corpus()

functions to generate a structured corpus object, so now the documents can be read into memory as PlainTextDocument. This corpus structure will be the foundation of following text processings inclusing tokenization, stopword removal, and document-term matrix creation.


fiction_001

As soon as Winston had dealt with each of the messages, he clipped his speak-written corrections to the appropriate copy of 'The Times' and pushed them into the pneumatic tube. Then, with a movement which was as nearly as possible unconscious, he crumpled up the original message and any notes that he himself had made, and dropped them into the memory hole to be devoured by the flames. What happened in the unseen labyrinth to which the pneumatic tubes led, he did not know in detail, but he did know in general terms. As soon as all the corrections which happened to be necessary in any particular number of 'The Times' had been assembled and collated, that number would be reprinted, the original copy destroyed, and the corrected copy placed on the files in its stead. This process of continuous alteration was applied not only to newspapers, but to books, periodicals, pamphlets, posters, leaflets, films, sound-tracks, cartoons, photographs – to every kind of literature or documentation which might conceivably hold any political or ideological significance. Day by day and almost minute by minute the past was brought up to date. In this way every prediction made by the Party could be shown by documentary evidence to have been correct, nor was any item of news, or any expression of opinion, which conflicted with the needs of the moment, ever allowed to remain on record. All history was a palimpsest, scraped clean and reinscribed exactly as often as was necessary. In no case would it have been possible, once the deed was done, to prove that any falsification had taken place. The largest section of the Records Department, far larger than the one on which Winston worked, consisted simply of persons whose duty it was to track down and collect all copies of books, newspapers, and other documents which had been superseded and were due for destruction. A number of 'The Times' which might, because of changes in political alignment, or mistaken prophecies uttered by Big Brother, have been rewritten a dozen times still stood on the files bearing its original date, and no other copy existed to contradict it. Books, also, were recalled and rewritten again and again, and were invariably reissued without any admission that any alteration had been made. Even the written instructions which Winston received, and which he invariably got rid of as soon as he had dealt with them, never stated or implied that an act of forgery was to be committed: always the reference was to slips, errors, misprints, or misquotations which it was necessary to put right in the interests of accuracy.

Task 3. Creating Document-Term Matrix

```
#preprocessing
intospace <- content_transformer(function(x,pattern) gsub(pattern," ",x))
docs <- tm_map(docs, content_transformer(tolower))

docs <- tm_map(docs, intospace, "-")
docs <- tm_map(docs, intospace, "\"")

docs <- tm_map(docs, removePunctuation)
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, removeWords, stopwords("english"))
docs <- tm_map(docs, stripWhitespace)
docs <- tm_map(docs, stemDocument, language = "english")

#create DTM
dtm <- DocumentTermMatrix(docs)
dim(dtm)

## [1]   20 1701
```

```r
#remove sparse terms
new_dtm <- removeSparseTerms(dtm, 0.75)
dim(new_dtm)
```

```
## [1] 20 22
```

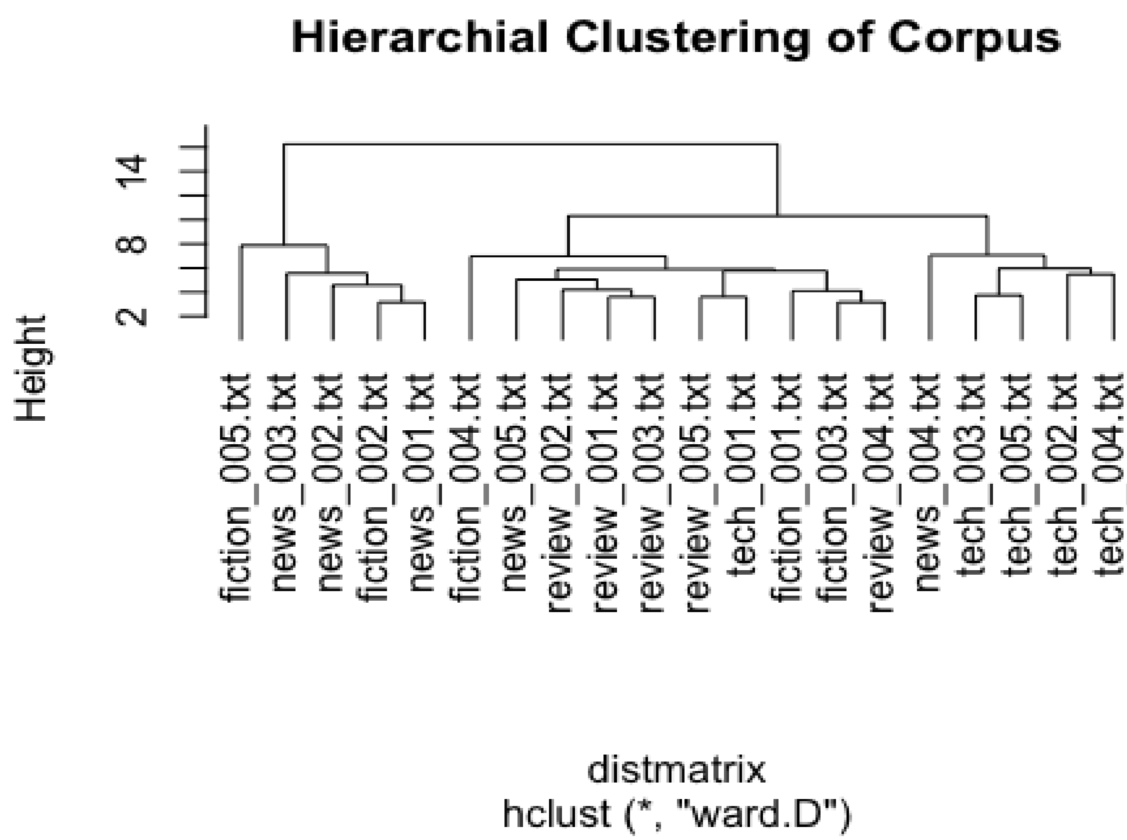
To prepare the text for analysis, I applied some standard text cleaning steps using tm package. These include: - Texts are converted to lowercase using “content_transformer(tolower)”. - Punctuation is removed using “removePunctuation”. - Numbers are removed using “removeNumbers”. - Stopwords like “the”, “a”, “is” are omitted using “stopwords(english)”. - I removed unwanted symbols like dash or quotation mark into space using customized text transformation. - Stemming is applied, and extra whitespace is removed as well. Then, I constructed a Document-Term Matrix (DTM). I used removeSparseTerms() to reduce dimensionality. We can observe that the dimension was originally (20,1705) which means 1705 unique terms. After fine tuning, I set the sparsity threshold of 0.75 so that the filtered matrix will contain only terms that appear in at least 25% of the documents. The new matrix became (19,22), containing 22 tokens.

```r
dtm_matrix <- as.matrix(new_dtm)
write.csv(dtm_matrix,"DTM.csv")
```

Task 4. Hierarchial Clustering

Cosine distance was computed between each pair of documents using the DTM, then ward’s method clustering was used to minimize within-cluster variance. The clustering dendogram illustrates the relationship between 20 text files.

```r
#cosine distance between documents
dtms <- as.matrix(new_dtm)
distmatrix <- proxy::dist(dtms, mehtod = "cosine")
#use ward's method
fit <- hclust(distmatrix, method = "ward.D")
#dendogram
plot(fit, hang = -1, main = "Hierarchial Clustering of Corpus")
```



The tree shows how the documents were grouped based on text similarity. We can see how the clusters did not exactly group documents by actual genre in an interpretable pattern.

```
#assign documents to clusters
cutfit <- cutree(fit, k=4)

topics <- c(
  rep("news",5),
  rep("fiction",5),
  rep("review",5),
  rep("tech",5)
)
#confusion matrix
cluster_table <- table(TrueLabel = topics, Cluster = cutfit)
cluster_table

##           Cluster
## TrueLabel 1 2 3 4
##   fiction 1 3 0 1
##   news    3 1 1 0
##   review  5 0 0 0
##   tech    1 0 0 4
```

```r
#rearrange
cluster_table <- cluster_table[,c(1,2,4,3)]
cluster_table
```

```
##          Cluster
## TrueLabel 1 2 4 3
##   fiction 1 3 1 0
##   news    3 1 0 1
##   review  5 0 0 0
##   tech    1 0 4 0
```

```r
#calculate accuracy
TA_matrix <- as.matrix(cluster_table)
accuracy <- sum(diag(TA_matrix))/sum(TA_matrix)
print(accuracy)
```

```
## [1] 0.1
```

I assigned each document to one of four clusters using cutree fit with k=4, and compared the labels with true genre labels of them. The accuracy was 0.10 which shows only 10% of data aligns with the actual cluster. The "review" genre texts seem to spread across multiple branches in the dendrogram, where as "tech" genre were formed in a tighter sub-cluster on the right side. This suggests that technical vocabularies in tech articles offer clearer separation.

Task 5. Sentiment Analysis

```r
file_paths <- list.files(path = "CorpusTexts", full.names=TRUE)
texts <- sapply(file_paths, function(x) {
  paste(readLines(x, warn=FALSE),collapse = " ")
})
#sentiment scores
sentiment_scores <- get_sentiment(texts, method = "syuzhet")
genres <- c(
  rep("news",5),
  rep("fiction",5),
  rep("review",5),
  rep("tech",5)
)
sentiment_df <- data.frame(
  genre = genres,
  sentiment = sentiment_scores
)
#Analysis
aggregate(sentiment ~ genre, data = sentiment_df, mean) #find mean
```

```
##     genre sentiment
## 1 fiction     -1.28
## 2    news     -1.08
## 3  review      2.80
## 4    tech      3.82
```

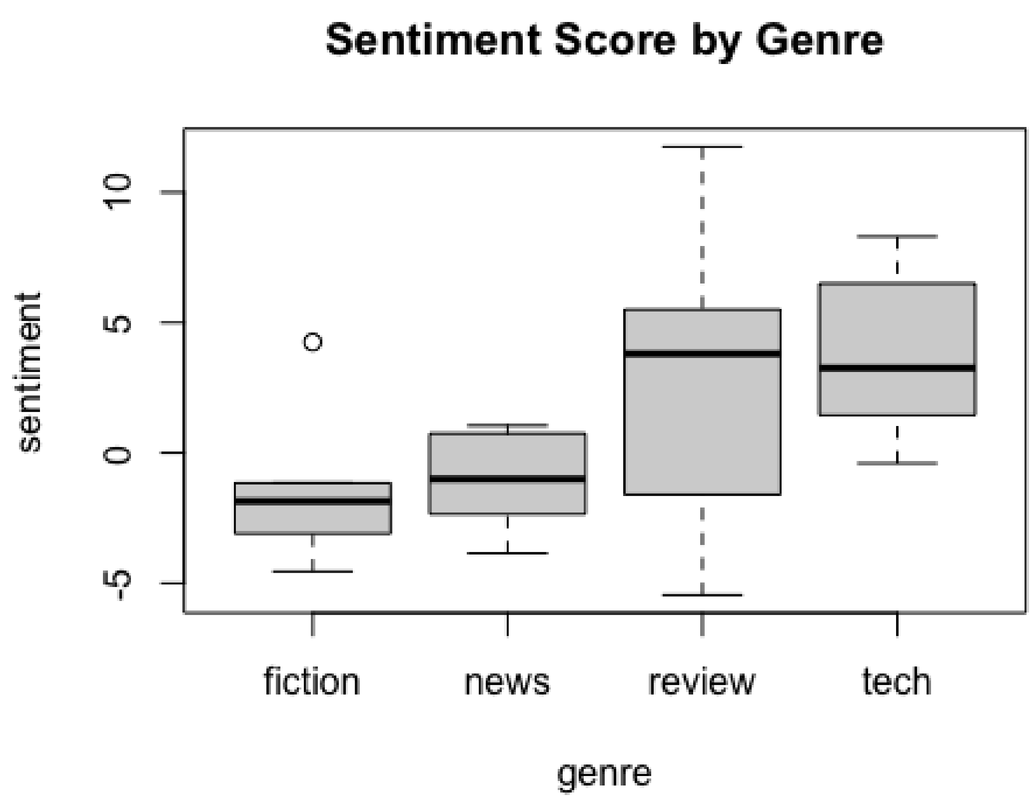
```r
aggregate(sentiment ~ genre, data = sentiment_df, sd) #find sd
```

```
##     genre sentiment
## 1 fiction  3.351418
## 2    news  2.072318
## 3  review  6.629762
## 4    tech  3.570819
```

To evaluate emotional tone of each document, sentiment analysis using syuzhet package is performed. get_sentiment() function is applied to each texts in the corpus file, computing a score per document. It gives a score ranging from negative to positive values. Then, I computed the mean and standard deviation value of sentiment scores for each type of genre.

Fiction (-1.28) and news (-1.08) both have negative mean sentiment, suggesting that these contain more emotionally negative language. The mean score for review genre is +2.8 and tech is +3.82, indicating more optimistic language used in the passages.

```r
#boxplot
boxplot(sentiment~genre, data = sentiment_df,
        main = "Sentiment Score by Genre")
```



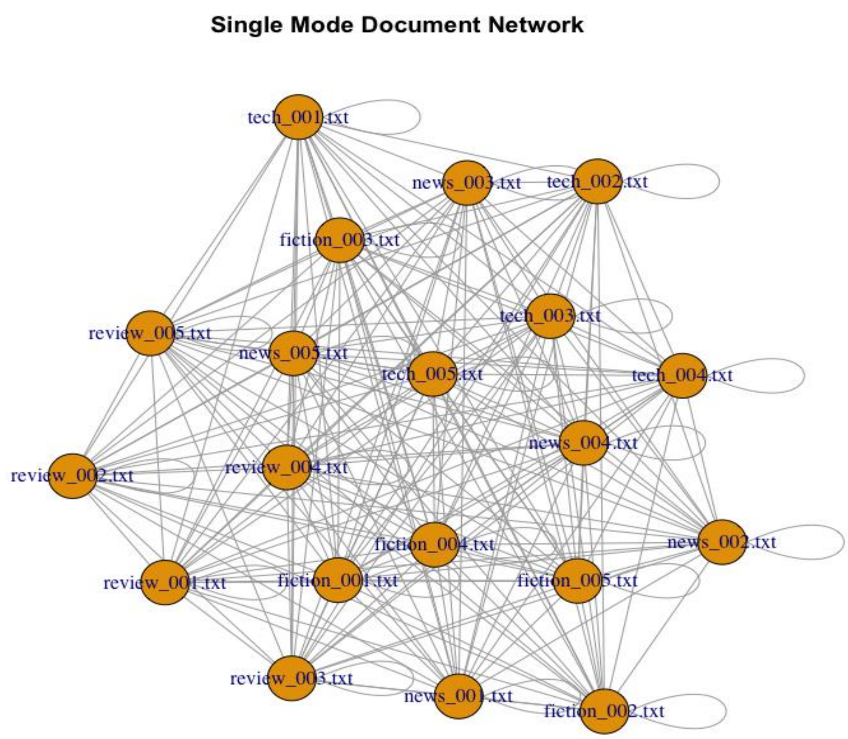
I also created a boxplot to demonstrate the sentiment scores. We can see that reviews have a very high variability. This is possibly because some reviews have harsh views towards a film while some has positive ones.

Task 6. Single-mode network between documents

```
dtm_matrix <- as.matrix(dtm)
dtm_binary <- (dtm_matrix > 0) + 0
adjacency_matrix <- dtm_binary %*% t(dtm_binary)
set.seed(35865377)
doc_network <- graph_from_adjacency_matrix(adjacency_matrix, mode =
"undirected", weighted = TRUE)
#setting edge width, color, node size
edge_weights <- E(doc_network)$weight
add_color <- colorRampPalette(c("pink","yellow","green"))
edge_colors <- add_color(length(edge_weights))[as.numeric(cut(edge_weights,
breaks = length(edge_weights)))]
node_size <- betweenness(doc_network)
```

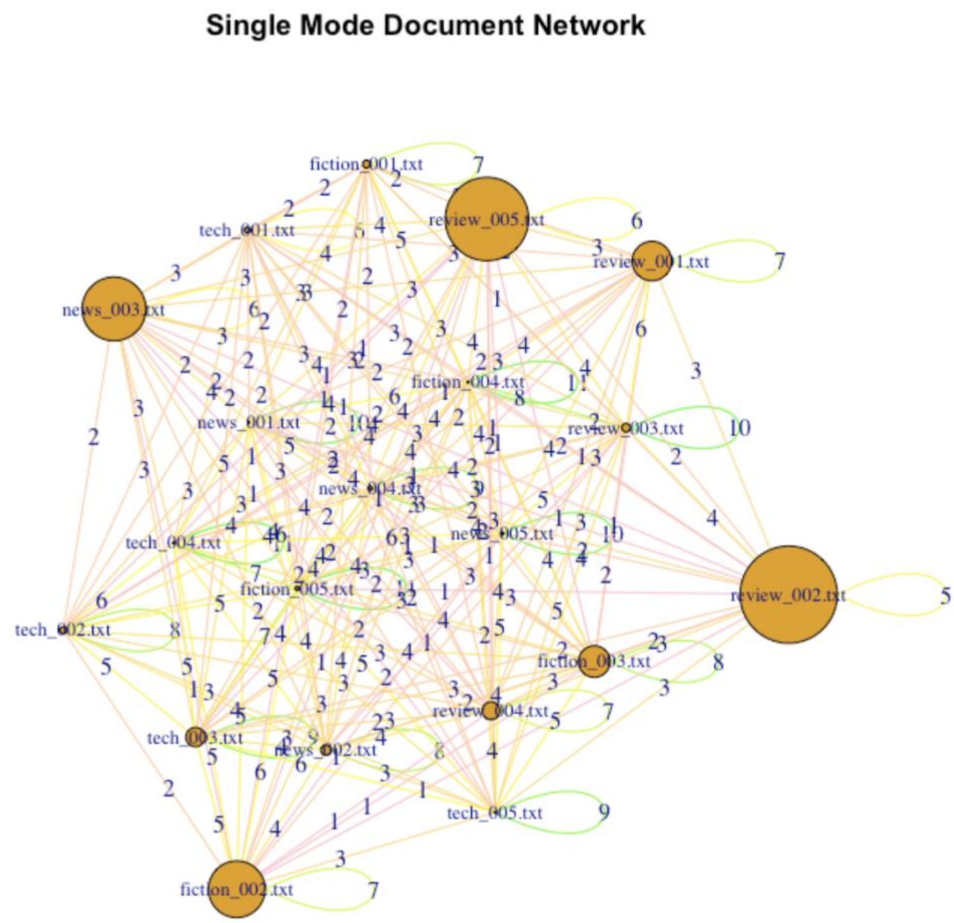
In this task, I created a single-mode network using the reduced DTM, and improved the model over progress. First, I converted the distance matrix to binary matrix (1=word present, 0=otherwise). Then, I multiplied the distance matrix to binary matrix, making leading diagonal zero.

```
plot(doc_network, main = "Single Mode Document Network")
```



To improve the single basic mode network, I added weight between documents, and have color codings based on the strength of weight. Low values of shared term count use pink, medium values use yellow, and high values use green.

```r
plot(doc_network,
     edge.label=edge_weights,
     edge.color = edge_colors,
     edge_width = edge_weights / 2,
     vertex.size = node_size,
     vertex.label.cex = 0.8,
     main = "Single Mode Document Network")
```

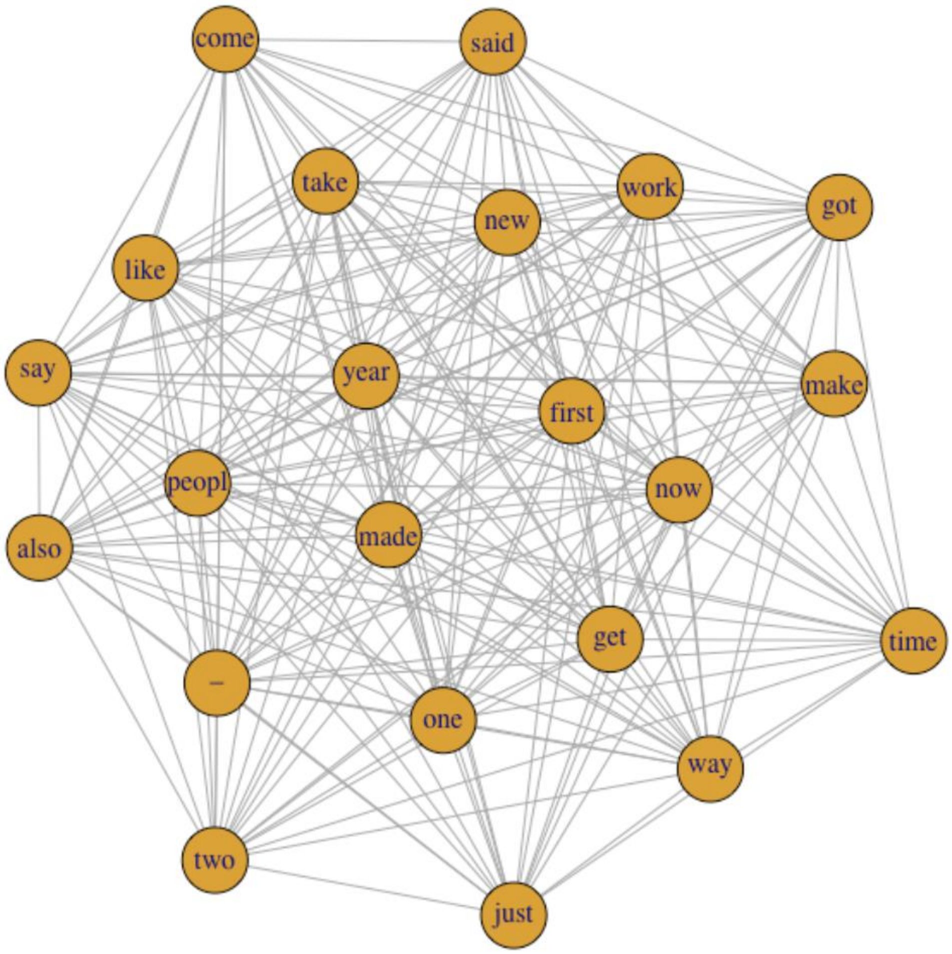


This is the improved single-mode network. Each node represents a document, and each edge indicates the number of shared terms between them. The edge weights are the strength of the connections and is color-coded as well. The graph looks densely connected, indicating many documents share vocabulary, although the connections are not equal. We can observe a distinct cluster on the left-bottom side of the network. There are many tech articles included in this cluster. The largest nodes in the network have the highest betweenness centrality. Documents like review_002.txt and review_005.txt may contain words that overlaps across multiple genres. Documents on the periphery (e.g. tech_002, fiction_002) may have fewer or weaker connections than others. This could be because of its use in more unique words or narrower topic.

Task 7. Single-mode network between tokens

```r
term_matrix <- t(dtm_binary) %*% dtm_binary
diag(term_matrix) <- 0
```

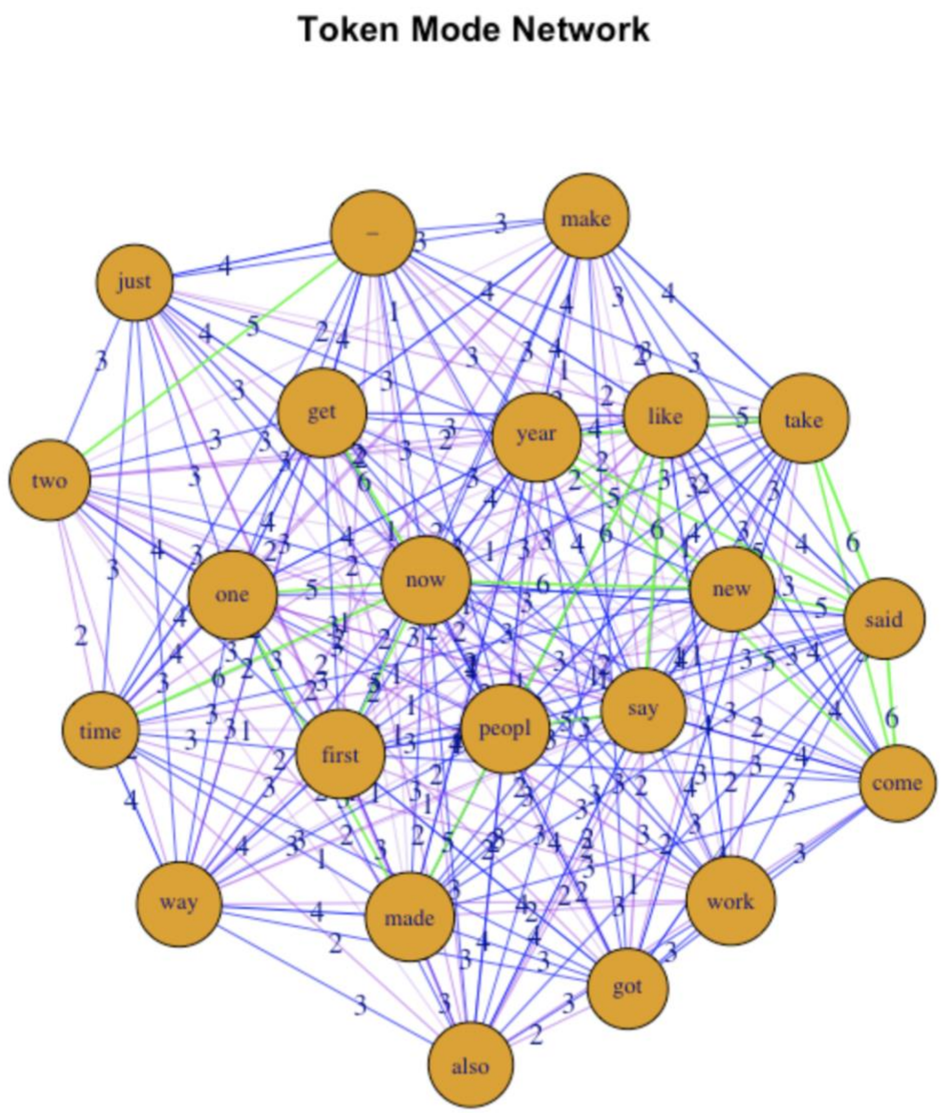
```
word_network <- graph_from_adjacency_matrix(term_matrix, mode = "undirected",
weighted = TRUE)

plot(word_network)
```



A similar process is applied, but this time we look at the relationship between terms. I removed any self loops using diag(term_matrix). To improve the simple-node network, again I added some customizations.

```
#setting edge width, color, node size
edge_weights <- E(word_network)$weight
edge_width <- edge_weights/2
edge_color <- ifelse(edge_weights > 4, "green",
                     ifelse(edge_weights > 2, "blue", "purple"))
node_size <- degree(word_network) * 2
plot(word_network,
     edge.label = edge_weights,
     vertex.label.cex = 0.8,
     vertex.size = node_size / 2,
```

```
    edge.width = edge_width / 2,
    edge.color = edge_color,
    main = "Token Mode Network")
```



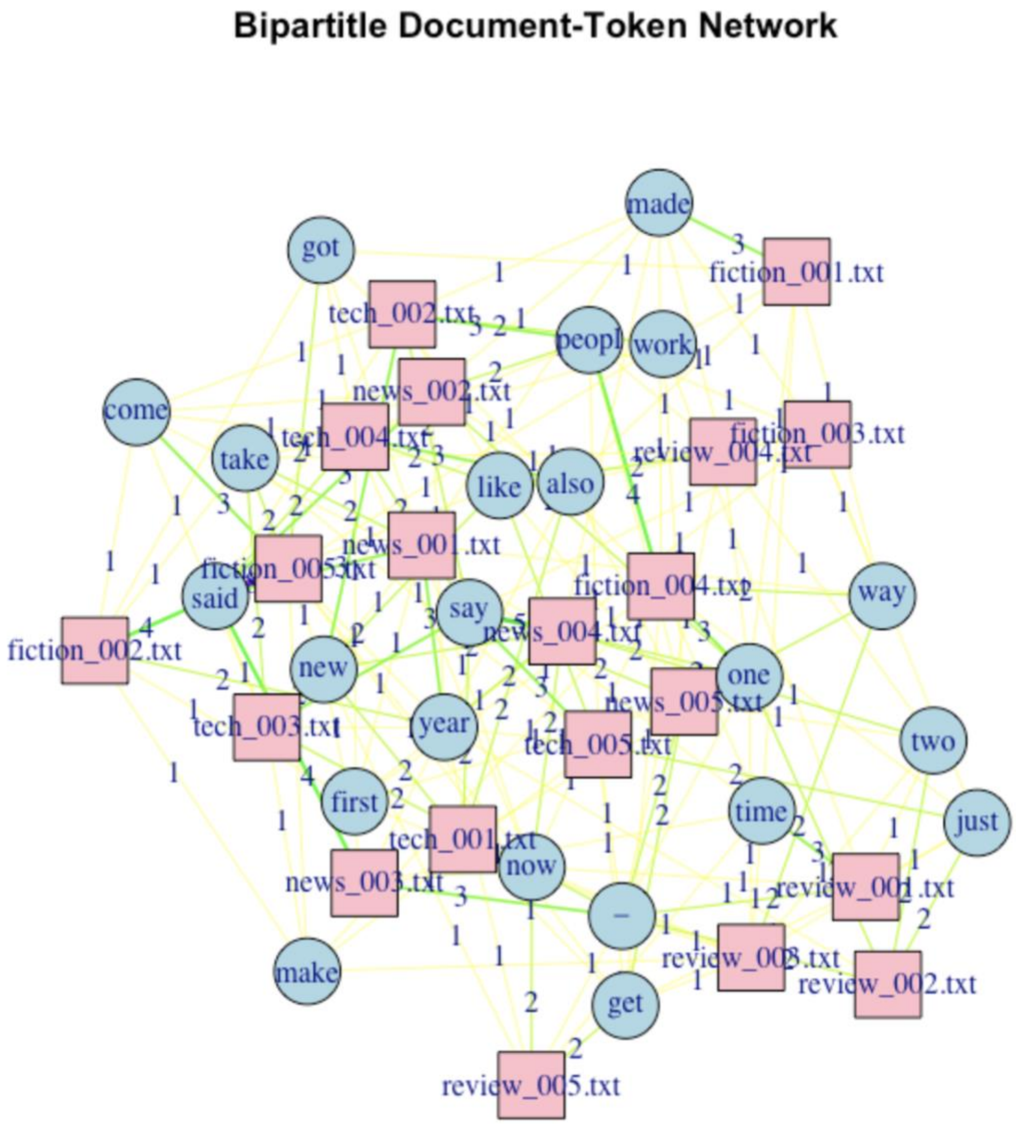
From this graph, we can observe the word co-occurence relationships and try to gain some meaningful linguistic insights. There are words like “now”, “people”, “get”, “say”, “come”, which are not very topic-specific but are conversational languages that commonly appear throughout all genres. Node sizes are similar among the tokens. Green edges indicate very frequent co-occurrence. For example, there is a strong connection between “come”, “said”, “take”.

Task 8. Bipartite Network

I transformed the DTM into a long-format edge list and built an undirected bipartite graph.

```
dtm_df <- as.data.frame(as.matrix(new_dtm)) #clone dtms
dtm_df$doc <- rownames(dtm_df) #add row names
```

```r
#convert to long format
dtm_long <- data.frame()

for (i in 1:nrow(dtm_df)) {
  for (j in 1:(ncol(dtm_df) - 1)) {
    weight <- dtm_df[i, j]
    if (weight > 0) {
      dtm_long <- rbind(dtm_long, data.frame(
        doc = dtm_df[i, "doc"],
        token = colnames(dtm_df)[j],
        weight = weight
      ))
    }
  }
}

graph <- graph_from_data_frame(dtm_long, directed = FALSE)

V(graph)$type <- bipartite_mapping(graph)$type
V(graph)$color <- ifelse(V(graph)$type, "lightblue", "pink")  # documents vs
tokens
V(graph)$shape <- ifelse(V(graph)$type, "circle", "square")
edge_weights <- E(graph)$weight
edge_colors <-
colorRampPalette(c("yellow","green","purple"))(length(edge_weights))[as.numer
ic(cut(edge_weights, breaks = length(edge_weights)))]

plot(graph,
     edge.label = edge_weights,
     edge.width = edge_weights / 2,
     edge.color = edge_colors,
     main = "Bipartitle Document-Token Network"
     )
```



As shown in the graph, the pink boxes represent the documents and light blue circles represent the words. The edges indicate whether a word appears in the document, and the edge weight reflect its frequency. We can observe that tech_002 and fiction_004 has strong connection wit the word "people". fiction_002 and texh_003 has strong connection with the word "said". However, there are no purple edges which should indicate strong connections. Although nodes seem to gather around the left-top part, it is still difficult to spot a clear cluster in this graph.

##Conclusion Across multiple analyses, I was able to find patterns in word and document structure that shows genre-related connections. review_002, review_005, news_003, fiction_002 stood out in the document network as central hubs, showing that they contain words that connect them to multiple genres. From the token network, words like "now", "say", "people" and "make" were among the most important ones. They frequently occur across different documents and genres. The accuracy was quite low (10%) for document clustering in task 4. This means that just purely unsupervised clustering on a small corpus may not separate genres. Thus, hierarchial clustering was simple to implement and good for initial grouping, but didn't perform well to identify genres. Network analysis provides richer insights by showing exact weights of connections by node size and edge colors. To improve this project, we should utilize Term Frequency-Inverse Document Frequency, as it gives greater weights to words that are important but rare across the corpus. In this project, the terms that were chosen were way too common, so this way, it could reduce the influence of common terms.