Document Clustering with R

Motivation

Analyze the the underlying structure of documents (text) in a quantitative manner.

The dataset I used is a Wikipedia pages of several Animation Movies. Clustering was performed to group the movies together. The output of such analysis can be used for Recommendations of similar movie titles.

Method

Required Libraries

We start by loading the libraries needed for Text Mining and Clustering, and WordCloud Library for visualization of Clustered Results

```
library(tm)
```

```
## Loading required package: NLP
```

```
library(proxy)
```

```
##
## Attaching package: 'proxy'
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

```
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
library(RTextTools)
```

```
## Loading required package: SparseM
```

```
##
## Attaching package: 'SparseM'
```

```
## The following object is masked from 'package:base':
##
##     backsolve
```

```
library(fpc)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(cluster)
library(tm)
library(stringi)
library(proxy)
library(wordcloud)
```

Reading Data

Next we load the data into the R Corpus Data Structure. In this case the data is set of Plot Summaries from various Animation Movies obtained from Wikipedia

```
path = "/home/saqib/ml_class/project/corpus/"
dir = DirSource(paste(path,"/Archive/",sep=""), encoding = "UTF-8")
#corpus = Corpus(dir, readerControl=list(reader=readPDF))
corpus = Corpus(dir)
summary(corpus)
```

```
##                                Length Class             Mode
## Despicable Me                  2      PlainTextDocument list
## Despicable Me 2                2      PlainTextDocument list
## Finding Dory                   2      PlainTextDocument list
## Finding Nemo                   2      PlainTextDocument list
## Frozen                         2      PlainTextDocument list
## Ice Age                        2      PlainTextDocument list
## Ice Age_ Dawn of the Dinosaurs 2      PlainTextDocument list
## Kung Fu Panda                  2      PlainTextDocument list
## Kung Fu Panda 2                2      PlainTextDocument list
## Kung Fu Panda 3                2      PlainTextDocument list
## Madagascar                     2      PlainTextDocument list
## Madagascar_ Escape 2 Africa    2      PlainTextDocument list
## Minions                        2      PlainTextDocument list
## Shrek 2                        2      PlainTextDocument list
## Shrek Forever After            2      PlainTextDocument list
## Shrek the Third                2      PlainTextDocument list
```

Document Term Matrix (DTM)

The next step is to create a Document-Term Matrix (DTM). DTM is a matrix that lists all occurrences of words in the corpus. In a DTM, documents are represented by rows and the terms (or words) by columns. If a word occurs in a particular document n times, then the matrix entry for corresponding to that row and column is n, if it doesn't occur at all, the entry is 0.

```
ndocs <- length(corpus)
# ignore extremely rare words i.e. terms that appear in less then 1% of the documents
minTermFreq <- ndocs * 0.01
# ignore overly common words i.e. terms that appear in more than 50% of the documents
maxTermFreq <- ndocs * .5

dtm = DocumentTermMatrix(corpus,
                         control = list(
                                      stopwords = TRUE,
                                      wordLengths=c(4, 15),
                                      removePunctuation = T,
                                      removeNumbers = T,
                                      #stemming = T,
                                      bounds = list(global = c(minTermFreq, maxTermFreq))

                                     ))

#dtm <- dtm[, names(head(sort(colSums(as.matrix(dtm))), 400))]
```

```
#dtm <- dtm[, names(sort(colSums(as.matrix(dtm))))]
#print(as.matrix(dtm))
write.csv((as.matrix(dtm)), "test.csv")



#head(sort(as.matrix(dtm)[18,], decreasing = TRUE), n=15)

dtm.matrix = as.matrix(dtm)

#wordcloud(colnames(dtm.matrix), dtm.matrix[28, ], max.words = 20)
```

Let's inspect the generate Document Term Matrix.

Notice that the Sparsity is fairly high. This is good, as this indicates that there are unique terms the documents (movie synopsis) that are not present in ALL the documents. This is useful for separating (distance) the documents.

```
inspect(dtm)
```

```
## <<DocumentTermMatrix (documents: 16, terms: 2324)>>
## Non-/sparse entries: 3687/33497
## Sparsity           : 90%
## Maximal term length: 15
## Weighting          : term frequency (tf)
## Sample             :
##                                Terms
## Docs                            alex charming dory fiona manny marlin
##   Despicable Me 2                  0        0    0     0     0      0
##   Finding Nemo                     0        0   14     0     0     18
##   Frozen                           0        0    0     0     0      0
##   Ice Age_ Dawn of the Dinosaurs   0        0    0     0    10      0
##   Kung Fu Panda                    0        0    0     0     0      0
##   Kung Fu Panda 2                  0        0    0     0     0      0
##   Kung Fu Panda 3                  0        0    0     0     0      0
##   Madagascar_ Escape 2 Africa     14        0    0     0     0      0
##   Minions                          0        0    0     0     0      0
##   Shrek the Third                  0       19    0     7     0      0
##                                Terms
## Docs                            nemo shen shifu shrek
##   Despicable Me 2                  0    0     0     0
##   Finding Nemo                    19    0     0     0
##   Frozen                           0    0     0     0
##   Ice Age_ Dawn of the Dinosaurs   0    0     0     0
##   Kung Fu Panda                    0    0    15     0
##   Kung Fu Panda 2                  0   27     4     0
##   Kung Fu Panda 3                  0    0     7     0
##   Madagascar_ Escape 2 Africa      0    0     0     0
##   Minions                          0    0     0     0
##   Shrek the Third                  0    0     0    22
```

Term frequency-invese document frequecy (tf-idf)

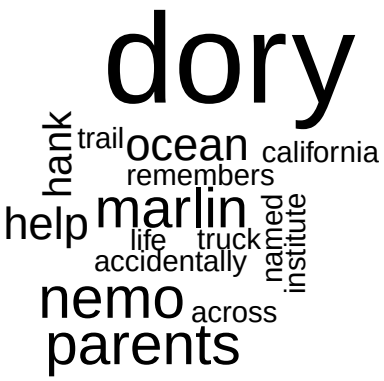
In the above DTM you will notice that terms that occur frequently have a high value associated. However, a certain term's high frequency within a document means little if that terms appear frequenty in other documents in the corpus. In other words, terms that occur frequently within a document but not frequently within the corpus receive a higher weighting as these words are assumed to contain more meaning in relation to the document.

To achieve this, we will downweight the terms that occur frequently accross the documents. This is done by computing the tf-idf Statistics.

```
#dtm <- weightTfIdf(dtm, normalize = TRUE)
dtm.matrix = as.matrix(dtm)
#wordcloud(colnames(dtm.matrix), dtm.matrix[28, ], max.words = 20)
#inspect(dtm)
write.csv((as.matrix(dtm)), "test.csv")
```

Below is the list of most important terms for the Finding Dory movie as determined using tf-idf weighting.

```
#head(sort(as.matrix(dtm)[1,], decreasing = TRUE), n=15)
wordcloud(colnames(dtm.matrix), dtm.matrix[3, ], max.words = 200)
```



Calculating Distance

Next we calculate the euclidean distance between the documents. This distance is what the Clustering algorithm uses to cluster documents.

First, DTM needs to be converted to a Standard R Matrix that can be consumed by dist

```
m  <- as.matrix(dtm)
# # # m <- m[1:2, 1:3]
distMatrix <- dist(m, method="euclidean")
#print(distMatrix)
#distMatrix <- dist(m, method="cosine")
#print(distMatrix)
```

Clustering

The first algorithm we'll look at is hierarchical clustering.

The R algorithm we'll use is hclust which does agglomerative hierarchical clustering. Here's a simplified description of how it works:

```
Assign each document to its own (single member) cluster
Find the pair of clusters that are closest to each other (dist) and merge them. So you now have one clus
Compute distances between the new cluster and each of the old clusters.
Repeat steps 2 and 3 until you have a single cluster containing all documents.
```

```r
groups <- hclust(distMatrix,method="ward.D")
plot(groups, cex=0.9, hang=-1)
rect.hclust(groups, k=5)
```