

# FIT3152 Data analytics

## Assignment 3

**Name:** Rhyme Bulbul

**Student ID:** 31865224

**AI statement:** No content generated by AI technologies has been used in this assessment.

### Task 1

I gathered a collection of 19 text documents from different sources that covered a range of themes for this task. Primarily these focus on either `linux`, or movie reviews. `fast`, `pirates` and `impossible` are the movie reviews included in these publications. These refer to the movie sequels “The Fast and The Furious”, “The Pirates of the Caribbean” and “Mission Impossible” respectively.

Text files were created by copying and pasting the document contents, and references can be found in the appendix.

### Task 2

Task 1 already had the document’s contents pasted into text files. The text files were assembled into the `text` folder within the working directory in order to generate the corpus. Next, the code that follows is executed. It installs and imports the necessary libraries, sets the seed and uses the `Corpus()` function from the `tm` package to construct the corpus.

```
rm(list = ls())
set.seed(31865224)

library(tm)
library(cluster)
library(igraph)

cname <- file.path(".", "text")
docs <- Corpus(DirSource((cname)))
list.files("text")
```

```
## [1] "fast01.txt"      "fast02.txt"      "fast03.txt"      "fast04.txt"
## [5] "fast05.txt"      "impossible01.txt" "impossible02.txt" "impossible03.txt"
## [9] "impossible04.txt" "impossible05.txt" "linux01.txt"      "linux02.txt"
## [13] "linux03.txt"     "linux04.txt"     "linux05.txt"     "pirates01.txt"
## [17] "pirates02.txt"   "pirates03.txt"   "pirates04.txt"   "pirates05.txt"
```

Documents are named based on the topic, suffixed by a number. Such as, `linux01.txt` is the first text document on linux systems, followed by `linux02.txt` the second text document on the same topic, and so fourth.

### Task 3

We start by text transforming the corpus, meanwhile replacing dashes and line breaks with spaces for consistency, removing numbers, punctuation, converting all characters to lowercase, and removing any extra white spaces. In addition, we also remove English stop words before finally stemming all words for consistency. This is required as we don’t want these characters creating an unwanted bias in our data, as it would be harder to work with, as well as inaccurate.

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

```
docs <- tm_map(docs, to_space, "\n")
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, removePunctuation)
docs <- tm_map(docs, content_transformer(tolower))
docs <- tm_map(docs, stripWhitespace)
docs <- tm_map(docs, removeWords, stopwords("english"))
docs <- tm_map(docs, stemDocument, language = "english")
```

With our unwanted terms removed, and desired keywords preserved, the corpus is now ready to create a document term matrix from.

```
dtm <- DocumentTermMatrix(docs)
```

Let's analyse the attributes of the document term matrix by inspecting a sample of 20 from its head and tail of most and least frequent terms in alphabetical order.

```
inspect(dtm)
```

```
## <<DocumentTermMatrix (documents: 20, terms: 5539)>>
## Non-/sparse entries: 13346/97434
## Sparsity          : 88%
## Maximal term length: 58
## Weighting          : term frequency (tf)
## Sample            :
##
##      Terms
## Docs      android devic fast  furious googl kernel linux pirat use version
## fast01.txt      0      0  70    68      0      0      0      4  0      3
## fast04.txt      0      0  85    57      0      0      0      6  1      1
## fast05.txt      0      0  76    68      0      0      0      4  2      6
## linux01.txt     14     37   5      0      8     283    244      0 69     76
## linux02.txt      0      4   0      0      3      8     42      0 52     27
## linux03.txt    468    178   1      0    262     36     42      3 104     77
## linux04.txt     17     41   0      0     10      2      3      0 17     38
## linux05.txt     96     39   0      0     28      1      3      0 19     29
## pirates01.txt    0      0   2      2      0      0      0     38  4      2
## pirates05.txt    0      0   2      2      0      0      0     53  1      1
```

```
freq <- colSums(as.matrix(dtm))
freq[head(order(freq), 20)]
```

```
##      actress      apr      assassin      bandolero
##          1          1          1          1
##      blast      bloodi      blu      campo
##          1          1          1          1
##      cara      chemistri claudiocarvalhodec      clown
##          1          1          1          1
##      conner      convoy      crawl      distributor
##          1          1          1          1
##      drivera      dwight      enemi      entwinn
##          1          1          1          1
```

```
freq[tail(order(freq), 20)]
```

```
##      user      man      imposs      mission      ubuntu      system      support      caribbean
##      187      188      194      203      204      218      219      235
##      develop      pirat      releas      version      use      devic      furious      googl
##      249      264      264      273      278      301      307      311
```

```
##      kernel      linux      fast      android
##        330        334        356        595
```

With 6062 terms, or as we call it, tokens; the DTM is highly sparse at 88%. Interestingly, the least occurring tokens seem to only appear once, while the most frequent token is **android**, occurring almost 500 times, although it is only expected to be used in 3 documents, pointing to it's sparsity.

This leads us to the removal of sparse tokens from the document term matrix. This time we set the sparsity to 9% as it gives us the best mix of efficiency, reliability, and observability. This figure was chosen as it allows the DTM to have around 20 tokens after removing sparse terms without sacrificing on other attributes, due to the nature of the corpus chosen.

```
dtms <- removeSparseTerms(dtm, 0.09)
inspect(dtms)
```

```
## <<DocumentTermMatrix (documents: 20, terms: 30)>>
## Non-/sparse entries: 582/18
## Sparsity           : 3%
## Maximal term length: 7
## Weighting          : term frequency (tf)
## Sample            :
##                  Terms
## Docs
## fast04.txt         1      2      2      2      7      1      1      1      5      2
## fast05.txt         2      2      2      2      3      2      4      1      5      2
## impossible01.txt   2      2      1      3      4      2      1      2      4      2
## linux01.txt        39      6     18      7     17      8      1     62      8     36
## linux02.txt        24      2     10     12     11     13      1     57      5     20
## linux03.txt        48     25     29     15     22     51     50     45     11     73
## linux04.txt        15      2     16     20      4     11      7     18      1     15
## linux05.txt        11     11     15      8      9     19      3     61      3     13
## pirates01.txt       2      2      3      1      6      2      1      1      5      2
## pirates05.txt       1      2      2      4      2      3      1      3      7      2
```

```
freqs <- colSums(as.matrix(dtms))
freqs[head(order(freqs), 20)]
```

```
##      sign connect  critic technic      date languag      help      color countri popular
##        22      23      28      29      31      34      35      36      36      44
##      unit  video      box  origin  known  offic      top  review product      like
##        50      50      52      52      58      60      66      71      73      76
```

```
freqs[tail(order(freqs), 20)]
```

```
##      unit  video      box  origin  known  offic      top  review product      like
##        50      50      52      52      58      60      66      71      73      76
## compani  play  offici      see      one  featur  open      also      user  releas
##        78      83      87      93     112     119     134     165     187     264
```

This time round, the least frequent tokens are **exact**, **speed**, **pass** and **care** while the most frequent is **use** with 259 uses. The Document term matrix in it's full length can be found attached in the appendix.

## Task 4

To determine which performs better, a hierarchical clustering of the corpus is carried out using two metrics: Euclidean distance and cosine distance. First, the DTM is transformed into a regular matrix format.

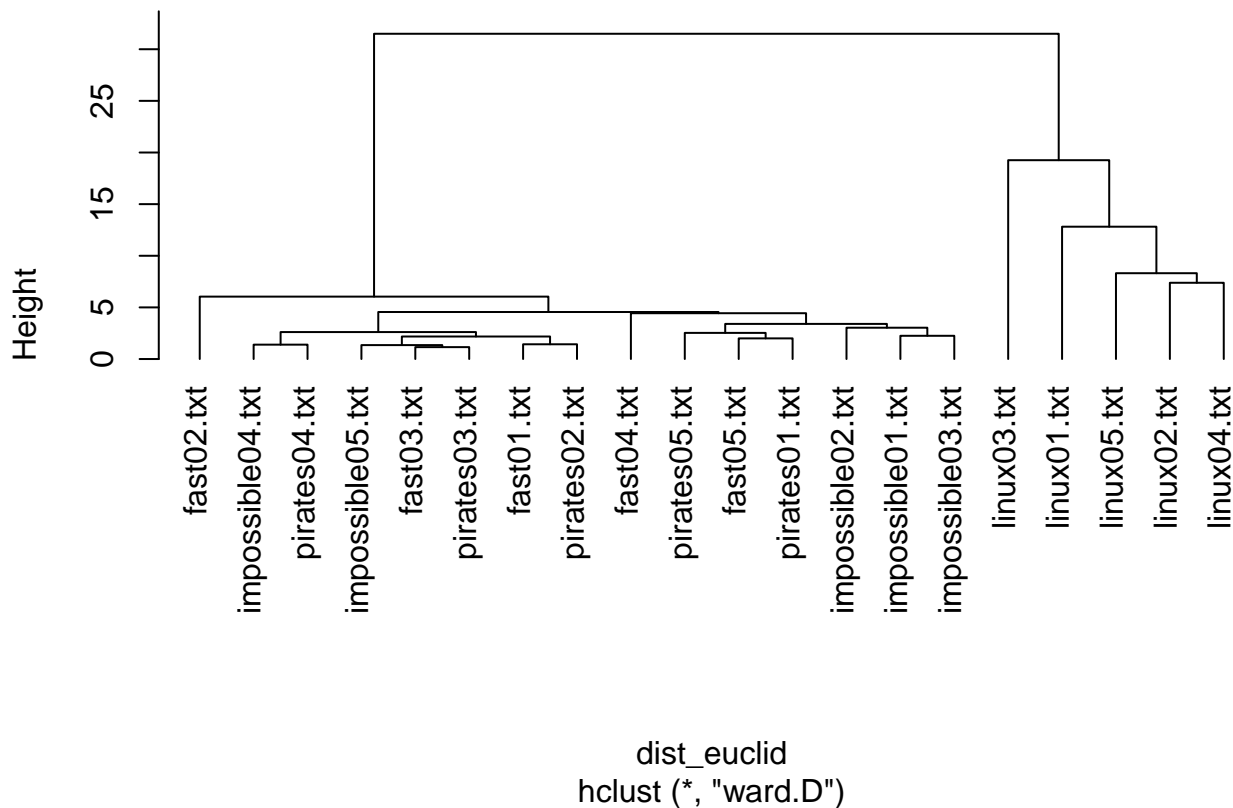
```
dtms_matrix <- as.matrix(dtms)
```

Each document in document clustering is represented as a vector with numerous dimensions that match the terms it contains. Euclidean distance classifies these vectors according to proximity by measuring the straight-line distance between them. Greater similarity is indicated by a shorter distance. The angle between vectors is measured by cosine distance; smaller angles indicate closer distances and more similarity. By pre-weighting the DTM with the term frequency-inverse document frequency (TF-IDF) statistic, which gives higher weights to terms that appear frequently within a document (implying importance) but infrequently across all documents (implying significance), clustering with cosine distance can be further enhanced.

This code is an adaptation of Lecture 10's clustering using Euclidean distance algorithm. After scaling and converting `dtms_matrix` to a Euclidean distance matrix, a dendrogram is shown.

```
dist_euclid <- dist(scale(dtms_matrix))
fit_euclid <- hclust(dist_euclid, method = "ward.D")
plot(fit_euclid, hang = -1)
```

## Cluster Dendrogram



The IDF for each term is first calculated, and a TF-IDF weighted matrix is generated by applying the cross product of the TFs and IDFs in order to cluster with TF-IDF weighting and cosine distance. Next, the matrix is subjected to the cosine distance formula to obtain a cosine distance matrix. After that, the dendrogram is plotted.

```
idf <- log(ncol(dtms_matrix) / (1 + rowSums(dtms_matrix != 0)))
idf <- diag(idf)
dtms_matrix_tfidf <- crossprod(dtms_matrix, idf)
colnames(dtms_matrix_tfidf) <- rownames(dtms_matrix)
```

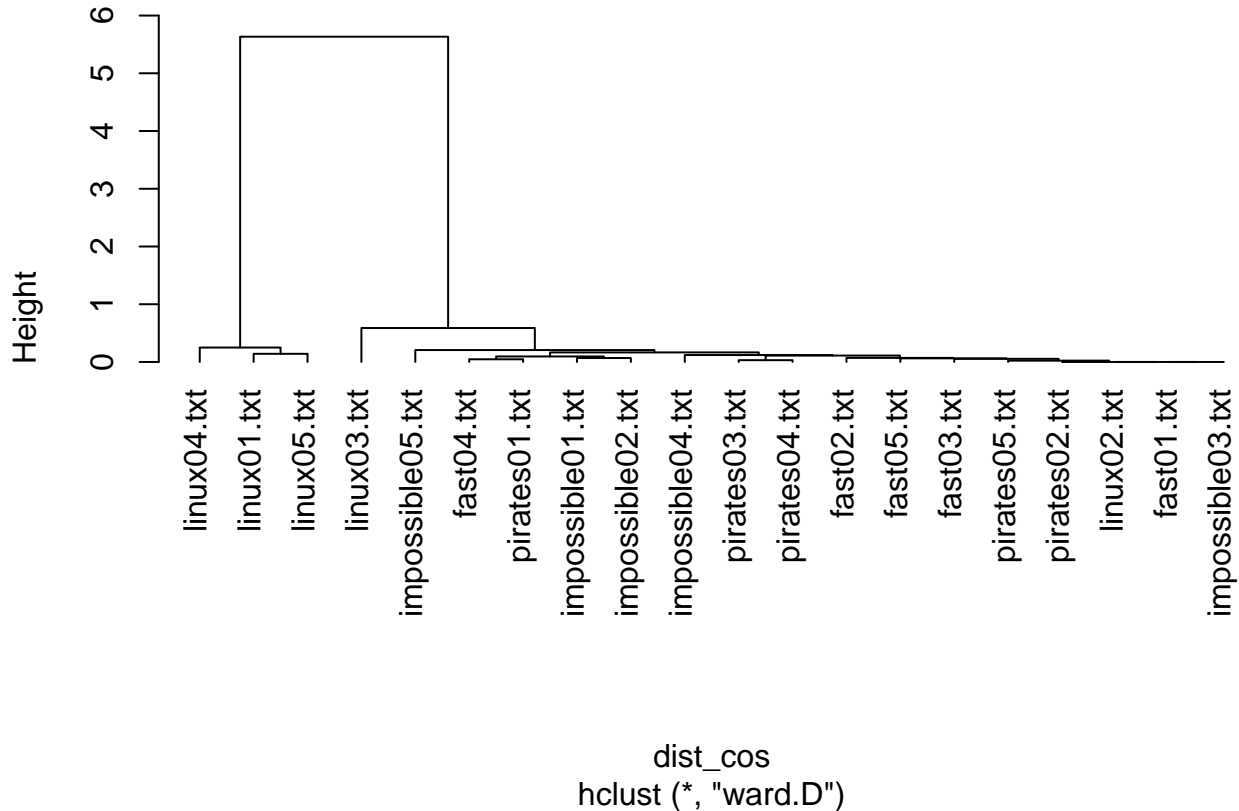
```

dist_cos <- 1 - crossprod(dtms_matrix_tfidf) / (sqrt(colSums(dtms_matrix_tfidf ** 2) %*%
  t(colSums(dtms_matrix_tfidf ** 2))))
dist_cos[is.na(dist_cos)] <- 0
dist_cos <- as.dist(dist_cos)

fit_cos <- hclust(dist_cos, method = "ward.D")
plot(fit_cos, hang = -1)

```

## Cluster Dendrogram



In terms of Euclidean distance clustering, the first divide is between two clusters, one of movie reviews including **fast**, **pirates** and **impossible**, and another of purely **linux**. This is pretty accurate, with the **linux** cluster branching off with a new document in each cluster. The movies reviews cluster is further branched off with **fast02** separated from all others. While unexpected, this points the document to be an outlier. The next two clusters branched do a worse job, with all movies reviews being split across the clusters, it also noted that with heights much lower than 20, we expect lower accuracy compared to the **linux** cluster.

Cosine Distance clustering is similar story, however only correctly identifying three **linux** documents into a cluster, and all the others into another. With an even lower height, less than one, for this cluster this time, even for the **linux** cluster, it is no surprise that the movies reviews cluster is highly inaccurate, with no real correlation between clustered documents.

By marking each document with its topic, creating a confusion matrix of the clustering, and calculating the accuracy, it is able to obtain a quantitative assessment of each clustering because each document's true topic is known. Owing to their length, the Appendix displays the confusion matrices and the cluster-topic assignments, and in order to reduce topic-cluster ambiguity for Euclidean distance, 15 clusters are produced.

```

short_name <- function(doc) {
  return(substr(doc, 1, nchar(doc) - 6))
}

doc_names <- list.files("text")
doc_names_short <- unlist(lapply(doc_names, short_name))

table(Topic = doc_names_short, Cluster = cutree(fit_euclid, k = 15))
table(Topic = doc_names_short, Cluster = cutree(fit_cos, k = 10))

```

The matrix accuracy is determined by hand. This is provided for clustering using Euclidean distance by

```
4 / 20
```

```
## [1] 0.2
```

And for Cosine distance is provided by

```
9 / 20
```

```
## [1] 0.45
```

Clustering with cosine distance has a considerably higher accuracy than clustering with Euclidean distance, which is consistent with the data.

Hierarchical clustering can be said to be the quicker and easier way to get an understanding of the relationships between documents and tokens in the corpus. It neatly groups documents into clusters at different heights, so groupings can be interpreted based on the desired cluster size. The accuracies of the clusterings performed in Task 4 are not very high, but an accuracy of around 45% for clustering with cosine distance can be considered strong for a small corpus (only 20 documents) and the genericness of the tokens in contrast to Euclidean Distance clustering, which did extremely poorly at 20%, but is justified given the sparsity, topic overlap, and small sample size.

Euclidean distance clustering has an accuracy of 20%, low homogeneity, poor cluster purity, and weak cohesion. On the other hand, cosine distance clustering has an accuracy of 45%, moderate homogeneity, better cluster purity, and improved cohesion. Both methods have their strengths and weaknesses, with cosine distance clustering demonstrating better accuracy, homogeneity, cluster purity, and cohesion compared to Euclidean distance clustering.

Clustering, however, is unable to recognise significant documents, tokens, or groupings. Social network analysis helps users quickly understand the relationships within the corpus by visualising the relationships between documents, tokens, and document-token pairs. Viewers can use computed metrics (such as proximity, betweenness, transitivity, etc.) to determine the significance of each document or token or the network's overall connectivity if necessary. Because of this, social networks provide a versatile means of locating significant clusters and connections within the data that may be leveraged by a larger number of users, whether they be technical or public viewers.

## Task 5

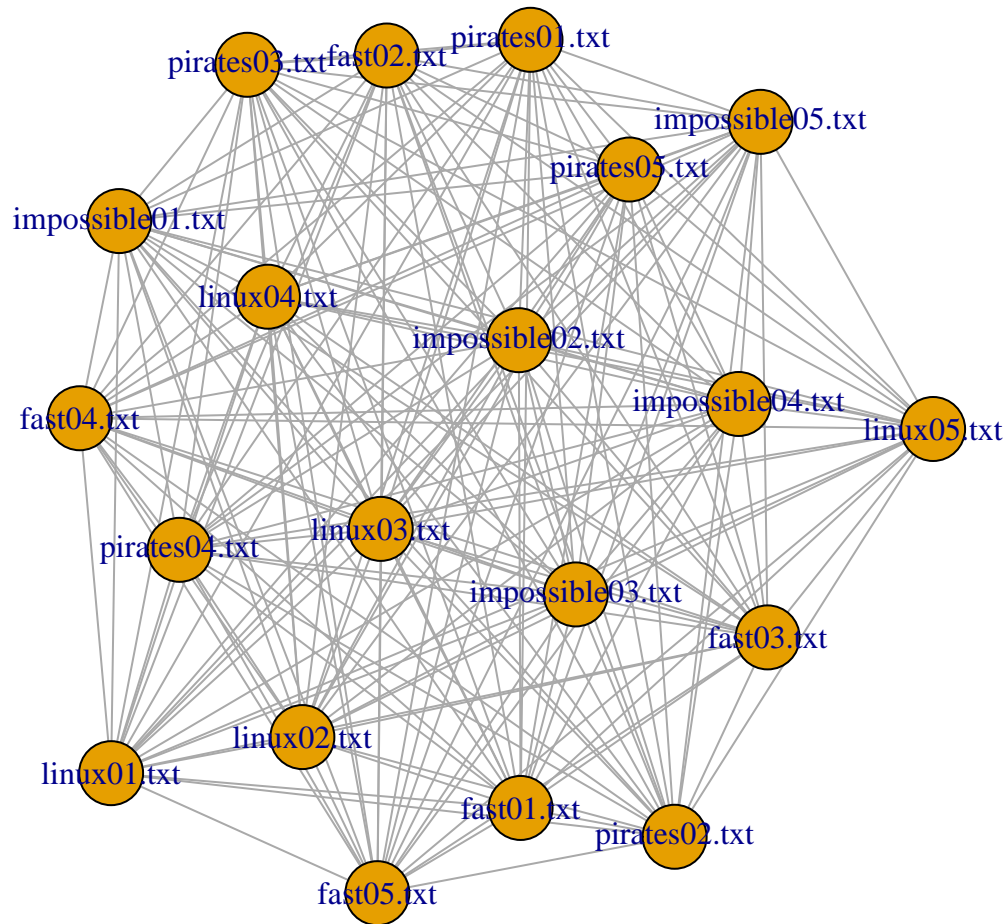
The DTM is first transformed into a binary matrix and then multiplied by its transpose in order to construct a single-mode network that visualises the connections between documents based on the quantity of shared phrases. Following the first phase, a matrix is created that has a record of 1 for each token that appears in the document of that row. Following multiplication, the number of shared tokens in every pair of documents is displayed in the resulting matrix.

```

dtms_mat_bin <- as.matrix((dtms_matrix > 0) + 0)
abs_mat <- dtms_mat_bin %*% t(dtms_mat_bin)
diag(abs_mat) <- 0

```

```
abs_net <- graph_from_adjacency_matrix(abs_mat, mode = "undirected", weighted = TRUE)
plot(abs_net)
```



The graph depicted is quite dense; practically every pair of vertices that might exist has an edge. This is demonstrated by computing the graph's density.

```
graph.density(abs_net)
```

```
## Warning: `graph.density()` was deprecated in igraph 2.0.0.
## i Please use `edge_density()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
## [1] 1
```

This demonstrates how each document is connected to nearly every other document to some degree based on common phrases. In comparison to other vertex pairings, **pirates03**, and **pirates01** are extremely close to **fast02**, suggesting a strong association, whereas **linux05** appears to have comparatively weaker ties to the other documents.

Finding the graph's transitivity is an excellent place to start when trying to find distinct groups in the data.

```
transitivity(abs_net)
```

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

The ratio of triangles to linked triples is known as transitivity, or clustering coefficient; a larger value of this ratio denotes more closely spaced groups. This graph has an extremely high transitivity of 1. There are numerous ways to divide the documents into groups, such as hierarchical clustering, but the `linux`, `pirates` and `fast` may be the most influential groups.

Because of their prominent locations, `impossible02`, `impossible03` and `linux03` initially appear to be the most significant documents. To obtain numerical measurements, execute the code provided. The outcomes are displayed in the Appendix due of their length.

```
sort(-closeness(abs_net))
sort(-betweenness(abs_net))
sort(evcent(abs_net)$vector)
```

```
## Warning: `evcent()` was deprecated in igraph 2.0.0.
## i Please use `eigen_centrality()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
sort(degree(abs_net))
```

Eigenvector centrality evaluates a document's quality of connections, degree counts the number of connections made by a vertex, closeness assesses a document's connectivity to others, betweenness evaluates a document's ability to act as an intermediary and thus have more influence over the network flow. The results for closeness and betweenness are negative because the weights of the vertices match the amount of common phrases. This means that when two documents are tightly related, there is a high distance (weight of the edge) between them.

Documents with values of 0 for betweenness have the most potential to be the "hubs" of the network. Just `linux`, `impossible03` and `pirates02` are disconnected from all other documents in terms of degree. Their degrees are all still at the maximum of 19, meaning that they are quite significant. Others, however, do not give us meaningful insights in contrast to each document.

The graph is enhanced by colouring the vertices according to subject, scaling the edge widths according to inter-document strength (number of common terms), and scaling the vertex sizes according to the closeness centrality of the document in order to visually represent the noteworthy aspects of the data. This code can be used to accomplish this.

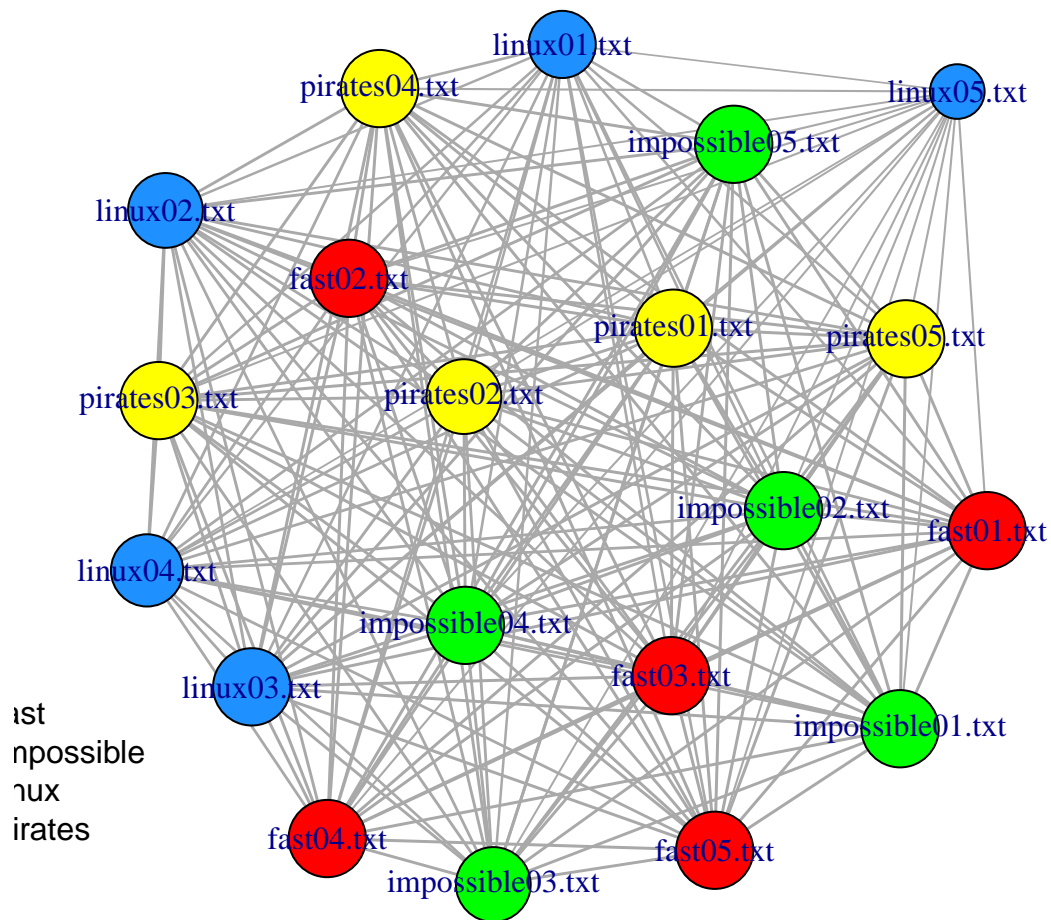
```
doc_colr <- c("red", "green", "dodgerblue", "yellow", "orange", "magenta", "pink",
              "chocolate", "gray", "cyan")
topics <- unique(doc_names_short)

for (doc in doc_names) {
  V(abs_net)[doc]$color <- doc_colr[match(short_name(doc), topics)]
}

V(abs_net)$size <- 1 / closeness(abs_net, mode = "all") / 30
E(abs_net)$width <- E(abs_net)$weight / 20

plot(abs_net)
legend(x = -1.5, y = -0.5, legend = topics, pch = 21, cex = 1,
      pt.bg = doc_colr, bty = "n", ncol = 1)
```





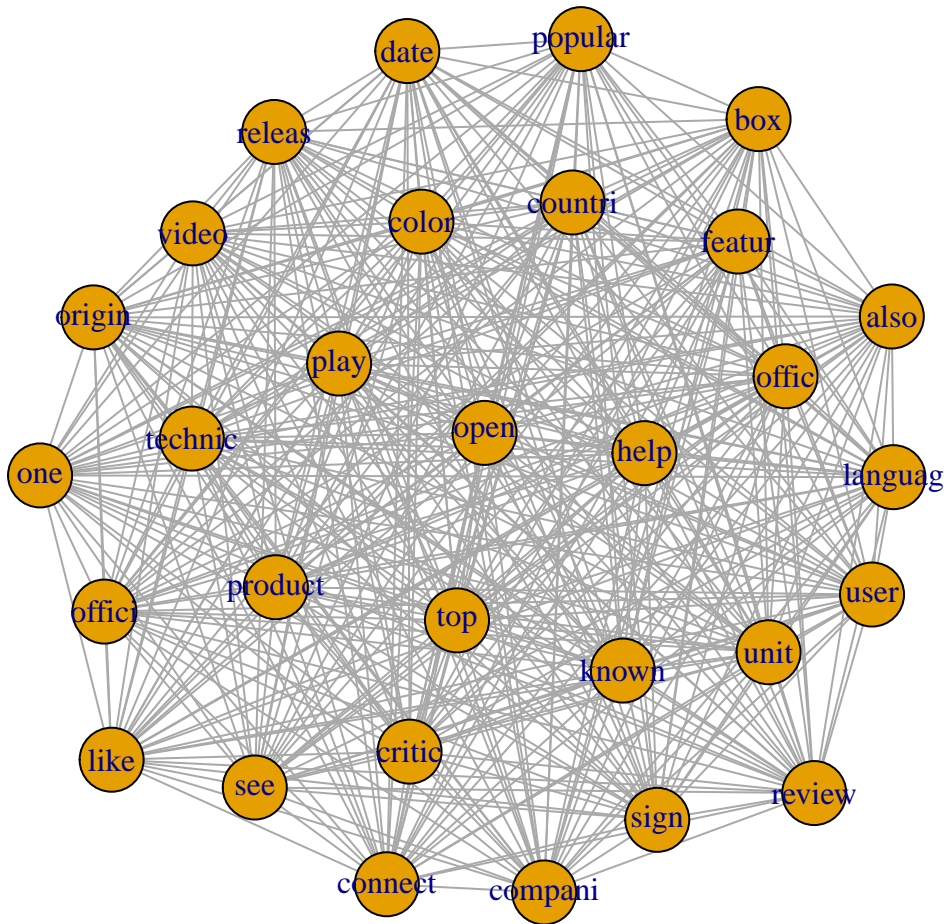
We can see how far apart the **fast** docs are from the others with greater clarity thanks to this graph. It is also simpler to see that some topics like **pirates** contain all of the papers closely together, whereas other topics have the reverse.

## Task 6

Most of the code is repurposed from Task 5 with minor modifications to create a single-mode network that shows the relationships between the tokens depending on the number of common documents they appear in. The binary matrix's transpose is multiplied by the original to get the tokens matrix. Next, the graph is plotted using the identical function.

```
tok_mat <- t(dtms_mat_bin) %*% dtms_mat_bin
diag(tok_mat) <- 0

tok_net <- graph_from_adjacency_matrix(tok_mat, mode = "undirected", weighted = TRUE)
plot(tok_net)
```



The graph depicted appears to be much denser than the one for documents, with many more connections between nodes.

```
graph.density(tok_net)
```

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

According to the observation, the network has a density of 1, indicating that every vertex is connected, making it denser. This indicates that the tokens are quite general terms that show up in all publications on different subjects.

The vertices of the graph appear to be rather uniformly distributed, making it challenging to discern distinct groups. **date**, **popular** and **box** at the top of the graph, which form a prominent triangle, are suspected clusters.

```
transitivity(tok_net)
```

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

The graph's transitivity is 1. This measure shows us that every three tokens can form a triangle, indicating a very large number of potential clusters, even though it is difficult to distinguish clusters by observation. The generic nature of the tokens may also be to blame for this.\*During DTM pre-processing, I tried manually eliminating a few generic terms, but it resulted in worse clustering performance. This is most likely due to the fact that, despite being widely used, generic phrases offer crucial contextual information that ties papers together.

**open**, **play**, **help** and **top** are the tokens that appear to be the most central, suggesting that they may hold some significance. To look into this, execute the code below. The Appendix displays the results.

```

sort(-closeness(tok_net))
sort(-betweenness(tok_net))
sort(evcent(tok_net)$vector)
sort(degree(tok_net))

```

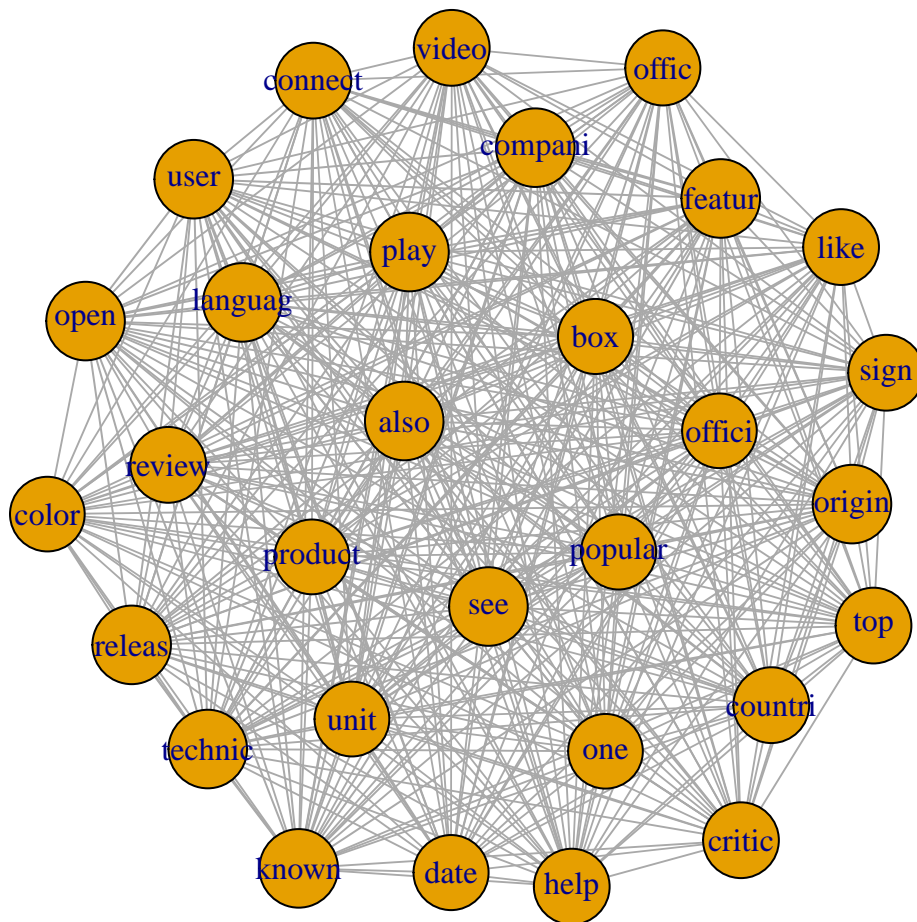
In order to make this graph better, the weights of the number of shared documents between tokens and the vertex sizes are scaled to their closest centralities.

```

V(tok_net)$size <- 1 / closeness(tok_net, mode = "all") / 30
E(tok_net)$width <- E(tok_net)$weight / 20

plot(tok_net)

```



## Task 7

```

dtms_dfa <- as.data.frame(dtms_matrix)
dtms_dfa$abstract <- rownames(dtms_dfa)
dtms_dfb <- data.frame()
for (i in seq_len(nrow(dtms_dfa))) {
  for (j in seq_len(ncol(dtms_dfa) - 1)) {
    to_use <- cbind(dtms_dfa[i, j], dtms_dfa[i, ncol(dtms_dfa)],
                    colnames(dtms_dfa[j]))
    dtms_dfb <- rbind(dtms_dfb, to_use)
  }
}

```

```

}
colnames(dtms_dfb) <- c("weight", "abstract", "token")

dtms_dfc <- dtms_dfb[dtms_dfb$weight != 0, ]
dtms_dfc <- dtms_dfc[, c("abstract", "token", "weight")]

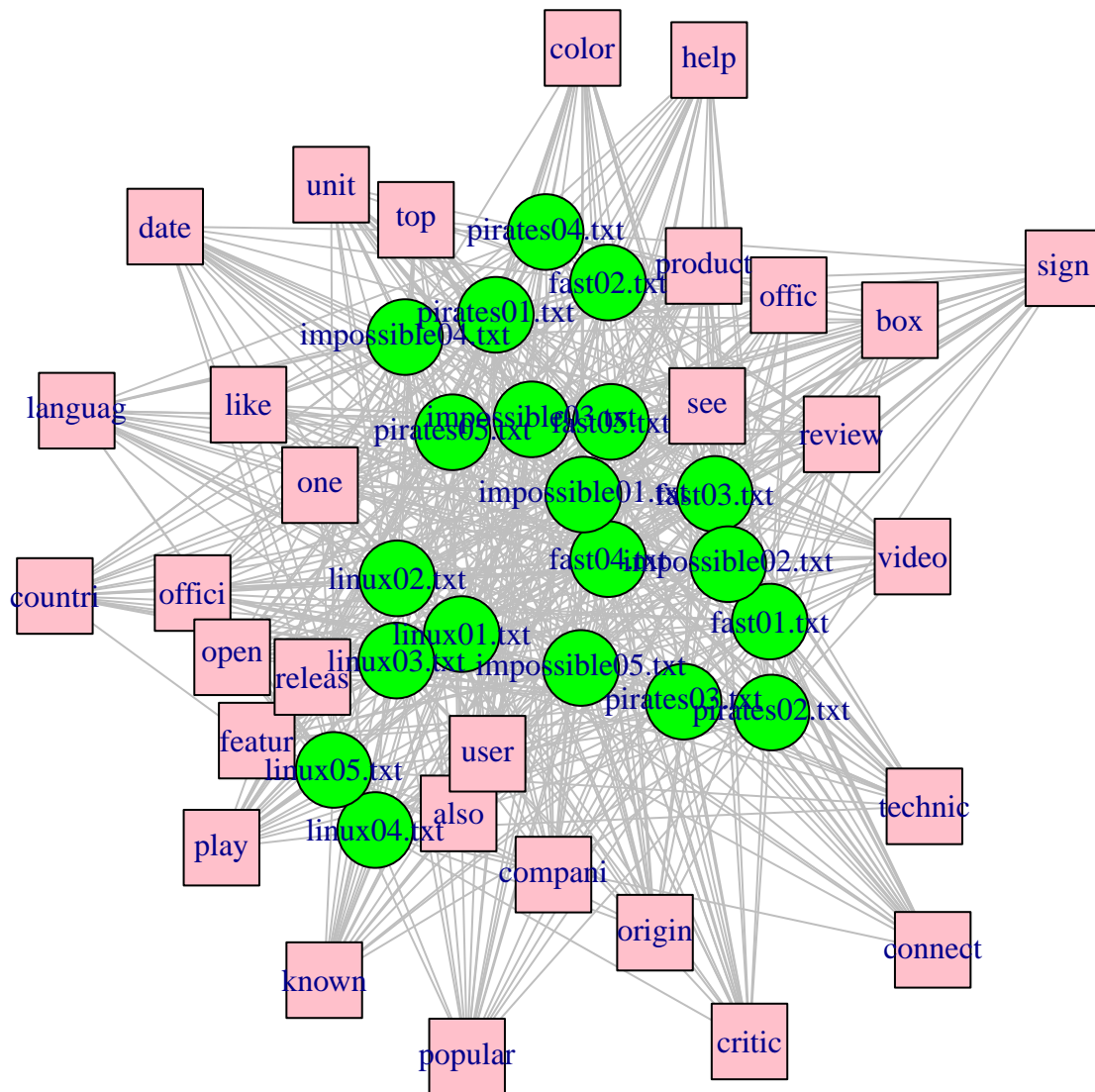
bipart <- graph.data.frame(dtms_dfc, directed = FALSE)

## Warning: `graph.data.frame()` was deprecated in igraph 2.0.0.
## i Please use `graph_from_data_frame()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

V(bipart)$type <- bipartite_mapping(bipart)$type
V(bipart)$color <- ifelse(V(bipart)$type, "pink", "green")
V(bipart)$shape <- ifelse(V(bipart)$type, "square", "circle")
E(bipart)$color <- "grey"

plot(bipart)

```



```

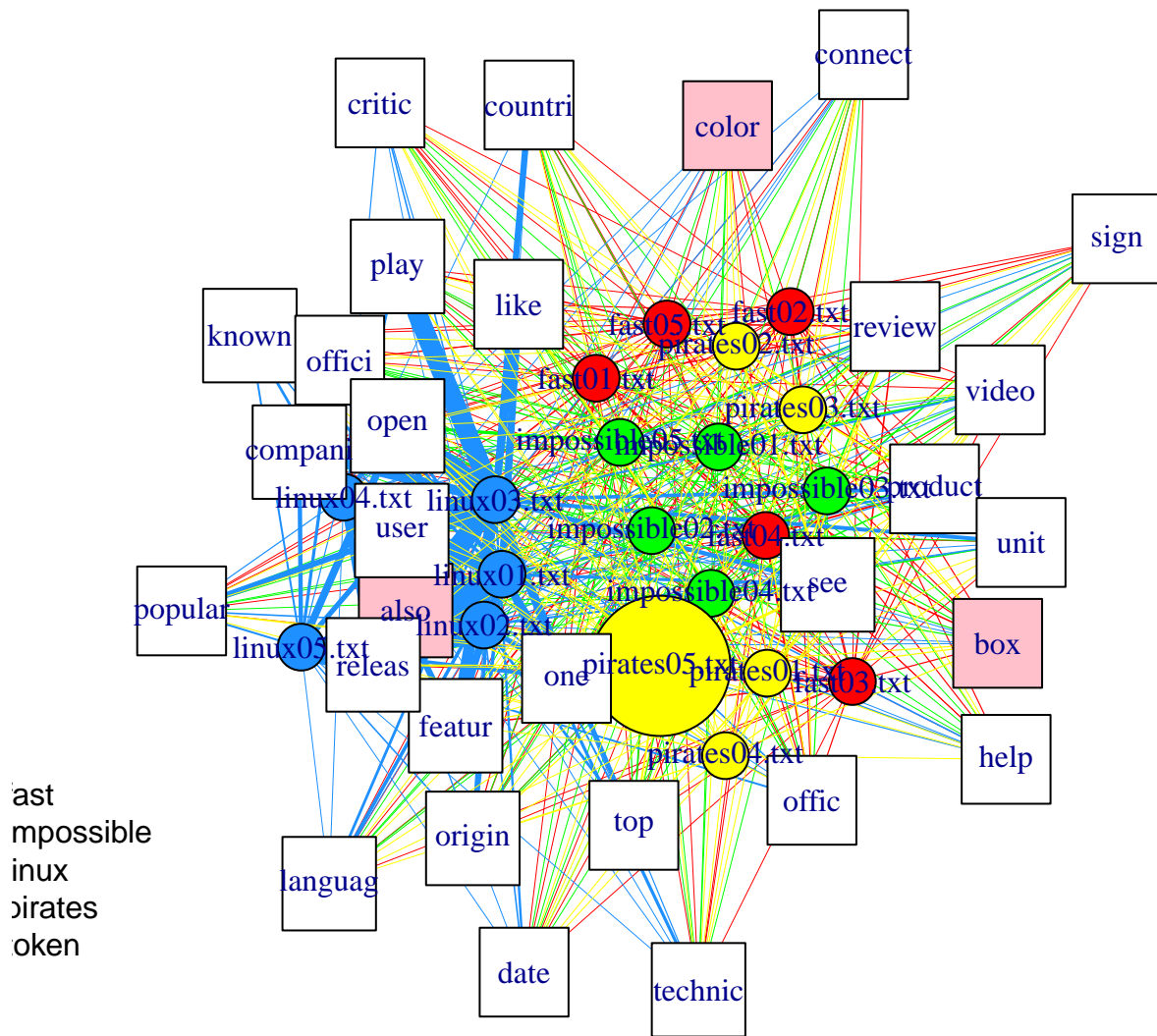
sort(degree(bipart))

for (i in seq_len(23)) V(bipart)$size[i] <- 10
V(bipart)$size[20:50] <- degree(bipart)[20:50]
for (j in seq_len(length(doc_names))) {
  V(bipart)[j]$color <- doc_colr[match(short_name(doc_names[j]), topics)]
}
for (k in c(24:length(V(bipart)))) V(bipart)[k]$color <- "white"
E(bipart)$width <- as.numeric(dtms_dfc$weight) / 5
E(bipart)$color <- tail_of(bipart, E(bipart))$color

plot(bipart)
legend(x = -1.5, y = -0.5, legend = c(topics, "token"), pch = 21, cex = 1,
      pt.bg = c(doc_colr, "white"), bty = "n", ncol = 1)

```





## Appendix

### References

- Wikimedia Foundation. (2024, May 14). Ubuntu. Wikipedia. <https://en.wikipedia.org/wiki/Ubuntu>
- Wikimedia Foundation. (2024c, May 18). Linux kernel. Wikipedia. [https://en.wikipedia.org/wiki/Linux\\_kernel](https://en.wikipedia.org/wiki/Linux_kernel)
- Wikimedia Foundation. (2024a, April 5). Lineageos. Wikipedia. <https://en.wikipedia.org/wiki/LineageOS>
- Wikimedia Foundation. (2024b, May 7). Android (Operating System). Wikipedia. [https://en.wikipedia.org/wiki/Android\\_\(operating\\_system\)](https://en.wikipedia.org/wiki/Android_(operating_system))
- Wikimedia Foundation. (2024b, May 7). Android (Operating System). Wikipedia. <https://en.wikipedia.org/wiki/CyanogenMod>
- IMDb.com. (2003, June 6). 2 fast 2 furious. IMDb. [https://www.imdb.com/title/tt0322259/?ref\\_=nv\\_sr\\_srg\\_0\\_tt\\_7\\_nm\\_1\\_q\\_2%2520fast](https://www.imdb.com/title/tt0322259/?ref_=nv_sr_srg_0_tt_7_nm_1_q_2%2520fast)
- IMDb.com. (2009, April 3). Fast & Furious. IMDb. [https://www.imdb.com/title/tt1013752/?ref\\_=nv\\_sr\\_srg\\_1\\_tt\\_7\\_nm\\_0\\_q\\_fast%2520and%2520fur](https://www.imdb.com/title/tt1013752/?ref_=nv_sr_srg_1_tt_7_nm_0_q_fast%2520and%2520fur)

IMDb.com. (2006, June 16). The Fast and the Furious: Tokyo Drift. IMDb. [https://www.imdb.com/title/tt0463985/?ref\\_=nv\\_sr\\_srgs\\_1\\_tt\\_7\\_nm\\_0\\_q\\_tokyo%2520d](https://www.imdb.com/title/tt0463985/?ref_=nv_sr_srgs_1_tt_7_nm_0_q_tokyo%2520d)

IMDb.com. (2011, April 29). Fast five. IMDb. [https://www.imdb.com/title/tt1596343/?ref\\_=fn\\_al\\_tt\\_6](https://www.imdb.com/title/tt1596343/?ref_=fn_al_tt_6)

IMDb.com. (2013). Fast & Furious 6. IMDb. [https://www.imdb.com/title/tt1905041/?ref\\_=nv\\_sr\\_srgs\\_0\\_tt\\_8\\_nm\\_0\\_q\\_fast%25206](https://www.imdb.com/title/tt1905041/?ref_=nv_sr_srgs_0_tt_8_nm_0_q_fast%25206)

IMDb.com. (2003b, July 9). Pirates of the Caribbean: The curse of the black pearl. IMDb. [https://www.imdb.com/title/tt0325980/?ref\\_=nv\\_sr\\_srgs\\_0\\_tt\\_8\\_nm\\_0\\_q\\_pirates](https://www.imdb.com/title/tt0325980/?ref_=nv_sr_srgs_0_tt_8_nm_0_q_pirates)

IMDb.com. (2006). Pirates of the Caribbean: Dead Man's Chest. IMDb. [https://www.imdb.com/title/tt0383574/?ref\\_=nv\\_sr\\_srgs\\_6\\_tt\\_5\\_nm\\_3\\_q\\_pirates%2520of%2520the%2520caribbean](https://www.imdb.com/title/tt0383574/?ref_=nv_sr_srgs_6_tt_5_nm_3_q_pirates%2520of%2520the%2520caribbean)

IMDb.com. (2003b, July 9). Pirates of the Caribbean: At World's End. IMDb. [https://www.imdb.com/title/tt0449088/?ref\\_=nv\\_sr\\_srgs\\_7\\_tt\\_5\\_nm\\_3\\_q\\_pirates%2520of%2520the%2520caribbean](https://www.imdb.com/title/tt0449088/?ref_=nv_sr_srgs_7_tt_5_nm_3_q_pirates%2520of%2520the%2520caribbean)

IMDb.com. (2003b, July 9). Pirates of the Caribbean: On Stranger Tides. IMDb. [https://www.imdb.com/title/tt1298650/?ref\\_=nv\\_sr\\_srgs\\_9\\_tt\\_5\\_nm\\_3\\_q\\_pirates%2520of%2520the%2520caribbean](https://www.imdb.com/title/tt1298650/?ref_=nv_sr_srgs_9_tt_5_nm_3_q_pirates%2520of%2520the%2520caribbean)

IMDb.com. (2017, May 26). Pirates of the caribbean: Dead men tell no tales. IMDb. [https://www.imdb.com/title/tt1790809/?ref\\_=nv\\_sr\\_srgs\\_3\\_tt\\_8\\_nm\\_0\\_q\\_pirates](https://www.imdb.com/title/tt1790809/?ref_=nv_sr_srgs_3_tt_8_nm_0_q_pirates)

IMDb.com. (1996, May 22). Mission: Impossible. IMDb. [https://www.imdb.com/title/tt0117060/?ref\\_=nv\\_sr\\_srgs\\_0\\_tt\\_8\\_nm\\_0\\_q\\_mission](https://www.imdb.com/title/tt0117060/?ref_=nv_sr_srgs_0_tt_8_nm_0_q_mission)

IMDb.com. (2000). Mission: Impossible II. IMDb. [https://www.imdb.com/title/tt0120755/?ref\\_=tt\\_sims\\_tt\\_t\\_1](https://www.imdb.com/title/tt0120755/?ref_=tt_sims_tt_t_1)

IMDb.com. (2006). Mission: Impossible III. IMDb. [https://www.imdb.com/title/tt0317919/?ref\\_=tt\\_sims\\_tt\\_t\\_2](https://www.imdb.com/title/tt0317919/?ref_=tt_sims_tt_t_2)

IMDb.com. (2011). Mission: Impossible - Ghost Protocol. IMDb. [https://www.imdb.com/title/tt1229238/?ref\\_=tt\\_sims\\_tt\\_t\\_3](https://www.imdb.com/title/tt1229238/?ref_=tt_sims_tt_t_3)

IMDb.com. (2015). Mission: Impossible - Rogue Nation. IMDb. [https://www.imdb.com/title/tt2381249/?ref\\_=tt\\_sims\\_tt\\_t\\_4](https://www.imdb.com/title/tt2381249/?ref_=tt_sims_tt_t_4)

At the conclusion of Task 3, the document-term matrix is printed as a data frame.

```
as.data.frame(as.matrix(dtms))
```

##	also	box	color	compani	connect	countri	critic	date	featur	help
## fast01.txt	1	3	2	2	1	1	1	2	1	1
## fast02.txt	2	3	2	2	1	1	1	1	3	3
## fast03.txt	1	3	2	2	1	1	1	1	4	1
## fast04.txt	1	4	2	2	2	1	1	1	2	1
## fast05.txt	2	3	2	2	1	2	2	1	2	3
## impossible01.txt	2	3	2	2	1	1	1	1	1	2
## impossible02.txt	1	3	2	3	1	1	1	1	1	1
## impossible03.txt	2	3	2	2	1	1	1	1	2	1
## impossible04.txt	3	3	2	2	1	1	1	2	2	2
## impossible05.txt	2	3	2	2	1	1	1	1	2	1
## linux01.txt	39	0	1	6	1	1	4	0	18	4
## linux02.txt	24	2	0	2	1	2	2	4	10	3
## linux03.txt	48	1	1	25	3	16	3	6	29	4
## linux04.txt	15	1	2	2	2	1	3	2	16	1
## linux05.txt	11	2	2	11	0	0	0	1	15	0
## pirates01.txt	2	3	2	2	1	1	1	1	3	2
## pirates02.txt	1	3	2	2	1	1	1	1	2	1
## pirates03.txt	3	3	2	2	1	1	1	1	2	1

```

## pirates04.txt      4   3   2   3   1   1   1   1   2   2
## pirates05.txt      1   3   2   2   1   1   1   2   2   1
##
##      known languag like offic offici one open origin play popular
## fast01.txt         1     1   1   3     2   3   1     3   1     1
## fast02.txt         1     1   3   3     2   2   1     1   1     1
## fast03.txt         1     1   2   3     2   1   1     2   1     1
## fast04.txt         1     1   3   5     2   7   1     2   1     1
## fast05.txt         1     1   2   3     2   3   2     3   4     1
## impossible01.txt   1     1   3   3     3   4   2     1   1     1
## impossible02.txt   1     1   2   3     1   4   3     3   1     2
## impossible03.txt   2     1   1   3     0   6   1     1   2     1
## impossible04.txt   1     1   5   3     1   3   4     1   1     1
## impossible05.txt   2     1   1   3     2   4   8     1   3     1
## linux01.txt        8     9  19   0     7  17   8     9   1     0
## linux02.txt        4     1   6   3    12  11  13     2   1     4
## linux03.txt       17     7  16   4    15  22  51     9  50    15
## linux04.txt        7     1   2   1    20   4  11     1   7     3
## linux05.txt        5     1   0   5     8   9  19     4   3     5
## pirates01.txt      1     1   4   3     1   6   2     1   1     1
## pirates02.txt      1     1   2   3     1   0   1     3   1     1
## pirates03.txt      1     1   1   3     1   2   1     1   1     2
## pirates04.txt      1     1   1   3     1   2   1     1   1     1
## pirates05.txt      1     1   2   3     4   2   3     3   1     1
##
##      product releas review see sign technic top unit user video
## fast01.txt         3     3     4   5   1     1   2   2   2     1
## fast02.txt         4     1     4   5   3     1   2   2   2     1
## fast03.txt         3     1     4   4   1     1   2   2   2     3
## fast04.txt         5     1     4   5   1     1   3   3   2     2
## fast05.txt         3     1     4   5   1     1   2   2   2     3
## impossible01.txt   4     2     4   4   1     1   4   2   2     6
## impossible02.txt   4     1     4   3   2     1   4   2   2     3
## impossible03.txt   4     1     4   3   1     1   3   4   2     4
## impossible04.txt   4     1     4   3   1     1   2   4   2     2
## impossible05.txt   3     1     4   3   1     1   3   2   2     2
## linux01.txt        6    62     3   8   1     8   5   2  36     2
## linux02.txt        3    57     1   5   1     1   7   1  20     2
## linux03.txt        9    45     4  11   1     3  11  10  73     5
## linux04.txt        0    18     3   1   1     1   1   0  15     2
## linux05.txt        1    61     0   3   0     1   0   1  13     0
## pirates01.txt      3     1     4   5   1     1   4   2   2     3
## pirates02.txt      4     1     4   6   1     1   2   2   2     2
## pirates03.txt      4     2     4   4   1     1   2   2   2     2
## pirates04.txt      3     1     4   3   1     1   2   3   2     2
## pirates05.txt      3     3     4   7   1     1   5   2   2     3

```

Confusion matrices utilised in Task 4 to quantify clustering.

```
table(Topic = doc_names_short, Cluster = cutree(fit_euclid, k = 10))
```

```

##      Cluster
## Topic      1 2 3 4 5 6 7 8 9 10
## fast       2 1 1 1 0 0 0 0 0 0
## impossible 2 0 0 0 3 0 0 0 0 0
## linux      0 0 0 0 0 1 1 1 1 1
## pirates    3 0 0 2 0 0 0 0 0 0

```



```
table(Topic = doc_names_short, Cluster = cutree(fit_cos, k = 10))
```

```
##           Cluster
## Topic      1 2 3 4 5 6 7 8 9 10
## fast       4 1 0 0 0 0 0 0 0 0
## impossible 1 0 2 1 1 0 0 0 0 0
## linux      1 0 0 0 0 1 1 1 1 0
## pirates    2 1 0 0 0 0 0 0 0 2
```

Cluster assignment based on Euclidean distance for every topic, as of Task 4.

- 1 fast
- 2 impossible
- 3 linux
- 4 pirates

Task 4 assigns a cosine distance cluster to each topic.

- 1 fast
- 2 impossible
- 3 linux
- 4 pirates

Results for the abstractions network for Task 5 in terms of proximity, betweenness, eigenvector centralities, and degree measures.

```
sort(-closeness(abs_net))
```

```
##      linux05.txt      linux01.txt      linux04.txt impossible03.txt
##      -0.002564103      -0.002083333      -0.001937984      -0.001872659
##      linux02.txt      pirates02.txt      fast01.txt      fast02.txt
##      -0.001872659      -0.001872659      -0.001811594      -0.001811594
##      fast03.txt      fast04.txt      fast05.txt impossible01.txt
##      -0.001811594      -0.001811594      -0.001811594      -0.001811594
## impossible02.txt impossible04.txt impossible05.txt      linux03.txt
##      -0.001811594      -0.001811594      -0.001811594      -0.001811594
##      pirates01.txt      pirates03.txt      pirates04.txt      pirates05.txt
##      -0.001811594      -0.001811594      -0.001811594      -0.001811594
```

```
sort(-betweenness(abs_net))
```

```
##      fast01.txt      fast02.txt      fast03.txt      fast04.txt
##      0              0              0              0
##      fast05.txt impossible01.txt impossible02.txt impossible03.txt
##      0              0              0              0
## impossible04.txt impossible05.txt      linux01.txt      linux02.txt
##      0              0              0              0
##      linux03.txt      linux04.txt      linux05.txt      pirates01.txt
##      0              0              0              0
##      pirates02.txt      pirates03.txt      pirates04.txt      pirates05.txt
##      0              0              0              0
```

```
sort(evcent(abs_net)$vector)
```

```
##      linux05.txt      linux01.txt      linux04.txt impossible03.txt
##      0.7148700      0.8755370      0.9382096      0.9692139
##      pirates02.txt      linux02.txt      linux03.txt      fast05.txt
##      0.9692139      0.9692139      1.0000000      1.0000000
```

```
## impossible04.txt    pirates04.txt    fast02.txt    fast03.txt
##      1.0000000      1.0000000      1.0000000      1.0000000
##      fast04.txt impossible02.txt impossible05.txt    pirates01.txt
##      1.0000000      1.0000000      1.0000000      1.0000000
##      pirates03.txt    pirates05.txt    fast01.txt impossible01.txt
##      1.0000000      1.0000000      1.0000000      1.0000000
```

```
sort(degree(abs_net))
```

```
##      fast01.txt    fast02.txt    fast03.txt    fast04.txt
##      19          19          19          19
##      fast05.txt impossible01.txt impossible02.txt impossible03.txt
##      19          19          19          19
## impossible04.txt impossible05.txt    linux01.txt    linux02.txt
##      19          19          19          19
##      linux03.txt    linux04.txt    linux05.txt    pirates01.txt
##      19          19          19          19
##      pirates02.txt    pirates03.txt    pirates04.txt    pirates05.txt
##      19          19          19          19
```

Results for the tokens matrix for Task 6 in terms of proximity, betweenness, eigenvector centralities, and degree measures.

```
sort(-closeness(tok_net))
```

```
##      color    offici    one    product    unit    box
## -0.001872659 -0.001872659 -0.001872659 -0.001869159 -0.001869159 -0.001862197
##      date    offic    popular    connect    countri    critic
## -0.001862197 -0.001862197 -0.001862197 -0.001845018 -0.001845018 -0.001845018
##      help    like    review    sign    top    video
## -0.001845018 -0.001845018 -0.001845018 -0.001845018 -0.001845018 -0.001845018
##      also    compani    featur    known    languag    open
## -0.001779359 -0.001779359 -0.001779359 -0.001779359 -0.001779359 -0.001779359
##      origin    play    releas    see    technic    user
## -0.001779359 -0.001779359 -0.001779359 -0.001779359 -0.001779359 -0.001779359
```

```
sort(-betweenness(tok_net))
```

```
##      also    box    color    compani    connect    countri    critic    date    featur    help
##      0      0      0      0      0      0      0      0      0      0
##      known    languag    like    offic    offici    one    open    origin    play    popular
##      0      0      0      0      0      0      0      0      0      0
## product    releas    review    see    sign    technic    top    unit    user    video
##      0      0      0      0      0      0      0      0      0      0
```

```
sort(evcent(tok_net)$vector)
```

```
##      one    color    offici    product    unit    box    popular    offic
## 0.9518295 0.9518295 0.9518295 0.9535132 0.9535132 0.9568986 0.9568986 0.9568986
##      date    critic    like    sign    video    help    connect    countri
## 0.9568986 0.9654682 0.9654682 0.9654682 0.9654682 0.9654682 0.9654682 0.9654682
##      review    top    technic    compani    user    also    origin    play
## 0.9654682 0.9654682 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
##      see    featur    known    languag    open    releas
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
```

```
sort(degree(tok_net))
```

##	also	box	color	compani	connect	countri	critic	date	featur	help
##	29	29	29	29	29	29	29	29	29	29
##	known	languag	like	offic	offici	one	open	origin	play	popular
##	29	29	29	29	29	29	29	29	29	29
##	product	releas	review	see	sign	technic	top	unit	user	video
##	29	29	29	29	29	29	29	29	29	29

Vertex degrees of Task 7's bipartite graph.

```
sort(degree(bipart))
```

##	box	color	connect	countri
##	19	19	19	19
##	critic	date	help	like
##	19	19	19	19
##	offic	offici	one	popular
##	19	19	19	19
##	product	review	sign	top
##	19	19	19	19
##	unit	video	also	compani
##	19	19	20	20
##	featur	known	languag	open
##	20	20	20	20
##	origin	play	releas	see
##	20	20	20	20
##	technic	user	linux05.txt	linux01.txt
##	20	20	21	26
##	linux04.txt	impossible03.txt	linux02.txt	pirates02.txt
##	28	29	29	29
##	fast01.txt	fast02.txt	fast03.txt	fast04.txt
##	30	30	30	30
##	fast05.txt	impossible01.txt	impossible02.txt	impossible04.txt
##	30	30	30	30
##	impossible05.txt	linux03.txt	pirates01.txt	pirates03.txt
##	30	30	30	30
##	pirates04.txt	pirates05.txt		
##	30	30		