CS 522

Homework Assignment 1

NAME: DHAYALINI NAGARAJ

A.NO :A20359686

20 NEWSGROUP DATA

The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned across 20 different newsgroups. It was collected by Ken Lang. It is the popular data set for experiments in text mining. This data set is organised into 20 different newsgroups which corresponds to different topics and some are related to each other.Eg: comp.sys.ibm.pc.hardware / comp.sys.mac.hardware. The following is the list of the 20 newgroups, partitioned according to the subject matter.

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

From analysing the above data, some groups are closely related to each other while the other groups are distinct to each other.

Version 1: Original 20 Newsgroups data set
Version 2: 20 Newsgroups sorted by date; duplicates and some headers removed (18846 documents)
Version 3: 20 Newsgroups; duplicates removed, only "From" and "Subject" headers (18828 documents).It does not include any cross-posts and includes only the “From” and the “Subject” headers.

Let us use version 2 “bydate” for our analysis purpose since cross-experiment comparison is easier (no randomness in train/test set selection). It is sorted by date into training (60%) and testing (40%) sets does not include cross-posts (duplicates) and does not include newsgroup-identifying headers (Xref, Newsgroups, Path, Follow-up-To, Date). In addition to that, newsgroup-identifying information has been removed and it's more realistic because the train and test sets are separated in time.

```
# Install
install.packages("tm") # for text mining
```

```
install.packages("SnowballC") # for text stemming
install.packages("wordcloud") # word-cloud generator
install.packages("RColorBrewer") # color palettes
# Load
library("tm")
library("SnowballC")
library("wordcloud")
library("RColorBrewer")

data<-c("D:\\20news-bydate-train\\talk.religion.misc")
data<- tm_map(data, tolower) ## Convert to Lower Case
data<- tm_map(data, removePunctuation) ## Remove Punctuations
data<- tm_map(data, removeNumbers) ## Remove Numbers
data<- tm_map(data, stripWhitespace) ## Eliminate Extra White Spaces
data<- tm_map(data, PlainTextDocument)
data <- tm_map(data, removeWords, stopwords("english")) ## Remove Stopwords
data <- tm_map(data, stemDocument) ## Stemming
```

Document matrix is a table containing the frequency of the words.

```
dtm <- DocumentTermMatrix(data,control=list(wordLengths=c(4,Inf)))
dtm
```

The unique number of terms for each document was collected by doing this data preprocessing steps. The unique terms count was collected after elimination white space, after removing stopwords and after stemming the data. The following table contains no. of documents and words corresponding to the groups. From the below table, we get to know the count of the terms gets decreased as the data is processed which makes the process more efficient.

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

Data Preprocessing and Experiments:

Data processing is commonly kmown as collection and manipulation of items of data to produce meaningful information. The change in the dataset will be detectable by an observer. Preprocessing of text documents is a series of simple steps to remove the un-important values from the text that can be alter our experiments. Text preprocessing steps usually have a big impact on the quality of the results for clustering and classification. Typical preprocessing steps include conversion to lower case, removal of stop words, stemming, removal of numbers, removal of punctuation. As further pre-processing, we usually remove words that are too infrequent

Data groups:

For our analysis I have considered two dataset. Each dataset is of three groups.

1. The first data set contains similar groups comp.os.ms-windows.misc, comp.sys.ibm. pc.hardware, comp.sys.mac.hardware. It is known as Data1.
2. The second data set contains groups which are not similar and they are sci.space , rec.motorcycles , soc.religion.christian. It is known as Data2.

Experiments:

Text mining and wordcloud packages are required. They can be installed and loaded using the R code mentioned above. Load the datasets Data1 and Data2 and let us analyse the data before processing it. The text is loaded using Corpus() function from text mining (tm) package. Corpus is a list of a document

```
data1<-c("D:\\20news-bydate-train\\comp.os.ms-windows.misc","D:\\20news-bydate-
train\\comp.sys.ibm.pc.hardware","D:\\20news-bydate-train\\comp.sys.mac.hardware")
data1 <- Corpus(DirSource(data1, recursive=TRUE),readerControl = list(reader=readPlain))

data2<-c("D:\\20news-bydate-train\\sci.space","D:\\20news-bydate-
train\\rec.motorcycles","D:\\20news-bydate-train\\soc.religion.christian")
```

data2 <- Corpus(DirSource(data2, recursive=TRUE),readerControl = list(reader=readPlain))

Before Processing:

Data 1

<<DocumentTermMatrix (documents: 1759, terms: 55136)>>
Non-/sparse entries: 179540/96804684
Sparsity : 100%
Maximal term length: 214
Weighting : term frequency (tf)

<<TermDocumentMatrix (terms: 55136, documents: 1759)>>
Non-/sparse entries: 179540/96804684
Sparsity : 100%
Maximal term length: 214
Weighting : term frequency (tf)

Data 2
<<DocumentTermMatrix (documents: 1790, terms: 55272)>>
Non-/sparse entries: 248844/98688036
Sparsity : 100%
Maximal term length: 80
Weighting : term frequency (tf)

<<TermDocumentMatrix (terms: 55272, documents: 1790)>>
Non-/sparse entries: 248844/98688036
Sparsity : 100%
Maximal term length: 80
Weighting : term frequency (tf)

Frequency of the words before processing

The following code can be used to generate frequent items:
m <- as.matrix(tdm)
v <- sort(rowSums(m), decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
head(d, 10)

Data 1

Max	3289
That	2807
With	2496
Have	2265
This	1864
From	1778
Subject	1773

Lines	1762
Organization	1708
From	1073

Data2

That	6678
This	3017
Have	2714
With	2542
from:	1814
subject:	1796
lines:	1794
They	1769
From	1765
Organisation	1692

Text Preprocessing and Analysis

Many of the words that are too frequent are called stop words and they will be removed in the stop word removal step. Other words can also be frequent in the document. If the data contains headers or metadata, we can remove that for better clustering and classification. The tm_map() function is used to remove white space, to convert the text to lower case, to remove common stopwords like 'the', "we". The information value of 'stopwords' is near zero due to the fact that they are so common in a language.

Most important preprocessing step is to make a text stemming which reduces words to their root form. In other words, this process removes suffixes from words to make it simple and to get the common origin. For example, a stemming process reduces the words "moving", "moved" and "movement" to the root word, "move".

Let us remove the headers and other less important words using the following commands :

```
data1 <- tm_map(data1, removeWords,"Subject")
data1 <- tm_map(data1, removeWords,"Organization")
 data1 <- tm_map(data1, removeWords,"writes")
 data1 <- tm_map(data1, removeWords,"From")
 data1 <- tm_map(data1, removeWords,"lines")
 data1 <- tm_map(data1, removeWords," NNTP-Posting-Host")
 data1 <- tm_map(data1, removeWords,"article")
data1 <- tm_map(data1, tolower)
 data1 <- tm_map(data1, removeWords, stopwords("english"))
 data1 <- tm_map(data1, removePunctuation)
 data1 <- tm_map(data1, removeNumbers)
 data1 <- tm_map(data1, stripWhitespace)
 data1 <- tm_map(data1 , PlainTextDocument)
 dtm1 <- DocumentTermMatrix(data1,control=list(wordLengths=c(4,Inf)))
```

Removing Stop Words:

Stopwords are defined as words in a language that are so common that their information value is practically null. Some common stopwords are words such as a, and, but, the, and if, among many

others. It is common practice in text analysis to reduce the number of 'noisy' words in the data by removing stopwords.

Document Term Matrix:

A document-term matrix or term-document matrix is a mathematical matrix that describes the frequency of terms that occur in a collection of documents. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms.

Data1
DocumentTermMatrix (documents: 1759, terms: 34045)>>
Non-/sparse entries: 125551/59759604
Sparsity : 100%
Maximal term length: 166
Weighting : term frequency (tf)

Data2
<<DocumentTermMatrix (documents: 1790, terms: 26502)>>
Non-/sparse entries: 176838/47261742
Sparsity : 100%
Maximal term length: 78
Weighting : term frequency (tf)

By analysing the document term matrix before processing and after processing, we can find that there are huge difference in the number of terms in the matrix. The less important words are removed which makes the dataset more useful for further experiments.

Stemming:

Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, baseor root form—generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. So let us do stemming for our data and reduce it for further process.

Data1
data1 <- tm_map(data1, stemDocument)
data1 <- tm_map(data1 , PlainTextDocument)
dtm1 <- DocumentTermMatrix(data1,control=list(wordLengths=c(4,Inf)))

<<DocumentTermMatrix (documents: 1759, terms: 30370)>>
Non-/sparse entries: 117253/53303577
Sparsity : 100%
Maximal term length: 166
Weighting : term frequency (tf)

Data2

<<DocumentTermMatrix (documents: 1790, terms: 18845)>>
Non-/sparse entries: 162079/33570471
Sparsity : 100%
Maximal term length: 78

Weighting : term frequency (tf)

Now after stemming , after comparing the results of document term matrix, it is clearly seen that the number of terms in the data has been reduced. So the terms have been stemmed and grouped with their root words. This dataset is processed and clustering and classification can be carried out using this dataset and the results of it would be more realistic that processing with an unprocessed data.

Frequency of word count

Data 1

Max	3318
Lines	1761
Windows	1033
Will	926
Scsi	912
Drive	886
nntppostinghost	791
Card	776
Problem	764
Know	731

Data2

Lines	1793
Will	1415
Space	1140
Just	1010
Like	966
People	900
Know	853
Think	819
Also	754
Time	642

Let us compare the frequent terms wordcount before processing and after processing. Since the headers are removed from the dataset, some of the repeated words are removed and the count of other important words have been calculated and displayed. This shows the frequent words in the dataset differs when the text has been processed.

Remove Sparse Data or Pruning words

removeSparseTerms(), sparsity refers to the threshold of relative document frequency for a term, above which the term will be removed. Relative document frequency here means a proportion.We have a large corpus of documents. We have some words that are less frequent. Since they show up only once they are either always associated to one class or the other. So inorder to reduce or remove the less frequent words we can do sparsing by giving a threshold value. So the words whose frequency are below the threshold will be removed.

Data1

dtm1sparse2 <- removeSparseTerms(dtm1, 0.9)

DocumentTermMatrix (documents: 1759, terms: 63)>>
Non-/sparse entries: 21025/89792
Sparsity : 81%
Maximal term length: 15
Weighting : term frequency (tf)

Data2

dtm2sparse2 <- removeSparseTerms(dtm2, 0.9)
DocumentTermMatrix (documents: 1790, terms: 110)>>
Non-/sparse entries: 32809/164091
Sparsity : 83%
Maximal term length: 15
Weighting : term frequency (tf)

After sparsing the document term matrix of both the data the less frequent items have been reduced and terms of both dataset have been reduced a lot and the counts are seen as 63 and 110. The clustering is always based on the frequency count and since the sparsity is reduced only the most important words of the document will be considered and the clustering will give more accuracy and the error in clustering and classification will be reduced.

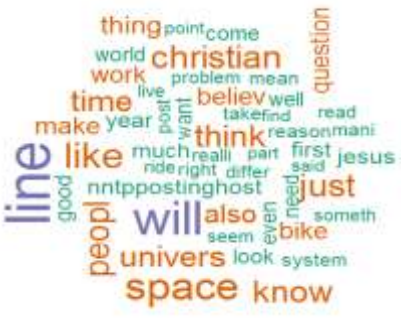
WordCloud:
Humans are generally strong at visual analytics. That is part of the reason that these have become so popular. So let us represent our frequent words using wordcloud after removing sparse the dataset.

freq1data2 = data.frame(sort(colSums(as.matrix(dtm1sparse2)), decreasing=TRUE))
wordcloud(rownames(freq1data2), freq1data2[,1], max.words=50, colors=brewer.pal(1, "Dark2"))

The importance of words can be illustrated as a word cloud as follow :

Arguments of the word cloud generator function :
- words : the words to be plotted
- freq : their frequencies
- min.freq : words with frequency below min.freq will not be plotted
- max.words : maximum number of words to be plotted • random.order : plot words in random order. If false, they will be plotted in decreasing frequency
- rot.per : proportion words with 90 degree rotation (vertical text) • colors : color words from least to most frequent. Use, for example, colors =“black” for single color

Word cloud data1



Word cloud data2

TF-IDF

TF-IDF stands for "Term Frequency, Inverse Document Frequency". The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. It is a way to score the importance of words (or "terms") in a document based on how frequently they appear across multiple documents. We can find the huge difference in the word cloud which makes that TF-IDF efficient.

dtm_tfxidf1<- weightTfIdf(dtm1)

Data1

DocumentTermMatrix (documents: 1759, terms: 30370)>>
Non-/sparse entries: 117253/53303577
Sparsity : 100%
Maximal term length: 166
Weighting : term frequency - inverse document frequency (normalized) (tf-idf)

Data2

DocumentTermMatrix (documents: 1790, terms: 18845)>>
Non-/sparse entries: 162079/33570471
Sparsity : 100%
Maximal term length: 78
Weighting : term frequency - inverse document frequency (normalized) (tf-idf)

Data1

Data2

Remove Sparse terms for TF-IDF : review_dtm_tfidf1 = removeSparseTerms(dtm_tfxidf1, 0.95)

Data1

DocumentTermMatrix (documents: 1759, terms: 210)>>
Non-/sparse entries: 38654/330736
Sparsity : 90%
Maximal term length: 15
Weighting : term frequency - inverse document frequency (normalized) (tf-idf)

Data2

DocumentTermMatrix (documents: 1790, terms: 328)>>
Non-/sparse entries: 59440/527680
Sparsity : 90%

Maximal term length: 18
Weighting : term frequency - inverse document frequency (normalized) (tf-idf)

Reduce TF-IDF data2

Clustering Analysis

K means clustering is a Partitional clustering approach. Each cluster is associated with a centroid (center point) and each point is assigned to the cluster with the closest centroid. Number of clusters, K, must be specified.

The document term matrix is used to perform K means clustering and the results are used to compare Latent Semantic Indexing using Single Value Decomposition. We need to map the document and the term in the semantic space. The datasets are clustered using K means and this clustering will be used to evaluate the performance of LSA on clustering after we perform SVD.

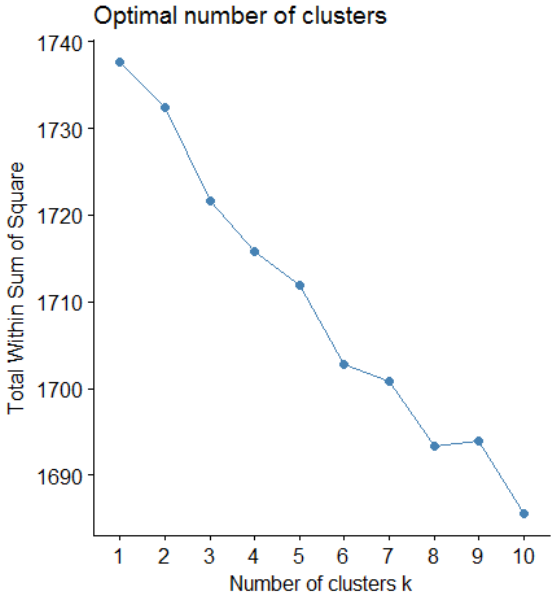
The following code can be used to install the required packages:

```
if(!require(devtools))
install.packages("devtools")
devtools::install_github("kassambara/factoextra")
pkgs <- c("cluster", "NbClust")
install.packages(pkgs)
library(factoextra)
library(cluster)
 library(NbClust)
```

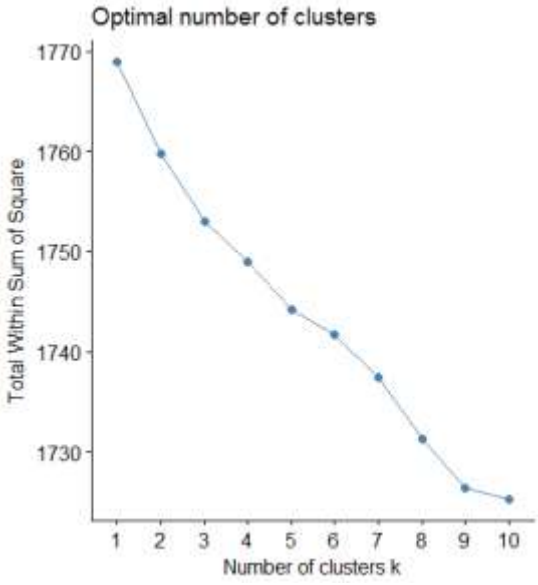
K-means Cluster

The following code can be used to find the cluster:

```
m <- as.matrix(dtm_tfxidf1)
 rownames(m) <- 1:nrow(m)
 #rownames(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, 3)
 table(cl$cluster)
plot(prcomp(m_norm)$x, col=cl$cl)
 inspect(data1[which(cl$cluster==3)])
```



Data1



Data2

The oldest method for determining the true number of clusters in a data set is inelegantly called the elbow method.
The Elbow method looks at the percentage of variance explained as a function of the number of clusters. We should choose the number of cluster in such a way that adding a new cluster does not give much modelling of the data. If one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the "elbow criterion".

The idea is that Start with K=2, and keep increasing it in each step by 1, calculating your clusters and the cost that comes with the training. At some value for K the cost drops dramatically, and after that it reaches a plateau when you increase it further. This is the K value you want.

This "elbow" cannot always be unambiguously identified. Percentage of variance explained is the ratio of the between-group variance to the total variance, also known as an F-test. A slight variation of this method plots the curvature of the within group variance.

Run k-means clustering on the dataset for a range of values of k (say, k from 1 to 10 in the examples above), and for each value of k calculate the sum of squared errors (SSE). Then, plot a line chart of the SSE for each value of k. If the line chart looks like an arm, then the "elbow" on the arm is the value of k that is the best. The idea is that we want a small SSE, but that the SSE tends to decrease toward 0 as we increase k. So our goal is to choose a small value of k that still has a low SSE, and the elbow usually represents where we start to have diminishing returns by increasing k.

The total within-cluster sum of square (wss) measures the compactness of the clustering and we want it to be as small as possible.

The NbClust is used to find the best number of clusters and from the above plot, we get to know that for Data1 there is no elbow. It means that the best cluster would be anything above cluster 10. Data2 has an elbow or bend like plot at 9. So the best cluster would be 9 or 10 for Data2.

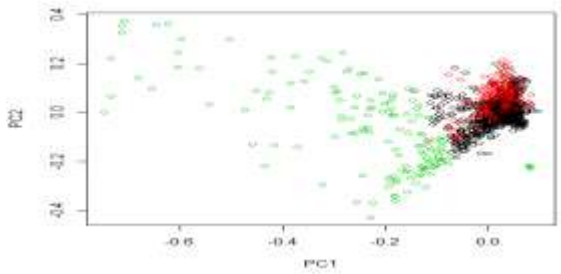
Data 1

K= 3

1 2 3

1307 296 156

> cl$withinss
[1] 1236.42627 43.26155 442.25990
> cl$tot.withinss
[1] 1721.948

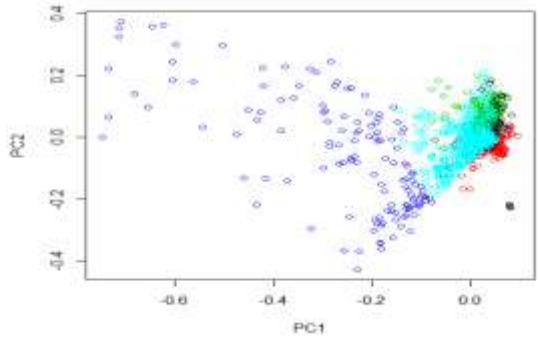


K=5

1 2 3 4 5

125 231 152 137 1114

> plot(prcomp(m_norm)$x, col=cl$cl)
> cl$withinss
[1] 269.63819 186.43179 53.61504 1150.83318 52.64158
> cl$tot.withinss
[1] 1713.16

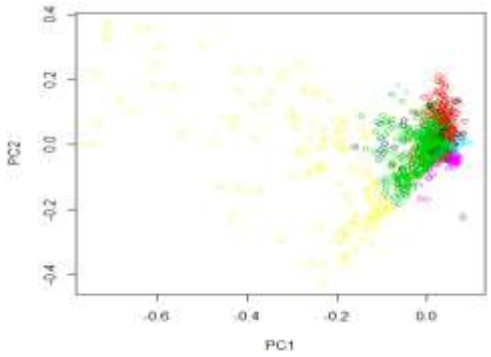


K= 10

1 2 3 4 5 6 7 8 9 10

25 139 1079 29 84 105 135 81 32 50

> cl$withinss
[1] 129.8783808 0.3862902 155.5414822 73.8531490 41.4015173 80.8818403
[7] 240.8917381 894.0392846 32.3351071 35.7556776
> cl$tot.withinss
[1] 1684.964



The k-means cluster is plotted for data 1. Data 1 consists of three groups which are similar. So from the graph plot we can see that the data points are merged in the dimensional space and there is a similarity between them. But one group is not is not merged. The data points of that group is scattered. From this we can conclude that though the three groups are taken from similar groups, two group have similarity and the other does not have any similarity.

Data2

K=3

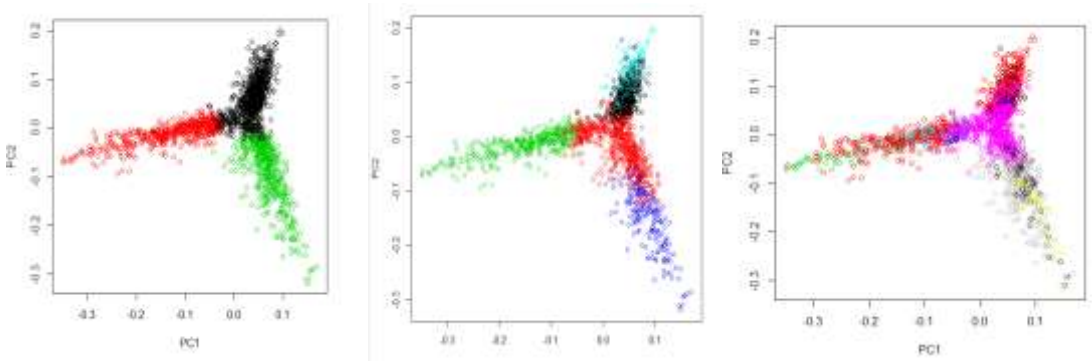
```
  1   2   3
826 491 473

>cl$withinss
[1] 484.0745 458.7841 798.7911
>cl$tot.withinss
[1] 1741.65
```

K=5

```
  1   2   3   4   5
452 623 403 190 122
>cl$withinss
[1] 484.0745 458.7841 798.7911
>cl$tot.withinss
[1] 1741.65
```

K=10

```
  1   2   3   4   5   6   7   8   9  10
 67 298  39  20  75 697  36 268  79 211
>cl$withinss
 [1] 506.22827 191.69323  69.61388 121.10622 254.28771  48.00233  46.92527  31.24282 362.58468
77.53160
>cl$tot.withinss
[1] 1709.216
```



K=3 k =5 k = 10

The k-means clustering is performed on the tf-idf of Data2. The k-means cluster is plotted. Data 2 was picked from three different news group. From the visuals it is seen that all the three groups are separate and they do not have any similarity. The data points are separate in the graph which has 3 clusters. When k=5 and k=10 the clusters have some data points merged. This means that there are very little or commons items similar between the three groups though they are dissimilar.

SVD and LSA Computation:

Latent semantic analysis (LSA) is a technique in natural language processing. It is used to analyse the relationships between the set of documents and the terms they contain by producing the set of concepts related to the documents and the terms. LSA is an application of reduced order SVD. A matrix containing word counts per paragraph rows represent unique words and columns represent each paragraph is constructed from a large piece of text and a mathematical technique called singular value decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns. The singular vectors and corresponding singular values produced by SVD allow words and documents to be mapped into the same "latent semantic space". Words are then compared by taking the cosine of the angle between the two vectors formed by any two rows. Values close to 1 in the matrix it is considered to be the similar word and values close to 0 is considered as the dissimilar words.

The singular value decomposition (SVD) is a factorization of a real or complex matrix. It is the generalization of the eigendecomposition of a positive semidefinite normal matrix to any M x N matrix via an extension of polar decomposition. It has many useful applications in signal processing and statistics.

The SVD matrix is computed using the formula and the feature space is varied for the same cluster and the SSE is measured and the values are tabulated.

The following code can be used for SVD computation

```
reduce <- function(A,dim) {
+ sing <- svd(A)
+ u<-as.matrix(sing$u[, 1:dim])
+ v<-as.matrix(sing$v[, 1:dim])
+ d<-as.matrix(diag(sing$d)[1:dim, 1:dim])
+ return(as.matrix(u%*%d%*%t(v),type='blue')) }
> rdtm<- reduce(dtm,50)
```

Document Vector

Data 1
Dimension d= 50

K Cluster

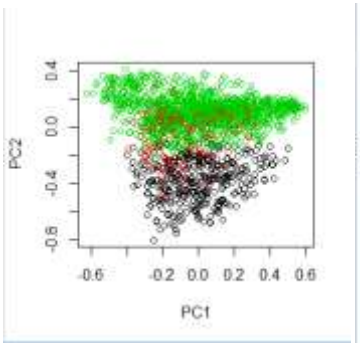
k=3
> cl$tot.withinss
[1] 1094.042

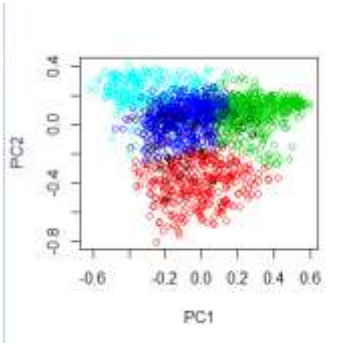
k=5

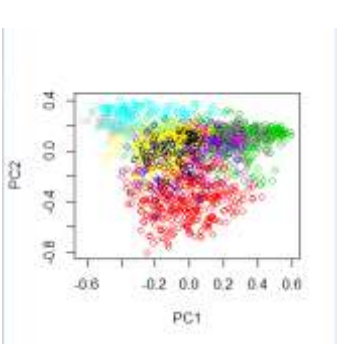
> cl$tot.withinss
[1] 989.3351

k=10

> cl$tot.withinss
[1] 871.8776

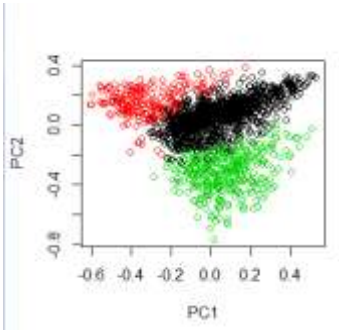


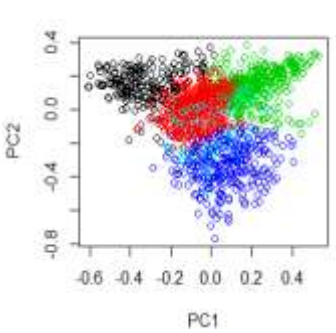


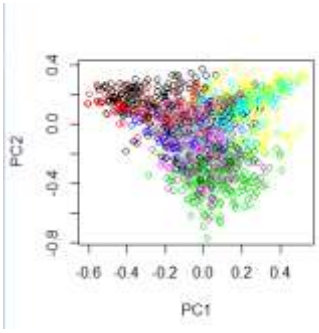


K=3 k= 5 k=10

Dimension d =100

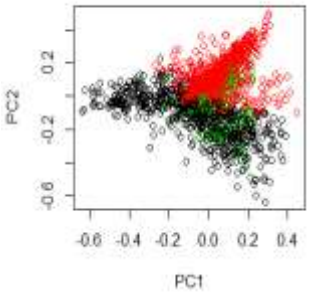




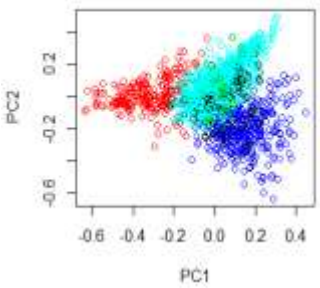


K=3 k=5 k=10

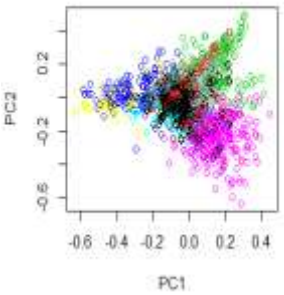
Dimension d =200



K=3



k=5

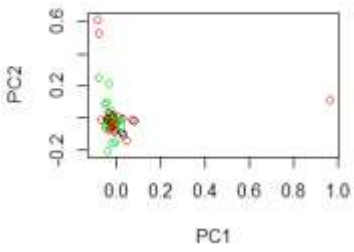


k=10

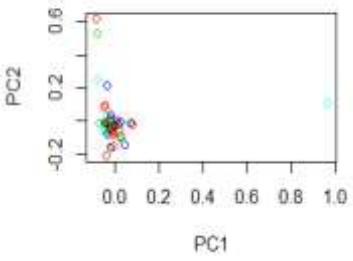
The document vector for data 1 has been computed and plotted using SVD and k-means. Three different dimensions 50,100,200 was given to SVD and the k-means clustering was for data 1 for cluster k=3, 5 and 10. When the dimension is 50 the clusters are scattered and for different clusters, data points are seen with different colours. But when the dimension of SVD increase, the data points are clustered together and similar documents and term are grouped together.

Word Vector

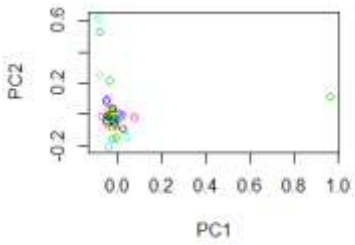
Dimension d=50



K=3

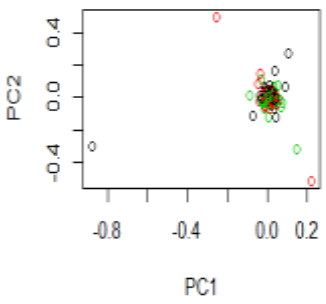


k=5

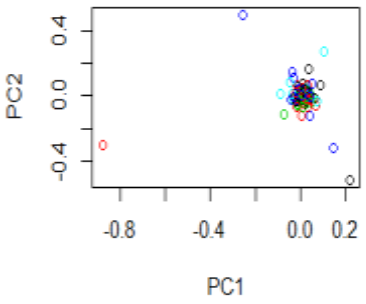


k=10

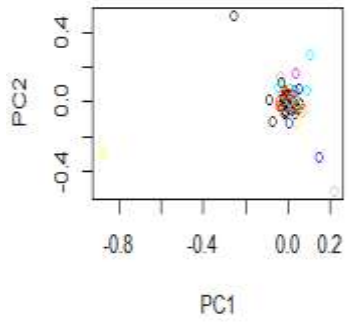
Dimension d=100



K=3

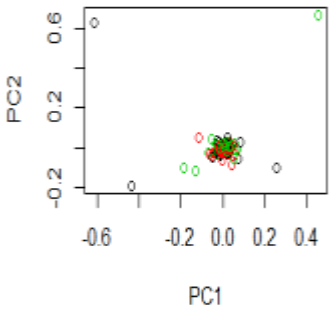


k=5

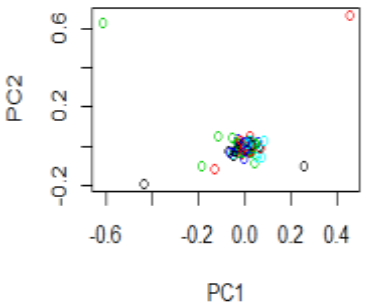


k=10

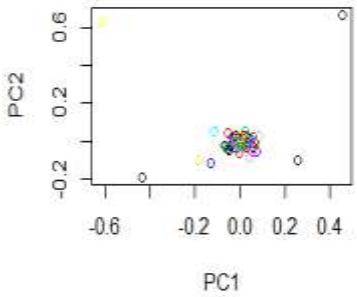
Dimension d=200



K=3



k=5



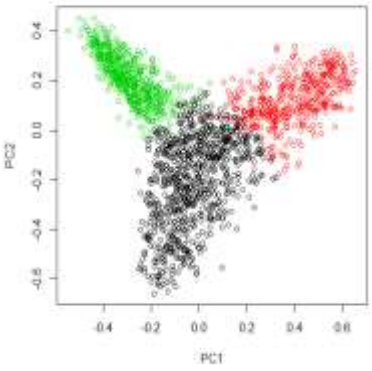
k=10

The word vectors are plotted for different dimensions for the cluster size to be three, five and ten. We get to know the words are scattered to some point when dimension is 50 and when it is 100 and 200, almost all the words are overlapped which strongly shows the similarity between the words.

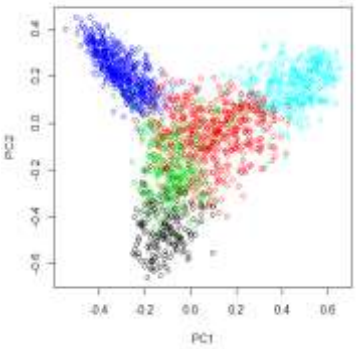
Data 2

Document Vector

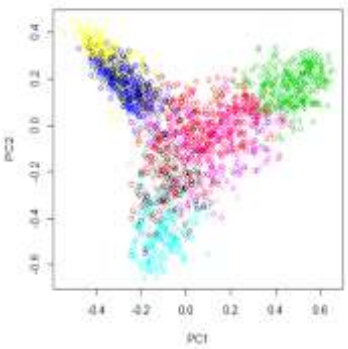
Dimension d =50



K=3

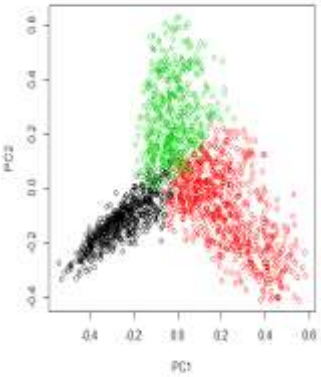


k=5

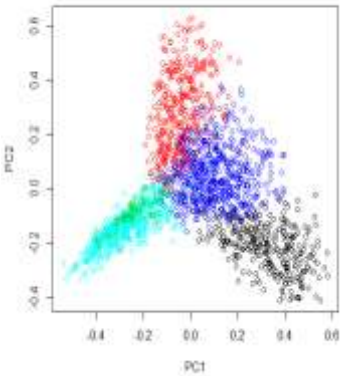


k=10

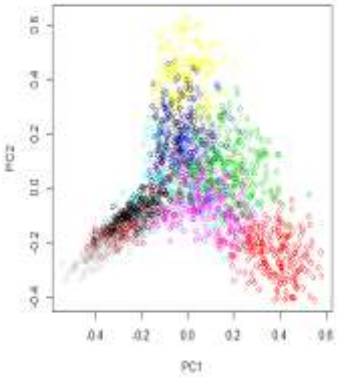
Dimension d=100



K=3

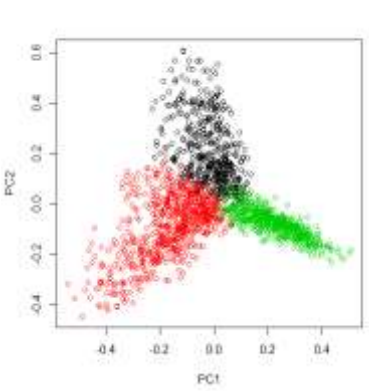


k=5

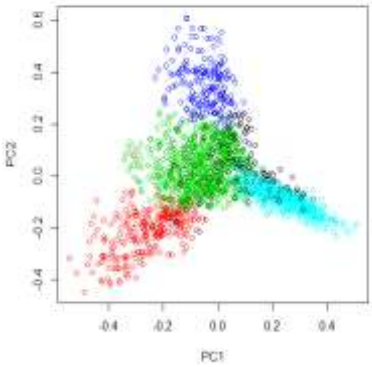


k=10

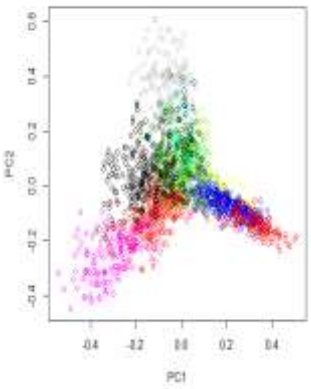
Dimension d= 200



K=3



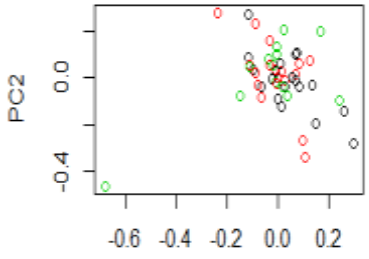
k=5



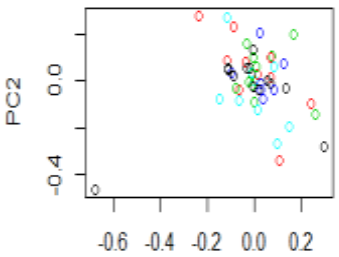
k=10

The SVD with different dimensions d =50,100 and 200 was computed for Data 2 which consists of dissimilar documents and the k-means was done using different values of K. From the plots it is seen that the documents are clustered in such a way showing the dissimilarity between them. The overlapping of the data point in the space is very less and it is because of some common words.

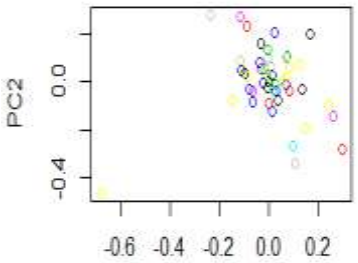
Word Vector
Dimension D=50



K=3

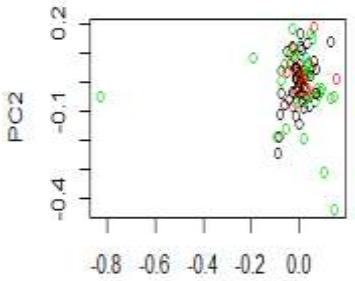


k=5

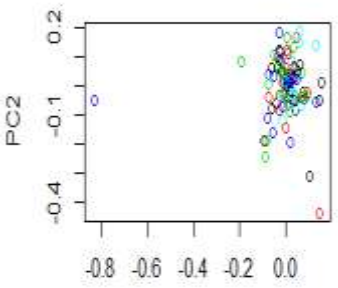


k=10

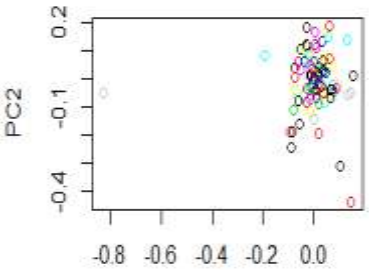
D=100



K=3

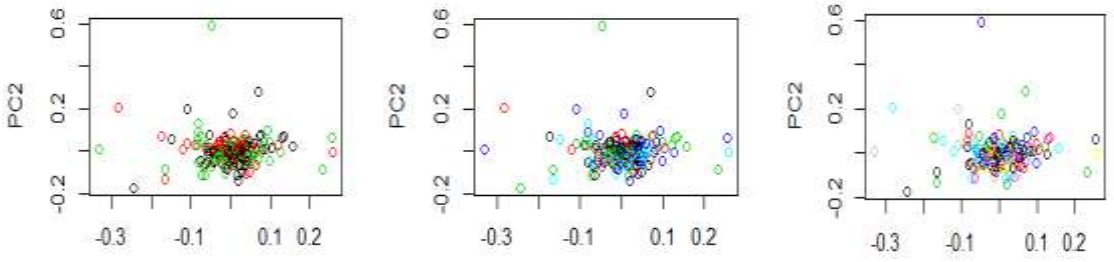


k=5



k=10

D=200



K=3 k=5 k=10

The clusters for different dimensions and different clusters are plotted. On analyzing the plot, we get to know how the data points are grouped together. The words are scattered and plotted for data 2 showing the dissimilarity. But as the dimensions increase similar words or common words in these documents are brought closer and plotted.

Performance Evaluation

In order to evaluate the clustering done using k-means and SVD we need certain terms and values. Sum of Squared Errors known as SSE and Accuracy plays important role in evaluating the clusters. SSE is used to compare clustering same data and for same number of clusters. SSE plays an important role in the performance evaluation. Residual sum of squares (RSS), also known as the sum of squared residuals (SSR) or the sum of squared errors of prediction (SSE), is the sum of the squares of residuals (deviations of predicted from actual empirical values of data). It refers to the measure of mismatch of data between the estimated models.When the clusters are more the SSE value will decrease and when there are more dimensions SSE value will increase.The SSE is also directly proportional to the features, so more the features the SSE measure is also more for the particular cluster. Percentage is defined as the ratio of (TotalSS – TotalwithinSS) to the TotalSS and this gives the better place for clustering and higher the percent, the more efficient clustering is done.

A graph plotted against cluster and SSE values can provide a useful graphical way to choose an approximated cluster solution. That is, an appropriate cluster solution could be defined as the solution at which the reduction in SSE slows dramatically. This produces an "elbow" in the plot of SSE against cluster solutions.

Accuracy can also be used since we know the topic labels for the documents. A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand.

Typical objective functions in clustering formalize the goal of attaining high intra-cluster similarity and low inter-cluster similarity. This is an internal critlse neerion for the quality of a clustering.

Rand index penalizes both false positive and false negative decisions during clustering. The F measure in addition supports differential weighting of these two types of errors.

A true positive (TP) decision assigns two similar documents to the same cluster, a true negative (TN) decision assigns two dissimilar documents to different clusters. There are two types of errors we can commit. A (FP) decision assigns two dissimilar documents to the same cluster. A (FN) decision

assigns two similar documents to different clusters. The Rand index ( ) measures the percentage of decisions that are correct. That is, it is simply accuracy

$$RI = TP+ TN/ TP+TN+FP+FN$$

The Rand index gives equal weight to false positives and false negatives. Separating similar documents is sometimes worse than putting pairs of dissimilar documents in the same cluster. We can use the F measure to penalize false negatives more strongly than false positives by selecting a value $\beta>1$, thus giving more weight to recall. Since the Rand Index gives equal weight to false positives and false negatives, evaluation of clusters is done by this.

Data	No. of Cluster	Feature Space	Total Within SS
Data1	3	Before SVD	1721.948
	3	50	1094.042
	3	100	1313.331
	3	200	1468.352
	5	Before SVD	1713.16
	5	50	989.3351
	5	100	1241.776
	5	200	1413.476
	10	Before SVD	1684.964
	10	50	871.8776
	10	100	1150.675
	10	200	1335.442
Data2	3	Before SVD	1741.694
	3	50	956.774
	3	100	1181.626
	3	200	1444.041
	5	Before SVD	1730.876
	5	50	878.4791
	5	100	1238.064
	5	200	1397.312
	10	Before SVD	1708.594
	10	50	786.748
	10	100	1085.063
	10	200	1334.75

We have performed many experiments using 20 Newsgroup dataset. The above table shows the SSE values obtained during various cluster and dimensional experiments. As discussed earlier, when the dimension increases the SSE value also increases. When the cluster increases the SSE value decreases.

Data 1

When the cluster for k-means is 3 the value of SSE is 1721.948. When the clusters are increased to 5 and 10, the SSE value is decreased which can be seen from the above table. Similarly when cluster value is 3 and dimension is 50, the SSE value is 1094.042. But when the dimensions are increased to 100 and 200, the SSE value is increased. The best cluster is usually identified using less SSE values. So

from the above table we can conclude that the best cluster solution is when k=10 and dimension d = 50. This solution is got from SVD experiment.

Data2

When the cluster for k-means is 3 the value of SSE is 1741.694. When the clusters are increased to 5 and 10, the SSE value is decreased which can be seen from the above table. Similarly when cluster value is 3 and dimension is 50, the SSE value is 956.774. But when the dimensions are increased to 100 and 200, the SSE value is increased. The best cluster is usually identified using less SSE values. So from the above table we can conclude that the best cluster solution is when k=10 and dimension d = 50. This solution is got from SVD experiment.

From the above results, we can see that we got better cluster solution and accuracy after performing LSA.

The accuracy increases when cluster increases and dimension decreases. It also depends on the data we use for analysis.

- LSA and SVD can be used to give us better representation of data by bringing out the hidden concepts or features of the dataset
- It also gives the similarity measures for the documents and the terms in the concept space which can be compared.
- Clustering after LSA gives better result than the clustering without LSA. SSE measure in the above table also describes the values of SSE before the SVD is higher than the value after SVD.

Most Representative Words

Representative words are used to get the concepts that best describe them based on the most repetitive words. Based on the repetition of the words we can describe what the concept in it is. The important words describe the concept. So that word which belongs to the concept can be found from V matrix after SVD if the matrix is DTM. It can be found from U matrix if it is TDM.

The code below is used to find the most representative words:

```
sv = sort.list(abs(sing$v[, i]), decreasing = TRUE)

dtm$dimnames$Terms[head(sv, 10)]
```

Where i in the R command is the index for the i'th concept , svd$v is the v matrix of the SVD object 'svd'. The representative words is used to find the underlying concept in the set of words. LSA don't give us the names or the labels so using the representative words we can understand the meaning of the concepts. We can find whether the concepts are correctly classified based on these words. The 10 most representative words for the concepts are given

Data1

Concept1: "mgvgvgvgvgvgvgvgvgvgvgvgvgvgv" "maxaxaxaxaxaxaxaxaxaxaxaxax" "maxaxaxaxaxaxaxaxaxaxaxaxaw" "window" "work" "like" "keyboard" "quadra" "vram" "know"

Concept 2: "reply" "board" "good" "need" "midi" "distribution" "find" "files" "much" "anyone"

Concept 3: "without" "tried" "keywords" "boot" "send" "lines" "running" "group" "connector" "hard"

Data2

Concept 1: "mass" "human" "constant" "research" "like" "center" "radius" "scripture" "orbit" "flight"

Concept 2: "doubt" "exhaust" "moon" "death" "also" "pressure" "mike" "black" "years" "science"

Concept 3: "reply" "even" "message" "part" "timer" "language" "loss" "make" "university" "another"

LDA

Latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics. LDA is an example of a topic model and was first presented as a graphical model for topic discovery.

LDA represents documents as mixtures of topics that spit out words with certain probabilities. It assumes that documents are produced in the following fashion: when writing each document, you

- Decide on the number of words N the document will have (say, according to a Poisson distribution).
- Choose a topic mixture for the document (according to a Dirichlet distribution over a fixed set of K topics). For example, assuming that we have the two food and cute animal topics above, you might choose the document to consist of 1/3 food and 2/3 cute animals.
- Generate each word w_i in the document by:
    - First picking a topic (according to the multinomial distribution that you sampled above; for example, you might pick the food topic with 1/3 probability and the cute animals topic with 2/3 probability).
    - Using the topic to generate the word itself (according to the topic's multinomial distribution). For example, if we selected the food topic, we might generate the word "broccoli" with 30% probability, "bananas" with 15% probability, and so on.

Assuming this generative model for a collection of documents, LDA then tries to backtrack from the documents to find a set of topics that are likely to have generated the collection.

Now let us calculate the Topic Probabilities and then combine it with k-means to find the clusters.

The following code can be used to find LDA:

```
install.packages("lda")

install.packages("topicmodels")

library("lda")

library("topicmodels")

lda <- LDA(dtm,50,method="Gibbs")
```

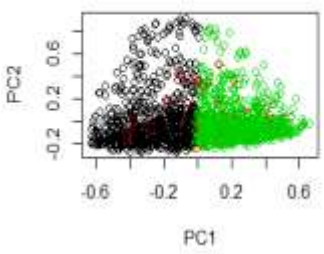
lda.topics<- as.matrix(topics(lda))

lda.terms <- as.matrix(terms(lda,10))

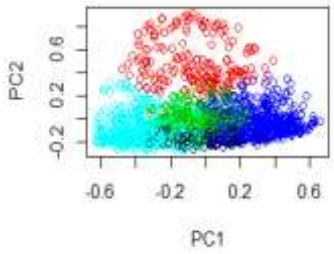
topicProbabilities <- as.data.frame(lda@gamma)

Data1

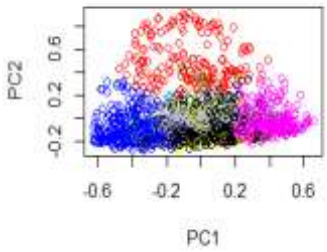
Dimension D=50



K=3

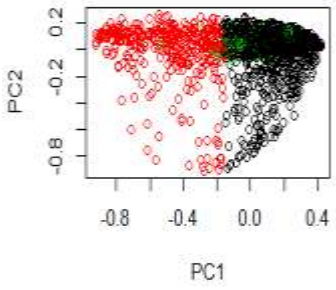


k=5

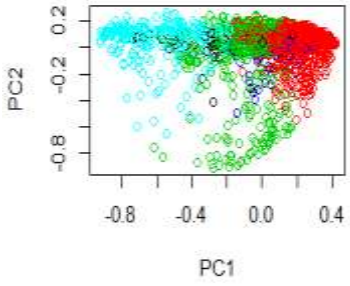


k=10

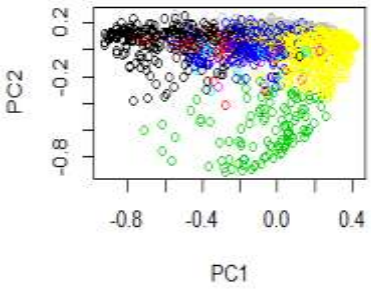
Dimension D =100



K=3

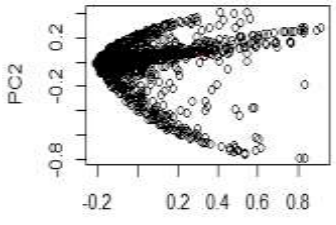


k=5

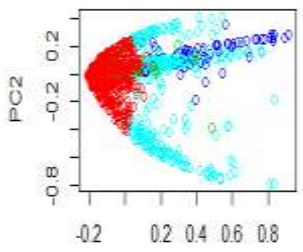


k=10

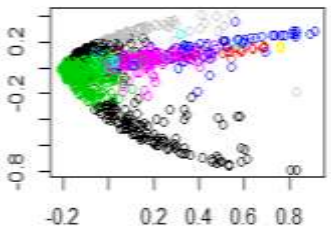
Dimension D=200



K=3



k=5



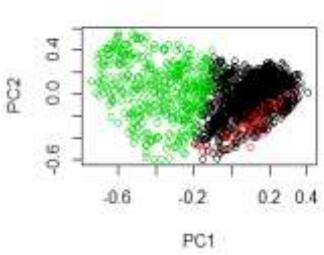
k=10

The LDA along with k-means has been computed and plotted. Three different dimensions 50,100,200 was given to LDA and the k-means clustering was for data 1 for cluster k=3, 5 and 10. When the dimension is 50 the clusters are clustered in such a way that when k=3 only two of the clusters are seen majorly. But when the dimension of LDA increase, the data points are clustered

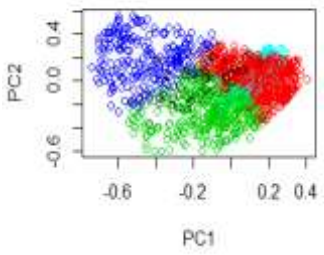
together and similar documents and term are grouped together and only some uncommon words are scattered. When dimension is 200 and k=3 almost one cluster covers the other two clusters which shows the similarity.

Data2

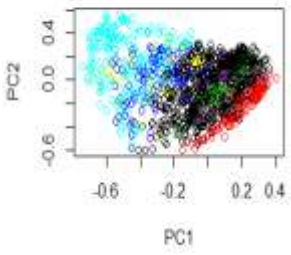
Dimension D=50



k=3

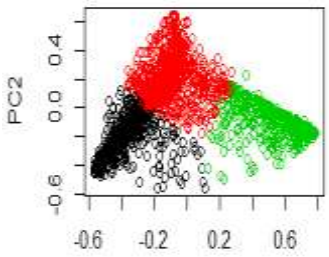


k=5

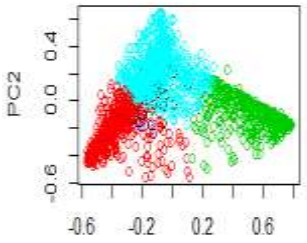


k=10

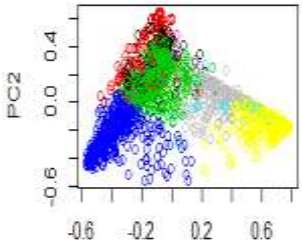
Dimension D=100



k=3

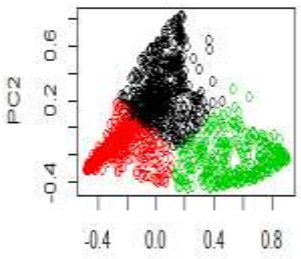


k=5

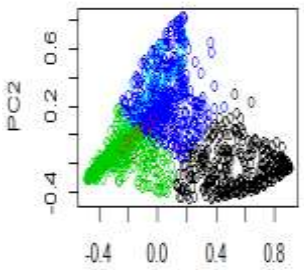


k=10

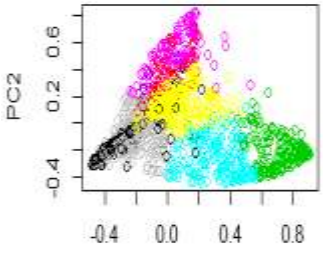
Dimension D=200



k=3



k=5



k=10

The LDA with different dimensions d =50,100 and 200 was computed for Data 2 which consists of dissimilar documents and the k-means was done using different values of K. From the plots it is seen that the documents are clustered in such a way showing the dissimilarity between them. The overlapping of the data point in the space is very less and it is because of some common words.

Performance Evaluation

Data	No. of Cluster	Feature Space	Total Within SS
Data1	3	Before LDA	1721.948
	3	50	777.6703
	3	100	659.6771
	3	200	505.5204
	5	Before LDA	1713.16
	5	50	690.0497
	5	100	591.8891
	5	200	434.8653
	10	Before LDA	1684.964
	10	50	600.5405
	10	100	517.5059
	10	200	352.0585
Data2	3	Before LDA	1741.694
	3	50	1148.763
	3	100	1028.085
	3	200	1018.602
	5	Before LDA	1730.876
	5	50	1081.11
	5	100	975.1704
	5	200	963.7322
	10	Before LDA	1708.594
	10	50	983.9595
	10	100	893.6909
	10	200	838.8688

We have performed many experiments using 20 Newsgroup dataset. The above table shows the SSE values obtained during various cluster and dimensional experiments. As discussed earlier, when the dimension increases the SSE value also decreases for LDA. When the cluster increases the SSE value decreases.

Data 1

When the cluster for k-means is 3 the value of SSE is 1721.948. When the clusters are increased to 5 and 10, the SSE value is decreased which can be seen from the above table. Similarly when cluster value is 3 and dimension is 50, the SSE value is 777.6703. But when the dimensions are increased to 100 and 200, the SSE value is decreased. The best cluster is usually identified using less SSE values. So from the above table we can conclude that the best cluster solution is when k=10 and dimension d = 200. This solution is got from LDA experiment.

Data2

When the cluster for k-means is 3 the value of SSE is 1741.694. When the clusters are increased to 5 and 10, the SSE value is decreased which can be seen from the above table. Similarly when cluster value is 3 and dimension is 50, the SSE value is 1148.763. But when the dimensions are increased to 100 and 200, the SSE value is decreased for LDA. The best cluster is usually identified using less SSE

values. So from the above table we can conclude that the best cluster solution is when k=10 and dimension d = 200. This solution is got from LDA experiment.

From the above results, we can see that we got better cluster solution and accuracy after performing LDA.

- The accuracy increases when cluster increases and dimension increases. It also depends on the data we use for analysis.
- Clustering after LDA gives better result than the normal k-means tf-idf clustering. SSE measure in the above table also describes the values of SSE before the LDA is higher than the value after LDA.

Comparison of k-means tf-idf, LSA and LDA:

When the cluster for k-means is 3 the value of SSE is 1721.948. When the clusters are increased to 5 and 10, the SSE value is decreased which can be seen from the above table. When SVD along with k-means is done,when cluster value is 3 and dimension is 50, the SSE value is 1094.042. But when the dimensions are increased to 100 and 200, the SSE value is increased. LDA along with k-means,when cluster value is 3 and dimension is 50, the SSE value is 777.6703. But when the dimensions are increased to 100 and 200, the SSE value is decreased. In SVD, k=10 and d=50 gives the best cluster. In LDA k=10 and d=200 gives the best cluster.Similar observation is seen for data 2 also.

When we see both the table of LSA and LDA, it is clearly seen from the SSE values that LDA is better than LSA which is better than k-means tf-idf. Because LSA uses SVD, and as a result the topics are assumed to be orthogonal whereas LDA treats topics as word distributions, uses probabilistic methods, and topics are allowed to be non-orthogonal with dirichlet priors for the document-topic and topic-word distributions. This prevents over-fitting, and gives better results.

So LDA clustering gives best clustering results when compared with LSA and k-means tf-idf.

Conclusion

On analyzing the 20 News group data set , we can arrive to the following conclusion:

Data pre-processing involved many steps. The following 3 steps were found to be significant.

- Stopwords – This helped remove all obvious stop words in English. It reduced the number of words to be examined by a significant percentage. This is done for better clustering.
- Tf-Idf – This increased variance on important and meaningful words. We saw in the wordcloud that stemming process brought forth some important words that would have otherwise be lost.
- Sparsity – Removing sparse terms or Pruning also contributed considerably to the reduction in the matrix size. The sparse index used was 0.9, it led to reduction in number of terms that are less frequent by more than 90% in almost all datasets.
- Word-cloud is a very convenient and effective way of representing the important words in the corpus.
- The document matrix can be normalized and scaled which in turn increase the performance.
- SVD helps in doing the Latent Semantic Analysis and brings out the hidden concepts / features from the document term matrix which is very useful in analysing the dataset and thereby to conclude some solutions on the dataset.

- SVD reduces the dimensionality of matrix and helps in analysis of large dataset because the unnecessary words / features can be removed thereby increasing the performance of overall process.
- Various experiments showed us that LSA is better than k-means tf-idf based on SSE values.
- LDA represents documents as mixtures of topics that spit out words with certain probabilities.
- Various experiments showed us that LDA is better than k-means tf-idf based on SSE values.
- Quality of analysis is heavily dependent on the quality of data. If the data is not cleaned properly or improper then it will impact negatively.
- Better clustering results are got when dimension or feature space is used along with clusters. From various results LDA is better than LSA which in turn is better than k-means tf-idf.
-

YELP DATA

Analysis on Yelp Data

We use dataset provided by Yelp Data challenge to identify relationship between most frequent words used in Reviewes and Reviewer star rating. Yelp review will receive using training data provided by Yelp in a recent Kaggle competition. Features were extracted from the review text with text mining techniques, and other features were created from supplementary user, business, and review data sets.

Yelp is a local business directory that includes social networking features. With the help of its users, Yelp provides a wealth of information about a business, such as its location, price range, and ambiance, among many other things. More interestingly, users are able to rate and write reviews about a business. The moderating of the reviews is largely left to the community, where users can judge the quality of a review by flagging it as being 'Useful', 'Funny', or 'Cool'. Other social networking features include the ability to 'check-in' to the business via the Yelp mobile app.

The data sets that Yelp provided consisted of 229,907 reviews of 11,537 businesses in the Phoenix, Arizona area by 43,873 users. A data set containing 8,282 sets of check-ins was also provided. The training data was collected from March 2005 up to January 2013. Due to the daily submission limit on the Kaggle competition, it was inconvenient for us to assess our model performance on their test set, as we were training many models.

For our Data Analysis we have extracted the reviews from review.csv based on the some categories and business id. The review extracted from the csv file are all positive review which have a start rating of 4 or 5. Though the data is grouped as different category all the review are mostly similar because we have taken positive reviews which would be easy for clustering.

Data Preprocessing

Data preprocessing is necessary, before we could extract features from text for further analysis. The data may contain words which does not provide any useful information while analysis and it may also have junk characters and null values which would alter the results during clustering and classification. The data cleaning process involved removing formatting, converting the data into plain text, stemming words, removing whitespace and uppercase characters, and removing stopwords.

Analysis before preprocessing

Document Term Matrix

<<DocumentTermMatrix (documents: 5223, terms: 27482)>>

Non-/sparse entries: 154878/143383608

Sparsity : 100%

Maximal term length: 139

Weighting : term frequency (tf)

<<**TermDocumentMatrix** (terms: 27482, documents: 5223)>>

Non-/sparse entries: 154878/143383608

Sparsity : 100%

Maximal term length: 139

Weighting : term frequency (tf)

We have taken nearly 5000 reviews and have removed null values and have processed the document term matrices. The documents are 5223 and the terms in documents are 27482. This is a huge data. Let us look see the frequent words in the data.

The	11272
And	7479
Was	3391
For	2718
that	1968
This	1829
With	1807
But	1746
They	1705
You	1592

Analysis after preprocessing

Since we read the data from a csv file there are no headers to be removed. We can do other steps for preprocessing the data. The following code is used to preprocess the text.

```
review<- tm_map(review, tolower)

review<- tm_map(review, removePunctuation)

review<-tm_map(review, removeNumbers)

review<- tm_map(review, stripWhitespace)

review<- tm_map(review, PlainTextDocument)

dtm<- DocumentTermMatrix(review)
```

Document Term Matrix.

The Document Term Matrix and the Term Document Matrix is a mathematical way of representing the number of documents and the terms in the documents in a matrix form.

DocumentTermMatrix (documents: 5223, terms: 17822)>>

Non-/sparse entries: 148043/92936263

Sparsity : 100%

Maximal term length: 91

Weighting : term frequency (tf)

Removing Stop Words:

Stopwords are defined as words in a language that are so common that their information value is practically null. Some common stopwords are words such as a, and, but, the, and if, among many others. It is common practice in text analysis to reduce the number of 'noisy' words in the data by removing stopwords.

review<- tm_map(review, removeWords, stopwords("english"))

<<DocumentTermMatrix (documents: 5223, terms: 17722)>>

Non-/sparse entries: 110341/92451665

Sparsity : 100%

Maximal term length: 91

Weighting : term frequency (tf)

Now after removing the stop words from documents we can find a difference of 100 terms. It clearly shows that the reviews collected had very few stopwords and now after removing that, the remaining terms will have high frequencies which would make a difference in clustering.

Role of stemming

Stemming refers to the process of erasing word suffixes to retrieve the root or stems of the words, which reduces the complexity of the data without significant loss of information in a bag-of-words representations of text data. For example, stemming reduces words such as fishing, fished, fish, fisher, and fishes to the stem word, fish. The stemming procedure reduces the number of words to consider and provides a better frequency representation in the DTM as a result.

review<- tm_map(review, stemDocument)

review<- tm_map(review, PlainTextDocument)

DocumentTermMatrix (documents: 5223, terms: 12821)>>

Non-/sparse entries: 94047/66870036

Sparsity : 100%

Maximal term length: 91

Weighting : term frequency (tf)

Frequency of words

The frequent words of the data after text preprocessing

place	1308
good	969
like	936
food	909
time	778
great	758
just	712
servic	562
realli	537
order	532

Pruning or Sparsity

For large set of data there is a high possibility of sparsity which means there are more less frequent values in the matrix. So to remove the very less frequent words from the DTM we use the following command. The command removes the words that are not in the 95% of the document.

dtms<- removeSparseTerms(dtm1, 0.97)

DocumentTermMatrix (documents: 5223, terms: 83)>>

Non-/sparse entries: 23490/410019

Sparsity : 95%

Maximal term length: 9

Weighting : term frequency (tf)

As a final step we reduced the sparsity of the DTM by removing uncommon words. There were many words that were very rare, and others that were grossly misspelled. These words appeared in only a handful of documents and offered little information value, so to reduce the number of noisy variables in our data further, we removed terms that appeared in less number of documents.

In the DTM after pruning the number of terms remaining are only 83 words. From 12821 words it has been reduced to 83 words which are the most important words.

TF-IDF

It has been reported that a weighted term frequency DTM provides more information, as opposed to just a term frequency DTM. We used the Term Frequency-Inverse Document Frequency (TF-IDF) weighting to provide a metric for the importance of a word in our DTM. As its name suggests, TFIDF

is the product of the normalized Term Frequency (TF) and the Inverse Document Frequency (IFT) statistics, where

$$TF\text{-}IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

In our data, t is word in our DTM and d is a single review in the set of all reviews, D. A high TF-IDF value is obtained when a term appears frequently in a given review but appears rarely in the collection of all reviews. When a term appears in many reviews, the IDF function will approach 0. Therefore, we can view the TF-IDF weights as filtering out common words.

DocumentTermMatrix (documents: 5223, terms: 16773)>>

Non-/sparse entries: 96576/87508803

Sparsity : 100%

Maximal term length: 91

Weighting : term frequency - inverse document frequency (normalized) (tf-idf)

Word cloud

Word cloud is a visual representation of the words in the document. The word cloud can be generated based on the frequency of the words.

The importance of words can be illustrated as a word cloud as follow :

Arguments of the word cloud generator function :

- words : the words to be plotted
- freq : their frequencies
- min.freq : words with frequency below min.freq will not be plotted
- max.words : maximum number of words to be plotted • random.order : plot words in random order. If false, they will be plotted in decreasing frequency
- rot.per : proportion words with 90 degree rotation (vertical text) • colors : color words from least to most frequent. Use, for example, colors ="black" for single color

Here we have generated word cloud for the yelp data before processing, after processing, after Pruning and for TF-IDF.

Before Preprocessing

After Preprocessing

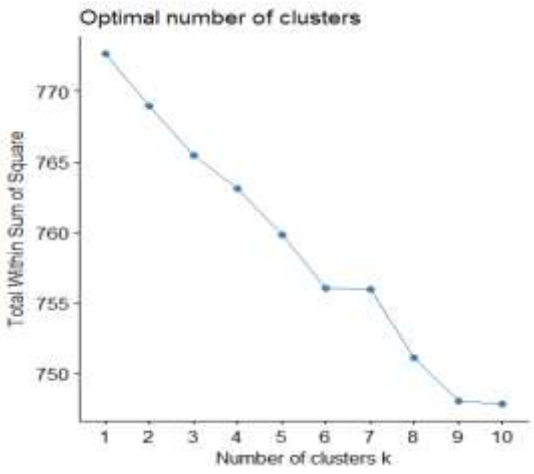
After Sparse

After TF-IDF

Clusterering Analysis

K means clustering is a Partitional clustering approach. Each cluster is associated with acentroid (center point) and each point is assigned to the cluster with the closest centroid.Number of clusters, K, must be specified.

The document term matrix is used to perform K means clustering and the results are used to compare Latent Semantic Indexing using Single Value Decomposition. We need to map the document and the term in the semantic space. The datasets are clustered using K means and this clustering will be used to evaluate the performance of LSA on clustering after we perform SVD



The oldest method for determining the true number of clusters in a data set is inelegantly called the elbow method.

The Elbow method looks at the percentage of variance explained as a function of the number of clusters. We should choose the number of cluster in such a way that adding a new cluster does not give much modelling of the data. If one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the "elbow criterion".

The idea is that Start with K=2, and keep increasing it in each step by 1, calculating your clusters and the cost that comes with the training. At some value for K the cost drops dramatically, and after that it reaches a plateau when you increase it further. This is the K value you want.

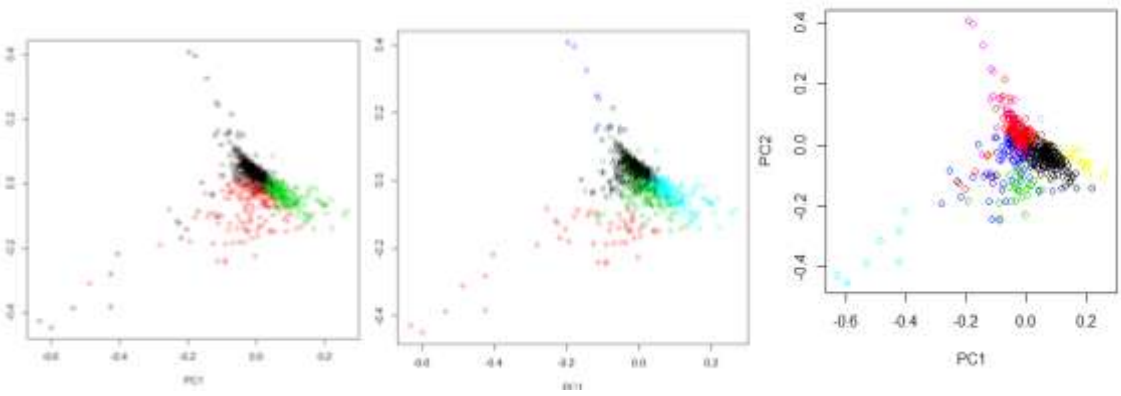
From the above graph plot k=9 is the point where there is a ‘elbow’ or ‘knee’ cutu below which the SSE values will always be less. So the best no.of cluster for the yelp dataset is 9.

K- means

K=3

table(cl$cluster)

1 2 3
438 160 183



K=3 k=5 k=9

The k-means clustering is performed on the tf-idf of Yelp data. The k-means cluster is plotted. From the visuals it is seen that all the reviews from each groups are closely related to each other and they have many similarity. The data points that are separated in the graph are some of the dissimilar words in the document. When k=5 and k=10 the clusters have most of the data points merged. This means that there are many commons items similar between the different groups.

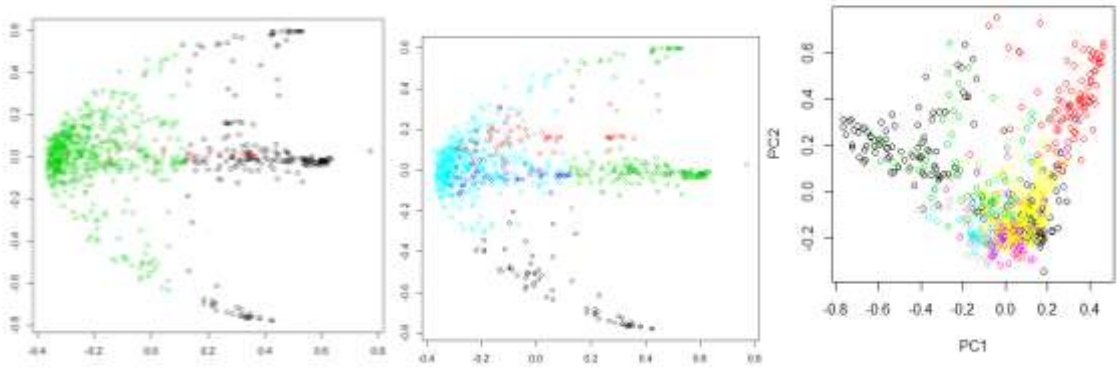
SVD and LSA Computation:

Latent semantic analysis (LSA) is a technique in natural language processing. It is used to analyse the relationships between the set of documents and the terms they contain by producing the set of concepts related to the documents and the terms. LSA is an application of reduced order SVD. A matrix containing word counts per paragraph rows represent unique words and columns represent each paragraph is constructed from a large piece of text and a mathematical technique called singular value decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns. The singular vectors and corresponding singular values produced by SVD allow words and documents to be mapped into the same "latent semantic space". Words are then compared by taking the cosine of the angle between the two vectors formed by any two rows. Values close to 1 in the matrix it is considered to be the similar word and values close to 0 is considered as the dissimilar words.

The SVD is computed for various dimensions and the results are combined with k-means and plotted.

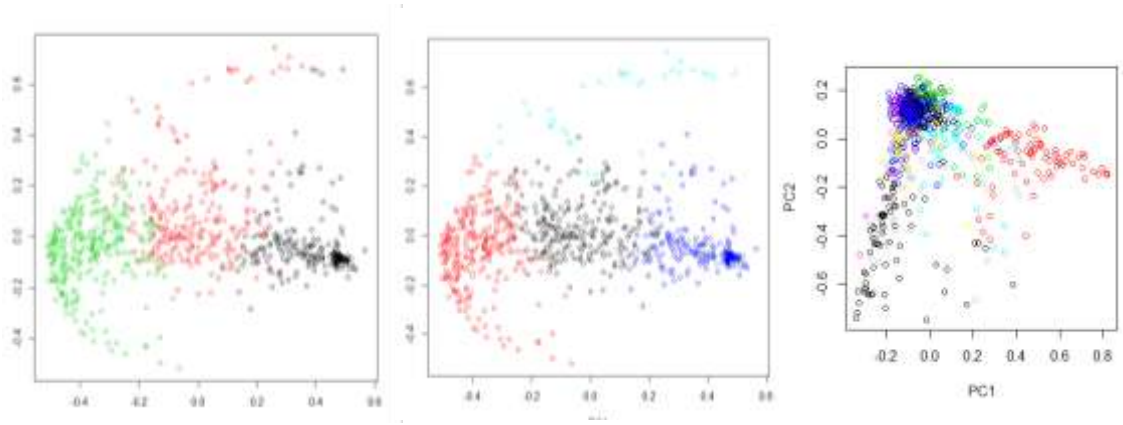
Document Vector

D=50



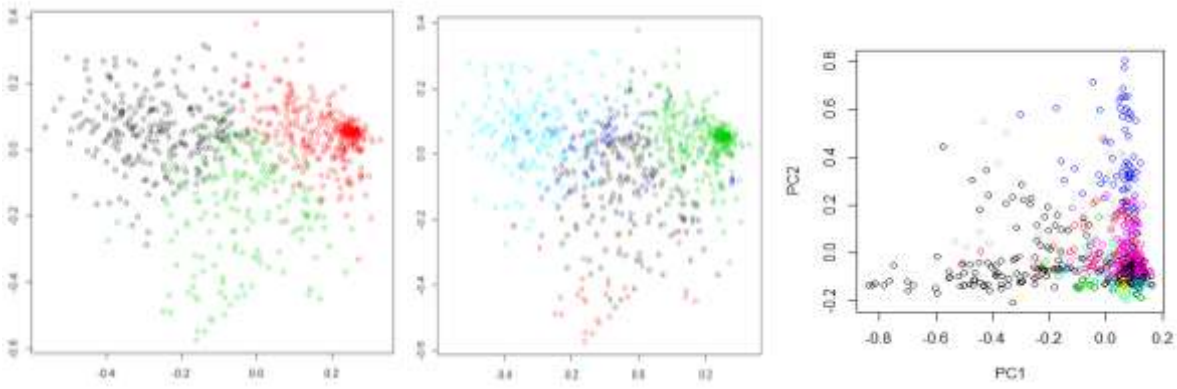
K=3 k=5 k=9

D=100



K=3 k=5 k=9

D=200

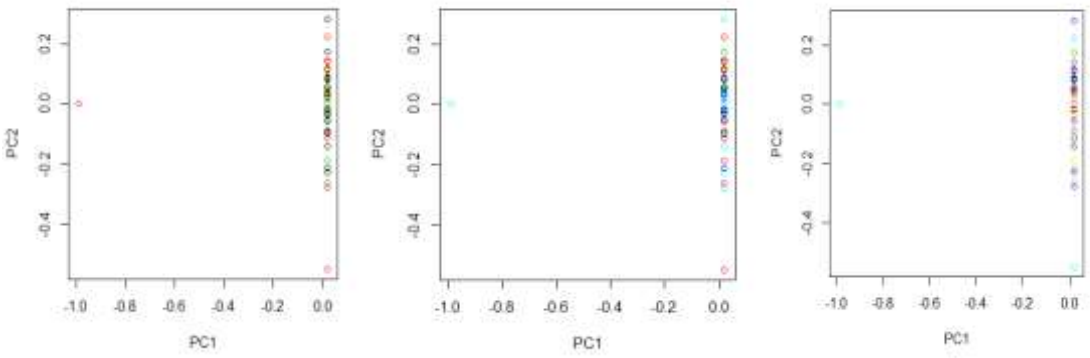


K=3 k=5 k=9

The SVD with different dimensions d =50,100 and 200 was computed for Yelp data which consists of dissimilar documents and the k-means was done using different values of K. From the plots it is seen that the documents are clustered in such a way showing the dissimilarity between them. The overlapping of the data point in the space is very less and it is because of some common words.

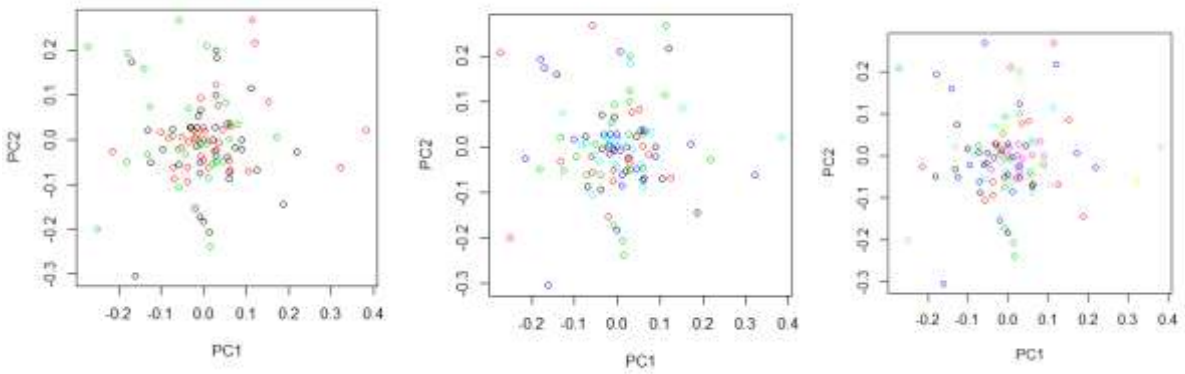
Word Vector

Dimension D=50



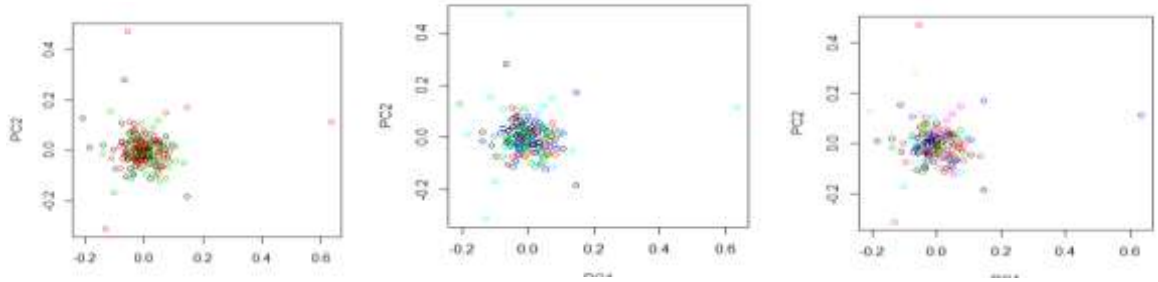
K=3 k=5 k=9

D= 100



K=3 k=5 k=9

D=200



K=3 k=5 k=9

The clusters for different dimensions and different clusters are plotted. On analysing the plot, we get to know how the data points are grouped together. The words are scattered and plotted for data showing the dissimilarity when the dimension is 50. But as the dimensions increase similar words or common words in these documents are brought closer and plotted. This shows though the reviews are grouped based on categories all the positive reviews have similar kind of words in the documents and their centroids are close to each other, thus forming as a cluster together.

Performance Evaluation

In order to evaluate the clustering done using k-means and SVD we need certain terms and values. Sum of Squared Errors known as SSE and Accuracy plays important role in evaluating the clusters. SSE is used to compare clustering same data and for same number of clusters. SSE plays an important role in the performance evaluation. Residual sum of squares (RSS), also known as the sum of squared residuals (SSR) or the sum of squared errors of prediction (SSE), is the sum of the squares of residuals (deviations of predicted from actual empirical values of data). It refers to the measure of mismatch of data between the estimated models.When the clusters are more the SSE value will decrease and when there are more dimensions SSE value will increase.The SSE is also directly proportional to the features, so more the features the SSE measure is also more for the particular cluster. Percentage is defined as the ratio of (TotalSS – TotalwithinSS) to the TotalSS and this gives the better place for clustering and higher the percent, the more efficient clustering is done.

A graph plotted against cluster and SSE values can provide a useful graphical way to choose an approximated cluster solution. That is, an appropriate cluster solution could be defined as the solution at which the reduction in SSE slows dramatically. This produces an "elbow" in the plot of SSE against cluster solutions.

No. of Clusters	Feature Space	Total Within SS
3	Before SVD	765.8798
3	50	388.5677
3	100	506.842
3	200	698.4963
5	Before SVD	759.5563
5	50	328.1802
5	100	486.8671
5	200	684.5945
9	Before SVD	747.8546
9	50	541.0305
9	100	622.4175
9	200	684.8654

We have performed many experiments using Yelp dataset. The above table shows the SSE values obtained during various cluster and dimensional experiments. As discussed earlier, when the dimension increases the SSE value also increases. When the cluster increases the SSE value decreases.

When the cluster for k-means is 3 the value of SSE is 765.8798. When the clusters are increased to 5 and 10, the SSE value is decreased which can be seen from the above table. Similarly when cluster value is 3 and dimension is 50, the SSE value is 388.5677. But when the dimensions are increased to

100 and 200, the SSE value is increased. The best cluster is usually identified using less SSE values. Since the yelp data is a huge set and we have collected some of its data from particular business and category id, the set may be seen as an improper set. This set contained many null values and during SVD we had to remove the null values inorder to give better results. So from the above table we can conclude that the best cluster solution is when k=5 and dimension d = 50. This solution is got from LSA experiment.

From the above results, we can see that we got better cluster solution and accuracy after performing LSA.

- The accuracy increases when cluster increases and dimension decreases. It also depends on the data we use for analysis.
- LSA and SVD can be used to give us better representation of data by bringing out the hidden concepts or features of the dataset
- It also gives the similarity measures for the documents and the terms in the concept space which can be compared.
- Clustering after LSA gives better result than the clustering without LSA. SSE measure in the above table also describes the values of SSE before the SVD is higher than the value after SVD.

Most Representative words

Representative words are used to get the concepts that best describe them based on the most repetitive words. Based on the repetition of the words we can describe what the concept in it is. The important words describe the concept. So that word which belongs to the concept can be found from V matrix after SVD if the matrix is DTM. It can be found from U matrix if it is TDM.

The code below is used to find the most representative words:

```
sv = sort.list(abs(sing$v[, i]), decreasing = TRUE)

dtm$dimnames$Terms[head(sv, 10)]
```

Where i in the R command is the index for the i'th concept , svd$v is the v matrix of the SVD object 'svd'. The representative words is used to find the underlying concept in the set of words.LSA don't give us the names or the labels so using the representative words we can understand the meaning of the concepts. We can find whether the concepts are correctly classified based on these words. The 10 most representative words for the concepts are given

Concept 1: "really" "didnt" "buffet" "found" "coffee" "lunch" "since" "thats" "pepper" "went"

Concept 2: "piano" "delicious" "recommend" "sweet" "sure" "order" "around" "meat" "elliot" "service"

Concept 3: "dessert" "fantastic" "rice" "excellent" "must" "never" "meat" "night" "club" "wasnt"

LDA

Latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a

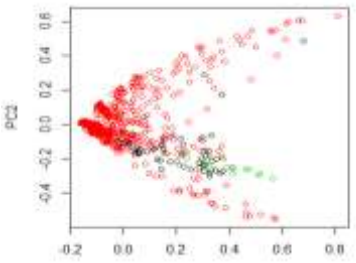
mixture of a small number of topics and that each word's creation is attributable to one of the document's topics. LDA is an example of a topic model and was first presented as a graphical model for topic discovery.

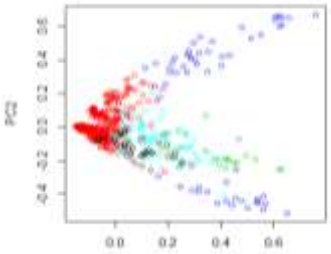
LDA represents documents as mixtures of topics that spit out words with certainprobabilities. Itassumes that documents are produced in the following fashion: when writing each document, you

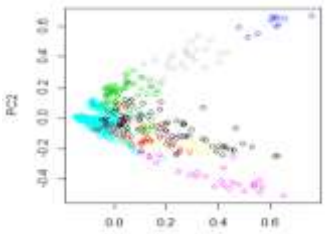
- Decide on the number of words N the document will have.
- Choose a topic mixture for the document
- Generate each word w in the document by:
  - First picking a topic
  - Using the topic to generate the word itself

The LDA for the selected Yelp data is executed and analysed.

Dimension D=50

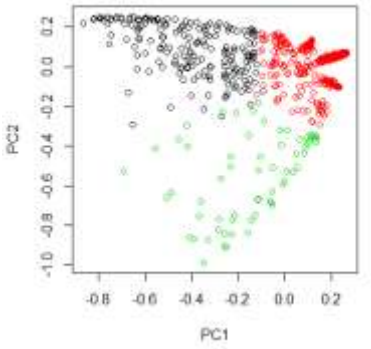


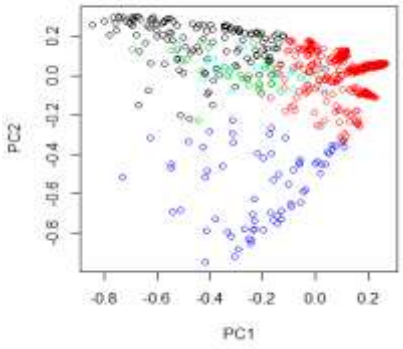


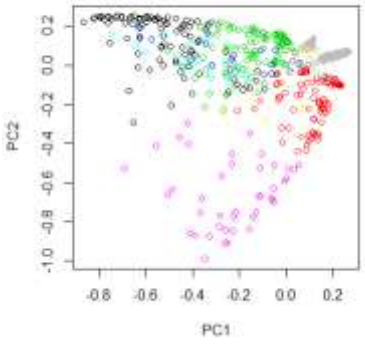


K=3 k=5 k=9

D=100

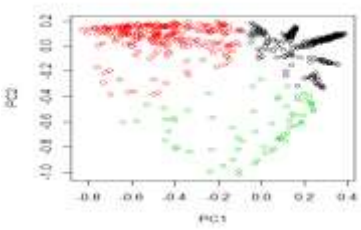


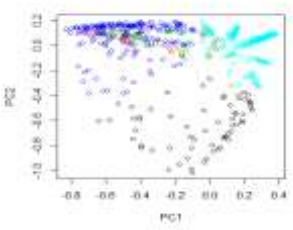


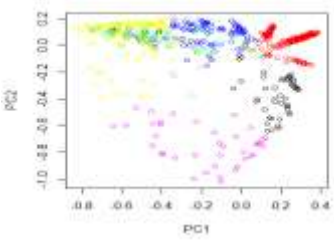


K=3 k=5 k=9

D=200







K=3 k=5 k=9

The LDA with different dimensions d =50,100 and 200 was computed for Yelp data which consists of dissimilar documents and the k-means was done using different values of K. From the plots it is seen that the documents are clustered in such a way showing the dissimilarity between them when the dimension increases. When the dimension is 50 most of the data points are clustered together and the dissimilar data points are scattered. But when the dimension increases to 100 and 200 most of the data points are scattered in the space. The overlapping of the data point in the space is very less and it is because of some common words.

Performance Evaluation

Sum of Squared Errors known as SSE and Accuracy plays important role in evaluating the clusters. SSE is used to compare clustering same data and for same number of clusters. SSE plays an important role in the performance evaluation.

No. of Cluster	Feature Space	Total Within SS
3	Before LDA	765.8798
3	50	176.3436
3	100	221.8196
3	200	266.5268
5	Before LDA	759.5563
5	50	150.4293
5	100	190.9917
5	200	242.8078
9	Before LDA	747.8546
9	50	127.3588
9	100	170.7984
9	200	216.4872

We have performed many experiments using Yelp dataset. The above table shows the SSE values obtained during various cluster and dimensional experiments. As discussed earlier, when the dimension increases the SSE value also increases. When the cluster increases the SSE value decreases.

When the cluster for k-means is 3 the value of SSE is 765.8798. When the clusters are increased to 5 and 10, the SSE value is decreased which can be seen from the above table. Similarly when cluster value is 3 and dimension is 50, the SSE value is 176.3436. But when the dimensions are increased to 100 and 200, the SSE value is increased. The best cluster is usually identified using less SSE values. Since the yelp data is a huge set and we have collected some of its data from particular business and category id, the set may be seen as an improper set. So from the above table we can conclude that the best cluster solution is when k=9 and dimension d = 50. This solution is got from LSA experiment.

From the above results, we can see that we got better cluster solution and accuracy after performing LDA.

- The accuracy increases when cluster increases and dimension decreases. It also depends on the data we use for analysis.

Comparison of k-means tf-idf, LSA and LDA:

When the cluster for k-means is 3 the value of SSE is 765.8798. When the clusters are increased to 5 and 10, the SSE value is decreased which can be seen from the above table. When SVD along with k-means is done,when cluster value is 3 and dimension is 50, the SSE value is 388.5677. But when the dimensions are increased to 100 and 200, the SSE value is increased. LDA along with k-means,when cluster value is 3 and dimension is 50, the SSE value is 176.3436. But when the dimensions are increased to 100 and 200, the SSE value is increased. In SVD, since the data review is improper data k=5 and d=50 gives the best cluster. In LDA k=9 and d=50 gives the best cluster. But when we see both the table of LSA and LDA, it is clearly seen from the SSE values that LDA is better than LSA which is better than k-means tf-idf. Because LSA uses SVD, and as a result the topics are assumed to be orthogonal whereas LDA treats topics as word distributions, uses probabilistic methods, and topics are allowed to be non-orthogona lwith dirichlet priors for the document-topic and topic-word distributions. This prevents over-fitting, and gives better results.

So LDA clustering gives best clustering results when compared with LSA and k-means tf-idf.

Conclusion

On analyzing the Yelp dataset , we can arrive to the following conclusion:

Data pre-processing involved many steps. The following 3 steps were found to be significant.

- Stopwords – This helped remove all obvious stop words in English. It reduced the number of words to be examined by a significant percentage. This is done for better clustering.
- Tf-Idf – This increased variance on important and meaningful words. We saw in the wordcloud that stemming process brought forth some important words that would have otherwise be lost.
- Sparsity – Removing sparse terms or Pruning also contributed considerably to the reduction in the matrix size. The sparse index used was 0.95, it led to reduction in number of terms that are less frequent by more than 95% in almost all datasets.
- Word-cloud is a very convenient and effective way of representing the important words in the corpus.
- The document matrix can be normalized and scaled which in turn increase the performance.
- SVD helps in doing the Latent Semantic Analysis and brings out the hidden concepts / features from the document term matrix which is very useful in analysing the dataset and thereby to conclude some solutions on the dataset.
- SVD reduces the dimensionality of matrix and helps in analysis of large dataset because the unnecessary words / features can be removed thereby increasing the performance of overall process.
- Various experiments showed us that LSA is better than k-means tf-idf based on SSE values.
- LDA represents documents as mixtures of topics that spit out words with certainprobabilities.
- Various experiments showed us that LDA is better than k-means tf-idf based on SSE values.
- Quality of analysis is heavily dependent on the quality of data. If the data is not cleaned properly or improper then it will impact negatively.
- Better clustering results are got when dimension or feature space is used along with clusters. From various results LDA is better than LSA which in turn is better than k-means tf-idf.

Reference:

https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

http://www.astrostatistics.psu.edu/datasets/2006tutorial/html/base/html/svd.html