NLP For Detection Of A Written Review Medium

Simeon Abrecht

2024-02-17

clear environment

```
rm(list = ls())
```

set random seed based on student ID:

```
set.seed(31931839)
```

Install relevant libraries:

```
library(slam)
library(tm)
library(SnowballC)
library(lsa)
library(caret)
library(igraph)
library(igraphdata)
```

For gathering data I have opted for copying text from webpages online. I have 3 separate document styles/topics. Them being Album Reviews, Movie Reviews and Restauraunt Reviews. The general idea behind this choice is that my assumption is that reviewers will use similar language, so trying to draw a distinction between them using text analysis will be an interesting task, as it will rely solely on key words. Also, as a side note all of the reviews are of a similar length, thus in theory no group of tokens should be disproportionately boosted because their respective document/s have a much higher word count.

In terms of gathering the data I have copied and pasted the text from the webpages/pdf documents into plain text documents. This in theory should speed up the pre processing requirement.

NOTE ON DDOCUMENT TITLE STRUCTURE: Each text document is formatted in the same way. ie;

title_reporter_topic. Title: The name of the respective restaurant, film or album.

Reporter: The news publication that has written the review. This is then abreviated into it's respective accronym . ie; Sydney Morning Herald is SMH.

Topic: The general type of review. This is also abbreviated. ie; Movie = M, Restaurant = R and Album = A.

Creating a corpus from the sourced documents:

```
cname = file.path("Text Documents")
docs = Corpus(DirSource(cname))
summary(docs)
```

```
##                             Length Class             Mode
## Bonny_SMH_R.txt             2      PlainTextDocument list
## DarkSideOfTheMoon_G_A.txt   2      PlainTextDocument list
## DemonDays_G_A.txt           2      PlainTextDocument list
## Fen_SMH_R.txt               2      PlainTextDocument list
## Harvie_T_R.txt              2      PlainTextDocument list
## Interstallar_HR_M.txt       2      PlainTextDocument list
## Interstellar_RE_M.txt       2      PlainTextDocument list
## Madvillainy_P_Album.txt     2      PlainTextDocument list
## Memento_RE_M.txt            2      PlainTextDocument list
## mmfood_P_A.txt              2      PlainTextDocument list
## Pirate Radio_RE _M.txt      2      PlainTextDocument list
## PirateRadio_HR_M.txt        2      PlainTextDocument list
## Soi38_TO_R.txt              2      PlainTextDocument list
## SourSoul_G_A.txt            2      PlainTextDocument list
## TheIrishman_HR_M.txt        2      PlainTextDocument list
## TheLastJar_T_R.txt          2      PlainTextDocument list
## Wilds_P_A.txt               2      PlainTextDocument list
```

Tokenisation:

```
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, removePunctuation)
docs <- tm_map(docs, content_transformer(tolower))
```

Filter words: Removing stop words, white space and all charachters that aren't alphabetical or numerical

```
docs <- tm_map(docs, stripWhitespace)
docs <- tm_map(docs, removeWords, stopwords("english"))
removeSpecialChars <- function(x) gsub("[^a-zA-Z0-9 ]","",x)
docs <- tm_map(docs, removeSpecialChars)
#the filter below is manual from words found in the end product that i didn't deem relevant. This was f
docs <- tm_map(docs, removeWords, c("can","one", "way", "back", "even", "also", "will", "frank", "just"
```

Stemming:

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

Creating initial document term matrix (DTM):

```
dtm <- DocumentTermMatrix(docs)
dtm = as.data.frame(as.matrix(dtm))
write.csv(dtm, "dtm.csv")
```

Removing sparse terms from Document Term Matrix so that it has around 20 tokens: The 0.47 figure used in the remove sparse terms function is manually tuned to help include around 20 tokens

```
dtm <- DocumentTermMatrix(docs)
freq = colSums(as.matrix(dtm))
dtms <- removeSparseTerms(dtm, 0.47)
```

Note that i have chosen to leave in tokens like 'like' and 'well' as they multiple meanings, they can be connective in grammar or be used to describe the quality of something. I think this is particularly important when for the analysis of reviews.

Analysing the remaining tokens:

```
inspect(dtms)
```

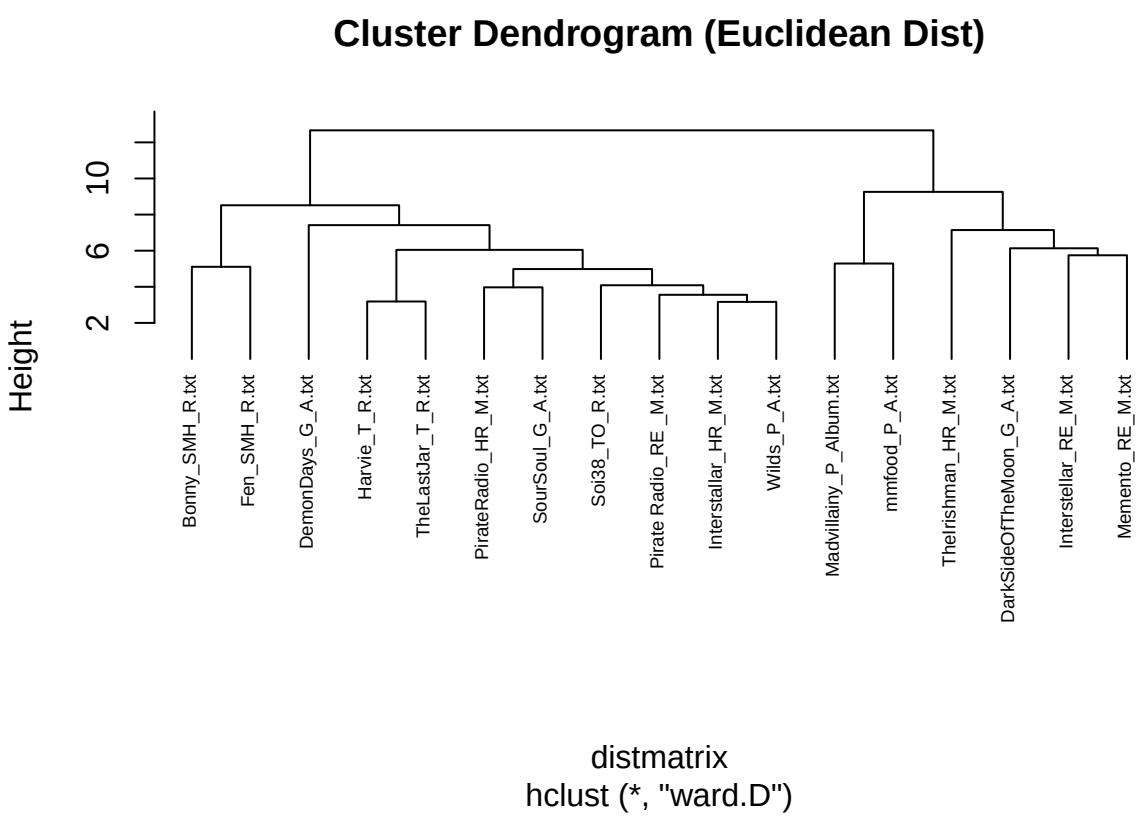
```
## <<DocumentTermMatrix (documents: 17, terms: 19)>>
## Non-/sparse entries: 211/112
## Sparsity           : 35%
## Maximal term length: 5
## Weighting          : term frequency (tf)
## Sample             :
##                          Terms
## Docs                      day first good like make new play time work year
##   Bonny_SMH_R.txt           0     0    7    2    4   0    0    0    3    2
##   DarkSideOfTheMoon_G_A.txt 1     1    0    1    2   1    1    3    3    4
##   DemonDays_G_A.txt         7     3    1    4    0   1    1    2    0    1
##   Interstallar_HR_M.txt     1     0    1    1    2   1    2    3    0    2
##   Interstellar_RE_M.txt     1     4    3    3    0   2    5    6    2    2
##   Madvillainy_P_Album.txt   3     2    1    8    5   2    3    0    4    2
##   Memento_RE_M.txt          0     1    2    2    3   0    5    7    3    0
##   mmfood_P_A.txt            0     2    5    7    5   2    1    2    1    3
##   PirateRadio_HR_M.txt      1     1    0    4    1   1    3    1    0    0
##   TheIrishman_HR_M.txt      3     5    1    0    4   4    5    5    3    3
```

Saving the dtms as csv:

```
dtms = as.matrix(dtms)
write.csv(dtms, "dtms.csv")
```

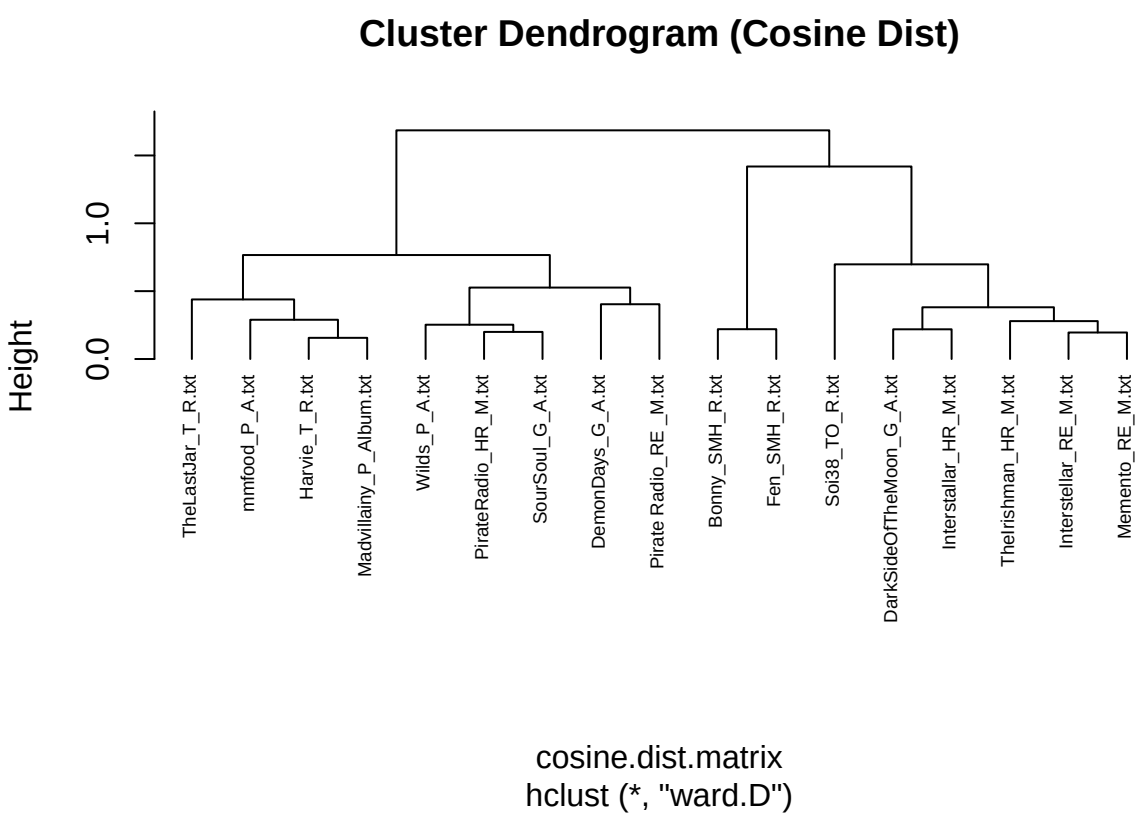
Creating a hierarchical cluster model based off of euclidean distance

```
distmatrix = dist(scale(dtms)) #euclidean distance
fit = hclust(distmatrix, method = "ward.D")
plot(fit, main = 'Cluster Dendrogram (Euclidean Dist)', cex = 0.6, hang = -1)
```



Creating a Heirarchichal cluster model based off of the cosine distance between documents:

```
dtms = as.matrix(dtms)
cosine.dist.matrix = cosine(t(dtms)) #finding cosine similarity
cosine.dist.matrix = 1 - cosine.dist.matrix #finding cosine distance
cosine.dist.matrix = as.dist(cosine.dist.matrix)
cosine.fit = hclust(cosine.dist.matrix, method = "ward.D")
plot(cosine.fit, main = 'Cluster Dendrogram (Cosine Dist)', cex = 0.6, hang = -1)
```



it appears as though the euclidean distance clustering was more effective

Quantitative analysis of the euclidean distance based cluster model:

```
euclid.groups = cutree(fit, k =3)
euclid.groups = (euclid.groups)
dtms
```

```
##                            Terms
## Docs                        find good like love make take well work year best
##   Bonny_SMH_R.txt              1    7    2    3    4    1    1    3    2    0
##   DarkSideOfTheMoon_G_A.txt    1    0    1    0    2    1    3    3    4    2
##   DemonDays_G_A.txt            0    1    4    0    0    1    0    0    1    0
##   Fen_SMH_R.txt                0    7    0    0    2    0    1    1    0    0
##   Harvie_T_R.txt               0    0    5    0    1    1    0    1    0    1
##   Interstallar_HR_M.txt        1    1    1    0    2    1    1    0    2    1
##   Interstellar_RE_M.txt        2    3    3    3    0    0    4    2    2    1
##   Madvillainy_P_Album.txt      1    1    8    0    5    3    1    4    2    2
##   Memento_RE_M.txt             1    2    2    1    3    1    1    3    0    0
##   mmfood_P_A.txt               1    5    7    1    5    4    1    1    3    2
##   Pirate Radio_RE _M.txt       1    1    1    1    0    0    2    2    1    1
##   PirateRadio_HR_M.txt         1    0    4    1    1    1    1    0    0    3
##   Soi38_TO_R.txt               1    0    0    1    0    0    1    0    0    0
##   SourSoul_G_A.txt             0    1    3    1    0    0    0    0    1    0
##   TheIrishman_HR_M.txt         0    1    0    2    4    2    0    3    3    3
##   TheLastJar_T_R.txt           0    2    2    0    0    2    0    0    0    3
##   Wilds_P_A.txt                1    0    3    1    0    1    1    1    1    0
```

```
##                            Terms
## Docs                        day first music new old play time name dont
##   Bonny_SMH_R.txt             0     0     0   0   0    0    0    0    0
##   DarkSideOfTheMoon_G_A.txt   1     1     3   1   1    1    3    0    0
##   DemonDays_G_A.txt           7     3     4   1   1    1    2    2    0
##   Fen_SMH_R.txt               1     0     0   3   0    0    0    0    0
##   Harvie_T_R.txt              1     0     0   0   0    0    0    0    1
##   Interstallar_HR_M.txt       1     0     1   1   1    2    3    1    1
##   Interstellar_RE_M.txt       1     4     1   2   2    5    6    0    0
##   Madvillainy_P_Album.txt     3     2     1   2   1    3    0    1    3
##   Memento_RE_M.txt            0     1     0   0   1    5    7    1    1
##   mmfood_P_A.txt              0     2     1   2   4    1    2    3    2
##   Pirate Radio_RE _M.txt      3     0     1   0   2    2    0    1    2
##   PirateRadio_HR_M.txt        1     1     2   1   2    3    1    3    0
##   Soi38_TO_R.txt              0     1     0   0   0    0    1    0    1
##   SourSoul_G_A.txt            1     0     1   1   2    1    1    2    0
##   TheIrishman_HR_M.txt        3     5     0   4   1    5    5    0    1
##   TheLastJar_T_R.txt          1     1     1   0   0    0    0    1    2
##   Wilds_P_A.txt               0     0     1   2   1    3    0    2    2
```

```
topics = c("Restaurant","Album","Album","Restaurant","Restaurant","Movie","Movie","Album","Movie","Albu
euclid.clust.table = table(GroupNames = topics, Clusters = euclid.groups)
euclid.clust.table
```

```
##             Clusters
## GroupNames   1 2 3
##   Album      3 1 2
##   Movie      3 3 0
##   Restaurant 5 0 0
```

Quantitative analysis of the cosine distance based cluster model:

```
cosine.groups = cutree(cosine.fit, k =3)
cosine.groups = (cosine.groups)
dtms
```

```
##                            Terms
## Docs                        find good like love make take well work year best
##   Bonny_SMH_R.txt              1    7    2    3    4    1    1    3    2    0
##   DarkSideOfTheMoon_G_A.txt    1    0    1    0    2    1    3    3    4    2
##   DemonDays_G_A.txt            0    1    4    0    0    1    0    0    1    0
##   Fen_SMH_R.txt                0    7    0    0    2    0    1    1    0    0
##   Harvie_T_R.txt               0    0    5    0    1    1    0    1    0    1
##   Interstallar_HR_M.txt        1    1    1    0    2    1    1    0    2    1
##   Interstellar_RE_M.txt        2    3    3    3    0    0    4    2    2    1
##   Madvillainy_P_Album.txt      1    1    8    0    5    3    1    4    2    2
##   Memento_RE_M.txt             1    2    2    1    3    1    1    3    0    0
##   mmfood_P_A.txt               1    5    7    1    5    4    1    1    3    2
##   Pirate Radio_RE _M.txt       1    1    1    1    0    0    2    2    1    1
##   PirateRadio_HR_M.txt         1    0    4    1    1    1    1    0    0    3
##   Soi38_TO_R.txt               1    0    0    1    0    0    1    0    0    0
##   SourSoul_G_A.txt             0    1    3    1    0    0    0    0    1    0
##   TheIrishman_HR_M.txt         0    1    0    2    4    2    0    3    3    3
```

```
##   TheLastJar_T_R.txt           0    2    2    0    0    2    0    0    0    3
##   Wilds_P_A.txt                1    0    3    1    0    1    1    1    1    0
##                             Terms
## Docs                         day first music new old play time name dont
##   Bonny_SMH_R.txt              0     0     0   0   0    0    0    0    0
##   DarkSideOfTheMoon_G_A.txt    1     1     3   1   1    1    3    0    0
##   DemonDays_G_A.txt            7     3     4   1   1    1    2    2    0
##   Fen_SMH_R.txt                1     0     0   3   0    0    0    0    0
##   Harvie_T_R.txt               1     0     0   0   0    0    0    0    1
##   Interstallar_HR_M.txt        1     0     1   1   1    2    3    1    1
##   Interstellar_RE_M.txt        1     4     1   2   2    5    6    0    0
##   Madvillainy_P_Album.txt      3     2     1   2   1    3    0    1    3
##   Memento_RE_M.txt             0     1     0   0   1    5    7    1    1
##   mmfood_P_A.txt               0     2     1   2   4    1    2    3    2
##   Pirate Radio_RE _M.txt       3     0     1   0   2    2    0    1    2
##   PirateRadio_HR_M.txt         1     1     2   1   2    3    1    3    0
##   Soi38_TO_R.txt               0     1     0   0   0    0    1    0    1
##   SourSoul_G_A.txt             1     0     1   1   2    1    1    2    0
##   TheIrishman_HR_M.txt         3     5     0   4   1    5    5    0    1
##   TheLastJar_T_R.txt           1     1     1   0   0    0    0    1    2
##   Wilds_P_A.txt                0     0     1   2   1    3    0    2    2
```

```
topics = c("Restaurant","Album","Album","Restaurant","Restaurant","Movie","Movie","Album","Movie","Albu
cosine.clust.table = table(GroupNames = topics, Clusters = cosine.groups)
cosine.clust.table
```

```
##             Clusters
## GroupNames   1 2 3
##   Album      0 1 5
##   Movie      0 4 2
##   Restaurant 2 1 2
```

From the initial inspection of the euclidean distance based clustering and the cosine distance based clustering. we see that the it is likely that the clusters 1, 2 and 3 are restaurants, Movies and Album respectively.

Using that information we can now calculate the accuracy of the two models.

function to determine clustering accuracy:

```
classification_accuracy = function(df){
  return ((df[1,3] + df[2,2] + df[3,1])/17)
}
```

Euclidean distance based clustering accuracy:

```
classification_accuracy(euclid.clust.table)
```

```
## [1] 0.5882353
```

cosine distance based clustering accuracy:

```
classification_accuracy(cosine.clust.table)
```

```
## [1] 0.6470588
```

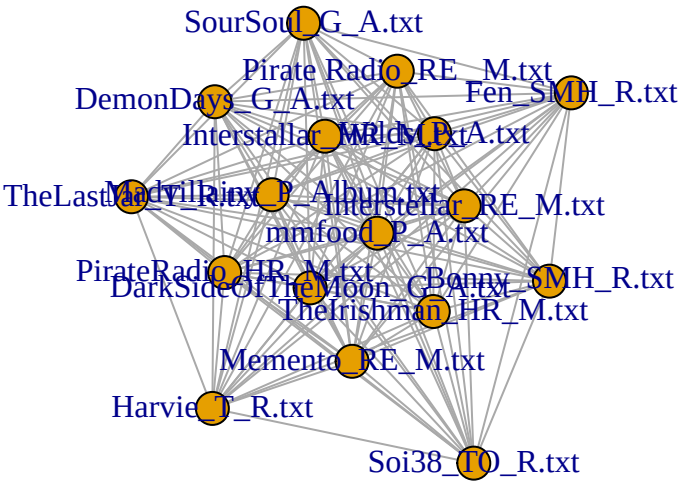
As predicted it appears that the cosine distance based clustering is more effective than the euclidean distance based clustering

Abstract matrix:

```
dtms.adj = as.matrix(dtms)
dtms.adj = as.matrix((dtms.adj > 0) + 0)
Abs.matrix = dtms.adj%*%t(dtms.adj)
Abs.matrix = Abs.matrix
diag(Abs.matrix ) = 0
```

creating a plot for the abstract matrix:

```
g.abs = graph_from_adjacency_matrix(Abs.matrix, mode = 'undirected', weighted = TRUE)
plot(g.abs, layout = layout.fruchterman.reingold)
```



We now have a naive network plot but it need to be improved.

Constructing a table representing the values of importance in the graph structure:

```
abs.closesness = as.table(closeness(g.abs))
abs.betweenness = as.table(betweenness(g.abs))
abs.degree = as.table(degree(g.abs))
abs.eigen = as.table(evcent(g.abs)$vector)

abstract.matrix = rbind(abs.closesness, abs.betweenness, abs.degree, abs.eigen)

abstract.matrix = rbind(abstract.matrix, colSums(abstract.matrix))

abstract.matrix = apply(abstract.matrix, 2, round)

rownames(abstract.matrix) = c('closeness','Betweenness','Degree','Eigen','Sum')
abstract.matrix = as.table(abstract.matrix)
```

unfortunately it seems that most of the data seems to have similar values. This is likely due to similar styles of documents used and frequency of the key words used. It may also have something to do with document size. all documents in the corpus, though still larger than the required size are still smaller than I would of liked. Hindsight is a powerful tool.

In order to gain a better visual insight into what my data is doing i have chosen to employ the following method.

improved abstract graph:

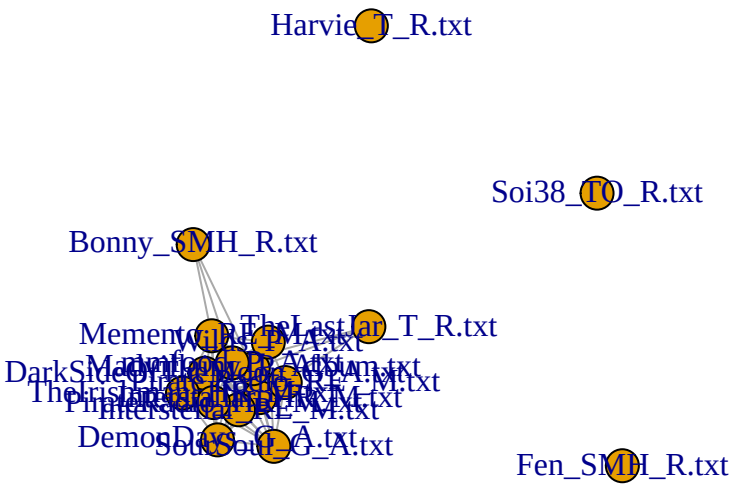
My approach here is to limit the amount of connectivity between the nodes in the graph by "Normalising" the Abstract matrix. To do this I have subtracted the average value in the matrix from every value in the matrix, then set the negative values as 0. This in theory should lead to a more sparse graph and in general give us some better insights into centrality measures.

manipulating the abstract matrix:

```
mean_vals = as.data.frame(apply(Abs.matrix, 2, mean))
mean_val = round(as.numeric(apply(mean_vals, 2, mean)))

Abs.matrix2 = Abs.matrix - mean_val
Abs.matrix2[Abs.matrix2 < 0] = 0
diag(Abs.matrix2) = 0
g.abs2 = graph_from_adjacency_matrix(Abs.matrix2, mode = 'undirected', weighted = TRUE)
```

naive plot of the new abstract matrix

```
plot(g.abs2, layout = layout.fruchterman.reingold)
```



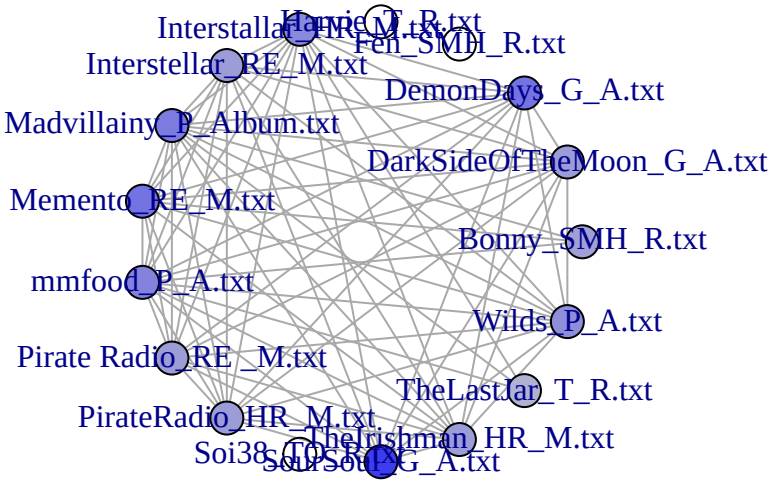
It looks a bit messy here but we can fix this

Finding centrality measures of the new matrix:

```
abs.closesness2 = as.table(closeness(g.abs2))
abs.betweenness2 = as.table(betweenness(g.abs2))
abs.degree2 = as.table(degree(g.abs2))
abs.eigen2 = as.table(evcent(g.abs2)$vector)

abstract.matrix2 = rbind(abs.closesness2, abs.betweenness2, abs.degree2, abs.eigen2)

abstract.matrix2 = rbind(abstract.matrix2, colSums(abstract.matrix2))

abstract.matrix2 = apply(abstract.matrix2, 2, round)

rownames(abstract.matrix2) = c('closeness','Betweenness','Degree','Eigen','Sum')
abstract.matrix2 = as.table(abstract.matrix2)
```

We can now apply these insights to help our initial plot.

```
V(g.abs2)$value = colSums(abstract.matrix2)
V(g.abs2)$col_values <- round( V(g.abs2)$value, 2)
colours <- colorRampPalette(c("gray80", "blue"))(100)
V(g.abs2)$color = colours[V(g.abs2)$col_values]
plot(g.abs2, layout = layout.circle)
```



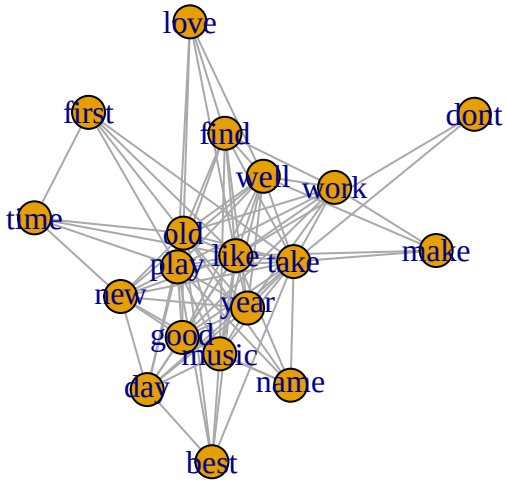
Here we have gained a visual insight into how central each document is in the matrix. With darker blues corresponding to an overall greater centrlaity meaure. The graph also helps in seeing the relative degree of all documents by analysing their incoming edges.

creating a token matrix using the same methodology as we used previously to help get better insights:

```
token.matrix = t(dtms.adj)%*%dtms.adj


token_mean_vals = as.data.frame(apply(token.matrix, 2, mean))
token_mean_val = round(as.numeric(apply(token_mean_vals, 2, mean)))

token.matrix = token.matrix - mean_val
token.matrix[token.matrix < 0] = 0
diag(token.matrix) = 0
g.token = graph_from_adjacency_matrix(token.matrix, mode = 'undirected', weighted = TRUE)
```

naive graph:

```
plot(g.token, layout = layout.fruchterman.reingold)
```

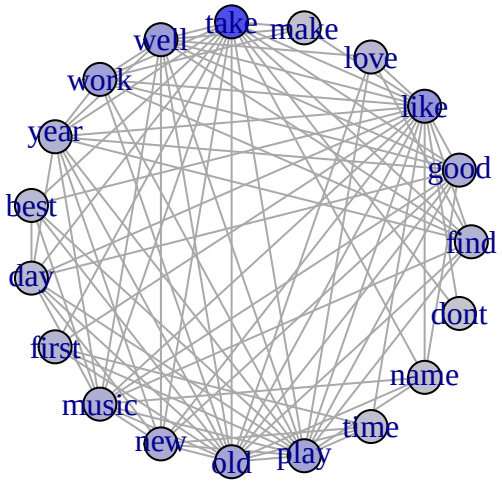


finding centrality measures of the token matrix:

```
token.closesness = as.table(closeness(g.token))
token.betweenness = as.table(betweenness(g.token))
token.degree = as.table(degree(g.token))
token.eigen = as.table(evcent(g.token)$vector)

token.cent.vals = rbind(token.closesness, token.betweenness, token.degree, token.eigen)

token.cent.vals = rbind(token.cent.vals, colSums(token.cent.vals))

token.cent.vals = apply(token.cent.vals, 2, round)

rownames(token.cent.vals) = c('closeness','Betweenness','Degree','Eigen','Sum')
```

new and imporved plot:

```
V(g.token)$value = token.cent.vals[5,]
V(g.token)$col_values <- round( V(g.token)$value, 2)
colours <- colorRampPalette(c("gray80", "blue"))(100)
V(g.token)$color = colours[V(g.token)$col_values]
plot(g.token, layout = layout.circle)
```



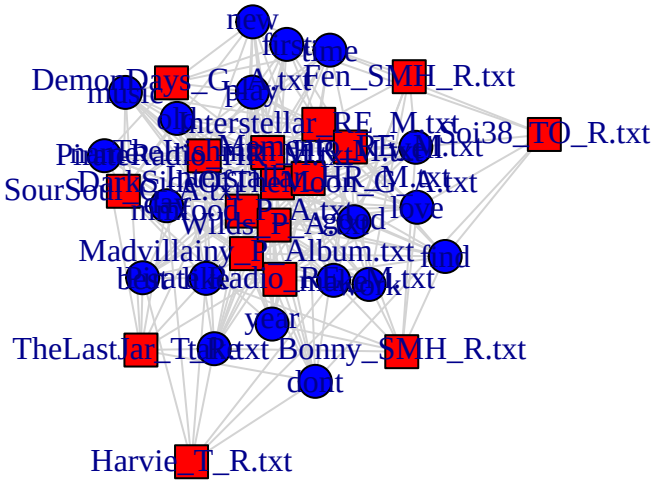
This plot, for simplicity, follows the same convention as the document based graph. ie; same colour scheme, layout etc.

constructing a bipartite matrix:

```
dtms.b = as.data.frame(dtms)
dtms.b$ABS = rownames(dtms.b)

dtms.c = data.frame()
  for (i in 1:nrow(dtms.b)){
    for (j in 1:(ncol(dtms.b)-1)){
        touse = cbind(dtms.b[i,j], dtms.b[i,ncol(dtms.b)],
        colnames(dtms.b[j]))
        dtms.c = rbind(dtms.c, touse ) } } # close loops

colnames(dtms.c) = c("weight", "abs", "token")
dtms.d = dtms.c[dtms.c$weight != 0,] # delete 0 weights
dtms.d = dtms.d[,c(2,3,1)]
```

Constructing the Bipartite graph from previously created matrix:

```
b.g = graph.data.frame(dtms.d, directed = FALSE)
V(b.g)$type = bipartite_mapping(b.g)$type
V(b.g)$color = ifelse(V(b.g)$type, "blue", "red")
V(b.g)$shape = ifelse(V(b.g)$type, "circle", "square")
E(b.g)$color = "lightgray"
plot(b.g)
```



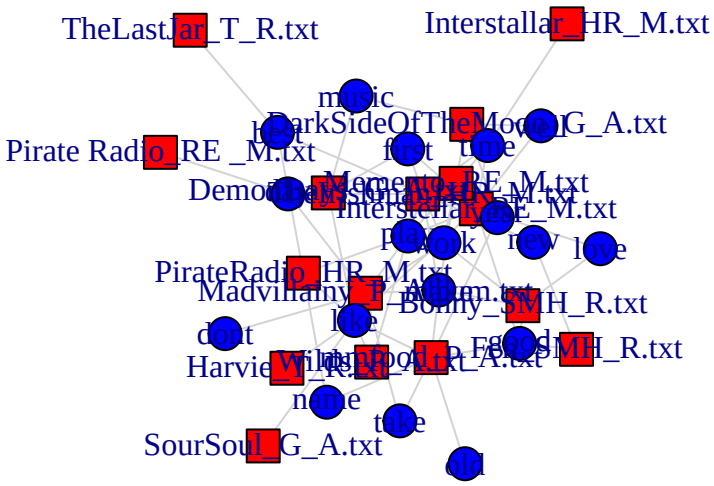
This is the simple graph but naturally, it leaves a lot to be desired. First of all it's very cluttered and i feel like it's not particularly readable. In order to remedy this I will use a similar method i used previously but this time i will scale the weights value in the bipartite matrix, to remove values until only those with the largest weighting are available

simplifying the matrix:

```
avg.weight = mean(as.numeric(dtms.d$weight))
avg.weight = round(avg.weight)
dtms.e = dtms.d
dtms.e$weight = as.numeric(dtms.e$weight) - avg.weight
dtms.e = dtms.e[dtms.e$weight > 0,]
```

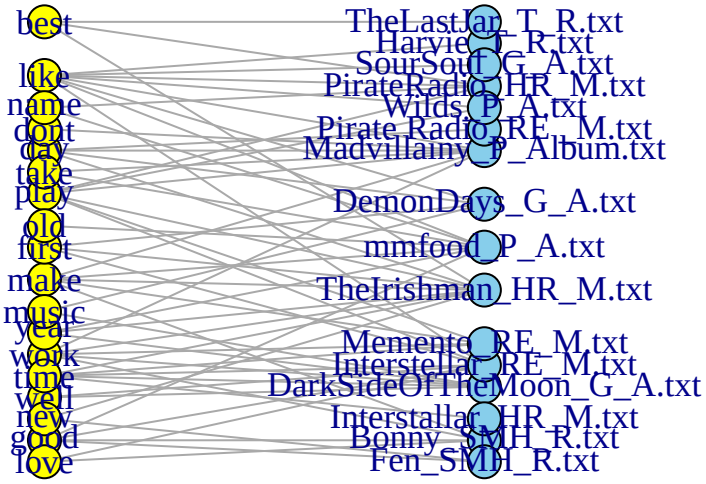
naive plot of scaled values:

```
b.g2 = graph.data.frame(dtms.e, directed = FALSE)
V(b.g2)$type = bipartite_mapping(b.g2)$type
V(b.g2)$color = ifelse(V(b.g2)$type, "blue", "red")
V(b.g2)$shape = ifelse(V(b.g2)$type, "circle", "square")
E(b.g2)$color = "lightgray"
plot(b.g2)
```



Using this method we can see that the plot has become more sparse but work is still needed to turn it into a truly good visual aid.

```
column_lo = layout_as_bipartite(b.g2, hgap = , maxiter = 1000)
column_lo = column_lo[,c(2,1)]
V(b.g2)$color = ifelse(V(b.g2)$type, "yellow", "skyblue")
V(b.g2)$shape = ifelse(V(b.g2)$type, "circle", "circle")
E(b.g2)$color = "dark gray"

plot(b.g2, layout = column_lo, vertex.label.cex = 1.1)
```



Using the above visualisation we can clearly see the separation between the tokens and the documents as well as the number of edges coming from each node. The use of shape seemed unnecessary as the two columns are separated based on type.