Clustering 20 News Channels

Irene Jacob

10/05/2018

Contents

Aim

In this project, we want to cluster documents from 20 different news sources based on the word-patterns found in them. Based on the results of the clustering, we may identify common expressions used in some of the news sources, the different categories covered in the data-set and popular stories. This project aims to data analysis to carry out clustering on a large data set sample, and to evaluate how clustering in a non-euclidean space can be used to identify similarities between news documents in the real world.

Environment Setup

The following libraries were used to help with the data analysis of this project:

```r
library(textmineR)
library(tm)
library(HistogramTools)
library(wordcloud)
library(factoextra)
library(cluster)
library(fpc)
library(slam)
```

Pre-processing the Data

We begin by reading the documents from the `train` folder and creating a word feature vector of these documents. An important variable to consider when creating the word vector is the ngram_window which is a vector of size two that takes two numbers, namely the minimum and maximum size of an ngram. An ngram is a sequence of `n` words taken (in order) from a document. We experiment with a few different lengths for the ngram_window, between 2 and 8 since we learned in class that in theory, a good shingle size for word documents is typically 8. Since an ngram works with words instead of characters as is the case with shingling, we ran the instruction to create the document term matrix (`dtm`) multiple times with different ngram lengths. We then plot the histogram of term frequencies from the document term matrix. Judging by eye, an ngram of length 4 seemed to have a good range in frequencies (following Zipf's law). Higher values of `n` resulted in many ngrams having a frequency of 1 (meaning the same ngram was not found in other documents) and lower values of `n` resulted in an extreme difference in frequencies between the first and second bin. However, it is clear that this method of deciding the size of the ngram_window is not optimal and in future, we should look into ways of deciding what the optimal size of an ngram is without judging by eye.

After running the hierarchical clustering algorithm once, we discovered a possible number of clusters that could occur in the `train` data set and we also found the top 5 words in those clusters. After observing the results of this summary table, we realise that some of the words in this table should be included as stop_words when we create the document term matrix. Therefore, we update the command to create `dtm` which accounts for common stopwords in the english language (as we recognised from the cluster summary table) and re-run the hierarchical clustering to achieve a more useful summary of the clustering algorithm.

```r
# Load the train data
setwd("./13_newsChannels/20news-bydate-train/")
train_filenames <- list.files(path='.', recursive = TRUE)
train_docs <- lapply(train_filenames, readLines, skipNul=TRUE)

train_docs <- lapply(train_docs,paste,collapse=" " )
train_doc_vector <- as.vector(train_docs)
```

The above commands are repeated on the test documents, to produce a document vector of the documents in the 20news-bydate-test folder. The code is redundant and therefore it is removed from the pdf. It can however be found in the HTML report.

```r
# With this for loop, we try to find a reasonable size for the ngram window.
for (i in 2:8) {
  train_dtm <-CreateDtm(doc_vec = train_doc_vector,
                        doc_names = train_filenames,
                        ngram_window = c(1, i),
                        lower = TRUE,
                        stopword_vec = c(tm::stopwords("english")),
                        remove_punctuation = TRUE,
                        remove_numbers = TRUE,
                        verbose = FALSE)
  # The plots have been removed from the pdf report, due to space constraints. They can be viewed in th
  hist.train <- hist(rowSums(train_dtm), xlab="Number of Terms in Term-Document Matrix",
  main="Number of Terms Per Document in News Articles Corpus", breaks=100, plot=FALSE )
}
```

As described earlier, an ngram window size of 4 was found ideal for this particualr dataset and so we create our document-term matrices for the train and test sets with this value. We also calculate the term frequencies for the train and test set.

```r
# Documents to word feature vectors
train_dtm <-CreateDtm(doc_vec = train_doc_vector,
                      doc_names = train_filenames,
                      ngram_window = c(1, 4),
                      lower = TRUE,
                      stopword_vec = c(tm::stopwords("english")),
                      remove_punctuation = TRUE,
                      remove_numbers = TRUE,
                      verbose = FALSE)

# number of terms in a document - enlarge size of bin
train_term_freq <- TermDocFreq(train_dtm)
```

Describing the data

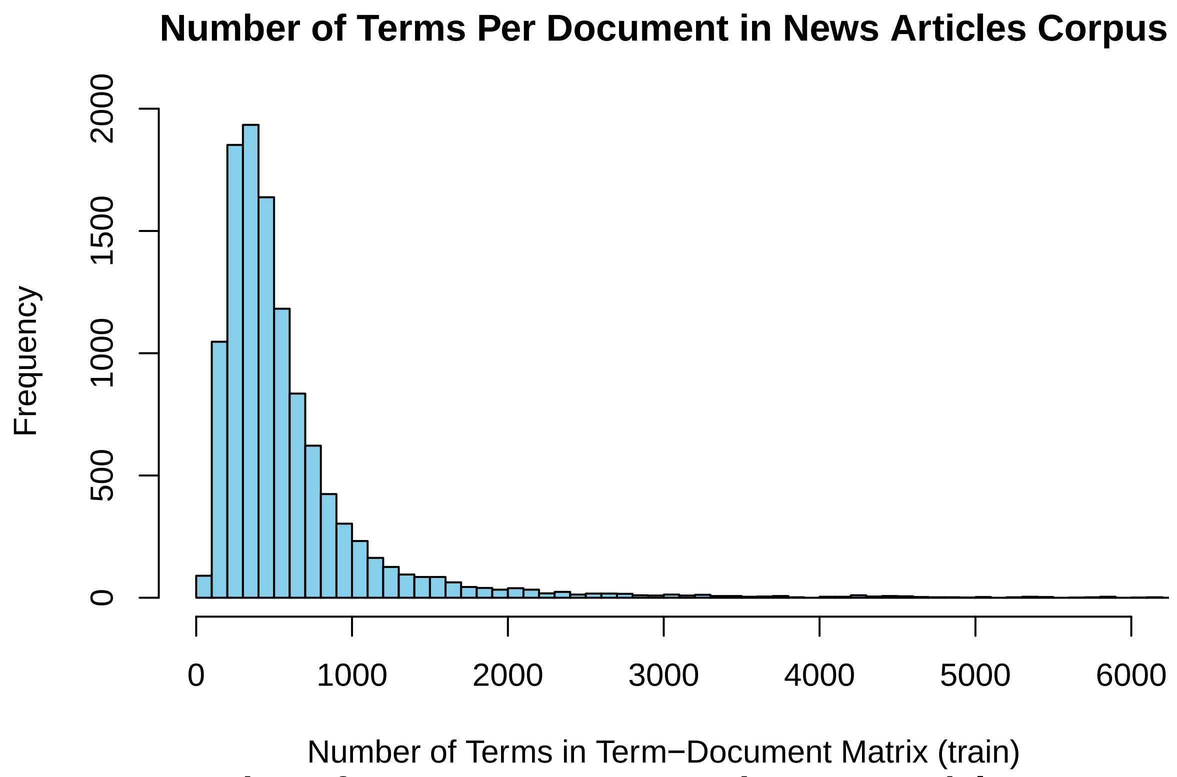
```r
dim(train_dtm)
```

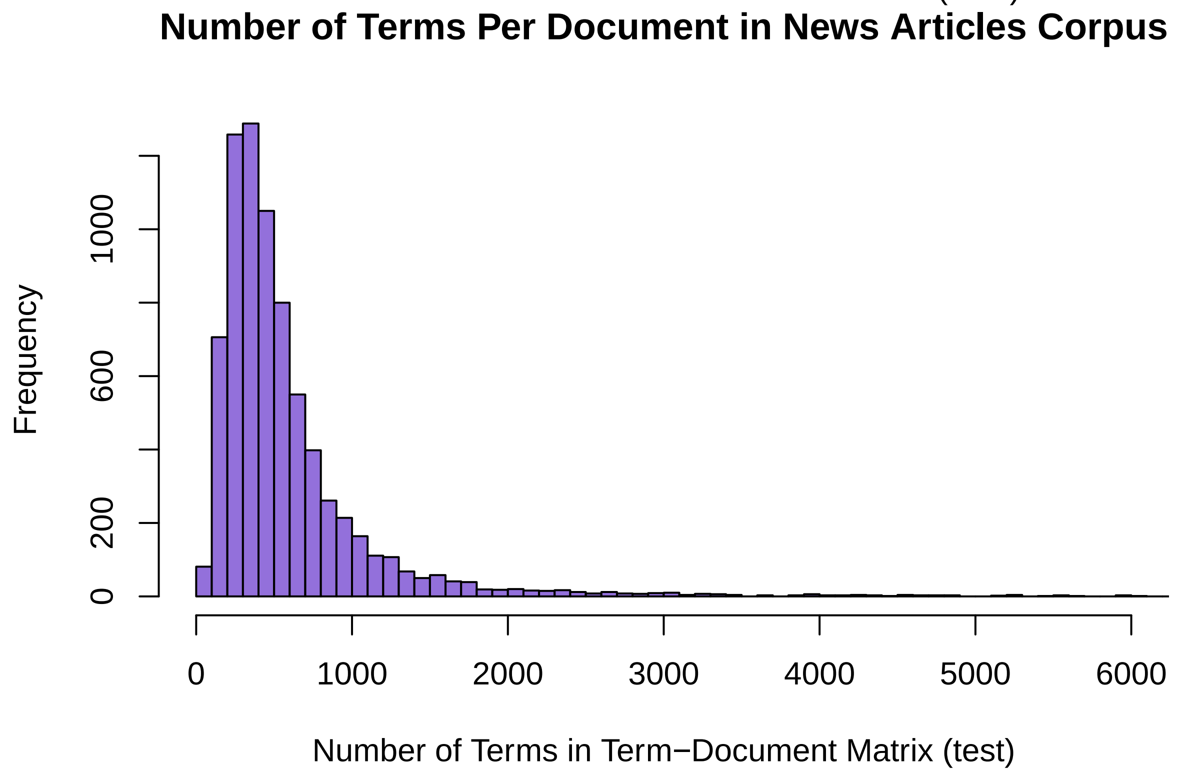
```
## [1]   11314 4035021
```

```r
dim(test_dtm)
```

```
## [1]    7532 2601958
```

Clearly, the document-term matrices are very large. Therefore, many computations may take a long time to execute, or we may not be able to execute them at all if the computation uses a lot of memory. Further ahead, we see how it is in fact not possible for us to cluster the document-term matrix due to its size. We will use the similarity matrices and cosine distances instead to approximate the clusters. A basic understanding of the document-term matrix can be found by plotting the histogram of the frequencies of the number of terms in the document-term matrix. In other words, we count how many other documents have a similar word-count. This information is useful since it allows us to get a crude understanding of the average length of the documents. From the two histograms below, we can see a clear similarity between the term-frequencies in the document-term-matrices of the train and test set. This is promising, since this assures us that finding the optimal `k` for the train data set, would likely result in finding the optimal `k` for the test data set.





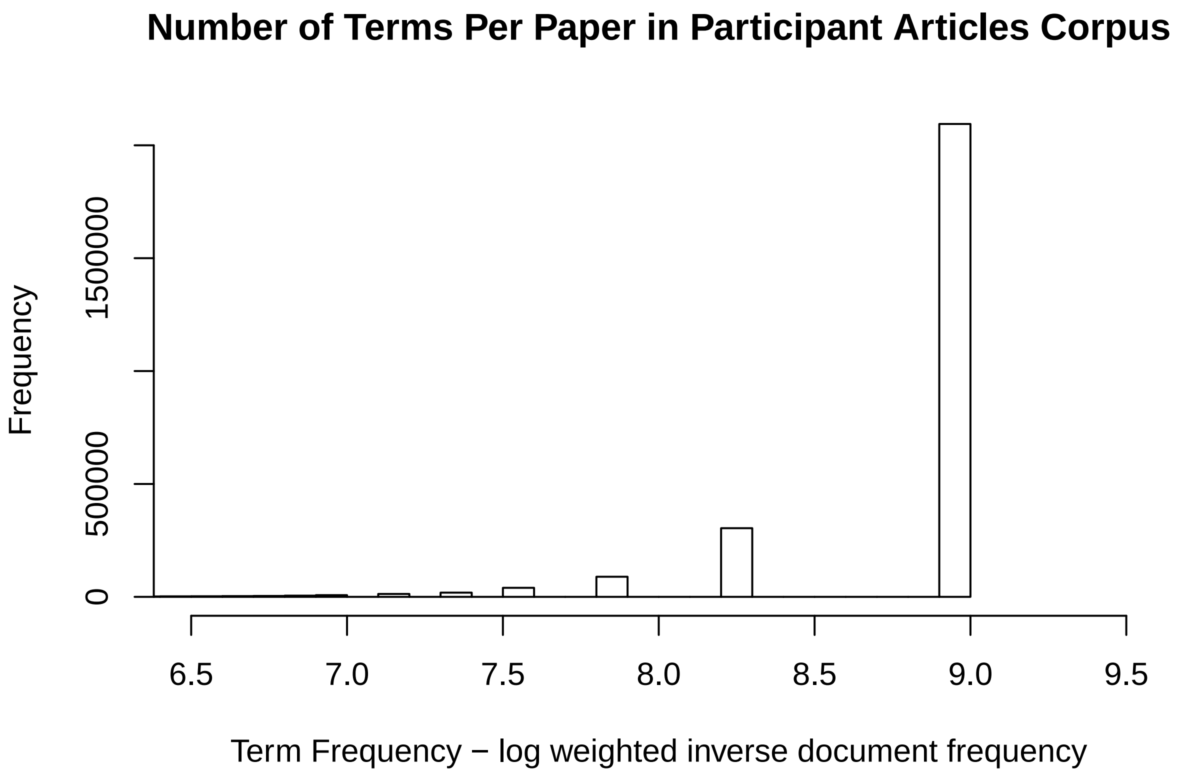
Term Frequency - Inverse document frequency

Essentially, the TF-IDF statistic reflects the importance of a term for a document in a collection of documents (also known as a Corpus). As we mentioned earlier, our document-term matrix is too large for clustering.

Therefore, we use the TF-IDF and we also calculate the cosine similarity between terms and do the clustering on the cosine similarity matrix instead.

```r
# TF-IDF and cosine similarity for train data
train_tf_idf <- t(train_dtm[ , train_term_freq$term ]) * train_term_freq$idf
train_tf_idf <- t(train_tf_idf)

train_csim <- train_tf_idf / sqrt(rowSums(train_tf_idf * train_tf_idf))
train_csim <- train_csim %*% t(train_csim)
train_cdist <- as.dist(1 - train_csim)


# TF-IDF and cosine similarity for test data
test_tf_idf <- t(test_dtm[ , test_term_freq$term ]) * test_term_freq$idf
test_tf_idf <- t(test_tf_idf)

test_csim <- test_tf_idf / sqrt(rowSums(test_tf_idf * test_tf_idf))
test_csim <- test_csim %*% t(test_csim)
test_cdist <- as.dist(1 - test_csim)
```

```r
hist(test_term_freq$idf, xlab="Term Frequency - log weighted inverse document frequency",
main="Number of Terms Per Paper in Participant Articles Corpus", breaks=100, xlim = c(6.5,9.5))
```



Hypothesis

From the graphs above, we can clearly see that the term frequency in the test data closely resembles that of the train data. Therefore, the clustering of the Train data set should predict the clustering of the Test data set. In other words, their clustering should look similar and we shall test this by clustering both the train and the test data set and comparing the results of the two. Since hierarchical clustring and k-means clustering are two different algorithms whichcluster the data in two different ways, it is not necessary that the number of clusters found by hierarchical clustering would be the optimal `k` for k-means clustering. We can test this out by clustering the data sets with both methods and comparing the number of clusters found in both.

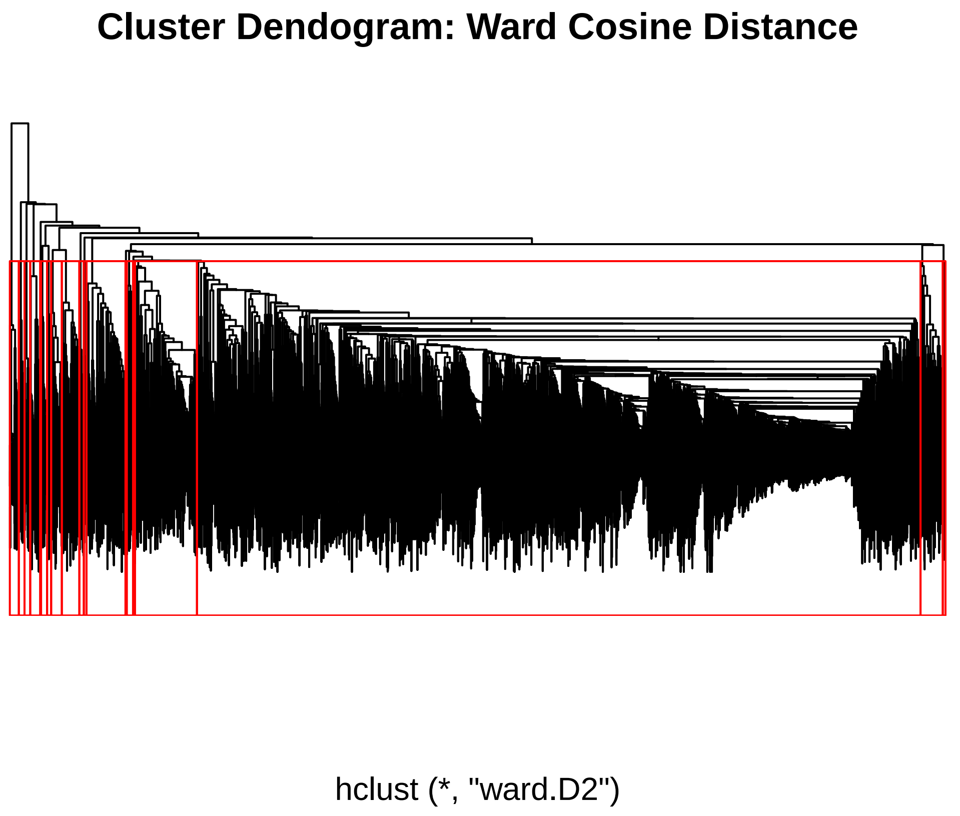
Data Analysis using Clustering Algorithm

Clustering Algorithm 1: Heirarchical Clustering

The first algorithm we try in order to cluster the data is hierarchical clustering. However, since the dimensions of the document-term matrix are so large, we cannot perform clustering on this matrix. Instead we will perform the clustering algorithm on the cosine similarity matrix of the terms that we calculated earlier. I also considered using other measures of similarity such as the euclidean similarity, however since we are working on a non-euclidean space with documents and words, the cosine similarity was the more accurate measure and so I opted to use it instead.

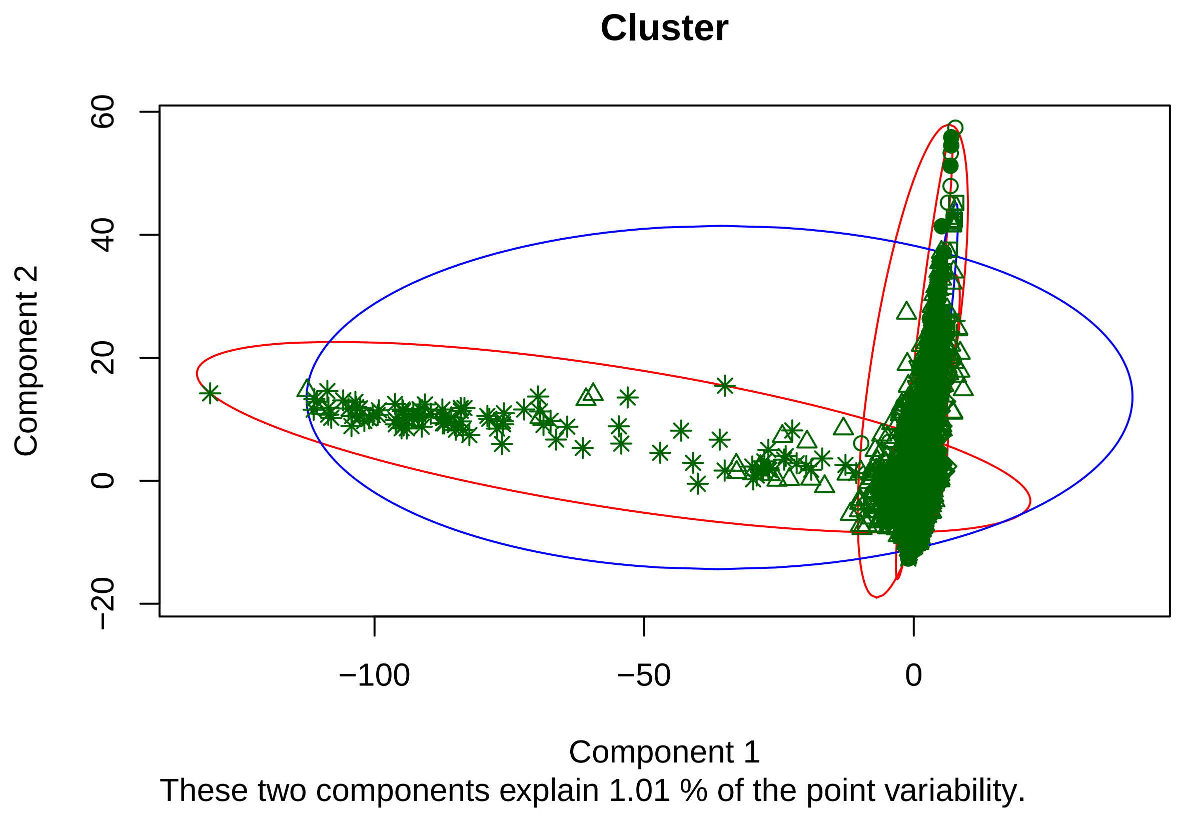
```r
train_hc <- hclust(train_cdist, "ward.D2")

clustering <- cutree(train_hc, 20)
```

```r
plot(train_hc, main = "Cluster Dendogram: Ward Cosine Distance",
     ylab = "", xlab = "", yaxt = "n", labels = FALSE)

rect.hclust(train_hc, 20, border = "red")
```



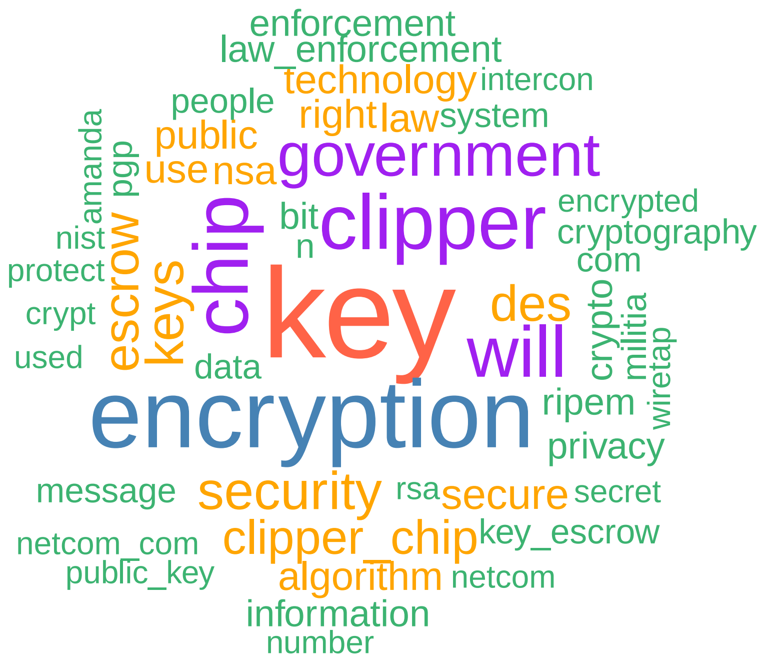
```r
p_words <- colSums(train_dtm) / sum(train_dtm)

cluster_words <- lapply(unique(clustering), function(x){
  rows <- train_dtm[ clustering == x , ]

  # for memory's sake, drop all words that don't appear in the cluster
  rows <- rows[ , colSums(rows) > 0 ]

  colSums(rows) / sum(rows) - p_words[ colnames(rows) ]
})
# create a summary table of the top 5 words defining each cluster
cluster_summary <- data.frame(cluster = unique(clustering),
                              size = as.numeric(table(clustering)),
                              top_words = sapply(cluster_words, function(d){
                                paste(
                                  names(d)[ order(d, decreasing = TRUE) ][ 1:50 ],
                                  collapse = ", ")
                              }),
                              stringsAsFactors = FALSE)
```

```
clusplot(as.matrix(train_csim), clustering, main="Cluster",color=TRUE, lines=0)
```



I tried colouring the wordcloud with different number of colours experimentally to see the average number of frequencies the words in that cluster fit into.

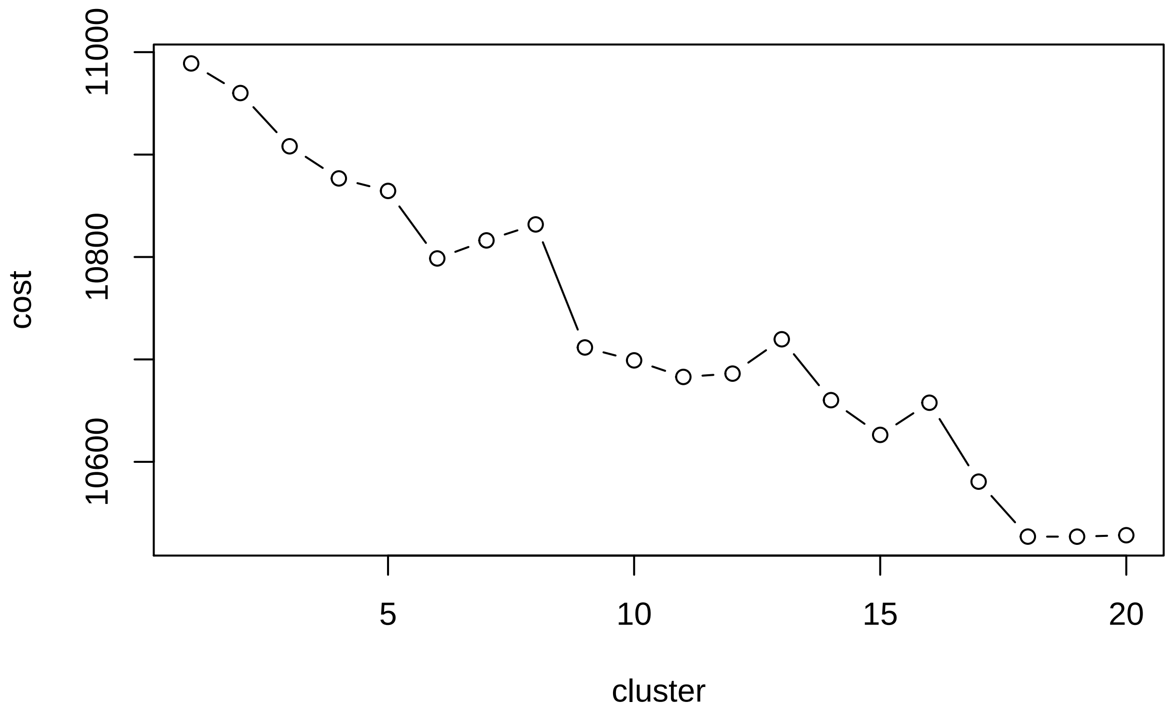
```
wordcloud::wordcloud(words = names(cluster_words[[ 14 ]]),
                     freq = cluster_words[[ 14 ]],
                     max.words = 50,
                     random.order = FALSE,
                     colors = c("mediumseagreen","orange","purple","steelblue",  "tomato"),
                     main = "Top words in cluster")
```

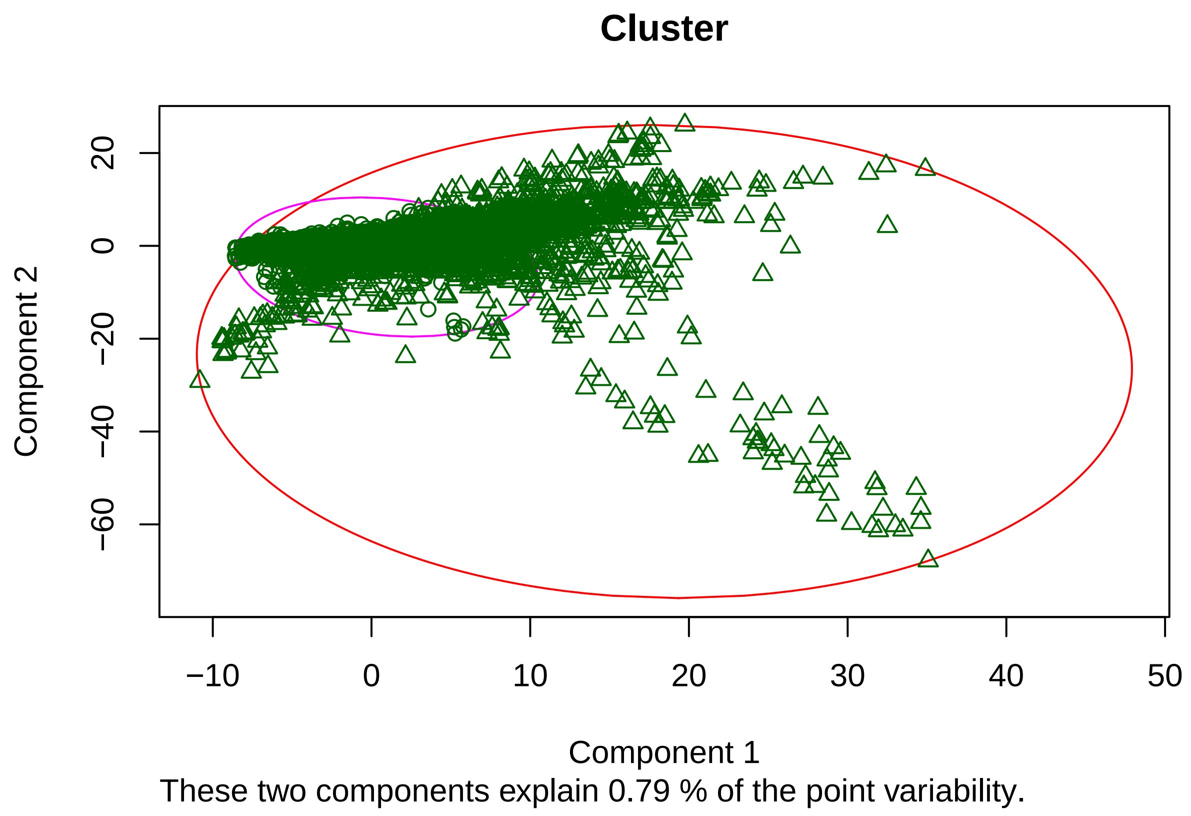


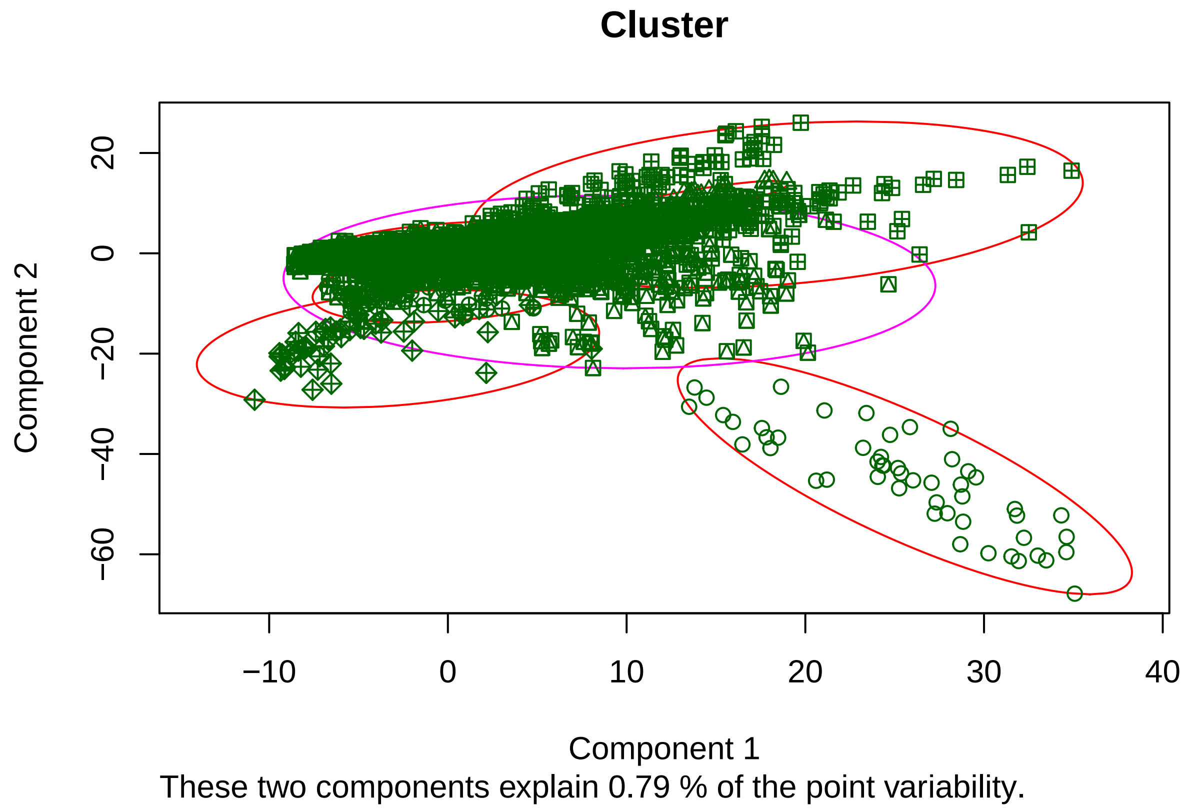
Clustering Algorithm 2: K-means clustering

Selecting K using Elbow method

```r
plot(cost_df,type = "b")
```




Cluster
Component 2
20
0
−20
−40
−60
−10
0
10
20
30
40
50
Component 1
These two components explain 0.79 % of the point variability.


Cluster
Component 2
20
0
−20
−40
−60
−10
0
10
20
30
40
Component 1
These two components explain 0.79 % of the point variability.

Conclusion

14 seems to be the optimal value of k for the Test data This shows that some documents are more or less similar (news channels with the same articles)

Evaluation

The first part of the hypothesis was partially correct, since the clusters of the train and test data set do look somewhat similar. However, it seems that the number of clusters in the train and test data set are not the same. This is understandable, since the train data set is much larger than the test data set.As predicted, the number of clusters found in hierarchical clustering was not the same as the optimal k found by the elbow method

Further Exploration

These were just some examples of data analysis that could be done with this data. A more in-depth analysis of the data would require us to divide the data further, and cluster each division separately. This would tell us many interesting features about individual news channels as well as the 20 news channels as a group.A further analysis would be to predict the clustering of the train and test data set together. In other words, given a new document from the test set, we should be able to cluster its terms in the current clustering.

References

1. Document Clustering: https://cran.r-project.org/web/packages/textmineR/vignettes/b_document_clustering.html
2. TF-IDF: https://en.wikipedia.org/wiki/Tf%E2%80%93idf
3. Kmeans Clustering: https://www.r-bloggers.com/clustering-search-keywords-using-k-means-clustering/
4. Document Clustering in R: http://faculty.washington.edu/jwilker/tft/Stewart.LabHandout.pdf
5. Dendograms: https://stackoverflow.com/questions/34819641/r-plot-upper-dendrogram-based-on-k
6. Zipf's law: https://en.wikipedia.org/wiki/Zipf%27s_law
7. Optimal k:

https://stackoverflow.com/questions/15376019/determining-optimum-number-of-clusters-for-k-means-with-a-large-dataset