FIT3152 Assignment 3 Report
Student name : Ng Chen Ting (31861148)

Question 2 – Movie Corpus

All the 15 documents used in this report are copied from online blogs and newspapers. The documents are show reviews where each of them fall in one of the 3 categories which are Romance, Horror and Law. All of the copied text are pasted into an empty text file labelled with the movie name for identification purpose in networks.

To create the corpus, all the txt files are stored in a folder which is accessed by defining the path to “NgChenTing31861148”. Corpus() function then use the path retrieved to create a corpus of all files in the folder by treating each txt file as in individual document.

Question 3 – Creating Document-Term Matrix

Pre-processing of documents are done to get the documents cleaned before preparing it’s Document-Term Matrix. This includes text transformations, tokenisation, word filtering and stemming.

Text transformations are made using the content_transformer() function which in this case turns any word specified into a space which is equivalent to a removal. Since the texts are copied from online, a few different punctuations have to be taken into consideration, like the difference in apostrophe used. These punctuation are undetected by tm_map(docs, removePunctuation) and therefore has to be explicitly transformed. Their differences can be seen below :

```
toSpace <- content_transformer(function(x, pattern) gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "”")
```

In addition, the dash symbol "–" is also removed to avoid it being interpreted as a token itself. After retrying removespareterms() multiple times, some tokens were unrelated which had to be explicitly removed in the same section to avoid it being used as a token again. The words include “one”, “two”, “also”, “get”, “just” and “see” which are then replaced by space instead.

Following up, Tokenization is used to remove numbers and punctuations in the documents. A content transformer is used to change all the characters to lowercase to avoid capital letters of the same word to be tokenized separately. Filtering is also done to filter out stop words in English such as “a”, “is”, “for” and many more. StripWhitespace is used to remove unnecessary whitespaces within the documents.

Lastly, the document is stemmed to reduce words to their root form by consolidating variations of the same word. This can help reduce the amount of tokens where words are just variations of each other.

The Document-Term matrix is then created but since this report only needs around 20 tokens. The number of tokens are shortened from 1707 tokens to only 19 tokens by removing sparse terms. This is done by using 0.45 in the removeSpareTerms() function that remove terms that have a document frequency under 0.45 or in other words, they appear in less than 45% of the document.

Question 4 – Cosine Distance

Cosine distance can be used to show the similarity between documents. Firstly, its Inverse Document Frequency is calculated which taken into the account the relative number of documents in which a word occurs.

Then the cosine distance is calculated with the following formula of

cosine_dist = 1 - crossprod(tf_idf) / (sqrt(colSums(tf_idf^2) %*% t(colSums(tf_idf^2))))

It measures the cosine of the angle between the two document vectors where the smaller the cosine distance value, the more similar the documents are.

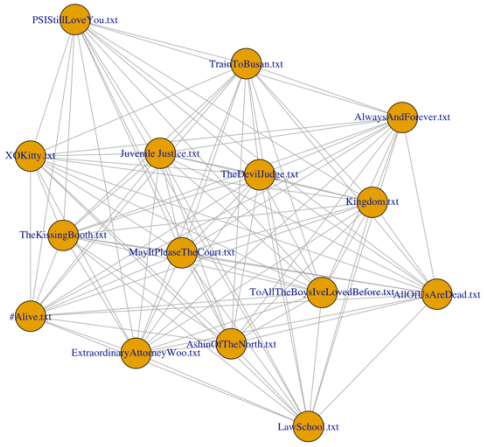
The dendrogram shows the hierarchical cluster of the documents and 3 clusters can be boxed using rect.hclust() and specifying k as 3. All the romance category are correctly identified which is the cluster on the far left. For Law cluster there seems to be one document mistakenly classified as Horror which is LawSchool.txt. Aside from that, all the clustering seems to be accurate for cosine distance clustering.

From a quantitative point of view, as mentioned previously only one of the law documents are wrongly classified. If we consider cluster 1 to be Horror category, cluster 2 to be Romance category and cluster 3 to be Law category, the confusion matrix below has an accuracy of 93.33% with only one wrong document out of 15 documents.

	Clusters		
GroupNames	1	2	3
Horror	5	0	0
Law	1	0	4
Romance	0	5	0

Question 5 – Single-mode Network for Documents

To create a single-mode network to show the connections between documents. The document term matrix is first converted to a binary matrix. This gives us a Abstracts network data where each number shows the number of tokens that exist between each pair of documents.



The highest number in the entire Abstracts Network Data is the pair between MayItPleaseTheCourt.txt and ExtraordinaryAttorneyWoo.txt with a number of 15 sharing tokens. This makes sense because both of them are series about law which makes their document writing style similar causing them to have many similar tokens in return.

Using the Abstracts Network Data, a Abstracts Network Plot can be generated where each edge represents the number of tokens shared among them.

By summing up all the columns, we can see for each document the number of tokens it shared with all of the other documents. MayItPleaseTheCourt has the most tokens shared with all of the documents in total.

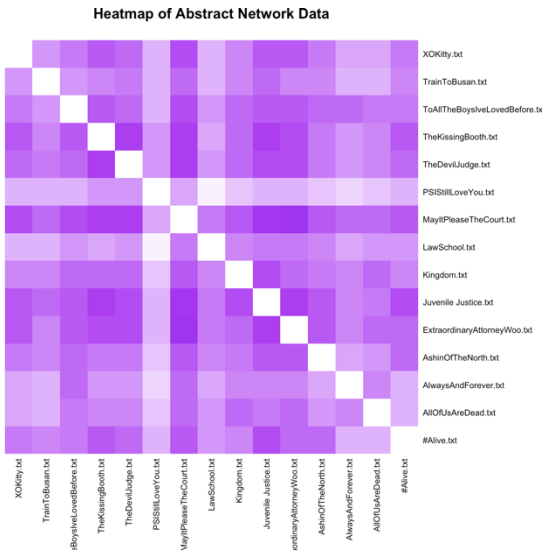
```
> colSums(ByAbsMatrix)
                    #Alive.txt                  AllOfUsAreDead.txt
                           120                                 106
              AlwaysAndForever.txt                AshinOfTheNorth.txt
                            97                                 122
     ExtraordinaryAttorneyWoo.txt               Juvenile Justice.txt
                           145                                 150
                   Kingdom.txt                       LawSchool.txt
                           126                                  94
           MayItPleaseTheCourt.txt                 PSIStillLoveYou.txt
                           157                                  66
                TheDevilJudge.txt                TheKissingBooth.txt
                           137                                 139
ToAllTheBoysIveLovedBefore.txt                     TrainToBusan.txt
                           131                                 103
                     XOKitty.txt
                           119
> # Important nodes
> max(colSums(ByAbsMatrix))
[1] 157
```

By measuring closeness, we can show how well connected is each node in the graph, here PSIStillLoveYou.txt has the highest closeness value meaning it has the highest ability to access and interact with other documents.

```
> closeness
                    #Alive.txt                  AllOfUsAreDead.txt
                      "0.0086"                           "0.0100"
              AlwaysAndForever.txt                AshinOfTheNorth.txt
                      "0.0111"                           "0.0090"
     ExtraordinaryAttorneyWoo.txt               Juvenile Justice.txt
                      "0.0078"                           "0.0076"
                   Kingdom.txt                       LawSchool.txt
                      "0.0088"                           "0.0135"
           MayItPleaseTheCourt.txt                 PSIStillLoveYou.txt
                      "0.0070"                           "0.0152"
                TheDevilJudge.txt                TheKissingBooth.txt
                      "0.0073"                           "0.0072"
ToAllTheBoysIveLovedBefore.txt                     TrainToBusan.txt
                      "0.0081"                           "0.0097"
                     XOKitty.txt
                      "0.0086"
```

It can also be modelled as heatmap instead of looking at the Abstract Network Data for a better visualisation. The figure shows a Heatmap where the shades of purple shows the amount of tokens being shared among each pair of documents. The darker the shade of purple, the more tokens are being shared.

A few observations can be made such as for row MayItPleaseTheCourt.txt, the darkest shade across the row is column JuvenileJustice.txt and ExtraordinaryAttorneyWoo.txt. The three of them fall under the same category of being a Law related series.

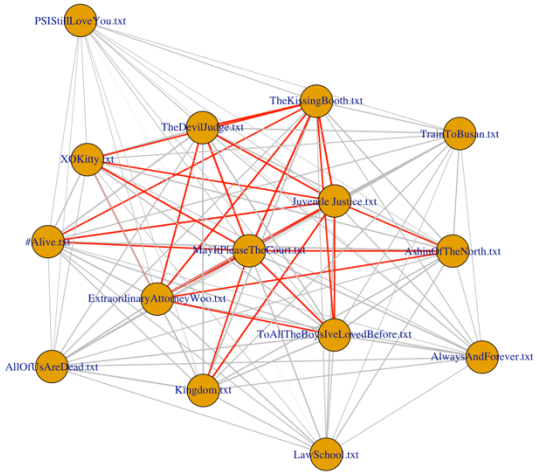


It can also be observed that the rows and columns for PSIStillLoveYou.txt has the lightest colour overall, this could mean that for the set of tokens, this document has the weakest relation to the rest of the document in terms of sharing the same tokens.

Furthermore, the same can also be observed from the Network graph where all the edges coming out form PSIStillLoveYou.txt are not coloured in red.

To get a clearer picture of which edges share most of the tokens, edges where documents are sharing more than 6 tokens are highlighted in red. While the rest are grey. The edges also have width according to their tokens.

The cut off point for an edge being red is 2 but due to rescaling by dividing all weights by 5 to make them look better in the graph, the cut off point for an edge to be red is when the pair shares a token of more than 6.



From the figure, PSIStillLoveYou.txt, TrainToBusan.txt, AlwaysAndForever.txt, AllOfUsAreDead.txt and LawSchool.txt seemed to have a low number of sharing tokens with all of the documents which are less than 6.

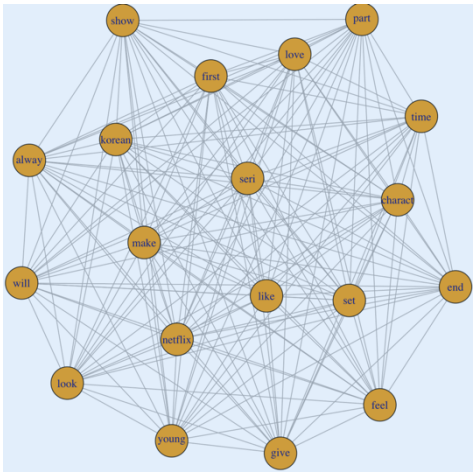
On the contrary, JuvenileJustice.txt seems to have the most edges highlighted in red, more specifically 10 edges. This means JuvenileJustice.txt shared more than 6 tokens with 10 other documents making it the most important central document. The Law Category seems to be the largest community being in the centre with many connections to other nodes.

In summary, aside from law, no other groups can be clearly formed. The relationship between documents are quite decent considering the average number of tokens shared between each pair is 8.6 out of 15.

```
> mean(E(ByAbs)$weight)
[1] 8.628571
```

Law category has the most relation with other documents making it the central documents in the network, more specifically MayItPleaseTheCourt.txt having the most shared tokens with all the documents in total and JuvenileJustice.txt having the most documents related with high number of tokens in each pair.

Question 6 – Single-mode Network for tokens



A single-mode network showing the connections between tokens can be generated by first creating a Token Matrix in a similar way as above.

The figure shows all the relationship between the 19 tokens in terms of their co-occurrences within the documents.

By summing the columns in Token Matrix, “Netflix” has the highest co-occurrence with 149 co-occurrences within the documents.

```
> colSums(ByTokenMatrix)
  alway charact     end    feel    give  korean    like    make netflix    part     set
    117     126     120     119     115     135     148     145     149     114     131
   time   young   first    look    love    seri    show    will
    120     126     146     120     109     142     117     121
> max(colSums(ByTokenMatrix))
[1] 149
```

But if we look at them individually, the highest number would be 11 with the pair of “Netflix” and “Korean”. This means both of them have a strong relationship and frequently occur together in documents.

From statistics retrieved from the graph, “love” has the highest closeness and Netflix has an eigenvector of 1. With a high closeness, it means “love” has a strong connectivity and plays an important part in interactions with other tokens. Making it an important token in the network.

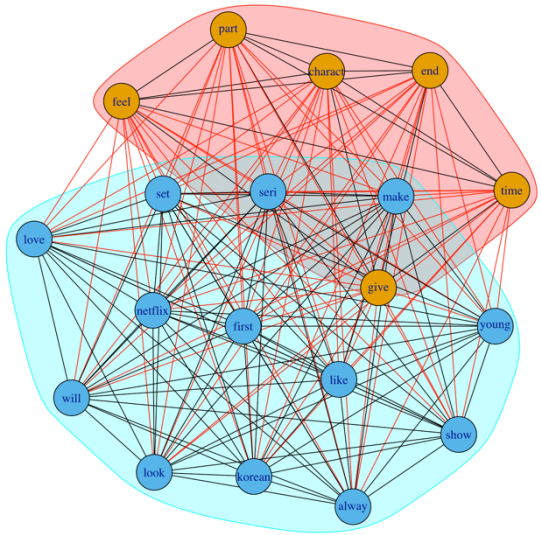
On the other hand, Eigenvector centrality not only consider the number of connections to a node but also the importance of connections, and in this case, “Netflix” has an eigenvector of 1 meaning it is the most central node making it the most influential in the network.

```
> stats[order(-stats$closeness),]
        degree betweenness   closeness eigenvector
love        18         0.0 0.009174312   0.7441054
part        18         0.5 0.008771930   0.7728789
give        18         0.0 0.008695652   0.7809838
alway       18         0.0 0.008547009   0.7954382
show        18         0.0 0.008547009   0.7973720
feel        18         0.0 0.008403361   0.8073123
end         18         0.0 0.008333333   0.8165950
time        18         0.0 0.008333333   0.8141028
look        18         0.0 0.008333333   0.8181623
will        18         0.0 0.008264463   0.8269168
charact     18         0.0 0.007936508   0.8497687
young       18         0.0 0.007936508   0.8555368
set         18         0.0 0.007633588   0.8880423
korean      18         0.0 0.007407407   0.9137099
seri        18         0.0 0.007042254   0.9557121
make        18         0.0 0.006896552   0.9722283
first       18         0.0 0.006849315   0.9788374
like        18         0.0 0.006756757   0.9924858
netflix     18         0.0 0.006711409   1.0000000
```

Both “love” and “Netflix” are important nodes in the network but they play different roles, “love” can easily engage in interactions while “Netflix” has a strong effect on network’s dynamic.

To cluster them into groups, greedy clustering is used. It’s a hierarchical clustering algorithm used to identify clusters of a network based on edge weight which in this case is number of co-occurrences.

cluster_fast_greedy() function from igraph package is used where it returns the network’s cluster for us to visualise in the figure.



Fast greedy is used here which starts with its own node and iteratively merge clusters to maximise modularity. It continues to merge pairs of tokens to lead to the largest increase in modularity greedily.

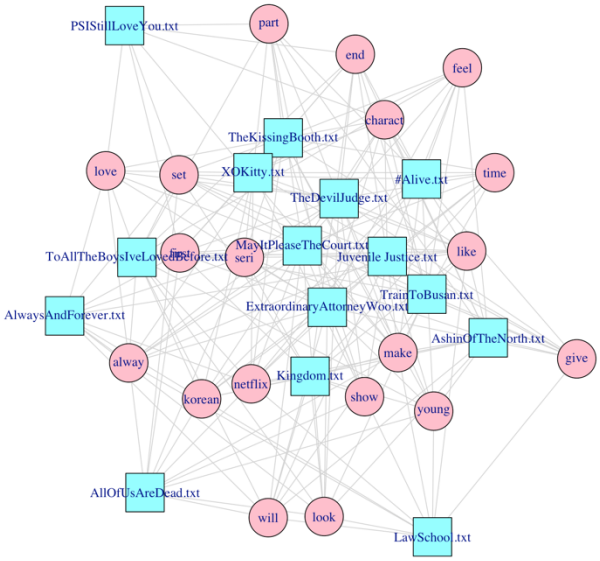
By using Greedy Clustering, two groups are formed in red and blue. The tokens in the blue clusters are mostly words using for documents related to “romance” which is an obvious group.

In summary, “love” and “Netflix” are the most important central nodes in the network, there are two clusters made from greedy clustering but they do not show much relation aside from categorising the romance category from the others.

Question 7 – Bipartite Network

A bipartite network of the corpus is created with document ID and token as nodes of their kinds. To create it, pre-processing has to be done which is to format as a set of edges between one node of each type and this includes converting the Document Term Matrix into a new dataframe that represents the relationships between documents, tokens and their weights. It then filters out rows with no weights and rearrange columns to have the order of abs, token, weight.

Interesting relationships can be observed from the bipartite graph.
For instance, all the 5 romance documents : PSIStillLoveYou.txt, AlwaysAndForever.txt, TheKissingBooth, ToAllTheBoysIveLovedBefore.txt and XOKitty are surrounding the token “love”.



As they are all documents about romance movie reviews, the term “love” is frequently mentioned which makes it such a clear indicator which separate romance documents with the other documents.

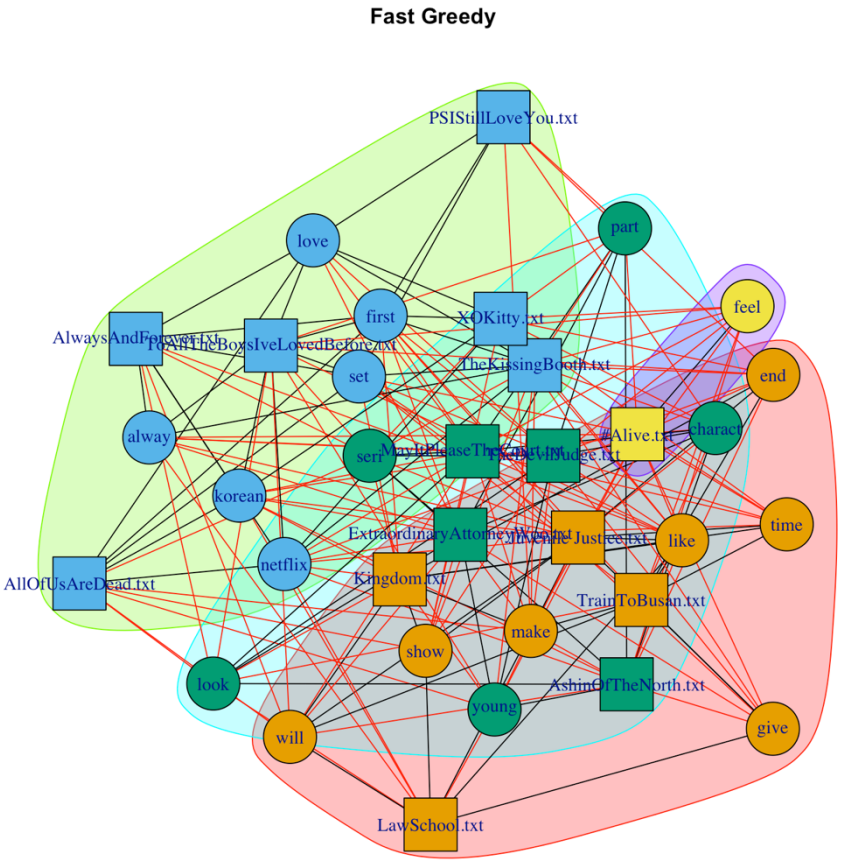
Aside from that, we can also easily identify by their tokens the groups of documents with similar tokens. From the token “Netflix”, we can easily tell that Kingdom.txt, AllOfUsAreDead.txt, LawSchool.txt and ExtraordinaryAttorneyWoo.txt are closely related to Netflix, this could be because it is their main streaming platform which is frequently mentioned during movie reviews.

```
> closeness(g)
                     #Alive.txt                     AllOfUsAreDead.txt
                     0.01515152                     0.01492537
                AlwaysAndForever.txt                AshinOfTheNorth.txt
                     0.01282051                     0.01538462
  ExtraordinaryAttorneyWoo.txt                Juvenile Justice.txt
                     0.01562500                     0.01408451
                    Kingdom.txt                    LawSchool.txt
                     0.01351351                     0.01369863
       MayItPleaseTheCourt.txt                PSIStillLoveYou.txt
                     0.01666667                     0.01204819
                TheDevilJudge.txt              TheKissingBooth.txt
                     0.01282051                     0.01587302
ToAllTheBoysIveLovedBefore.txt                 TrainToBusan.txt
                     0.01515152                     0.01176471
                    XOKitty.txt                          alway
                     0.01333333                     0.01408451
                        charact                            end
                     0.01250000                     0.01408451
                            feel                           give
                     0.01470588                     0.01515152
                          korean                           like
                     0.01492537                     0.01470588
                            make                        netflix
                     0.01408451                     0.01538462
                            part                            set
                     0.01408451                     0.01492537
                            time                          young
                     0.01315789                     0.01428571
                           first                           look
                     0.01408451                     0.01587302
                            love                           seri
                     0.01234568                     0.01408451
                            show                           will
                     0.01315789                     0.01538462
> max(closeness(g))
[1] 0.01666667
```

From the closeness of bipartite graph, we can see that MayItPleaseTheCourt.txt has the highest closeness out of all the nodes. Closeness in a bipartite graph is different from how it’s previously interpreted In the sense that here, the closeness centrality of a node quantifies how close it is to nodes in the opposite bipartite set.

This means MayItPleaseTheCourt.txt has a strong connection to a large portion of the nodes in the other bipartite set.

Using the same greedy clustering method as previously, it splits the nodes into four cluster but the purple cluster can be ignored since it's very likely to be an outlier. Three large clusters can be seen and we can actually categorise them according to their own categories.



The green cluster has the tokens "Love", "First" and "Always" which explains why documents that are about romance and clustered in the same group which are PSIStillLoveYou.txt, AlwaysAndForever.txt, ToAllTheBoysIveLovedBefore.txt, XOKitty.txt and TheKissingBooth.txt. This cluster correctly captures all the documents about romance show reviews with only one error being AllOfUsAreDead.txt which should belong to the Horror cluster.

Unlike the green cluster, the blue and red cluster seems to be overlapping, this shows that although there are two categories, they often have overlapping relations, to be more specific, the horror and law categories. But we can still tell that the blue cluster is meant to represent the category of law while the red represent the category for horror.

Overall the clustering is close to correctly grouping the 3 categories with one or two wrong documents in each cluster.

Comparison with Network Analysis and Clustering
For this set of documents and tokens, hierarchical clustering seems to do better at categorising them into the 3 categories but Network Analsyis gives more information about the relationship between documents and tokens. Especially for Bipartite where we could observe visually the clusters with documents and tokens in Romance.

References

Budowski, Jade. “Stream It or Skip It: “#Alive” on Netflix, a South Korean Zombie Thriller That Leans into the Anxieties of Isolation.” *Decider*, Decider, 11 Sept. 2020, decider.com/2020/09/11/alive-netflix-review/. Accessed 9 June 2023.

Budowski, Jade. “Stream It or Skip It: “Kingdom: Ashin of the North” on Netflix, a Thrilling Special Episode of the Korean Drama.” *Decider*, Decider, 26 July 2021, decider.com/2021/07/26/kingdom-ashin-of-the-north-netflix-review/. Accessed 9 June 2023.

Dumaraog, Ana. “Why to All the Boys Is so Much Better than the Kissing Booth.” *ScreenRant*, 27 Aug. 2021, screenrant.com/to-all-boys-vs-kissing-booth-netflix-movies-comparison/#:~:text=One%20of%20the%20main%20advantages,and%20easier%20to%20root%20for. Accessed 9 June 2023.

Hartford, Charles. “Extraordinary Attorney Woo Season 1 Review - but Why Tho?” *But Why Tho?*, 19 Aug. 2022, butwhytho.net/2022/08/extraordinary-attorney-woo-season-1-review-netflix/. Accessed 9 June 2023.

Horton, Adrian. “To All the Boys: Always and Forever Review – Enjoyable Enough Netflix Threequel.” *The Guardian*, The Guardian, 12 Feb. 2021, www.theguardian.com/film/2021/feb/11/to-all-the-boys-always-and-forever-review-enjoyable-enough-netflix-threequel. Accessed 9 June 2023.

Horton, Adrian. “To All the Boys: Always and Forever Review – Enjoyable Enough Netflix Threequel.” *The Guardian*, The Guardian, 12 Feb. 2021, www.theguardian.com/film/2021/feb/11/to-all-the-boys-always-and-forever-review-enjoyable-enough-netflix-threequel. Accessed 9 June 2023.

Keller, Joel. “Stream It or Skip It: “Kingdom” on Netflix, Where a Korean Kingdom Suffers from a Zombie Virus.” *Decider*, Decider, 25 Jan. 2019, decider.com/2019/01/25/kingdom-netflix-stream-it-or-skip-it/. Accessed 9 June 2023.

Keller, Joel. “Stream It or Skip It: “Law School” on Netflix, a K-Drama about a Group of Law Students Trying to Solve a Murder at Their School.” *Decider*, Decider, 14 Apr. 2021, decider.com/2021/04/14/law-school-netflix-review/. Accessed 9 June 2023.

Lee, Ann. “To All the Boys: PS I Still Love You Review – Entertaining Romcom Sequel.” *The Guardian*, The Guardian, 11 Feb. 2020, www.theguardian.com/film/2020/feb/11/to-all-the-boys-ps-i-still-love-you-review-romcom-sequel#:~:text=To%20All%20the%20Boys%3A%20PS%20I%20Still%20Love%20You%20doesn,likable%20central%20performance%20from%20Condor. Accessed 9 June 2023.

Loftus, Johnny. “Stream It or Skip It: “All of Us Are Dead” on Netflix, Where Zombies Overrun a Korean High School.” *Decider*, Decider, 28 Jan. 2022, decider.com/2022/01/28/all-of-us-are-dead-netflix-review/. Accessed 9 June 2023.

Sánchez, Kate. “May It Please the Court Review - but Why Tho?” *But Why Tho?*, 28 Oct. 2022, butwhytho.net/2022/10/review-may-it-please-the-court-is-an-ace-legal-drama/. Accessed 9 June 2023.

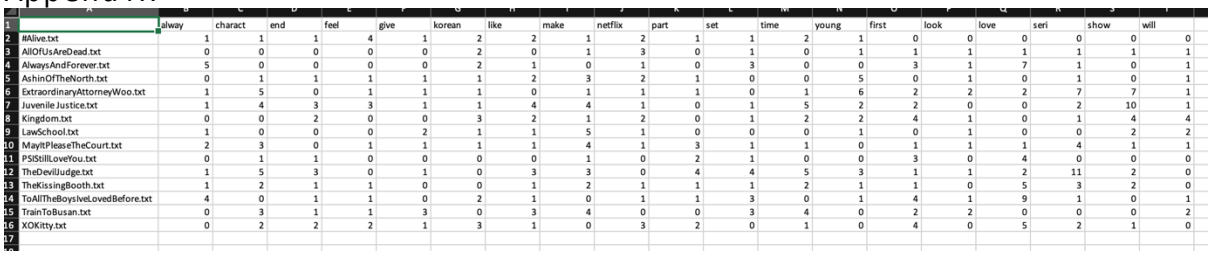
Tallerico, Brian. “Train to Busan Movie Review & Film Summary (2016) | Roger Ebert.” *Rogerebert.com*, Roger Ebert, 2016, www.rogerebert.com/reviews/train-to-busan-2016. Accessed 9 June 2023.

View. “The Devil Judge | Series Review.” *Ahjummamshies.com*, WordPress.com, Sept. 2021, ahjummamshies.com/2021/09/01/the-devil-judge-series-review/. Accessed 9 June 2023.

Wheeler, Greg. “Juvenile Justice Season 1 Review - a Solid Law Drama That Fizzles out at the End.” *The Review Geek*, 27 Feb. 2022, www.thereviewgeek.com/juvenilejustice-s1review/. Accessed 9 June 2023.

Zara, Janelle. “XO, Kitty Review – Convoluted but Charming Netflix Teen Series.” *The Guardian*, The Guardian, 18 May 2023, www.theguardian.com/tv-and-radio/2023/may/18/xo-kitty-review-to-all-the-boys-netflix-spinoff. Accessed 9 June 2023.

Appendix



	alway	charact	end	feel	give	korean	like	make	netflix	part	set	time	young	first	look	love	seri	show	will
#Alive.txt	1	1	1	4	1	2	2	1	2	1	1	2	1	0	0	0	0	0	0
AllOfUsAreDead.txt	0	0	0	0	0	2	0	1	3	0	1	0	1	1	1	1	1	1	1
AlwaysAndForever.txt	5	0	0	0	0	2	1	0	1	0	3	0	0	3	1	7	1	0	1
AshinOfTheNorth.txt	0	1	1	1	1	1	2	3	2	1	0	0	5	0	1	0	1	0	1
ExtraordinaryAttorneyWoo.txt	1	5	0	1	1	1	0	1	1	1	0	1	6	2	2	2	7	7	1
Juvenile Justice.txt	1	4	3	3	1	1	4	4	1	0	1	5	2	2	0	0	2	10	1
Kingdom.txt	0	0	2	0	0	3	2	1	2	0	1	2	2	4	1	0	1	4	4
LawSchool.txt	1	0	0	0	2	1	1	5	1	0	0	0	1	0	1	0	0	2	2
MayItPleaseTheCourt.txt	2	3	0	1	1	1	1	4	1	3	1	1	0	1	1	1	4	1	1
PSIStillLoveYou.txt	0	1	1	0	0	0	0	1	0	2	1	0	0	3	0	4	0	0	0
TheDevilJudge.txt	1	5	3	0	1	0	3	3	0	4	4	5	3	1	1	2	11	2	0
TheKissingBooth.txt	1	2	1	1	0	0	1	2	1	1	1	2	1	1	0	5	3	2	0
ToAllTheBoysIveLovedBefore.txt	4	0	1	1	0	2	1	0	1	1	3	0	1	4	1	9	1	0	1
TrainToBusan.txt	0	3	1	1	3	0	3	4	0	0	3	4	0	2	2	0	0	0	2
XOKitty.txt	0	2	2	2	1	3	1	0	3	2	0	1	0	4	0	5	2	1	0

```
# set working directory
setwd("/Users/aliciang/Downloads/FIT3152-DATA-
ANALYTICS/FIT3152_Assignment3")

# clean up the environment before starting
install.packages("ggplot2")
install.packages("igraph")
rm(list = ls())
library(slam)
library(tm)
library(SnowballC)
library(ggplot2)
library(igraph)

#Get file path to folder "NgChenTing31861148" where the
documents are located
cname = file.path(".", " NgChenTing31861148")
# Turn all documents into a Corpus
docs = Corpus(DirSource((cname)))

# Specific Transformations
toSpace <- content_transformer(function(x, pattern)
gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "’")
toSpace <- content_transformer(function(x, pattern)
gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "-")
toSpace <- content_transformer(function(x, pattern)
gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "–")
toSpace <- content_transformer(function(x, pattern)
gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "“")
toSpace <- content_transformer(function(x, pattern)
gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "”")
toSpace <- content_transformer(function(x, pattern)
gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "one")
toSpace <- content_transformer(function(x, pattern)
gsub(pattern, " ", x))
```

```
docs <- tm_map(docs, toSpace, "two")
toSpace <- content_transformer(function(x, pattern)
gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "also")
toSpace <- content_transformer(function(x, pattern)
gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "get")
toSpace <- content_transformer(function(x, pattern)
gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "just")
toSpace <- content_transformer(function(x, pattern)
gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "see")

#Tokenisation
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, removePunctuation)
docs <- tm_map(docs, content_transformer(tolower))

#Filter words
# Remove stop words and white space
docs <- tm_map(docs, removeWords, stopwords("english"))
docs <- tm_map(docs, stripWhitespace)

# Stem
docs <- tm_map(docs, stemDocument, language = "english")

# Create DTM
dtm = DocumentTermMatrix(docs)
dtm = as.data.frame(as.matrix(dtm))
write.csv(dtm, "OriginalDTM.csv")

dim(dtm)                              # Orginial DTM has 1707
Tokens

dtm = DocumentTermMatrix(docs)
dtms <- removeSparseTerms(dtm, 0.45)
dtms = as.matrix(dtms)
write.csv(dtms,"SparsedMovie.csv")
dim(dtms)                               # 20 columns means 19
words

freq <- colSums(as.matrix(dtms))
ord = order(freq)
freq[tail(ord,20)]


### QUESTION 4
#############################################################
##
```

```
# Calculates Inverse Document Frequency
( idf <- log( ncol(dtms) / ( 1 + rowSums(dtms != 0) ) ) )
# Converts the IDF values into a diagonal matrix
( idf <- diag(idf) )
# Calculates the TF-IDF matrix
tf_idf <- crossprod(dtms, idf)
colnames(tf_idf) <- rownames(dtms)

# Calculates the cosine distance between documents
cosine_dist = 1-crossprod(tf_idf) /
  (sqrt(colSums(tf_idf^2)%*%t(colSums(tf_idf^2))))

# remove NaN using 0
cosine_dist[is.na(cosine_dist)] <- 0

# Converts the cosine distance matrix into a distance object
cosine_dist <- as.dist(cosine_dist)

# Creating dendrogram
cluster1 <- hclust(cosine_dist, method = "ward.D")
plot(cluster1)
# Boxing clusters
rect.hclust(cluster1, k = 3, border = "red")

## QUANTITATIVE MEASURE using confusion matrix##
topics =
c("Horror","Horror","Romance","Horror","Law","Law","Horror",

"Law","Law","Romance","Law","Romance","Romance","Horror","Roma
nce")
groups = cutree(cluster1, k = 3)
table(GroupNames = topics, Clusters = groups)


### QUESTION 5
#################################################################
##

# Convert Document Term Matrix to Binary Matrix
dtmsx = as.matrix(dtms)
dtmsx = as.matrix((dtmsx > 0) + 0)

# Multiply Binary Matrix by its transpose
ByAbsMatrix = dtmsx %*% t(dtmsx)

# Make leading diagonal zero to complete Abstract Network Data
diag(ByAbsMatrix) = 0

# Important nodes
max(colSums(ByAbsMatrix))
closeness = format(closeness(ByAbs), digits = 2)
```

```
degree = as.table(degree(ByAbs))
betweenness = as.table(betweenness(ByAbs))

# Create an Abstracts Network Plot
ByAbs = graph_from_adjacency_matrix(ByAbsMatrix, mode =
"undirected", weighted = TRUE)
plot(ByAbs)

mean(E(ByAbs)$weight)
ReScaledWidth <- c(E(ByAbs)$weight)/5

# Only highlight red for edges where sharing tokens > 6
plot(ByAbs, edge.width = ReScaledWidth,
     edge.color = ifelse(ReScaledWidth > 2, "red", "grey"))
E(ByAbs)$weight


# Heatmap of Network Matrix
palf <- colorRampPalette(c("white","purple"))
heatmap(ByAbsMatrix[,15:1], Rowv = NA, Colv = NA, col =
palf(100),
        scale="none", margins=c(10,10), main = "Heatmap of
Abstract Network Data" )


### QUESTION 6
#################################################################
##

# start with original document-term matrix
dtmsx = as.matrix(dtms)

# convert to binary matrix
dtmsx = as.matrix((dtmsx > 0) + 0)

# multiply transpose binary matrix by binary matrix
ByTokenMatrix = t(dtmsx) %*% dtmsx

# make leading diagonal zero
diag(ByTokenMatrix) = 0

# Network Plot for Tokens
ByAbs = graph_from_adjacency_matrix(ByTokenMatrix, mode =
"undirected",
                                    weighted = TRUE)
plot(ByAbs)

# Highest value in Token Matrix
max(colSums(ByTokenMatrix))

# Importance of nodes
```

```
d = as.table(degree(ByAbs))
b = as.table(betweenness(ByAbs))
c = as.table(closeness(ByAbs))
e = as.table(evcent(ByAbs)$vector)
stats = as.data.frame(rbind(d,b,c,e))
stats = as.data.frame(t(stats))
colnames(stats) = c("degree", "betweenness", "closeness",
"eigenvector")

# Clustering using Greedy
cfg = cluster_fast_greedy(as.undirected(ByAbs))
g_cfg = plot(cfg,

as.undirected(ByAbs),vertex.label=V(ByAbs)$role,main="Fast
Greedy
             for Token Clustering")

### QUESTION 7
##############################################################
##

## PRE_PROCESSING ##
dtmsa = as.data.frame(dtms) # clone dtms
dtmsa$ABS = rownames(dtmsa) # adds a new column named "ABS" to
contain names
dtmsb = data.frame()        # empty data frame

# Iterate over DTM and extract their corresponding weight,
document name
# and token name to sppend to the dtmsb data frame
for (i in 1:nrow(dtmsa)){
  for (j in 1:(ncol(dtmsa)-1)){
    touse = cbind(dtmsa[i,j], dtmsa[i,ncol(dtmsa)],
                  colnames(dtmsa[j]))
    dtmsb = rbind(dtmsb, touse ) } } # close loops
colnames(dtmsb) = c("weight", "abs", "token")
dtmsc = dtmsb[dtmsb$weight != 0,] # delete 0 weights

# put colunms in order: abs, token, weight
dtmsc = dtmsc[,c(2,3,1)]

# Importance of nodes
d = as.table(degree(ByAbs))
b = as.table(betweenness(ByAbs))
c = as.table(closeness(ByAbs))
e = as.table(evcent(ByAbs)$vector)
stats = as.data.frame(rbind(d,b,c,e))
stats = as.data.frame(t(stats))
colnames(stats) = c("degree", "betweenness", "closeness",
"eigenvector")
```

```
# Creating Bipartite Network
g <- graph.data.frame(dtmsc, directed=FALSE)
bipartite.mapping(g)
# Adding type, colour, shape to vertices and edges
V(g)$type <- bipartite_mapping(g)$type
V(g)$color <- ifelse(V(g)$type, "pink", "darkslategray1")
V(g)$shape <- ifelse(V(g)$type, "circle", "square")
E(g)$color <- "lightgray"
plot(g)

# Closeness of each node in Network
max(closeness(g))

# Clustering using Greedy
cfg = cluster_fast_greedy(as.undirected(g))
g_cfg = plot(cfg,

as.undirected(g),vertex.label=V(g)$role,main="Fast Greedy")
```