

Clustering Analysis of Text Data with K-means and DB Scan Algorithms

Hieu Nguyen, Nithya Devadoss, Ramesh Simhambhatla, Ramya Mandava

Sep 13, 2018

Abstract

Text Clustering is an important part of the data mining and clustering of input data. Clustering is the process of separating the similar type of samples into the same class. It has applications in automatic document organization, topic extraction and fast information retrieval or filtering. This paper briefly discusses data analysis of a corpus of 1500 documents using 2 kinds of text clustering algorithms – K-Means and DBSCAN –based on the word frequency provided from those documents, analysis and comparison with testing of different cluster sizes, and findings from our testing.

1 Introduction

With advent of digital era, a large amount of unstructured data is being generated by various devices & applications. This data being generated can be stored in various systems and is often hard to extract and analyze manually. Clustering can be very useful to simplify this problem and to bind similar documents together. The data we try to analyze here is corpus of 1500 documents containing 12420 unique words, on which we would like to try **K-means** and **DBSCAN** clustering techniques which will further be explained in section 2.

However, in order to perform clustering, the unstructured data will have to be quantified. This can be achieved by looking at the words that make up the document and calculating the corresponding term frequency – inverse document frequency(**tf-idf**) scores.

Term frequency (**Tf**) is a measure of how important a word can be within a given document and it is calculated by looking at a word's frequency in a given document and is calculated as shown in formula 1, where **count** is the frequency of a word per document. It is important to note that a document can have highly frequent but insignificant words such as 'in', 'the', 'is' etc., which are also called stop words that might have to be removed to better understand the content and context of the document. On the other hand, Inverse document frequency (**idf**) is the inverse of how many times a word repeats across all the documents within a corpus. **Idf** decreases the score for most commonly used words and increases the weight for those that are not very common across the corpus and is calculated as shown in formula 2, where **N** = total number of documents in the corpus and **n**= number of documents a given word appears in.

In our approach we would like to use the **tf-idf** scores which is a combination of **tf** and **idf** and as defined in formula 3.

$$\mathbf{tf} = \ln(1 + \mathbf{count}) \quad (1)$$

$$\mathbf{idf} = \ln\left(1 + \frac{N}{n}\right) \quad (2)$$

$$\mathbf{tf_idf} = \mathbf{tf} * \mathbf{idf} \quad (3)$$

Tf and idf formulas are slightly modified to normalize the data and also to remove ‘undefined’ values arising due to sparse representations. Once the tf-idf scores are ready, these values are then assigned to a matrix called ‘*tfidf*’ of dimensions: *i by j* where, *i* is the number of documents and *j* is the total number of unique words or vocab size of the entire corpus. This newly derived matrix is then ready for clustering in order to club similar documents together.

2 Methods

Distance measure is needed to define similarity between documents, and then a criterion function to compute the quality of our clusters and finally a criterion optimization algorithm. Distance based partitioned clustering is widely used in database literature for achieving distance measure defined above. Partitioned clustering divides the data set into k disjoint clusters by using the distance measurements. Each k cluster contains the homogeneous data. One of the most widely used distance based partitioned clustering algorithm is k-means [2].

2.1 Clustering with k-mean algorithm

The goal of k-means algorithm is based on the input parameters k, in which the data set is divided into k clusters. Algorithm uses iterative update in each round, based on k point of references which were grouped around k clusters. Each cluster centroid will be used as a reference point for next round of iteration. Iteration makes the selected reference point closer to the true cluster centroid, so the clustering effect gets better.

K-means has been applied on tfidf matrix calculated in section 1, for various cluster sizes (5, 10, 20) and similar documents are plotted against their clusters as shown in fig 2.1.

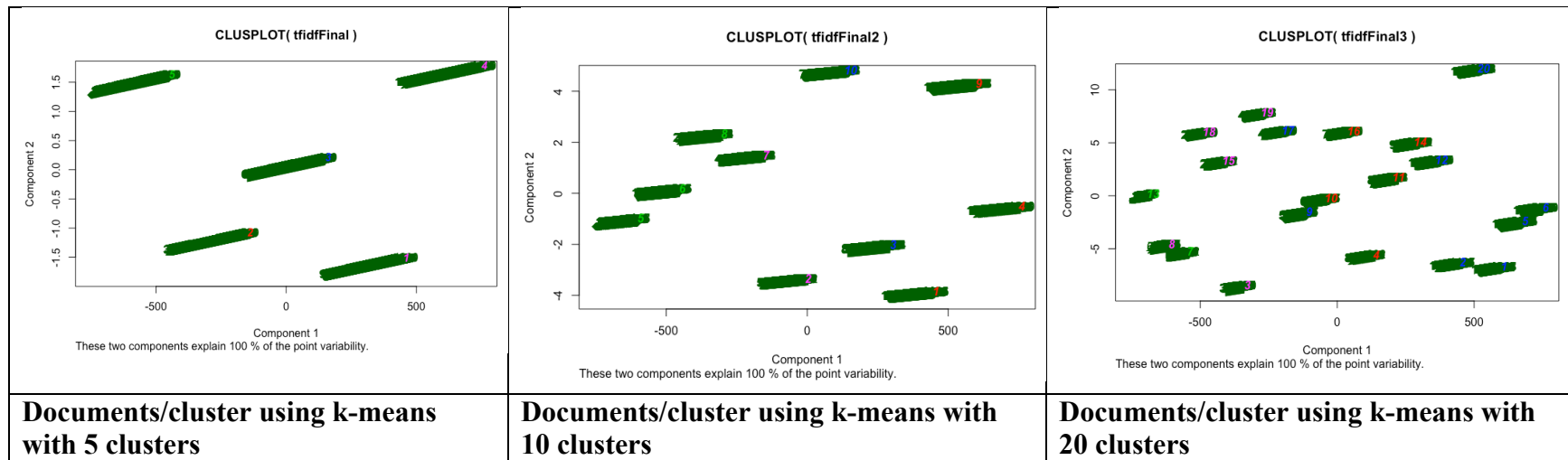


Figure 2.1

We have noticed that the data is better partitioned with about 10 clusters. K=20 yielded few overlapping clusters and splitting some of similar documents across different clusters.

Components of k-means cluster for k=10:

Clusters	Centers	Between_ss/total_ss	iter	ifault	Mean withinss	Mean betweenss
K=10	10	97.1%	5	0	845791.9	278489799

Within cluster sum of squares by cluster:

[1] 853093.7 852990.1 795595.1 857502.0 832778.8 837137.8 863845.6 837673.6 866151.5 861150.4
(between_SS / total_SS = 97.1 %)

Clusters size for each cluster:

[1] 149 151 151 150 150 149 148 151 150 151

The above table and results show the parameters being returned by k-means, where the algorithm took 5 iterations with $\text{between_ss}/\text{total_ss}$ ratio being at 97.1%.

So, our final model for K-means clustering uses a value of 10 for the number of clusters. And the below histogram (figure 2.2) shows the frequency of documents across all 10 clusters.

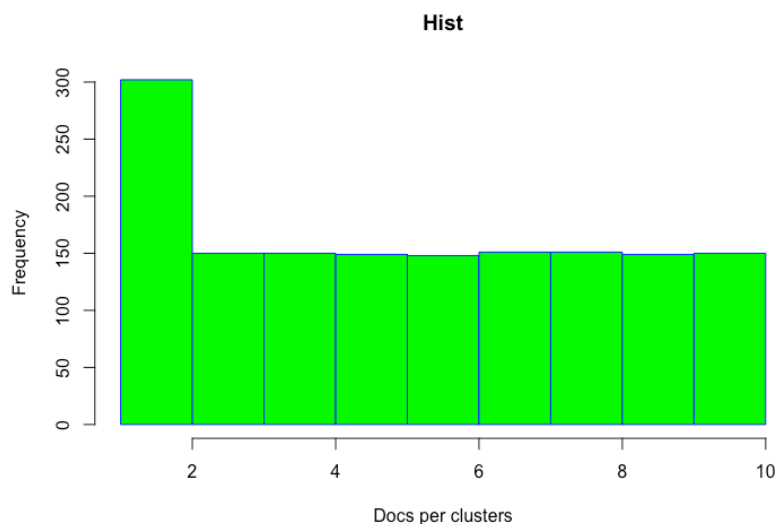


Figure 2.2: Histogram of Documents/cluster using k-means with 10 clusters

2.2 Clustering with Density Based Clustering (DBSCAN)

Density based clustering algorithm is very useful to classify nonlinear data, in density-based clustering algorithm cluster is defined as area of higher density that is the remainder of the dataset. Density based clustering considers density and boundary area of the cluster. One of the most widely used density based spatial clustering with the application of noise is DBSCAN.

DBSCAN has been applied on *tfidf* matrix calculated in section 1, for various [eps and min points] combinations ([65,25], [70,16], [77,16], [90,16]) and similar documents are plotted against their clusters as shown in fig 2.3.

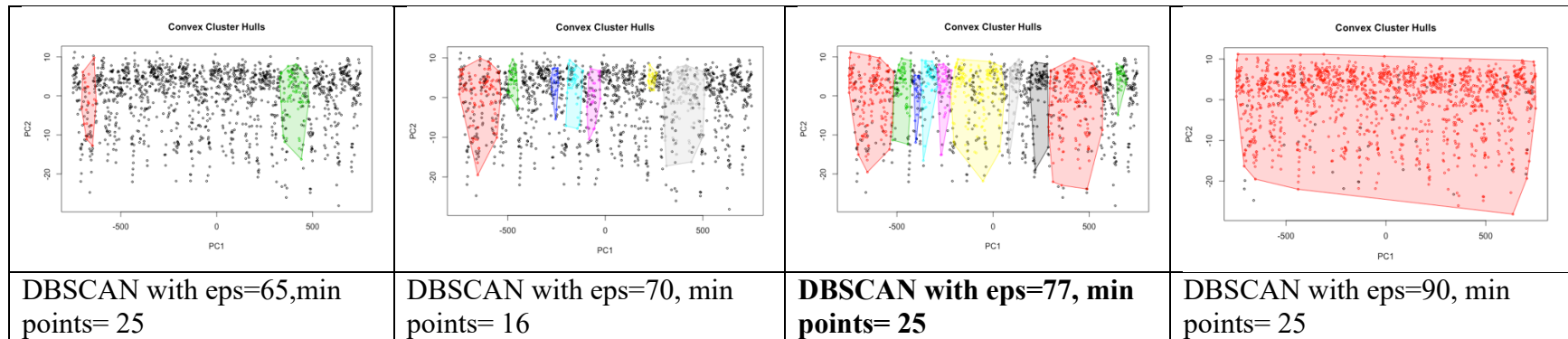


Figure 2.3

Due to lack of domain knowledge on our corpus, KNNdistplot has been used with $k = \log(\text{number of data points})$ in order to obtain the optimal eps and min points values to be used for DBSCAN algorithm. Best value for eps has been identified to be around 77 (knee point) as shown in figure 2.4 and min points to be 16(which is $\log(\text{number of data points})$). A knee corresponds to a threshold where a sharp change occurs along the k-distance curve.

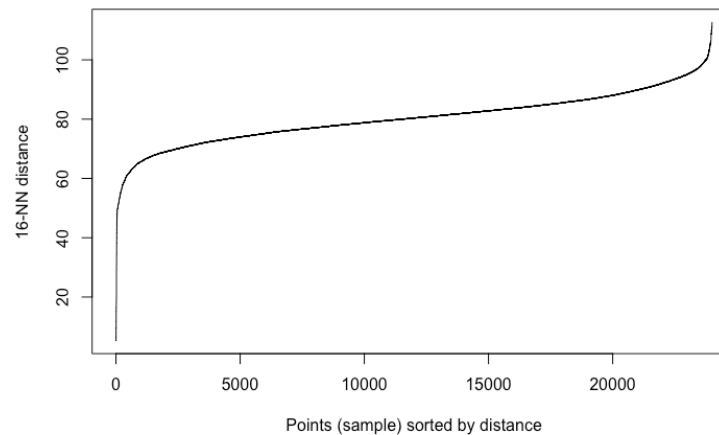


Figure 2.4 kNN dist plot

We have noticed that the data is better partitioned with about 10 clusters. So, our final model for DBSCAN clustering uses $\text{eps}=77$ and $\text{min points}=16$.

Key components of DBSCAN:

`dbscan(x, eps, minPts = 5, weights = NULL, borderPoints = TRUE, ...)`

Key Arguments used in the test:

x	a data matrix or a dist object.
eps	size of the epsilon neighborhood
minPts	number of minimum points in the eps region (for core points). Default is 5 points.

3 Results

K-means algorithm has been applied with an initial choice of 5, 10 and 20 clusters. After visual exploration of the results as shown in section 2, k-means with 10 clusters yielded best results – clusters are more evenly distributed, and no overlapping has been observed.

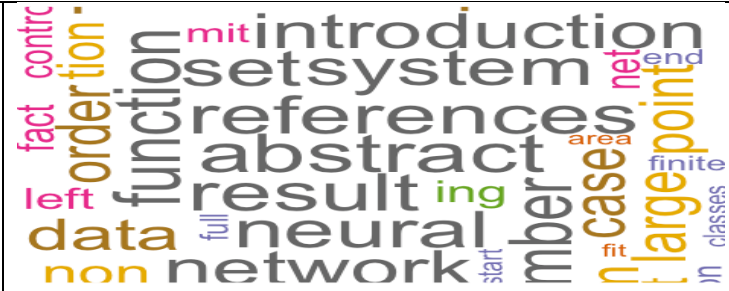
We further tested our data with DBSCAN across a range of eps and min-point values and the combination of [77, 16] yielded the best results which also gave us 10 clusters as can be seen in figure 2.3 from section 2. ***In all the DBSCAN results, we have noticed several data points within cluster 0 which are regarded as noise points.***

After running our data through k-means and DBSCAN algorithms, k-means is a better fit for our data, as we observed that there are too many noise data points in DBSCAN, which means they are not properly classified. This is also the main disadvantage of DBSCAN algorithm where it can't cluster data with large density. However, K-means tends to assume that our clusters will be simple, due to its partitional approach: it tries to decompose the dataset into non-overlapping subsets. k-means has a good effect on the convex cluster and is a good fit between the two for our data (corpus).

In order to further understand the context of each cluster, word clouds have been constructed using clusters from k-means algorithm with $k=10$.

Cluster 1: Word Cloud

Based on the word cloud, we could infer that cluster 1 has documents related to neural networks and have references to *MIT papers*.



Cluster 2: Word Cloud

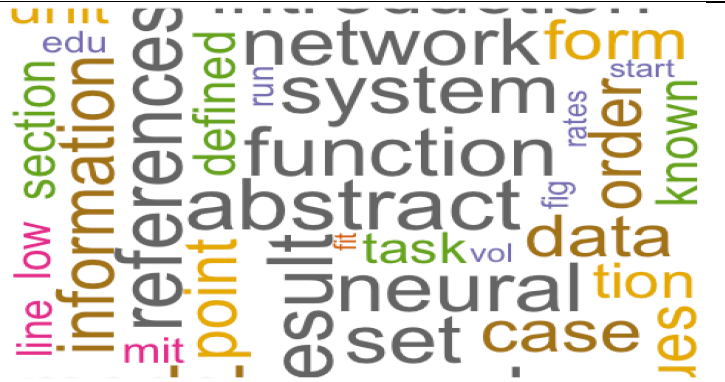


Based on the word cloud, we could infer that cluster 2 has documents related to neural networks and have references to *IEEE papers*.



Cluster 3: Word Cloud

Based on the word cloud, we could infer that cluster 3 has documents related to neural networks and *data modeling* and have references to *MIT papers*.



<p>Cluster 4: Word Cloud</p> <p>Based on the word cloud, we could infer that cluster 4 has documents related to neural networks and have references to <i>MIT papers</i>.</p>	 <p>A word cloud for Cluster 4 featuring terms like 'network', 'system', 'function', 'abstract', 'neural', 'data', 'case', 'set', 'result', 'point', 'defined', 'run', 'form', 'start', 'order', 'known', 'rates', 'fig', 'task', 'vol', 'fit', 'mit', 'line', 'low', 'section', 'information', 'references', 'es', 'case', 'jes', 'tion', 'data', 'vol', 'fit', 'task', 'neural', 'set', 'result', 'point', 'defined', 'run', 'form', 'start', 'order', 'known', 'rates', 'fig', 'task', 'vol', 'fit', 'mit', 'line', 'low', 'section', 'information', 'references'.</p>
<p>Cluster 5: Word Cloud</p> <p>Based on the word cloud, we could infer that cluster 5 has documents related to neural networks and have references to <i>nature</i> journals and <i>MIT</i> papers. Also, looks like these documents refer to many articles by Vincent st. Amour. His significant work is in image processing.</p>	 <p>A word cloud for Cluster 5 featuring terms like 'network', 'system', 'neural', 'result', 'model', 'case', 'set', 'function', 'abstract', 'references', 'introduction', 'data', 'takes', 'um', 'ber', 'form', 'part', 'sense', 'condition', 'lead', 'fig', 'unit', 'algorithm', 'nature', 'pair', 'differ', 'san', 'amour', 'forward', 'mit', 'part', 'sense', 'um', 'ber', 'form', 'part', 'sense', 'condition', 'lead', 'fig', 'unit', 'algorithm', 'nature', 'pair', 'differ', 'san', 'amour', 'forward', 'mit', 'part', 'sense'.</p>
<p>Cluster 6: Word Cloud</p> <p>Based on the word cloud, we could infer that cluster 6 has documents related to neural networks and have references to IEEE papers.</p>	 <p>A word cloud for Cluster 6 featuring terms like 'function', 'input', 'data', 'result', 'small', 'case', 'set', 'abstract', 'references', 'neural', 'system', 'order', 'parameter', 'left', 'task', 'field', 'word', 'Z', 'due', 'forward', 'tion', 'note', 'cell', 'tro', 'duction', 'ite', 'form', 'values', 'ing', 'red', 'takes', 'non', 'and', 'int', 'run', 'nce', 'tro', 'duction', 'ite', 'form', 'values', 'ing', 'red'.</p>

<p>Cluster 7: Word Cloud</p> <p>Based on the word cloud, we could infer that cluster 7 has documents related to neural networks and have references to MIT papers.</p>	 <p>Word cloud for Cluster 7. The most prominent words are 'learning', 'number', 'set', 'data', 'test', 'form', 'fact', 'cell', 'fig', 'allow', 'result', 'mode', 'network', 'function', 'error', 'task', 'hidden', 'point', 'cost', 'real', 'brain', 'term', 'selected', 'references', 'neural', 'system', 'input', 'log', 'bound', 'mit', 'takes', 'te', 'st', 'form', 'fact', 'cell', 'fig', 'allow', 'result', 'mode', 'network', 'function', 'error', 'task', 'hidden', 'point', 'cost', 'real', 'brain', 'term', 'selected'.</p>
<p>Cluster 8: Word Cloud</p> <p>Based on the word cloud, we could infer that cluster 8 has documents related to neural networks and have references to IEEE papers and MIT papers.</p>	 <p>Word cloud for Cluster 8. The most prominent words are 'function', 'data', 'neural', 'system', 'set', 'result', 'model', 'paper', 'references', 'small', 'ieee', 'on', 'values', 'case', 'number', 'unit', 'sum', 'network', 'system', 'function', 'abstract', 'result', 'neural', 'input', 'set', 'line', 'order', 'basis', 'short', 'references', 'large', 'form', 'studies', 'size', 'non', 'rate', 'area', 'full', 'fast', 'field', 'factor', 'seen', 'paper', 'area', 'non', 'rate', 'rule'.</p>
<p>Cluster 9: Word Cloud</p> <p>Based on the word cloud, we could infer that cluster 9 has documents related to neural networks.</p>	 <p>Word cloud for Cluster 9. The most prominent words are 'learning', 'case', 'model', 'part', 'system', 'function', 'abstract', 'result', 'neural', 'input', 'set', 'line', 'order', 'basis', 'short', 'references', 'large', 'form', 'studies', 'size', 'non', 'rate', 'area', 'full', 'fast', 'field', 'factor', 'seen', 'paper', 'area', 'non', 'rate', 'rule'.</p>
<p>Cluster 10: Word Cloud</p> <p>Based on the word cloud, we could infer that cluster 8 has documents related to neural networks and have references to MIT papers.</p>	 <p>Word cloud for Cluster 10. The most prominent words are 'learning', 'case', 'model', 'part', 'system', 'function', 'abstract', 'result', 'neural', 'input', 'set', 'line', 'order', 'basis', 'short', 'references', 'large', 'form', 'studies', 'size', 'non', 'rate', 'area', 'full', 'fast', 'field', 'factor', 'seen', 'paper', 'area', 'non', 'rate', 'rule'.</p>

4 Conclusion

Our corpus of 1500 documents have been analyzed and the associated tf-idf scores have been calculated. On which, we further performed clustering with k-means and DBSCAN algorithms.

K-Means is useful when you have an intuitive idea of how many clusters actually exists in your space. Its main benefit is its speed. There is a relationship between attributes and the number of observations in a dataset.

We observed that k-means clustering with $k=10$ gave us best results with proper segregation and grouping of documents while DBSCAN still showed several noise points even when tested across range of eps and min point combinations.

The quality of DBSCAN depends on the distance measure used in the function. Most commonly used is Euclidean. Especially for high-dimensional data, this metric can be rendered almost useless due to the so-called "Curse of dimensionality", making it difficult to find an appropriate value. DBSCAN cannot cluster data sets well, with large differences in densities, since the minPts-\var epsilon combination cannot then be chosen appropriately for all clusters.

5 References

- Prof Slater's Jupyter Notebook sample and class material
- <https://pdfs.semanticscholar.org/c643/6c6e262ee2ec9bd453aab5434c4b5b90bace.pdf>
- <https://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html>

A CODE

```
library(dplyr)
library(sqldf)
library(cluster)
library(dbSCAN)
library(HSAUR)
library("tm")
library("SnowballC")
library("wordcloud")
library("RColorBrewer")
library(caroline)
library(plyr)
```

```
N <- 1500
```

```
data <- read.csv('/Users/ramya/Documents/SMU/QTW/Week1/CaseStudy6_2/docword.nips.txt', sep = ' ', skip=3,
header=F)
colnames(data) <- c("doc", "word", "count")
head(data, n=10)
```

```
docs<-max(data[,1]) #same as above, but store for later
words=max(data[,2]) #find number of unique words
max_count=max(data[,3]) # find the max count, so we can normalize later
tfidf = data.frame(c(1:docs)) #create a dataframe with just the document numbers
```

```
head(tfidf)
head(data)
```

```
for (i in 1:words){
  label = paste("word_",toString(i),sep="")
  tfidf[label] <- 0}
```

```
head(tfidf[,20:30])
```

```
dim(tfidf)
```

```
for (i in 1:746315){  
  row = data[i,1]  
  col = data[i,2]  
  val = data[i,3]  
  tfidf[row,col+1] = val  
  if (i %% 20000 == 0){print(toString(i))}  
}
```

```
dim(tfidf[, colSums(tfidf != 0) > 0])
```

```
tfidf<-tfidf[, colSums(tfidf != 0) > 0]
```

```
dim(tfidf)
```

```
#head(tfidf, n=1)  
tfidf2 <- tfidf  
tfidf[1:5, 1:5]  
dim(tfidf2)
```

```
tfidf3 <- apply(tfidf[2:12376],1:2, function(x) log(1+x))
```

```
tfidf2[2:12376]<- tfidf3
```

```
#tfidf2$docs <- tfidf[, 1]  
tfidf2[1:5, 1:5]  
#tfidf2$c.1.docs. <- seq.int(from=1, to=1500, by=1)
```

```
idf<-colSums(tfidf2[-1] != 0)
```

```
idf[1]  
idf[2]  
idf[39]
```

```
idf = log(1+ (1/idf)*N)
```

```
max(idf)
```

```
head(idf)
```

```
new_words<-dim(tfidf2)[2] - 1  
new_words
```

```
for (i in 1:new_words){  
  tfidf2[,i+1] <- tfidf2[,i+1] * idf[i]}
```

```
tfidf2[1:5, 1:5]
```

```
#kmeans 5  
set.seed(5)  
clusters1 <- kmeans(tfidf2, 5)
```

```
tfidfFinal <- tfidf2[1]  
head(tfidfFinal)  
colnames(tfidfFinal) <- "doc"
```

```
tfidfFinal$kcluster <- as.factor(clusters1$cluster)  
head(tfidfFinal, n=10)  
tail(tfidfFinal, n=10)
```

```
clusplot(tfidfFinal, clusters1$cluster, color=TRUE, shade=TRUE,  
         labels=2, lines=0, plotchar=TRUE, span=TRUE)
```

```
#kmeans 10  
set.seed(10)  
clusters2 <- kmeans(tfidf2, 10)
```

```
tfidfFinal2 <- tfidf2[1]  
head(tfidfFinal2)  
colnames(tfidfFinal2) <- "doc"
```

```
tfidfFinal2$kcluster <- as.factor(clusters2$cluster)  
head(tfidfFinal2, n=10)  
tail(tfidfFinal2, n=10)
```

```
clusplot(tfidfFinal2, clusters2$cluster, color=TRUE, shade=TRUE,  
         labels=2, lines=0, plotchar=TRUE, span=TRUE)
```

```
hist(as.integer(tfidfFinal2$kcluster),  
     main="Hist",  
     xlab="Docs per clusters",  
     border="blue",  
     col="green",  
     breaks=10)
```

```
#kmeans 20  
set.seed(20)  
clusters3 <- kmeans(tfidf2, 20)
```

```
tfidfFinal3 <- tfidf2[1]  
head(tfidfFinal3)  
colnames(tfidfFinal3) <- "doc"
```

```
tfidfFinal3$cluster <- as.factor(clusters3$cluster)
head(tfidfFinal3, n=10)
tail(tfidfFinal3, n=10)
```

```
clusplot(tfidfFinal3, clusters3$cluster, color=TRUE, shade=TRUE,
         labels=2, lines=0, plotchar=TRUE, span=TRUE)
```

```
#DBSCAN
```

```
kNNdistplot(tfidf2, k = 16)
abline(h=.5, col = "red", lty=2)
```

```
set.seed(1234)
res <- dbscan(tfidf2, eps = 77, minPts = 16)
hullplot(tfidf2, res$cluster)
```

```
#Reading vocab
```

```
dataTxt <- read.csv('/Users/ramya/Documents/SMU/QTW/Week1/CaseStudy6_2/vocab.nips.txt', header=F)
colnames(dataTxt) <- c("wordtext")
head(dataTxt, n=10)
```

```
dataTxt2 <- dataTxt
dataTxt2$word <- seq.int(from=1, to=12419, by=1)
```

```
data2 <- data
dataTxtFinal <- merge(x = data2, y = tfidfFinal2, by = "doc", all = FALSE)
dim(dataTxtFinal)
```

```
dataTxtFinal2 <- dataTxtFinal
dataTxtFinal2 <- merge(x = dataTxtFinal2, y = dataTxt2, by = "word", all = FALSE)
dim(dataTxtFinal2)
```

```
nCount <- count(data, "word")  
head(nCount, n=10)
```

```
clouddata <- merge(x = dataTxtFinal2, y = nCount, by = "word", all = FALSE)  
head(clouddata)
```

```
#Creating Data Cloud for 1st cluster
```

```
clouddata2<- clouddata[dataTxtFinal2$kcluster == 1,]
```

```
tail(clouddata2, n=10)
```

```
wordcloud1 <- sqldf("select max(freq) as freq, wordtext from clouddata2 group by wordtext order by freq desc")
```

```
set.seed(1234)
```

```
wordcloud(words = wordcloud1$wordtext, freq = wordcloud1$freq, min.freq = 200,  
          max.words=500, random.order=FALSE, rot.per=0.35,  
          colors=brewer.pal(8, "Dark2"))
```

```
# Data cloud for cluster =2
```

```
#Creating Data Cloud for 2nd cluster
```

```
clouddata2<- clouddata[dataTxtFinal2$kcluster == 2,]
```

```
tail(clouddata2, n=10)
```

```
wordcloud1 <- sqldf("select max(freq) as freq, wordtext from clouddata2 group by wordtext order by freq desc")
```

```
set.seed(1234)
```

```
wordcloud(words = wordcloud1$wordtext, freq = wordcloud1$freq, min.freq = 200,  
          max.words=500, random.order=FALSE, rot.per=0.35,  
          colors=brewer.pal(8, "Dark2"))
```

```
# Data cloud for cluster =3
```

```
#Creating Data Cloud for 2nd cluster
```

```
clouddata2<- clouddata[dataTxtFinal2$kcluster == 3,]
```



```
tail(clouddata2, n=10)
wordcloud1 <- sqldf("select max(freq) as freq, wordtext from clouddata2 group by wordtext order by freq desc")
```

```
set.seed(1234)
wordcloud(words = wordcloud1$wordtext, freq = wordcloud1$freq, min.freq = 200,
          max.words=500, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```

```
# Data cloud for cluster =4
#Creating Data Cloud for 2nd cluster
clouddata2<- clouddata[dataTxtFinal2$kcluster == 4,]
tail(clouddata2, n=10)
wordcloud1 <- sqldf("select max(freq) as freq, wordtext from clouddata2 group by wordtext order by freq desc")
```

```
set.seed(1234)
wordcloud(words = wordcloud1$wordtext, freq = wordcloud1$freq, min.freq = 200,
          max.words=500, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```

```
# Data cloud for cluster =5
#Creating Data Cloud for 2nd cluster
clouddata2<- clouddata[dataTxtFinal2$kcluster == 5,]
tail(clouddata2, n=10)
wordcloud1 <- sqldf("select max(freq) as freq, wordtext from clouddata2 group by wordtext order by freq desc")
```

```
set.seed(1234)
wordcloud(words = wordcloud1$wordtext, freq = wordcloud1$freq, min.freq = 200,
          max.words=500, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```

```
# Data cloud for cluster =6
#Creating Data Cloud for 2nd cluster
```

```
clouddata2<- clouddata[dataTxtFinal2$kcluster == 6,]  
tail(clouddata2, n=10)  
wordcloud1 <- sqldf("select max(freq) as freq, wordtext from clouddata2 group by wordtext order by freq desc")
```

```
set.seed(1234)  
wordcloud(words = wordcloud1$wordtext, freq = wordcloud1$freq, min.freq = 200,  
          max.words=500, random.order=FALSE, rot.per=0.35,  
          colors=brewer.pal(8, "Dark2"))
```

```
# Data cloud for cluster =7  
#Creating Data Cloud for 2nd cluster  
clouddata2<- clouddata[dataTxtFinal2$kcluster == 7,]  
tail(clouddata2, n=10)  
wordcloud1 <- sqldf("select max(freq) as freq, wordtext from clouddata2 group by wordtext order by freq desc")
```

```
set.seed(1234)  
wordcloud(words = wordcloud1$wordtext, freq = wordcloud1$freq, min.freq = 200,  
          max.words=500, random.order=FALSE, rot.per=0.35,  
          colors=brewer.pal(8, "Dark2"))
```

```
# Data cloud for cluster =8  
#Creating Data Cloud for 2nd cluster  
clouddata2<- clouddata[dataTxtFinal2$kcluster == 8,]  
tail(clouddata2, n=10)  
wordcloud1 <- sqldf("select max(freq) as freq, wordtext from clouddata2 group by wordtext order by freq desc")
```

```
set.seed(1234)  
wordcloud(words = wordcloud1$wordtext, freq = wordcloud1$freq, min.freq = 200,  
          max.words=500, random.order=FALSE, rot.per=0.35,  
          colors=brewer.pal(8, "Dark2"))
```

```
# Data cloud for cluster =9
#Creating Data Cloud for 2nd cluster
clouddata2<- clouddata[dataTxtFinal2$kcluster == 9,]
tail(clouddata2, n=10)
wordcloud1 <- sqldf("select max(freq) as freq, wordtext from clouddata2 group by wordtext order by freq desc")

set.seed(1234)
wordcloud(words = wordcloud1$wordtext, freq = wordcloud1$freq, min.freq = 200,
          max.words=500, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))

# Data cloud for cluster =10
#Creating Data Cloud for 2nd cluster
clouddata2<- clouddata[dataTxtFinal2$kcluster == 10,]
tail(clouddata2, n=10)
wordcloud1 <- sqldf("select max(freq) as freq, wordtext from clouddata2 group by wordtext order by freq desc")

set.seed(1234)
wordcloud(words = wordcloud1$wordtext, freq = wordcloud1$freq, min.freq = 200,
          max.words=500, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```