Homework 1 analysis report

1. **Description of datasets** :
   a. 20NewsGroups data :

   One of the popular dataset for text mining. Has documents of 20 news groups. Each group has about 1000 documents within them. Some of these new groups are closely related while some of them totally unrelated. Below table summarizes the related groups:

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

I took the 20news-19997.tar.gz

I took the original dataset which has contains duplicate documents and header in each documents.

Upon, looking into the documents in the groups I found that header has fields like subject, date, organization etc. Since each document has these terms, we have to remove these words during preprocessing as these does not give any information if the document itself.

   b. Yelp dataset :
   This dataset is recent real time dataset collected from the yelp user reviews. This has 6 parts, namely, business, review, user, checkin and tip(each of which are json files).
   Upon examining, reviews can be grouped as positive, negative, sarcastic, suggestive etc
   We were provided with the csv files created from the jsons of reviews and business for simplicity and I am using these csv files.
   Business csv contains business ids and categories files from the business json
   Reviews csv contains business ids and reviews from the review json

Topics	No. of Documents	No. of Unique terms	No. of unique terms after removing stop words	No. of unique terms after Stemming
alt.atheism	480	10879	10808	7427
comp.graphics	584	12076	12010	9400
comp.os.ms-windows.misc	591	24271	24206	22401
comp.sys.ibm.pc.hardware	590	9234	9168	7305
comp.sys.mac.hardware	578	8557	8490	6560
comp.windows.x	593	13265	13199	10658
misc.forsale	585	9734	9666	7875
rec.autos	594	10475	10405	7844
rec.motorcycles	598	10490	10422	7884
rec.sport.baseball	597	9225	9157	7122
rec.sport.hockey	600	11399	11329	8833
sci.crypt	595	13545	13475	9610
sci.electronics	591	10219	10153	7638
sci.med	594	14413	14342	10566
sci.space	593	13804	13735	9921

soc.religion.christian	599	12800	12728	8697
talk.politics.guns	546	12905	12835	9152
talk.politics.mideast	564	15458	15387	10838
talk.politics.misc	465	12705	12634	8712
talk.religion.misc	377	10815	10746	7616

2. **Experiments** :

a. Data Processing :

For 20NG dataset I am Using below groups:
rec.sport.hockey
rec.sport.baseball
sci.space

Among these rec.sport.hockey and rec.sport.baseball and similar to each other
And sci.space is dissimilar from both rec.sport.hockey and rec.sport.baseball

For Yelp dataset I am choosing Infer Categories subproblem and using data from below categories:
American
Asian
Thai
Sports

Among these American, Asian and Thai are similar as these are types of restaurants
Sports is dissimilar from all three.

```
#loading selected data from 20NG data set from the folders
data2<-
c("20_newsgroups\\rec.sport.hockey","20_newsgroups\\rec.sport.baseball","20_newsgroups\\s
ci.space") #Read the text files for 3 groups

news <- Corpus(DirSource(data2, recursive=TRUE),readerControl = list(reader=readPlain))# Load
the data as a corpus
```

```
#loading selected Yelp data from the business.csv and reviews.csv files provided too us
business.category <- read.csv  (file="yelp_academic_dataset_business.csv", header=TRUE)

business.reviews <- read.csv  (file = "yelp_academic_dataset_review.csv", header=FALSE ,nrows
= 5000000,sep="," )

#Selecting Asian , American, Thai and Sports categories
business.seletedCategories <-
business.category[grepl("American",business.category$categories)
                        | grepl("Asian",business.category$categories)
                        | grepl("Thai",business.category$categories)
                        | grepl("Sports",business.category$categories),]

#removing b' at the beginning and ' at the end of business_id in business.csv
business.category$business_id <- gsub("^b'", "", business.category$business_id)
business.category$business_id <- gsub("'$", "", business.category$business_id)

#merging both categories and reviews in the common business_id field
filtered <- merge ( x = business.seletedCategories , y = business.reviews, by.x=c("business_id"),
by.y=c("business_id"))

#collapsing the filtered data on the categories, so that reviews for similar categories are placed
together
collapsed <- aggregate(text ~categories, data = filtered, toString)

t <- data.frame(collapsed$text, stringsAsFactors = FALSE)
corpus <- Corpus(DataframeSource(t))
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeWords, stopwords("english"))
corpus <- tm_map(corpus, stemDocument)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus,content_transformer(PlainTextDocument))
```

Creating DocumentTermMatrix :

```
dtm <- DocumentTermMatrix(news,control=list(wordLengths=c(4,Inf)))#considering only terms
whose length is greater thann 4
```

Creating TermDocumentMatrix :

```
tdm <- TermDocumentMatrix(news, control=list(wordLengths=c(4,Inf)))
```

Document-term matrix is an M * N matrix , where M is number of documents and N is number is words i n the entire corpus, and hold number of times a word is occurring in a particular document.
Term-document matrix is transpose of this matirx.

Below is a snapshot of DTM created from the corpus(Showing for 20Groups Data):

```
> dtm #document matrix for NewsGroups I have selected
        <<DocumentTermMatrix (documents: 3000, terms: 84623)>>
        Non-/sparse entries: 424546/253444454
        Sparsity           : 100%
        Maximal term length: 295
         Weighting          : term frequency (tf)
```

Top 10 words from the Term-document matrix :

```
                                word freq
that                            that 7055
have                              have 3868
this                              this 3726
with                              with 3322
from:                            from: 3041
date:                            date: 3019
subject:                      subject: 3015
newsgroups:                newsgroups: 3011
message-id:                message-id: 3009
path:                            path: 3003
```

In the preprocessing steps we remove all unwanted data which do not give any information and clean the data i.e, converting the text too lower case, remove punctuations, stopwords, remove some keywords repeating in all the documents, do stemming to reduce the words to their root forms, remove white spaces etc.

Below are the snapshots of DTM are processing(before stemming):

```
> dtm.preprocessed.notstemmed <- DocumentTermMatrix(news,control=list(wordLen
gths=c(4,Inf)))
> dtm.preprocessed.notstemmed
<<DocumentTermMatrix (documents: 3000, terms: 36649)>>
Non-/sparse entries: 278371/109668629
Sparsity           : 100%
Maximal term length: 234
Weighting          : term frequency (tf)
```

Here we can clearly see the difference in the number of terms. Preprocessing (removing stopwords, common words (headers for example), punctuations, whitespaces) is an important step and has a significant impact on the data that we are operating on.

Top 10 words from TDM after preprocessing and before stemming :

```
                                word freq
will                            will 2257
game                            game 1810
space                          space 1765
university                university 1599
team                            team 1584
like                            like 1472
just                            just 1468
think                          think 1349
year                            year 1344
first                          first 1209
```

Below is the code snippet from my solution which does preprocessing as mentioned above :

```
#Preprocessing the data for 20NG dataset
skipWords <- function(x) removeWords(x,
c("xref","path","from","newsgroups","subject","message-id","messageid","date","article-
id","article","references","organization","nntp-posting-host", "nntppostinghost", "lines",
"expires", "reply-to", "followup-to", "distribution" , "keywords", "summary" , "sender", "writes",
stopwords("english")))
funcs <- list(removePunctuation, stripWhitespace, removeNumbers,
skipWords,content_transformer(tolower))
news <- tm_map(news, FUN = tm_reduce, tmFuns = funcs)
news <- tm_map(news,content_transformer( PlainTextDocument))

#preprocessing for yelp dataset
t <- data.frame(collapsed$text, stringsAsFactors = FALSE)
corpus <- Corpus(DataframeSource(t))
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeWords, stopwords("english"))
corpus <- tm_map(corpus, stemDocument)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus,content_transformer(PlainTextDocument))
```

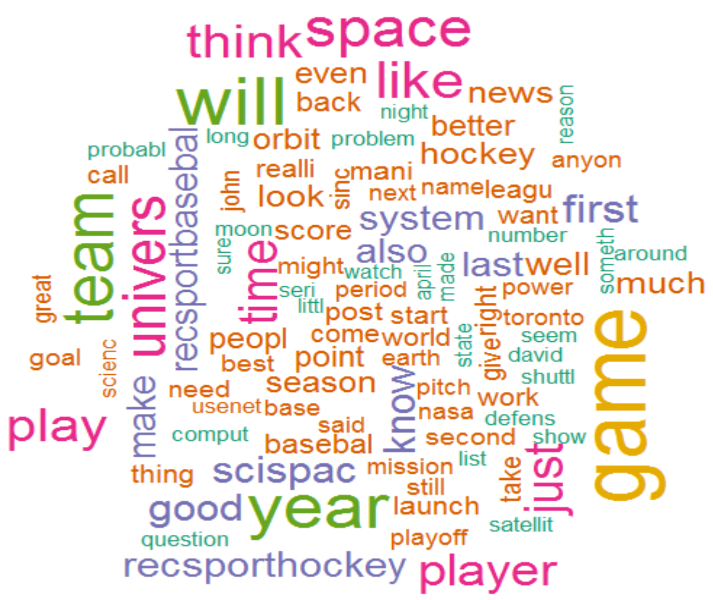
Below are the snapshots of DTM affter Stemming) and the word cloud for the same:

```
        > dtm.stemmed <- DocumentTermMatrix(news,control=list(wordLengths=c(4,
Inf)))
> dtm.stemmed
<<DocumentTermMatrix (documents: 3000, terms: 28608)>>
Non-/sparse entries: 257729/85566271
Sparsity           : 100%
Maximal term length: 234
Weighting          : term frequency (tf)
```

Here we can see the number of terms been decreased even more. What stemming does is converts words into their root forms , for ex. Education will be converted to educate.

Top 10 words fromm TDM after stemming :

```
                           word freq
game                       game 2843
will                       will 2325
team                       team 2117
year                       year 2112
space                     space 1771
univers                 univers 1722
like                       like 1678
play                       play 1507
think                     think 1499
time                       time 1479
```



Typically even after preprcessing and stemming DTM will be very sparse (has 0 or some low value) as many words occur only in few documents(We can observe the same from the above snapshot of my DTM for 20Groups data).
So we do remove sparse terms to drop those very less frequent words by doing below step :

```
dtm.sparse = removeSparseTerms(dtm, 0.75)# we are removing terms which are which are not occurring in 97% of the documents atleast.
```

Many common words like articles, conjunctions occur more frequently in documents than other words. Therefore, using counts can be deceiving. To overcome these problem and punish these words for occurring in all the documents we do term frequency-inverse document frequency weight instead of counts as shown below :

```
tdm.tf.idf.stemmed <- weightTfIdf(tdm.stemmed)
```

After all these steps we have our data ready for experimenting and finding the hidden information.

3. **Experiments** :

Here I am presenting the results I got when by plotting the clusters directly from the data, using Latent Semantic Analysis(A Dimensionality Reduction Technique) with Singular Value Decomposition and Latent Dirichlet Allocation(another DR technique).

Latent Semantic Analysis with SVD:

This technique decomposes our DTM into 3 matrices and uncovers the space represented by it. This method find hidden concepts in the data.

A = U d Vtranspose

Concepts are represented by the singular vectors

Each document and word are reprented in terms of the concept vectors, i.e. in terms of projection on the singular

vector representing **.**

Then we reduce all three matrices into k dimension, by selecting only k largest eigen values in the diagonal matrix d.

Below is the R code for the same :

```
svd <-svd(dtm.tf.idf.sparsed)

D <- diag(svd$d)

numberOfterms = 200#,100,200

diagD <- D[1:numberOfterms,1:numberOfterms]

documentVector<-svd$u[,1:numberOfterms] %*% diagD #creating the document vector

wordVector<-diagD %*%t(svd$v[,1:numberOfterms]) #creating the word vector
```

Since U contains the eigen values of document-document matrix, this when multiplied with the diagonal matrix containing the eigen values, gives the importance of document

Since V contains the eigen values of document-document matrix, this when multiplied with the diagonal matrix containing the eigen values, gives the importance of term

K-means Clustering :

Is a prototype based, partitional based clusteinng technique that attemps to find the user-specified number of clusters in the data, which are represented by their centroids.

We must have an objective function to measure the quality if clustering. In this case we use Sum of Squares Error (SSE), which is also known as SCATTER. In SSE we calculate the error of each data point, i.e its Euclidean distance to the closest centroid and them compute the total sum of squared error by summing over for all datapoints in all clusters.

Plotting k-means clusters in R:

```
m <- as.matrix(dtm.tf.idf.stemmed)

rownames(m) <- 1:nrow(m)

norm_eucl<- function(m)m/apply(m,MARGIN=1, FUN=function(x)sum(x^2)^.5)

m_norm<-norm_eucl(m)

cl<-kmeans(m_norm,2)

table(cl$cluster)

plot(prcomp(m_norm)$x, col=cl$cl)
```

Latent Dirichlet Allocation :

LDA is a generative probabilistic model of the corpus.

LDA models how the data was generated in order to categorize a signal. LDA asks the question to itself : based on my generation assumption, which categories are more likely to generate this signal.

Basic idea on which LDA operates is that : documents are generated as random mixtures over latent topics, where each topic is characterzed by a distribution over words.

LDA involves 3-levels, and notably the topic node is sampled repeatedly within the document. Under this model documents can be associated with multiple topics.

My Implementation in R :

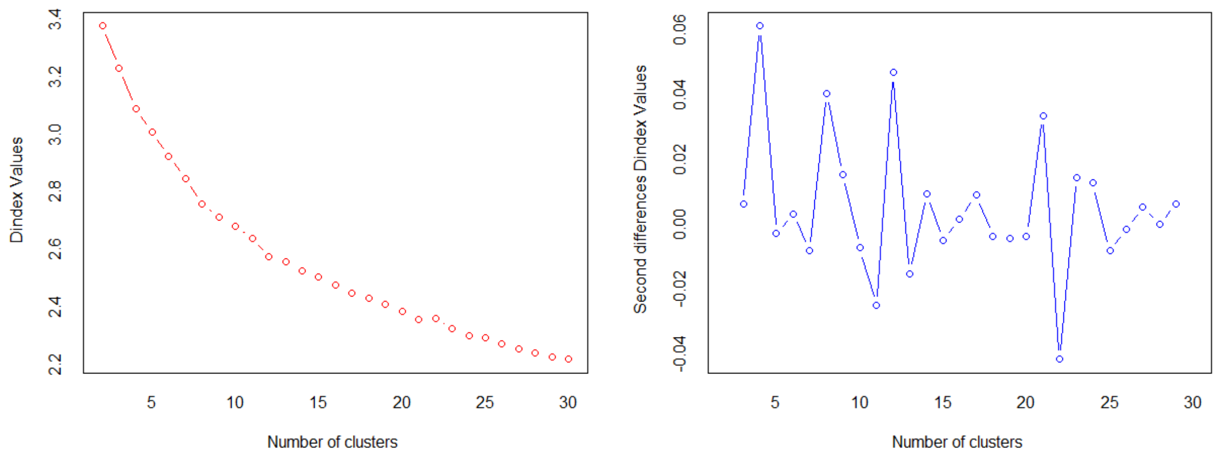
```
lda <- LDA(dtm,300,method="Gibbs")

lda.topics.300 <- as.matrix(topics(lda))

head(lda.topics.300,20)

lda.terms <- as.matrix(terms(lda,))

head(lda.terms,20)

topicProbabilities <- as.data.frame(lda@gamma)



m <- as.matrix(topicProbabilities)

rownames(m) <- 1:nrow(m)

norm_eucl <- function(m) m/apply(m, MARGIN=1, FUN=function(x) sum(x^2)^.5)

m_norm <- norm_eucl(m)

cl<- kmeans(m_norm, 21)

table(cl$cluster)

plot(prcomp(m_norm)$x, col=cl$cl)

cl$withinss

cl$tot.withinss
```

Results for 20 Groups data :

Number of clusters is 2, based on NBClust

```
nc <- NbClust(dtms, min.nc=2, max.nc=30, method="kmeans")
table(nc$Best.n[1,])
```

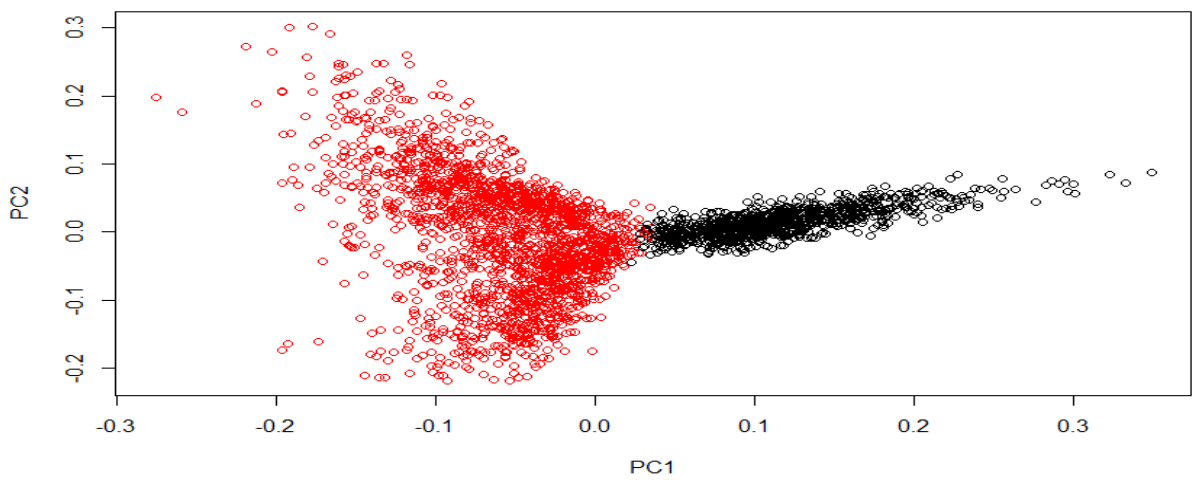
```
 0  2  3  5  7 12 17 20 21 22 30
 2  6  5  1  2  1  1  1  4  1  2
```



Plotting data (Term Frequency – Inverse Document Frequency):

```
table(cl$cluster)
```

```
   1    2
 987 2013
SSE : 2941.527
There exists overlapping across clusters
```



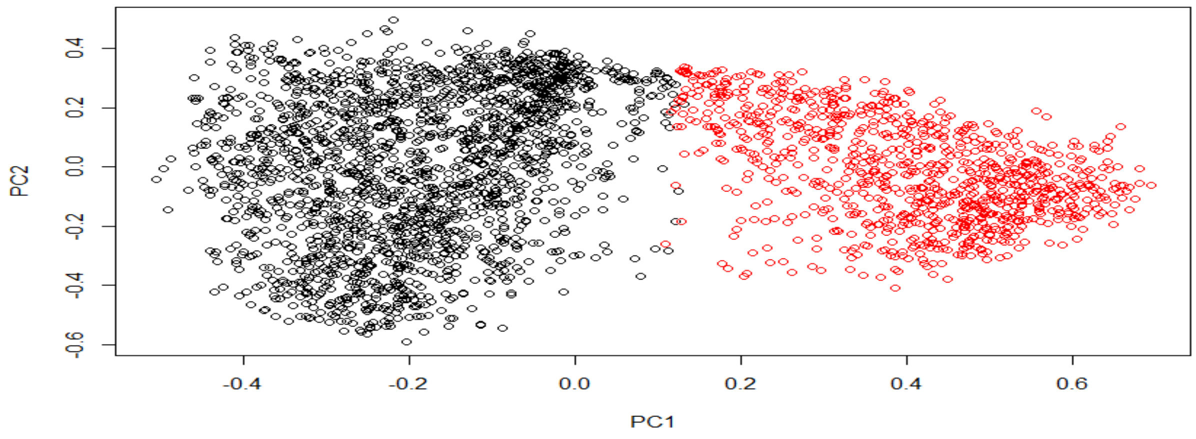
LSA : Plotting document vector from SVD :

D = 50 (Reducing the dimension to 50 and considering only highest 50 Eigen values in diagonal bector D)

```
>table(cl$cluster)

  1    2
997 2003
```

```
SSE : 2062.121
```

```
There clusters are separated out
```

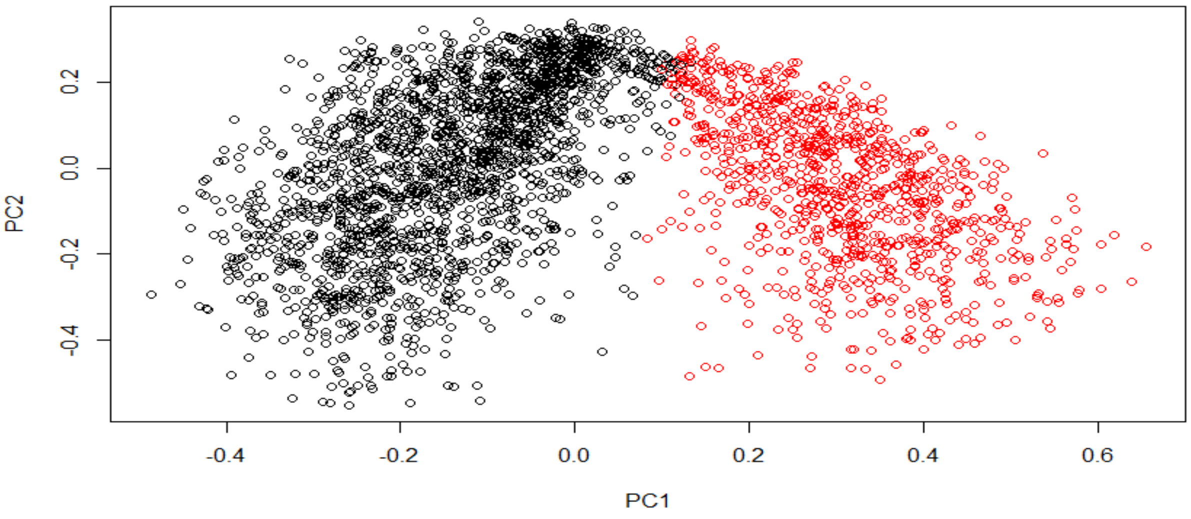


D = 100(Reducing the dimension to 100 and considering only highest 100 Eigen values in diagonal vector D)

```
>> table(cl$cluster)

  1    2
2052  948
```

```
SSE : 2470.836
```

There is some overlapping among the two clusters, but better than tf-idf.

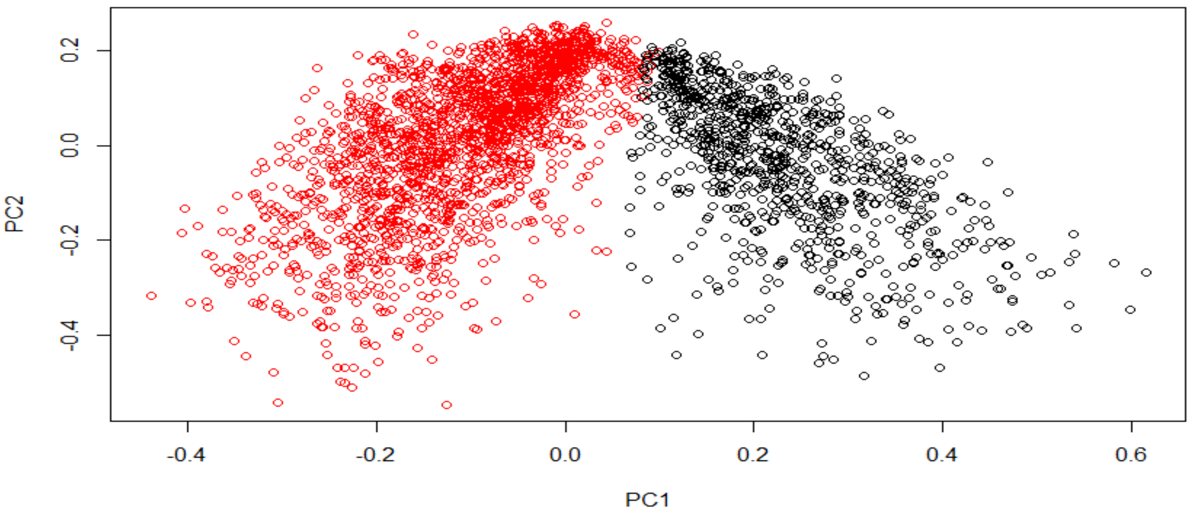


D = 200(Reducing the dimension to 200 and considering only highest 20 Eigen values in diagonal vector D)

```
> table(cl$cluster)

  1    2
947 2053
```

```
SSE : 2696.556
```

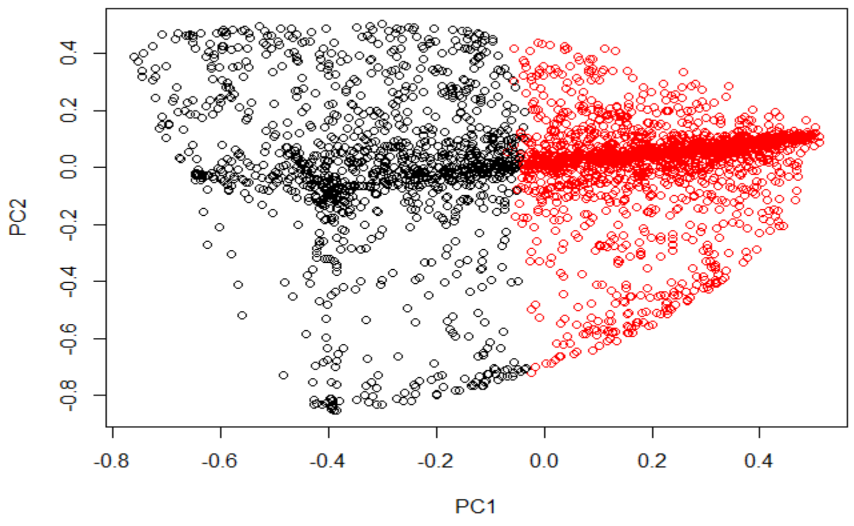
There is some overlapping among the two clusters, but better than tf-idf.



LDA (Plotting the topic probabilities) :

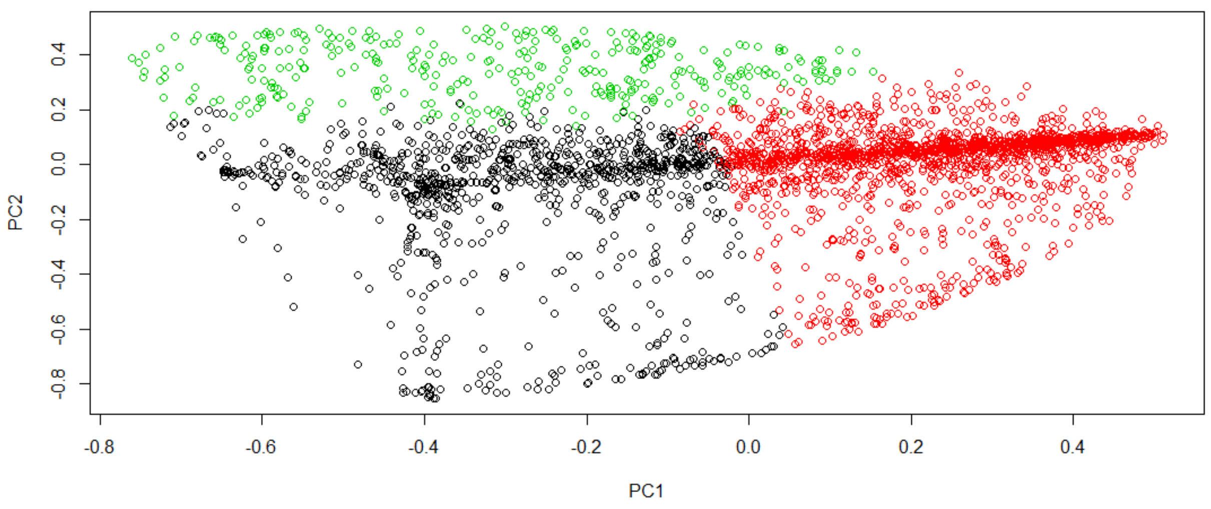
K=2

```
> table(cl$cluster)

   1    2
1216 1784

SSE : 1933.129
```



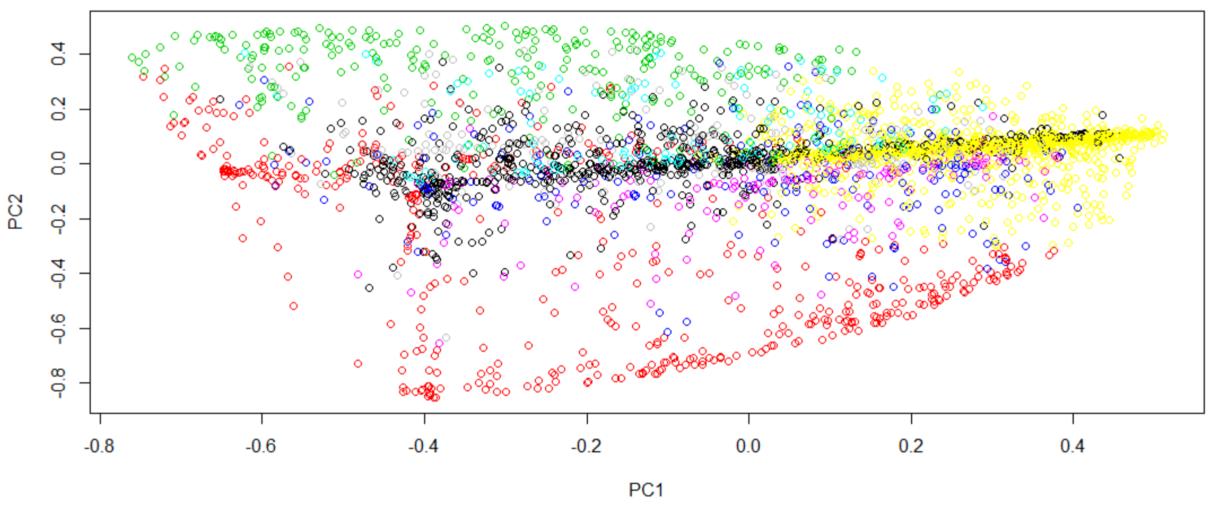
K =3

```
table(cl$cluster)

   1    2    3
 962 1694  344

SSE :  1831.981
```



K=21

```
> table(cl$cluster)

  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19 
 20  21 
453 175  23  94  57 109 950 104 196  33 225  32  11  57   8  81  15 255  23 
 52  47 
SSE : 1224.862
```



```
> lda.terms <- as.matrix(terms(lda,))
> head(lda.terms,20)#top terms in the topics learnt
          [,1]
Topic 1   "benifit"
Topic 2   "broadcast"
Topic 3   "dri"
Topic 4   "ottawa"
Topic 5   "ether"
Topic 6   "jon"
Topic 7   "game"
Topic 8   "aplandmalabcca"
Topic 9   "sat"
Topic 10  "clutch"
Topic 11  "xnewsread"
Topic 12  "gonna"
Topic 13  "coegalonlarcnasagov"
Topic 14  "bondracot"
Topic 15  "pgfsrlcacsusledu"
Topic 16  "ulf"
Topic 17  "figur"
Topic 18  "gray"
Topic 19  "german"
Topic 20  "perkyacsbuedu"
```

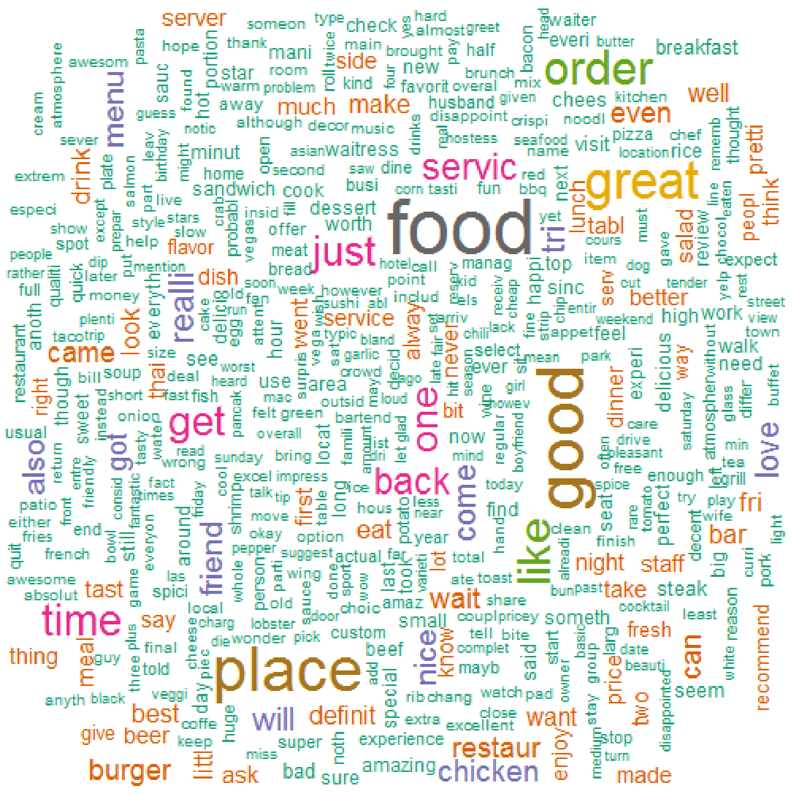
Comparison and Results for 20NG datasets :

We can clearly see that the SSE of the LDA is lower that that of LSA and tf-tdf

Model		SSE
Tf-idf		2941.527
LSA	D = 50	2062.121
	D = 100	2470.836
	D = 200	2696.556
LDA	K = 2	1933.129

For Yelp Dataset :

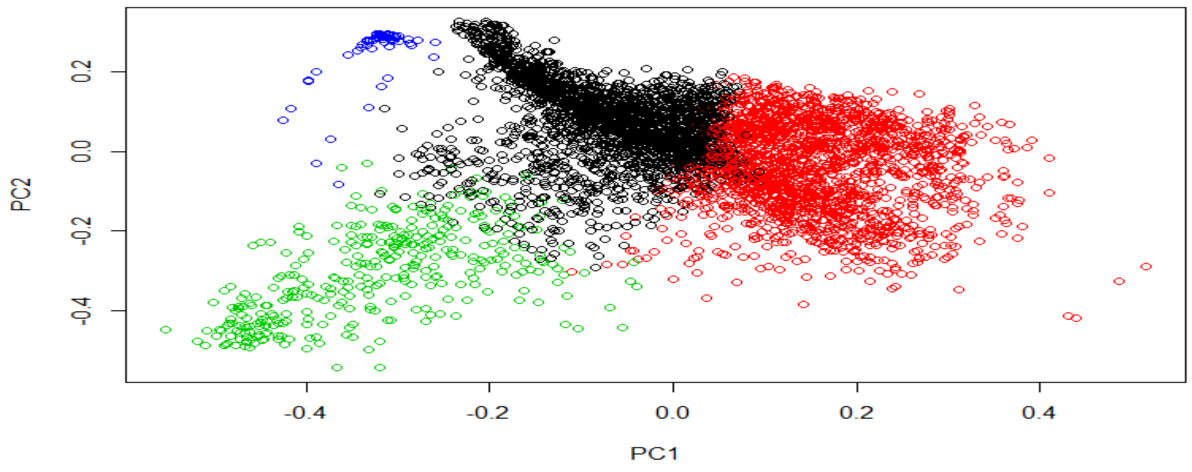
Worcloud :



Number of clusters is set to 4.

Plotting data (Term Frequency – Inverse Document Frequency):

```
> table(cl$cluster)

  1   2   3   4 
736 547  31  46 

SSE : 3654.87
```

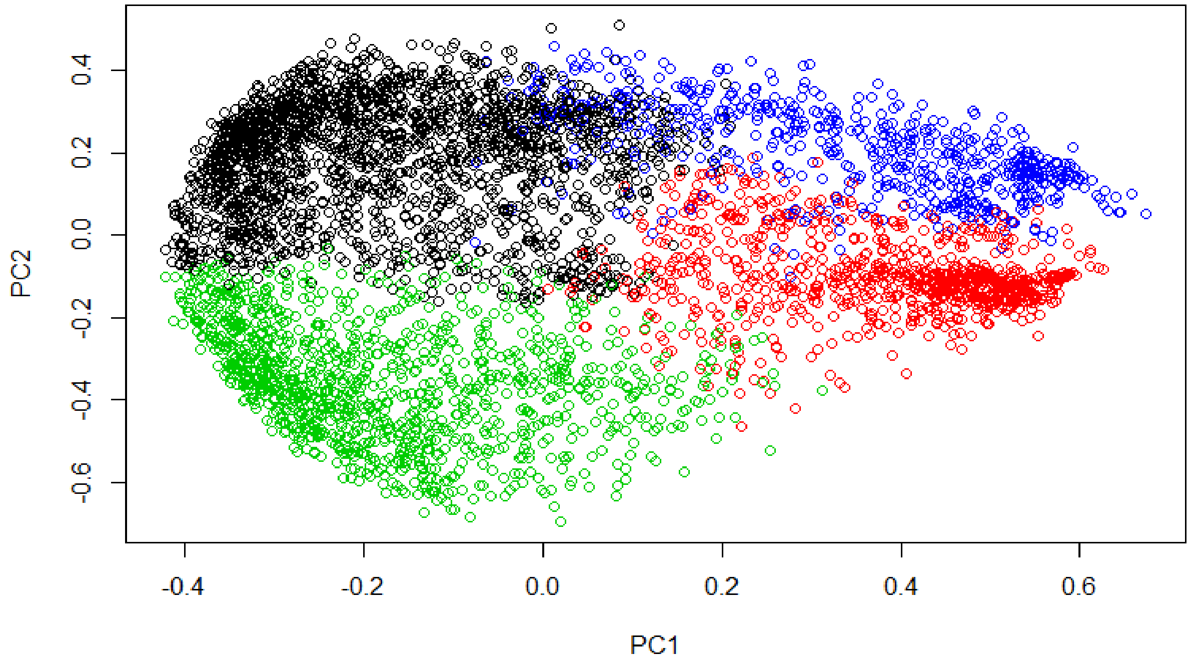


LSA :

D = 50

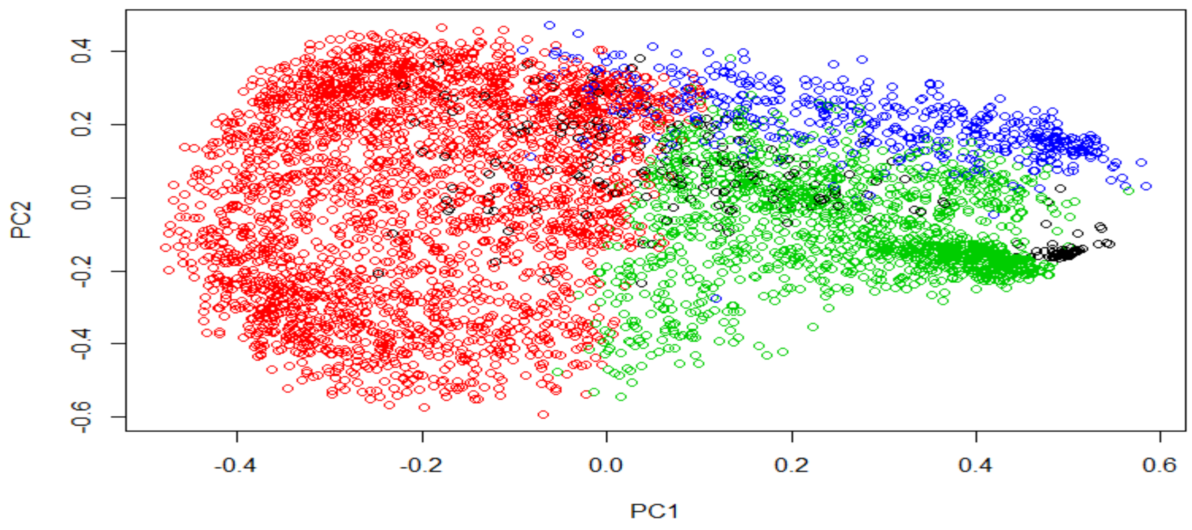
```
SSE: 2201.57
```

```
 Most Representative words :
        "war"      "und"      "laden"    "den"      "bin"      "mein"     "a
 mbient" "gym"      "buffet"   "asia
```



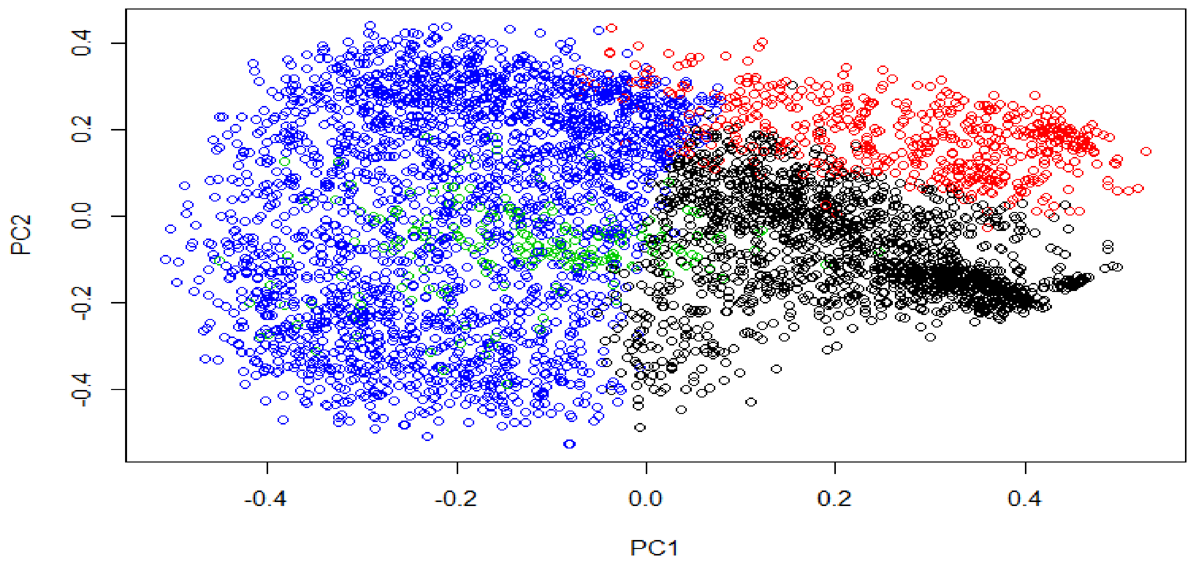
D = 100

```
SSE : 2954.02
Most Representative  words :
"war"     "und"     "laden"   "den"     "bin"     "mein"    "ambient" "gym"
"buffet"  "asia"
```



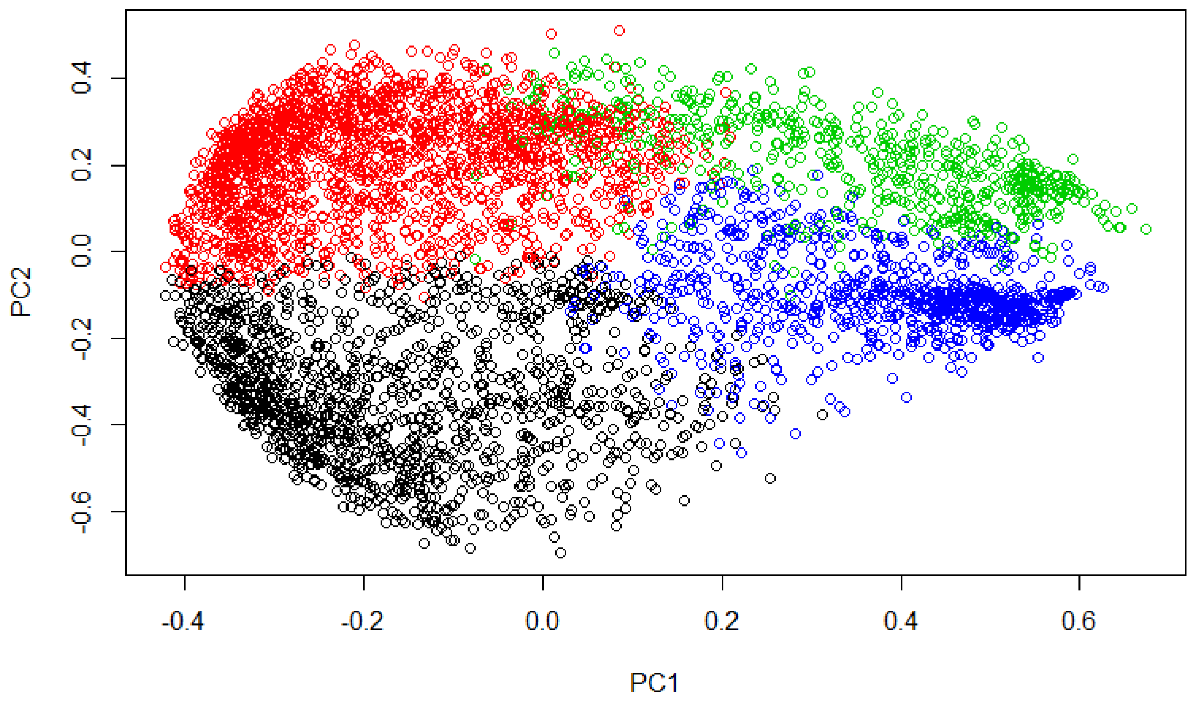
D = 200

```
  SSE : 3304.342
  Most Representative words : "war"     "und"     "laden"   "den"     "b
in"     "mein"    "ambient" "gym"     "buffet"  "asia"
```



LDA(Plotting the topic probabilities) :

```
SSE : 1869.126
```



Comparison and Results for 20NG datasets :

We can clearly see that the SSE of the LDA is lower that that of LSA and tf-tdf

Model		SSE
Tf-idf		3654.87
LSA	D = 50	2201.57
	D = 100	2959.02
	D = 200	3304.342
LDA	K = 4	1869.126

Results Summary:

Data	Model		SSE
20NewsGroups	Tf-idf		2941.527
	LSA	D = 50	2062.121
		D = 100	2470.836
		D = 200	2696.556
	LDA	K = 2	1933.129
Yelp	Tf-idf		3654.87
	LSA	D = 50	2201.57
		D = 100	2959.02
		D = 200	3304.342
	LDA	K = 4	1869.126

Conclusion:

I can conclude from the experiments above that LDA perform better than LSA which performs better than TF-IDF.

TF-IDF just uses the term frequencies and the inverse document frequencies of terms , so unrelated words or documents can fall into the same category just because they have similar frequencies across the corpus.

LSA is an effective technique that brings down the hidden information be plotting the data into lower dimensions as this process DE noises the data and makes it more cleaner and meaningful.

LDA takes few steps ahead and assign probabilities to terms and topics such that there are chances of a term belonging to various topics. Similarly to the topics in the model This provides more flexibility to the model.

I understood that LDA is a better model than LSA. I also learnt to implement these techniques in R. It was a good practice, learning and understanding process.