Text Mining 2: Clustering

300958 Social Web Analysis

Week 8 Lab Solutions

Set up the document-term matrix.

```
library("twitteR")
library("tm")
library("SnowballC")

load("Tweets.RData")
df1 = twListToDF(tweets1)
df2 = twListToDF(tweets2)
df3 = twListToDF(tweets3)
tweet.text = c(df1$text, df2$text, df3$text)

tweet.corpus = Corpus(VectorSource(tweet.text))
tweet.corpus = tm_map(tweet.corpus, function(x) iconv(x, to='ASCII'))
tweet.corpus = tm_map(t
tweet.corpus = tm_map(t
tweet.corpus = tm_map(tweet.corpus, stripWhitespace)
tweet.corpus = tm_map(tweet.corpus, tolower)
tweet.corpus = tm_map(tweet.corpus, removeWords, stopw
tweet.corpus = tm_map(tweet.corpus, stemDocument)
tweet.corpus = tm_map(tweet.corpus, PlainTextDocument)

tweet.dtm = DocumentTermMatrix(tweet.corpus)
tweet.wdtm = weightTfIdf(tweet.dtm)
tweet.matrix = as.matrix(tweet.wdtm)
## remove empty tweets
empties = which(rowSums(abs(tweet.matrix)) == 0)
tweet.matrix = tweet.matrix[-empties,]
```


- Write a for loop that performs k-means for 1 to 15 clusters and stores the SSW value for each clustering in the variable `SSW`.

```
n = 15
SSW = rep(0, n)
for (a in 1:n) {
  ## use nstart to reduce the effect of the random initialisation
  K = kmeans(tweet.matrix, a, nstart = 10)
  SSW[a] = K$tot.withinss
```

```
}
## plot the results
plot(1:15, SSW, type = "b")
```

- Determine the number of clusters when using Cosine distance with tweet vectors.

```
## first normalise all tweets to unit length (divide by their norm)
norm.tweet.matrix = diag(1/sqrt(rowSums(tweet.matrix^2))) %*% tweet.matrix
## then create the distance matrix
D = dist(norm.tweet.matrix, method = "euclidean")^2/2
## perform MDS using 100 dimensions
mds.tweet.matrix <- cmdscale(D, k=100)
n = 15
SSW = rep(0, n)
for (a in 1:n) {
  ## use nstart to reduce the effect of the random initialisation
  K = kmeans(mds.tweet.matrix, a, nstart = 20)
  SSW[a] = K$tot.withinss
}
## plot the results
plot(1:15, SSW, type =
```


' b We can see an elbow

clusters in clusters (a hierarchy of clusters).

We will compute three clusters.

```
K = kmeans(mds.tweet.matrix, 3, nstart = 20)
```

and visualise the clusters by projecting the data into a 2D space and colouring the clusters.

```
mds2.tweet.matrix <- cmdscale(D, k=2)
plot(mds2.tweet.matrix, col = K$cluster)
```

- Examine the words associated to each cluster.

```
cluster.number = 3
## find position of tweets in cluster
clusterTweetsId = which(K$cluster == cluster.number)
## extract tweets vectors for cluster
clusterTweets = tweet.matrix[clusterTweetsId,]
```

```
## combine the tweets into a mean tweet
clusterTermWeight = colMeans(clusterTweets)
## show the top 10 weighted words
sort(clusterTermWeight, decreasing = TRUE)[1:10]
```

- Perform clustering using single linkage clustering instead. Which of the two method provides more reasonable clusters?

First find only the terms that appear in at least 50 tweets

```
frequent.words = which(colSums(tweet.matrix > 0) > 50)
term.matrix = tweet.matrix[,frequent.words]
```

We want to compute the cosine between each of the terms (not documents), so we must make sure that all term vectors (columns of matrix) have a norm of 1.

```
norm.term.matrix = term.matrix %*% diag(1/sqrt(colSums(term.matrix^2)))
## preserve column names (terms associated to each column)
colnames(norm.term.matrix) = colnames(term.matrix)
```


We then compute the

transposing the matri


```
## then create the distance matrix
D = dist(t(norm.term.matrix), method = "euclidean")^2/2
```

Then perform the hierarchical clustering using single linkage clustering.

```
## hierarchical clustering
h = hclust(D, method="single")
plot(h)
```

And then complete linkage clustering.

```
## hierarchical clustering
h = hclust(D, method="complete")
plot(h)
```

Single linkage shows a greater similarity between Labour and Liberal compared to the Greens. Complete linkage clustering shows that all three parties are separate.