Name: Brijesh Mavani CWID: A20406960 Assignment: 1

20 NewsGroups Data Set

I Data:

The 20 Newsgroups data set is divided into 20 different newsgroups each consisting nearly equal number of documents. The data set consists of approximately 20K documents in total. Each newsgroup corresponds to a different topic. Some groups are closely related and some are unrelated. Below table showcase the 20 newsgroups grouped together in 6 larger 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

- **Analysis:**
- As this dataset is labeled data it is suitable for the supervised learning as well as helpful in cluster evaluations.
- The documents are nearly divided equally into different groups. Hence, we do not have to additional data preparation steps to make dataset evenly distributed.
- As there are groups from similar subject matter, we can expect overlapping of content between them.
- The following table depicts the overview about number of documents and words from each of the newsgroups (Code Reference: Appendix A):

Sr No	NewsGroup	# of Documents	# of unique words	Sr No	NewsGroup	# of Documents	# of unique words
1	alt.atheism	799	13814	11	rec.sport.hockey	999	13492
2	comp.graphics	973	14963	12	sci.crypt	991	16306
3	comp.os.ms-windows.misc	985	38055	13	sci.electronics	981	12912
4	comp.sys.ibm.pc.hardware	982	12010	14	sci.med	990	18863
5	comp.sys.mac.hardware	961	11440	15	sci.space	987	17090
6	comp.windows.x	980	18713	16	soc.religion.christian	997	16911
7	Misc. for sale	972	12542	17	talk.politics.guns	910	16528
8	rec.autos	990	13097	18	talk.politics.mideast	940	19313
9	rec.motorcycles	994	13073	19	talk.politics.misc	775	16132
10	rec.sport.baseball	994	11499	20	talk.religion.misc	628	13820

- From the dataset, we can see that most of the topics have a high number of documents.
- Also, data is balanced as each group has significantly similar number of unique words with exception to comp.os.ms-windows.misc. This shows that the data is balanced and no topic will overwhelm the others.

II Experiments:

II.A Data Preprocessing:

It is the most important step in data mining. There are many steps in data preprocessing steps, but at high levels it can be summarized as the extraction, transformation and loading of the data. To be more precise modifying the source data into a different format which:

(a) enables data mining algorithms to be applied easily
(b) improves the effectiveness and the performance of the mining algorithms
(c) represents the data in easy and understandable format for both humans and machines

We will use only 5 groups (alt.atheism, comp.windows.x, misc.forsale, rec.sport.hockey and talk.politics.guns) for further processing and analysis. These groups are from very diverse subjects. They belong to religion, technology, advertisement, sports and politics respectively. As they are from different subject matter, we expect them to have different content and to form distinct clusters during clustering.We should also expect LDA/LSA performance to be good on them. Code Reference: Appendix B

Document Term Matrix Before Preprocessing:

Documents	Terms	Non-/sparse entries	Sparsity	Maximal term length	Weighting
4657	49157	473022/228451127	100%	79	tf

Document Term Matrix After Preprocessing : The standard text preprocessing steps will be used here. We used steps for numeric removal, stop words removal, punctuation removal, etc.

Documents	Terms	Non-/sparse entries	Sparsity	Maximal term length	Weighting
4657	40454	363937/188030341	100%	150	tf

Document Term Matrix after removing sparse terms: It provides significant numbers of terms(785) to perform the clustering. As we have a good number of terms, we can perform LSA and LDA at high dimensions(200-300 or more) as well to see how clustering works at high dimensions.

Documents	Terms	Non-/sparse entries	Sparsity	Maximal term length	Weighting
4657	785	186611/3469134	95%	18	tf

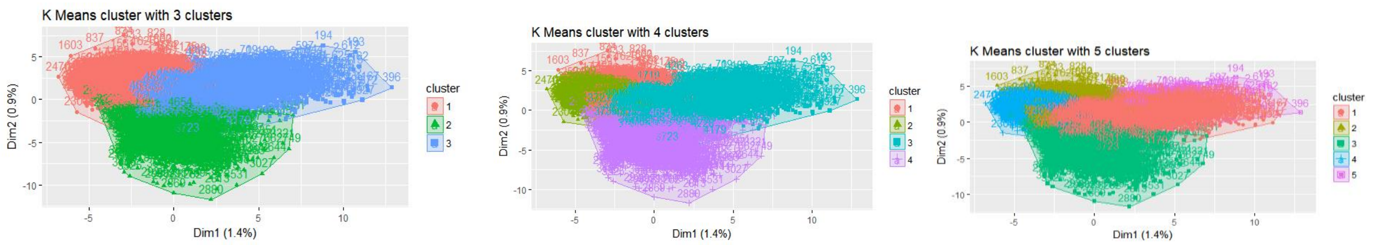
We extract top 10 most frequent words from the term document matrix.

	word	freq
will	will	3512
write	write	3366
articl	articl	2504
like	like	2455
peopl	peopl	2283
dont	dont	2280
think	think	2012
just	just	2011
game	game	2004
know	know	1895

Another popular visualization technique for text data is word cloud (shown above). It highlights the most trend terms in documents based on the frequency.

II.B Clustering Experiments and II.C Evaluation :

Below are the cluster formation for K=3 to 5.



# of Clusters	SSE	Total WithinSSE	withinSSE Ratio
3	4469.924	4351.294	97.34%
4	4469.924	4309.661	96.41%
5	4469.924	4282.354	95.8%

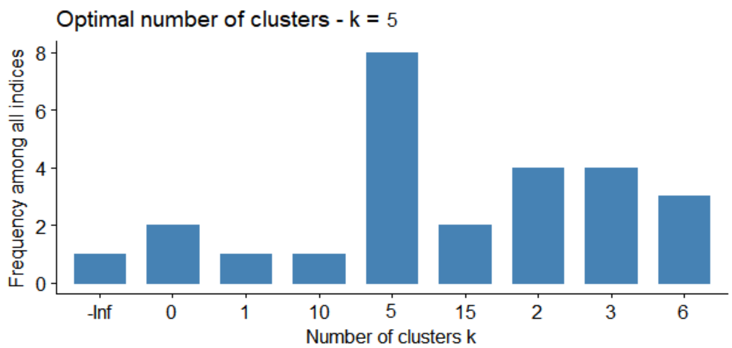
withinSSE Ratio can be calculated with : ((kmeans5$tot.withinss/kmeans5$totss)*100)

Analysis: Here, we know the number of groups are 5. Hence, cluster with 5 makes more sense and above figures indicate same. We can also see that withinSSE Ratio is smaller for k=5.In cases where we don't now the best number of clusters, NBclust package of R can be used to estimate the number of clusters. Code reference: Appendix C.

NbClust Prediction:

```
> nb <- NbClust(dtm_norm, distance = "euclidean", method = "complete", index ="all")
> fviz_nbclust(nb, kmeans, method="wss")
```

Among all indices:
===================
* 1 proposed -Inf as the best number of clusters
* 2 proposed 0 as the best number of clusters
* 1 proposed 1 as the best number of clusters
* 4 proposed 2 as the best number of clusters
* 4 proposed 3 as the best number of clusters
* 3 proposed 6 as the best number of clusters



* 1 proposed 10 as the best number of clusters
* 8 proposed 5 as the best number of clusters
* 2 proposed 15 as the best number of clusters

Conclusion
=========================
* According to the majority rule, the best number of clusters is 5 .

Confusion Matrix:
As the number clusters = 5 based on the clustering algorithm, we showcase the confusion matrix for k= 5.
The cluster assignment to each document and the actual group label of that document taken into the picture while constructing the confusion matrix. If both labels are same then the prediction of clustering is accurate as algorithm made mistake in cluster assignment. We need to assign the news group name to each cluster as Kmeans algorithm only assign the cluster id to each document. We do it using the news groups assignment of the documents in each cluster. Based on the most frequent true label from the documents in the cluster, we assign the majority news group label to that cluster.

category/Cluster #	1	2	3	4	5
alt.atheism	210	7	0	2	580
comp.windows.x	47	920	0	12	0
misc.forsale	71	70	18	811	0
rec.sport.hockey	127	20	849	2	1
talk.politics.guns	896	3	1	7	3

Analysis: The confusion matrix confirms the visual analysis we did from clustering analysis above. The documents from each of the groups are well separated, the only significantly off diagonal entries are 210 and 127 – the number of times documents from a news group alt.atheism and rec.sport.hockey respectively are assigned to the cluster for the news group talk.politics.guns. Also, clustering is 87.0947% accurate. The precision, recall and F1 are calculated below:

	alt.atheism	comp.windows.x	misc.forsale	rec.sport.hockey	talk.politics.guns
Precision	0.9931507	0.9019608	0.9724221	0.9781106	0.6632124
Recall	0.7259074	0.9397344	0.8360825	0.8498498	0.9846154
F1	0.8387563	0.9204602	0.8991131	0.9094804	0.7925697

Overall, we can see high values for the F1 score for 3 groups(close to 90), but we see that there are lower for values for group 1 and 5 because of misclassification as mentioned above.

LSA using SVD:
The singular value decomposition is a matrix decomposition algorithm.The SVD decomposes a matrix A into the produce of three specially formed matrices U, D and V each of these three matrices represents a different interpretation of the original data as shown below:
$A=U * D * V^T$
Computation of SVD for Document Term Matrix, gives us 3 values in SVD.
U describes the relationship between the terms(rows) and features(columns)
D is the diagonal matrix describes about the relative strength of features.
V^T describes relation between features(rows) and documents(columns)

We used following code to get K-dimensional LSA document vectors and LSA word vectors from the SVD.

```
LSA<-function(input,dim){
  s<-svd(input)
  u<-as.matrix(s$u[,1:dim])
  v<-as.matrix(s$v[,1:dim])
  d<-as.matrix(diag(s$d)[1:dim,1:dim])
  return(as.matrix(u%*%d%*%t(v),type="green"))
  # return(as.matrix(u%*%d,type="green"))  # uncomment  to get document vector only
  # return(as.matrix(d%*%t(v),type="green"))  #  uncomment to get word vector only
}
#SVD With 50 Dimensions
svd_dtm50<-LSA(dtmspostproc,50)
svd_norm50<-norm_eucl(svd_dtm50)
#Clustering with 50 dimensions
lsacluster50 <- kmeans(svd_norm50, 5)
```

The value of k is maintained at 5 to get the most effective results.

No. of concept	# of Clusters	SSE	Total WithinSSE	withinSSE Ratio	Accuracy
50	5	3535.2	2799.776	79.20%	79.36%

100	5	3909.7	3358.719	85.91%	73.69%
200	5	4149.2	3799.763	91.58%	70.84%
300	5	4237.9	4031.461	95.13%	87.82%

Analysis: Above table indicates the number of concepts vs withinSSE ratio similar to the tf-idf evaluation done earlier. We can see that as the no of dimensions are increasing SSE measure also increases. We also see that the withinSSE ratio increases, which indicates less cohesive clusters, although the accuracy based on labels increases(at d=300). We see better accuracy at dimension 300. This is a significant reduction in the number of dimensions. Even though each group has more than 10K unique words, LSA allows us to represent the data with only 300 dimensions or less. Code reference: Appendix D.

Confusion matrix:

D=50

category	1	2	3	4	5
alt.atheism	56	4	730	9	0
comp.windows.x	103	33	83	758	2
misc.forsale	74	696	59	102	39
rec.sport.hockey	96	3	140	22	738
talk.politics.guns	109	6	774	18	3

D=100

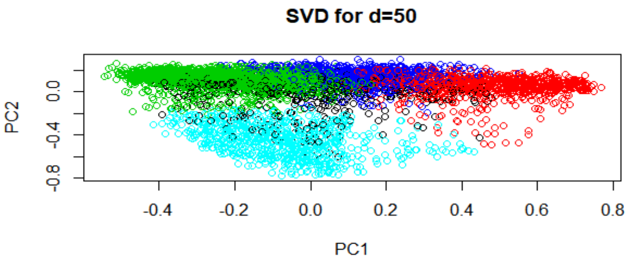
category	1	2	3	4	5
alt.atheism	1	250	0	13	535
comp.windows.x	12	96	0	836	35
misc.forsale	696	38	35	167	34
rec.sport.hockey	3	116	757	43	80
talk.politics.guns	5	274	3	20	608

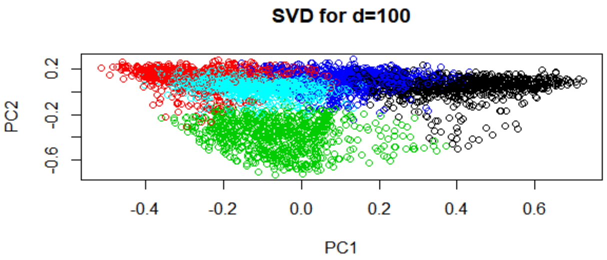
D =200

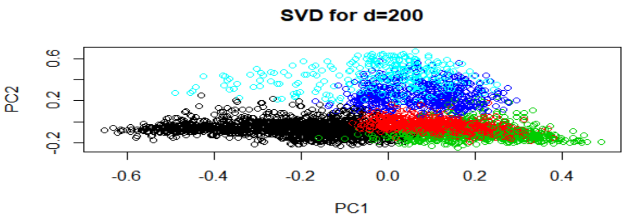
category	1	2	3	4	5
alt.atheism	15	539	244	0	1
comp.windows.x	774	59	145	1	0
misc.forsale	852	25	38	1	54
rec.sport.hockey	54	65	95	521	264
talk.politics.guns	26	613	268	2	1

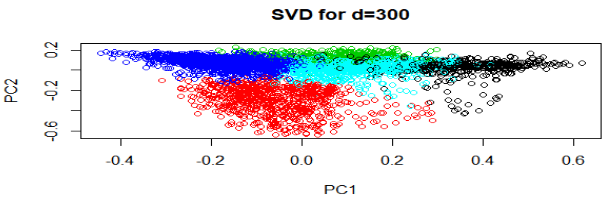
D=300

category	1	2	3	4	5
alt.atheism	0	1	0	737	61
comp.windows.x	0	0	535	84	360
misc.forsale	469	30	8	43	420
rec.sport.hockey	1	754	0	138	106
talk.politics.guns	5	3	2	783	117









We can get the most frequent terms by sorting output of V in decreasing order. Below are the list of frequent words for 5 concepts:

Sr no.	No of Concepts	Frequent Words
1	1	forsal,dividian,brand,keithccocaltechedu,allan,cdtswstratuscom,tavar,schneider,ranch,livesey
2	2	file,firearm,bill,amend,control,handgun,state,unit,hous,titl
3	3	window,widget,version,server,system,avail,applic,program,softwar,includ
4	4	atheist,peopl,believ,dont,exist,religion,mani,christian,belief,atheism
5	5	program,rule,line,build,info,return,must,widget,sourc,read

LDA:

We will perform unsupervised analyis topic modeling using lda function in R. LDA is aprobabilistic model which determines which words are likely to be generated from specific topic and then determine the topic of a document by examining these probabilities. LDA will also give a guess at the name of a topic. Like k-means, we need to supply the number of topics. Below parameters are required by lda function:

burnin : Starting point. Usually we discard first few steps as they don't reflect distribution properties.
iter : # of iterations to perform
thin : We will take every 500th iteration to avoid correlations between samples
seed : a seed integer for each starting point, for reproducibility
nstart : We do 5 independent runs at different starting points
best : return results of the run with highest posterior probability
Note : LDA takes the DocumentTermMatrix because the algorithm requires regular frequency weighting (not tf-Idf).

Code reference: Appendix E.

Below are the list of frequent words for K=5:

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
[1,]	write	right	game	peopl	window
[2,]	like	peopl	will	believ	file
[3,]	articl	state	team	reason	includ
[4,]	just	will	year	mani	program
[5,]	dont	fire	play	mean	sale
[6,]	know	bill	hockey	make	email
[7,]	think	weapon	first	moral	system
[8,]	time	govern	player	differ	work
[9,]	want	kill	last	claim	also
[10,]	good	control	goal	exist	applic

D=5

category	1	2	3	4	5
alt.atheism	7	37	233	1	521
comp.windows.x	638	3	333	0	5
misc.forsale	466	16	483	4	1
rec.sport.hockey	2	7	279	707	4
talk.politics.guns	5	631	237	2	35

D=50

category	1	2	3	4	5
alt.atheism	119	11	3	0	666
comp.windows.x	962	0	7	2	8
misc.forsale	319	0	622	3	26
rec.sport.hockey	159	2	2	823	13
talk.politics.guns	153	387	3	1	366

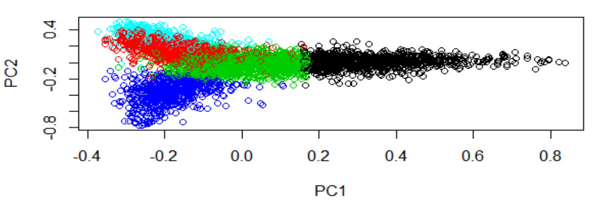
D=100

category	1	2	3	4	5
alt.atheism	6	592	196	4	1
comp.windows.x	3	48	918	10	0
misc.forsale	0	28	360	577	5
rec.sport.hockey	1	24	201	1	772
talk.politics.guns	219	479	209	3	0

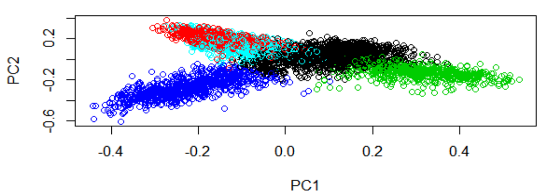
D=200

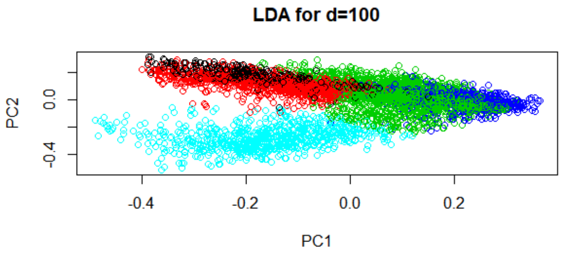
category	1	2	3	4	5
alt.atheism	1	0	261	1	536
comp.windows.x	3	0	343	589	44
misc.forsale	521	7	381	13	48
rec.sport.hockey	2	702	235	0	60
talk.politics.guns	1	2	251	0	656

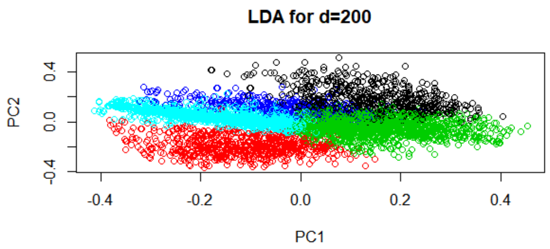
LDA for d=5



LDA for d=50







D	No of cluster	SSE	Total WithinSSE	withinSSE Ratio	Accuracy
5	5	632.4615	190.5777	30.13%	63.99%
50	5	2106.613	1711.723	81.25%	74.30%
100	5	2771.143	2456.379	88.64%	71.68%
200	5	3408.206	3167.345	92.93%	64.50%

D=5	alt.atheism	comp.windows.x	misc.forsale	rec.sport.hockey	talk.politics.guns
Precision	0.9204947	0.5706619	0.3086262	0.9901961	0.9092219
Recall	0.6520651	0.6516854	0.4979381	0.7077077	0.6934066
F1	0.76337	0.6084883	0.3810651	0.8254524	0.786783

- **Analysis:**
- We can see from the frequent words that LDA discovers the semantic topics for sports, politics and technology news groups. The representative words for the topic are hockey, weapon and email. The other topic's title is less informative.
- Other topics in remaining groups are much general so that it can appear any post of the new group. So, these topics does not separate the posts from different news groups that well.
- That is reflected in the confusion matrix and in the P/R/F1 shown above.
- It has an accuracy of only 63.99%. Hence, for this subset of the 20NG dataset and In this set of experiment, LSA vector were more helpful for clustering than LDA vectors.

Yelp Data Set

I Data:

The Yelp data set consists of 6 JSON files which consists data for more than 77K businesses. These JSON files are of size 2.2GB in compressed format and 8.95 GB in uncompressed form. These files contain lots of useless data. Hence, we cannot directly use raw data without extracting useful data. This data is less clean compared to 20NG dataset. Also, it consists several natural topics and groups of topics. Once we have sub-problem fixed we can extract corresponding data from raw JSON files and perform the experiments.

II Experiments:

II.A Data Preprocessing:

Similar to 20NG dataset, data preprocessing steps are carried here. We decided to work on the "infer categories" sub-problem, so we just need to extract information about the categories and customer reviews for the same. We can extract this information from business.json and review.json files. Business_id is the primary key in business file which will be used to merge the results.from review file. We decided to work on 5 categories: Restaurants, Automotive, Gyms, Doctors and Fashion. We extracted all customer reviews for these categories and then selected first 5000 reviews for our experiments and analysis. All the analysis is performed on these 5 categories. As they all are different, we expect them to have different content and to form distinct clusters during clustering. Code reference: Appendix F

Document Term Matrix Before Preprocessing:

Documents	Terms	Non-/sparse entries	Sparsity	Maximal term length	Weighting
5000	18751	257703/93497297	100%	28	tf

Document Term Matrix After Preprocessing : The standard text preprocessing steps are used here.

Documents	Terms	Non-/sparse entries	Sparsity	Maximal term length	Weighting
4986	15063	205952/75109048	100%	78	tf

Document Term Matrix after removing sparse terms: It provides significant numbers of terms(429) to perform the clustering.

Documents	Terms	Non-/sparse entries	Sparsity	Maximal term length	Weighting

4986	429	127035/2017965	94%	10	tf

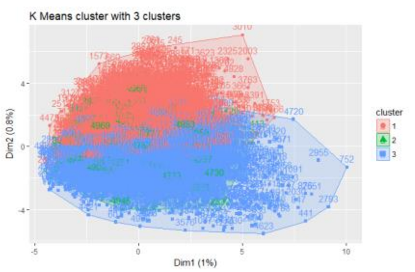
We extract top 10 most frequent words from the term document matrix.

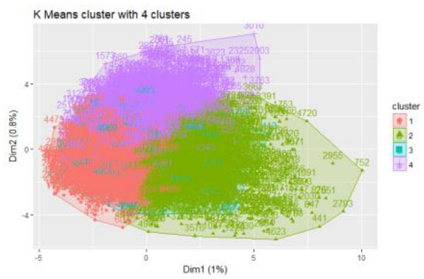
	word	freq
food	food	3509
place	place	3099
good	good	3094
order	order	2346
great	great	2242
like	like	2172
servic	servic	2045
time	time	2033
just	just	1838
back	back	1563

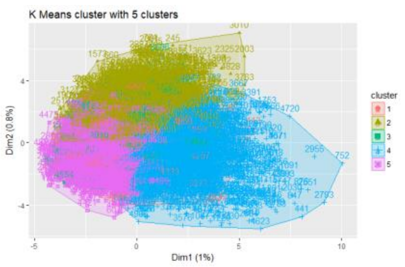
Another popular visualization technique for text data is word cloud (shown above). It highlights the most trend terms in documents based on the frequency.

II.B Clustering Experiments and II.C Evaluation:

Below are the cluster formation for K=3 to 5.







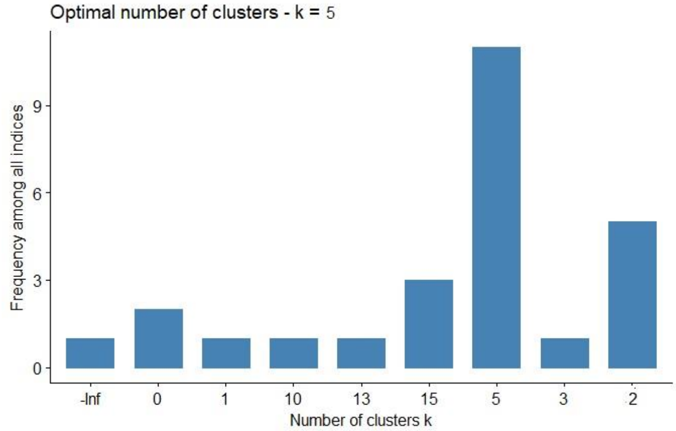
# of Clusters	SSE	Total WithinSSE	withinSSE Ratio
3	4708.355	4636.233	98.46%
4	4708.355	4607.303	97.85%
5	4708.355	4575.919	97.19%

withinSSE Ratio can be calculated with : ((kmeans5$tot.withinss/kmeans5$totss)*100)

Analysis: Here, we know the number of categories are 5. Hence, cluster with 5 makes more sense and above figures indicate same. When we choose K equivalent to no of topics, it results in better clustering. Cluster with k=5 has low withinSSE Ratio. NBclust package of R can be used to estimate the number of clusters.

NbClust Prediction:

```
> nb <- NbClust(dtm_norm, distance = "euclidean", method = "complete", index ="all")
> fviz_nbclust(nb, kmeans, method="wss")
Among all indices:
===================
* 1 proposed  -Inf as the best number of clusters
* 2 proposed  0 as the best number of clusters
* 1 proposed  1 as the best number of clusters
* 11 proposed  5 as the best number of clusters
* 1 proposed  3 as the best number of clusters
* 5 proposed  2 as the best number of clusters
* 1 proposed  10 as the best number of clusters
* 1 proposed  13 as the best number of clusters
* 3 proposed  15 as the best number of clusters
```



Conclusion

========================

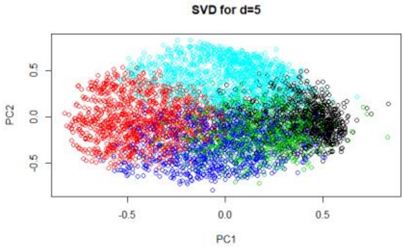
* According to the majority rule, the best number of clusters is 5.

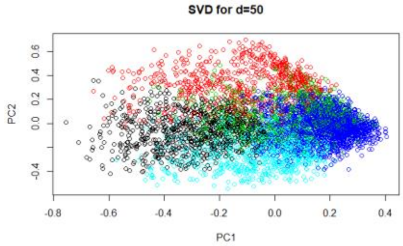
SVD:

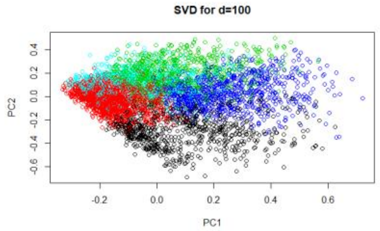
The value of k is maintained at 5 to get the most effective results.

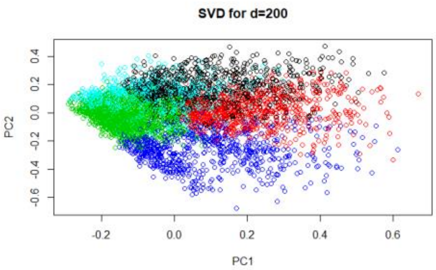
No. of concept	# of Clusters	SSE	Total WithinSSE	withinSSE Ratio
5	5	1930.201	828.6759	42.93%
50	5	3860.428	3365.421	87.18%
100	5	4144.358	3770.748	90.99%
200	5	4337.642	4043.196	93.21%
300	5	4413.69	4167.434	94.42%

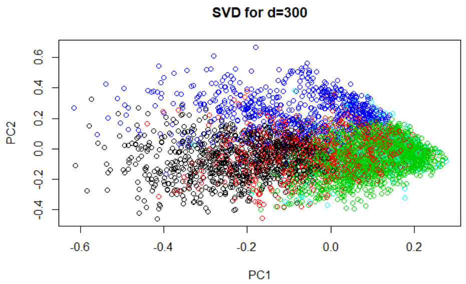
- **Analysis:**
- In line with the research community that 200-500 LSA dimensions give the best representation i.e. higher the number of LSA concepts gives better results.
- Post d=300, I also executed for 400, but the results were not improving further.
- The best result we received is almost similar to the results of Tf-Idf results. Hence, for this dataset and sample data extracted by us, LSA is not improving performance compared to the baseline of using tf-idf representation.











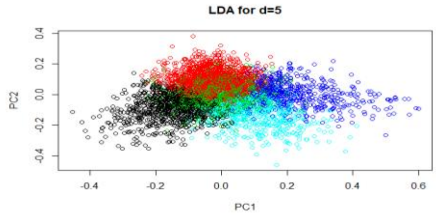
We can get the most frequent terms by sorting output of V in decreasing order.

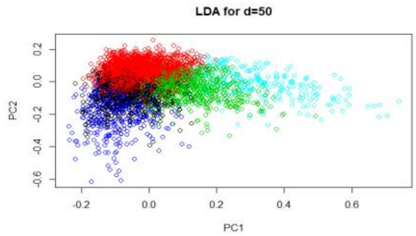
Sr no.	No of Concepts	Frequent Words
1	1	pricey,auto,cost,trip,sale,incred,unfortun,choos,terribl, servic
2	2	food,good,great,place,chicken,restaur,realli,price,lunch,dish
3	3	place,like,cloth,good,trend,love,also,great,just,nice
4	4	great,place,gyms,time,pound,back,love,will,work,wait
5	5	good,like,doctor,real,just,dont,time,want,peopl,hospital

LDA:

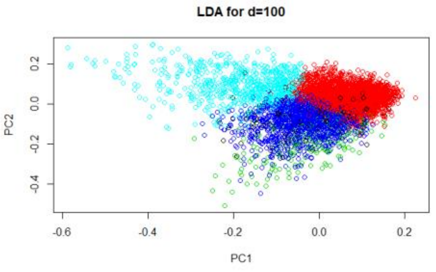
Below are the list of frequent words for K=5:

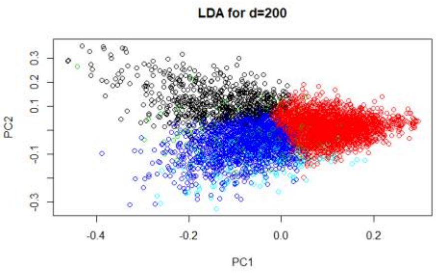
	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
[1,]	place	time	good	order	chicken
[2,]	great	back	like	cloth	also
[3,]	gyms	want	sale	servic	tast
[4,]	friend	just	just	love	menu
[5,]	love	will	price	wait	good
[6,]	best	never	littl	trend	sauc

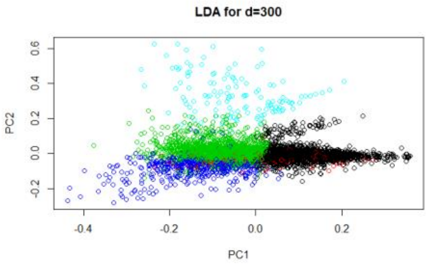




[7,]	servic	make	auto	time	dish
[8,]	alway	doctor	servic	nice	salad
[9,]	drink	dont	better	recommend	pizza
[10,]	will	good	much	even	fresh







No of Topics (d)	No of cluster	SSE	Total WithinSSE	withinSSE Ratio
5	5	243.3371	113.2767	46.55%
50	5	1401.743	1280.004	91.32%
100	5	2101.714	1992.228	94.79%
200	5	2875.982	2770.903	96.35%
300	5	3309.374	3209.459	96.98%

- **Analysis:** We can see from the frequent words that LDA discovers the semantic topics for gym, restaurants, doctors, clothing/fashion and automobiles. The representative words for the topic are doctor,gym,auto,cloth/trend, and pizza.
- Topics have few common terms such good, will,great, also, time, etc which are much general so that it can appear any review.

II.D Results Summary:

In this assignment, we used two datasets, 20 NewsGroup and Yel p Data. These both are a very different from each other. We used 3 different document representations, tf-idf, LSA, LDA and evaluated the same. The tf-idf representation uses normalized word frequency counts as features for the document vectors. LSA and LDA compute lower dimensional semantic spaces and represent documents as vectors in those spaces. Due to different features of both dataset, we could evaluate the all three document representations on both of them. We also analyzed and learned how the performance of LSA and LDA depends on the data set.

Comparison of clustering performance for 2 data sets and 3 representations:

Dataset	Algorithm	# of dimension/topics	# of cluster	wSSE Ratio	Accuracy
20 News Group	Kmeans	NA	5	95.80%	87.09%
	LSA	50	5	79.20%	79.36%
	LSA	100	5	85.91%	73.69%
	LSA	200	5	91.58%	70.84%
	LSA	300	5	95.13%	87.82%
	LDA	5	5	30.13%	63.99%
	LDA	50	5	81.25%	74.30%
	LDA	100	5	88.64%	71.68%
	LDA	200	5	92.93%	64.50%
Yelp	Kmeans	NA	5	97.19%	NA
	LSA	5	5	42.93%	NA
	LSA	50	5	87.18%	NA
	LSA	100	5	90.99%	NA
	LSA	200	5	93.21%	NA
	LSA	300	5	94.42%	NA
	LDA	5	5	46.55%	NA
	LDA	50	5	91.32%	NA

	LDA	100	5	94.79%	NA
	LDA	200	5	96.35%	NA
	LDA	300	5	96.98%	NA

III. Analysis of usefulness of LSA/LDA :

The experiment carried out in this assignment involves two very different sets of data.

- News group have different categories of data in it and I chosen 5 very different categories for experiments so it became easy for clustering. The result also gave us appropriate result.
- For 20NG the performance of LSA was better than the normal tf-idf. This might be for multiple reason. We obtained the best clustering accuracy with LSA vectors when we use 300 dimensional LSA representations.
- With Yelp data set the main challenge was to preprocess the whole data into a similar format for reusability of code.
- Once the data extracted, rest was similar to 20 NG dataset in terms of experiments.
- LDA performs better than LSA in yelp dataset. It might be for input parameters that was passed or the preprocessing of data.
- The usefulness of LDA mainly depends on the question we need to answer. If we just want to have a common idea about what documents we have, then we can say that LDA worked perfectly for our case. If we need the absolute surety then we need to validate LDA output which may need more work. In either case, this unsupervised preliminary analysis likely to have some usefulness.

IV LSA Derivation:

a) A is n x m term document matrix.

Let U be the n x n matrix whose columns are orthogonal eigenvectors of AA^T, and V be the m x m matrix whose columns are the orthogonal eigenvectors of A^TA. Denote by A^T the transpose of matrix A.
An eigenvector e of A is a vector that is mapped to a scaled version of itself, i.e., Ae = λe, where λ is the corresponding eigenvalue. For a matrix A of rank r, we can group the r non-zero eigenvalues in a r x r diagonal matrix Λ and their eigenvectors in a n x r matrix E, and we have ,
$AE = E\Lambda$
$A = E\Lambda E^{-1}$ (1)
Singular value decomposition of complex matrix A is a factorization of the form $U\Sigma V^T$, where U is a n x n unitary matrix, Σ is a n x m rectangular diagonal matrix with non-negative real number on the diagonal and V is a m x m unitary matrix. The diagonal entries σ_i of Σ is known as the singular values of A. The columns of U and the columns of V are called the left-singular vector and right singular vectors of A respectively.

$A = U \Sigma V^T$ (2)

We observed that equation (1) is similar to (2) By multiplying the above equation with its transposed, we have

$AA^T = U \Sigma V^T V \Sigma U^T = U \Sigma^2 U^T$
$A^TA = V \Sigma U^T U \Sigma V^T = V \Sigma^2 V^T$
Thus, the U in SVD on the original matrix A are the eigenvectors of AA^T, the term similarity matrix. And V in SVD are the eigenvectors A^TA, the document similarity matrix.

b) We performed clustering experiment using d=50,100,200, 300-dimensional.

```
LSA<-function(input,dim){
  s<-svd(input)
  u<-as.matrix(s$u[,1:dim])
  v<-as.matrix(s$v[,1:dim])
  d<-as.matrix(diag(s$d)[1:dim,1:dim])
  return(as.matrix(u%*%d%*%t(v),type="green"))
}
#SVD With 50 Dimensions
svd_dtm50<-LSA(dtmsyelppostproc,50)
svd_norm50<-norm_eucl(svd_dtm50)
#Clustering with 50 dimensions
lsacluster50 <- kmeans(svd_norm50, 5)
```

KEY NOTES:

- ✓ Careful Data Preprocessing
- ✓ Calculation Euclidean Distance and normalization
- ✓ Less SSE value for clusters
- ✓ LDA better accuracy
- ✓ LSA – Dimensionality Reduction using SVD

Appendix

A

```
# Importing all library which will be used in this code

library(e1071)
library(NbClust)
library(cluster)
library(ggplot2)
library(FunCluster)
library(tm)
library(SnowballC)
library(wordcloud)
library(RColorBrewer)
library(factoextra)
library(lsa)
library(topicmodels)
library(caret)

dataset<- c("C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/alt.atheism",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/comp.graphics",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/comp.os.ms-windows.misc",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/comp.sys.ibm.pc.hardware",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/comp.sys.mac.hardware",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/comp.windows.x",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/misc.forsale",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/rec.autos",
       "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/rec.motorcycles",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/rec.sport.baseball",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/rec.sport.hockey",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/sci.crypt",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/sci.electronics",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/sci.med",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/sci.space",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/soc.religion.christian",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/talk.politics.guns",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/talk.politics.mideast",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/talk.politics.misc",
        "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/talk.religion.misc")

news <- Corpus(DirSource(dataset, encoding = "UTF-8"), readerControl=list(reader=readPlain,language="en"))
dtmpreproc <- DocumentTermMatrix(news,control=list(wordLengths=c(4,Inf)))
dtmpreproc

# Load Data for experiments:
dataset<- c(
  "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/alt.atheism",
  "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/comp.windows.x",
  "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/misc.forsale",
  "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/rec.sport.hockey",
  "C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/20news-18828/talk.politics.guns")

news <- Corpus(DirSource(dataset, encoding ="UTF-8"), readerControl=list(reader=readPlain,language="en"))

category <- vector("character", length(news))
category[1:799] <- "alt.atheism"
category[800:1778] <- "comp.windows.x"
category[1779:2748] <- "misc.forsale"
category[2749:3747] <- "rec.sport.hockey"
category[3748:4657] <- "talk.politics.guns"
table(category)
```

B

```
#processing the data
news<-tm_map (news, content_transformer(tolower))
news<-tm_map (news, removePunctuation)
news<-tm_map (news, stripWhitespace)
```

```
news<-tm_map (news, removeNumbers)


#Transforming data by performing basic actions like removing white spaces , stop words etc.

myStopwords<-stopwords('english')
news<-tm_map (news, removeWords,myStopwords)
news<-tm_map (news, stemDocument)
news<-tm_map(news,removeWords,"Subject")
news<-tm_map(news,removeWords,"subject")
news<-tm_map(news,removeWords,"Organization")
news<-tm_map(news,removeWords,"writes")
news<-tm_map(news,removeWords,"From")
news<-tm_map(news,removeWords,"lines")
news<-tm_map(news,removeWords,"NNTP-Posting-Host")
news<-tm_map(news,removeWords,"article")

news<-tm_map (news, content_transformer(tolower))
news<-tm_map (news, removePunctuation)
news<-tm_map (news, stripWhitespace)
news<-tm_map (news, removeNumbers)

#Stop Words: words which do not contain important significance to be used in Search Queries.
#Usually these words are filtered out from search queries because they return vast amount of unnecessary information

myStopwords<-stopwords('english')
news<-tm_map (news, removeWords,myStopwords)

#stemming : Reduce the count of terms occurring in Document Term matrix which helps to delete the sparse items
#Simplifying them int to single words.
news<-tm_map (news, stemDocument)

#Document Term Matrix
dtmpostproc <- DocumentTermMatrix(news,control=list(wordLengths=c(4,Inf)))
dtmpostproc

# Term Document Matrix
tdmpostproc <- TermDocumentMatrix(news,control=list(wordLengths=c(4,Inf)))
tdmpostproc
#Using TDM to find frequency of words:

tdmspostproc <- removeSparseTerms(tdmpostproc, 0.98)
tdmspostproc
m <- as.matrix(tdmspostproc)
v <- sort(rowSums(m), decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
head(d, 10)


dtmspostproc <- removeSparseTerms(dtmpostproc, 0.98)
dtmspostproc
dtm_tfidf <- weightTfIdf(dtmspostproc)
dtm_tfidf1 <- as.matrix((dtm_tfidf))

dms <- as.matrix(dtmspostproc)
rownames(dms) <- 1:nrow(dms)

freq <- sort(colSums(dms),decreasing = TRUE)
dark2 <- brewer.pal(8, "Dark2")
wordcloud(names(freq), freq, max.words=200, rot.per=0.3,colors=dark2)

#Euclidean distance
norm_eucl <- function(dtm_tfidf1) dtm_tfidf1/apply(dtm_tfidf1, MARGIN=1, FUN=function(x) sum(x^2)^.5)
dtm_norm <- norm_eucl(dtm_tfidf1)
```

C

```
Kmeans5 <- kmeans((dtm_norm), centers=5,nstart=25)
table(kmeans5$cluster)
kmeans5$withinss
kmeans5$tot.withinss
kmeans5$totss
```

```
fviz_cluster(kmeans5, data=dtm_norm,centers=5, nstart=25,main="K Means cluster with 5 clusters")
Confm <- table(category, kmeans5$cluster)
confmmatrix<- as.matrix(Confm)
(sum(apply(Confm, 1, max))/sum(kmeans5$size))*100

#20NG Precision,Recall,F1
num_instances = sum(confmmatrix) # number of instances
num_classes = nrow(confmmatrix) # number of classes
correct_classifier = diag(confmmatrix) # number of correctly classified instances per class
n_inst_pCLass = apply(confmmatrix, 1, sum) # number of instances per class
n_pred_pCLass = apply(confmmatrix, 2, sum) # number of predictions per class
actual = n_inst_pCLass / num_instances # distribution of instances over the actual classes
predicted = n_pred_pCLass / num_instances # distribution of instances over the predicted classes
NG20accuracy = sum(correct_classifier) / num_instances
NG20accuracy
NG_20Precision = correct_classifier / n_pred_pCLass
NG_20Precision
NG_20Recall = correct_classifier / n_inst_pCLass
NG_20Recall
NG_20F1 = 2 * NG_20Precision * NG_20Recall / (NG_20Precision + NG_20Recall)
NG_20F1
data.frame(NG_20Precision, NG_20Recall, NG_20F1)
```

D

```
LSA<-function(input,dim){
  s<-svd(input)
  u<-as.matrix(s$u[,1:dim])
  v<-as.matrix(s$v[,1:dim])
  d<-as.matrix(diag(s$d)[1:dim,1:dim])
  return(as.matrix(u%*%d%*%t(v),type="green"))
  # return(as.matrix(u%*%d,type="green"))  # uncomment  to get document vector only
  # return(as.matrix(d%*%t(v),type="green"))  #  uncomment to get word vector only

}
#SVD With 5 Dimensions
svd_dtm5<-LSA(dtmspostproc,5)
svd_norm5<-norm_eucl(svd_dtm5)
#Clustering with 5 dimensions
lsacluster5 <- kmeans(svd_norm5, 5)
lsa_Confm5 <- table(category, lsacluster5$cluster)
#SSE
lsacluster5$totss
lsacluster5$tot.withinss
(lsacluster5$tot.withinss/lsacluster5$totss)*100
#Accuracy
(sum(apply(lsa_Confm5, 1, max))/sum(lsacluster5$size))*100
#plot
pr<-prcomp(svd_norm5)$x
plot( pr,col=lsacluster5$cluster,main='SVD for d=5')


# Find representative words
num=5
concept<-function(num){
  sv<-sort.list((svd(dtmspostproc))$v[,num],decreasing = TRUE)
  dm<-dtmspostproc$dimnames$Terms[head(sv,10)]
  dm
}
i<-(1:num)
lapply(i,concept)
```

E

```
#LDA
burnin <- 4000
iter <- 1000
thin <- 500
seed <-list(2003,5,63,100001,765)
nstart <- 5
best <- TRUE
```

```
lda5 <-LDA(dtmspostproc,5, method="Gibbs", control=list(nstart=nstart, seed = seed, best=best, burnin = burnin, iter = iter, thin=thin))
#top 10 terms in each topic
lda5.terms <- as.matrix(terms(lda5,10))
lda5.terms
lda_data<-as.data.frame(lda5@gamma)
lda_data_matrix<-as.matrix(lda_data)
rownames(lda_data_matrix)<-1:nrow(lda_data_matrix)
norm_eucl <- function(m) m/apply(m, MARGIN=1, FUN=function(x) sum(x^2)^.5)
lda_norm<-norm_eucl(lda_data_matrix)


# LDA with 5 clusters
lda_cluster5<-kmeans(lda_norm,centers=5,nstart=50)
#Plot:
plot(prcomp(lda_norm)$x,col=lda_cluster5$cluster,main='LDA for d=5')
lda_Confm <- table(category, lda_cluster5$cluster)
#Confusion Matrix
lda_Confm
#SSE
lda_cluster5$totss
lda_cluster5$tot.withinss
(lda_cluster5$tot.withinss/lda_cluster5$totss)*100
#Accuracy
(sum(apply(lda_Confm, 1, max))/sum(lda_cluster5$size))*100


#Precision/Recall/F1 Calculations:
num_instances = sum(lda_Confm) # number of instances
num_classes = nrow(lda_Confm) # number of classes
correct_classifier = diag(lda_Confm) # number of correctly classified instances per class
n_inst_pCLass = apply(lda_Confm, 1, sum) # number of instances per class
n_pred_pCLass = apply(lda_Confm, 2, sum) # number of predictions per class
actual = n_inst_pCLass / num_instances # distribution of instances over the actual classes
predicted = n_pred_pCLass / num_instances # distribution of instances over the predicted classes
NG20accuracy = sum(correct_classifier) / num_instances
NG20accuracy
NG_20Precision = correct_classifier / n_pred_pCLass
NG_20Precision
NG_20Recall = correct_classifier / n_inst_pCLass
NG_20Recall
NG_20F1 = 2 * NG_20Precision * NG_20Recall / (NG_20Precision + NG_20Recall)
NG_20F1
data.frame(NG_20Precision, NG_20Recall, NG_20F1)
```

<u>**F**</u>

```
install.packages("jsonlite")
library("jsonlite")
yelp <- stream_in(file("C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/yelp_dataset/dataset/business.json"))
yelp_flat<-flatten(yelp)
yelp_tbl<-as.data.frame(yelp_flat)
yelp1 <- stream_in(file("C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/yelp_dataset/dataset/review.json"))
yelp_flat1<-flatten(yelp1)
yelp_tbl2<-as.data.frame(yelp_flat1)
#Combine two attributes of review json file
rev<-yelp_tbl2[,c('business_id','text')]
#fetching cities
yelp_tbl$city
yelp_tbl$categories
#grepl compares the attribute
business_category<-which(grepl("Restaurants",yelp_tbl$categories) | grepl("Automotive",yelp_tbl$categories)
|grepl("Gyms",yelp_tbl$categories)|grepl("Doctors",yelp_tbl$categories)|grepl("Fashion",yelp_tbl$categories))

business_indx<-yelp_tbl$business_id[business_category]
business_indx
business_df<-data.frame(business_indx)
colnames(business_df)<-"business_id"
business_df<-data.frame(lapply(business_df,as.character),stringsAsFactors=FALSE)
merge_table<-merge(rev,business_df,by='business_id')
merge_table<-merge_table[sample(nrow(merge_table),nrow(merge_table)),]
```

```
merge_sample<-merge_table[sample(1:nrow(merge_table),5000,replace=FALSE),]
write.csv(merge_sample,"C:/STUDY/MS/Spring18/ADM/Assignments/Assignment1/DataSet/yelp_dataset/dataset/Yelp.csv")
```