
Contents

- **Abstract**
- **Introduction**
- **Motivation**
- **Objectives**
- **Variable list and data type, Software used**
- **Method of data collection**
- **Data pre-processing**
- **Methods and terminologies used**
- **Construction of IR system**
- **Construction of GUI**
- **Conclusion**
- **Reference**
- **Appendix**

Abstract

Objective:

Text-based classifications algorithm for facilitating shelf-allotment of books in the library of department of statistics spu.

To create a information retrieval system for departmentallibrary also to create GUI for the given system.

Methods:

In order to create this system 2392 distinct books from departmental library were selected and they are collected manually one by one and text was extracted using the titles of books in R and under the concepts of text mining it is executed under Nlp further IR system was constructed using vector space models and finally GUI was created using Tkinter module in python.

Output: The system looks as in figure below,

Book no.	Name of Book	Author
575	Introduction To Linear Regression Analysis	D. C. Montgomery, E. A. Peck
700	Regression Analysis by Example	Samprit Chatterjee
564	Applied Multivariate Statistical Analysis	R. A. Johnson And D. Wichern
619	Data mining methods and models	Daniel. T. Latose
699	Introduction to Applied Statistics	----
521	Statistical Inference	G. Casela, R. L. Berger
684	Survival Analysis	Miller
636	Biostatistics Principles and Practice	A Antonisamy

Introduction

The word '**Library**' comes from the Anglo-French word '*Libraire*' which means bookshop which later evolved into the word library. A library is a collection of sources of information and similar resources. The history of libraries can be dated back to 26 century BC when the art of writing was developed. The early libraries consist of archives of material on which the early humans used to write such as clay pots, stones, bones etc.

Even in this 21st Era still the world has come closer libraries have maintained their importance in the society. As the development of human the size of libraries have increase extensively making it difficult to manage these libraries.

The Department of Statistics Sardar Patel University, vidya nagar have a well developed departmental library placed near to the computer lab. It consists of total 19 cupboards with 2392 books with research papers, thesis are include in it under my consideration among that 3 cupboards are project journals which are not considered under our project.

Text mining is a new and exciting research area that tries to solve the information overload problem by using techniques from machine learning, natural language processing (NLP), data mining, information retrieval (IR), and knowledge management. Text mining involves the pre-processing of document collections such as information extraction, term extraction, text categorization, and storage of intermediate representations. The techniques that are used to analyse these intermediate representations such as clustering, association rules visualisation of the results, classification models, comparisons of binary logistic regression model and based on this a conclusion is drawn.

Text mining can be referred as a knowledge intensive process in which using a various suites of analysis tools, user interacts with a document collection. The text mining also extracts the useful information from data sources through the explorations and identifications of interesting patterns, which are similar or analogous to data mining. In this case of text mining, the data sources are document collections, and patterns are not found among formalised database records but in the unstructured textual data in the documents in these collections.

Certainly, from seminal research on data mining the text mining derives much of its direction and inspiration. So, it is not surprising to find that data mining and text mining systems have many high-level architectural similarities. For instance, both types of systems rely or based on pattern-discovery algorithms, presentation-layer elements and pre-processing routines such as visualisation tools to enhance the output data. Further, text mining adopts many of the specific types of patterns in its core knowledge discovery operations that were first introduced and vetted in data mining research.

Motivation

The departmental library has 2392 books dedicated different topics related to statistics although some one might be interested to have a particular book which have a particular keyword (e.g some one just want a book having information about say markov chain) in such case he have either of two choices ask someone for the relevant book or to go through the similar of all the books thus it can be helpful if there is a system which can find relevant books related to a query (keyword).

Objectives

Text-based classifications algorithm for facilitating shelf-allotment of books in the library of department of statistics SPU.

- To create a Information Retrieval system using NLP
- Under the text-mining package in R and python
 - Finding books relevant to single keyword
 - Finding books relevant to both keywords when two keywords are provided (AND)
 - Finding books relevant to either of keywords when two keywords are provided (OR)
- To create a Graphical User Interface (GUI) for this system.

Method of Data Collection

➤ Variable list and data type, Software used

Out of the total 2392 books in the departmental library 2392 distinct books were taken into the study. It was insured to collect the whole library books data related to each paper in the course of M.sc statistics, Applied statistics and MQPM has been included in the sample. The book data of each of the selected book was such as Shelf no , Row no, Book no, No of authors, 1st author name, Publication, Edition, Bound, Color, No of pages, Book type, Year, No of Copies, Remarks and Title of Books are collected in Excel sheets for our further Analysis part . The whole Book data collection time took is 1month by two project members.

Softwares used in it are Ms-Excel, R-Studio, Python

Data Variables:

LIBRARY_DATA_INTERNAL.xlsx - Microsoft Excel

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	SHELF NO	ROW NO	BOOK NO	NO OF AUTHORS	1ST AUTHOR	PUBLICATION	EDITION	BOUND	COLOUR	NO OF PAGES	TYPE	YEAR	NO. OF COPIES	REMARK	
2	2	1		2	S.C. MALIK	NEW AGE	4	SOFT	YELLOW	870	MATHEMATICAL	2010			
3	2	1	PNJM9/34	1	EARL A.CODDINGTON	PRENTICE HALL INDIA	1	PAPER	BLACK	292	MATHEMATICAL	1961			
4	2	1	PNJMS/30	1	N.PISKUNOV	PEACE	1	PAPER	YELLOW	895	MATHEMATICAL				
5	2	1	PNJM8/33	1	JAMES B.SCARBOROUGH	OXFORD & IBH	6	HARD	ORANGE	600	MATHEMATICAL	1964			
6	2	1	233493	2	SURESH CHANDRA	NAROSA	1	SOFT	BLUE	254	RESEARCH	2013			
7	2	1	233494	2	SURESH CHANDRA	NAROSA	1	SOFT	BLUE	254	RESEARCH	2013		2 SRNO-5	
8	2	1	221547	2	PETER MORTERS	CAMBRIDGE UNIVERSITY	1	SOFT	BLUE	403	MATHEMATICAL	2010			
9	2	1	224055	2	SANJEEV KULKARNI	A JOHN WILEY & SONS	1	HARD	BLACK	209	STATISTICAL	2011			
10	2	1	PNJM1/26	1	J.V. USPENSKY	MCGRAW-HILL	1	HARD	BROWN	353	MATHEMATICAL	1948			
11	2	1	242433	1	CHARLES L. BYRNE	CRC	1	HARD	PURPLE	279	MATHEMATICAL	2014			
12	2	1	PNJM6/31	1	TOM M.APOSTOL	WESLEY	1	HARD	BROWN	559	MATHEMATICAL	1957			
13	2	1	5/MS/2	1	S.S. SASTRY	PRENTICE HALL INDIA	2	SOFT	GREEN	296	MATHEMATICAL	1993			
14	2	1	18/G/2	1	JOSEPH GIBALDI	AFFILIATED EAST-WEST	4	SOFT	YELLOW	293	HANDBOOK	1996			
15	2	1	227864	2	MANFRED DENKER	BIRKHAUSER	1	SOFT	BLACK	509	STATISTICAL	1998			
16	2	1	28/ECO/1	2	B.C. MEHTA	SULTAN CHAND	1	SOFT	GREEN	493	MATHEMATICAL	1973			
17	2	1	250230	4	ROBERT COE	SAGE	2	SOFT	BLUE	372	RESEARCH	2017			
18	2	1	250237	1	LAURA RUTH JOHNSON	SAGE	1	SOFT	BLUE	187	RESEARCH	2017			
19	2	1	250241	1	SCOTT A. MILLER	SAGE	5	SOFT	WHITE	466	RESEARCH	2018			
20	2	1	250233	2	PARVI ERIKSSON	SAGE	2	SOFT	WHITE	363	RESEARCH	2016			
21	2	1	204629	1	ETHEM ANPAYDIN	PRENTICE HALL INDIA	1	HARD	ORANGE	415	STATISTICAL	2004			
22	2	1	2A/ECO/1	2	B.C. MEHTA	SULTAN CHAND	1	SOFT	GREEN	493	MATHEMATICAL	1973		2 SRNO-15	
23	2	1	19/G/2	2	G.S.DIWAN	THE POPULAR BOOK DEPOT	2	HARD	GREEN	324	MATHEMATICAL	1949			
24	2	1	PNJG4/19	2	J.F. KENNEY	AFFILIATED EAST-WEST	3	HARD	BLACK	348	MATHEMATICAL	1974			
25	2	1	249249	1	MICHEL KERN	WILEY	1	HARD	GREEN	212	MATHEMATICAL	2016			
26	2	1	245828	1	M.L. THIVAGAR	STUDERA PRESS	1	HARD	BLACK	232	REPORT	2016			
27	2	1	245823	2	WALTER G.KELLEY	SPRINGER	2	SOFT	YELLOW	423	MATHEMATICAL	2010			
28	2	2	202251	2	ANGELA DEAN	SPRINGER	1	SOFT	ORANGE	740	STATISTICAL	2006			
29	2	2	233495	1	MICHAEL J. CRAWLEY	WILEY	2	HARD	GREEN	1051	STATISTICAL	2013			
30	2	2	213201	5	JOSEPH F. HAIR, JR.	PEARSON	6	SOFT	GREEN	923	STATISTICAL	2006			

Text-Encoding

It is necessary to pre-process the text documents and store the information in a data structure for mining large document collections, which is more suitable for further processing than a plain text file. Various methods exist that try to exploit also the syntactic structure and semantics of text document, most text mining approaches are based on the idea that a text document can be represented by a set of words, which means a text document is described based on the set of words contained in it

Data pre-processing



The demo R script and demo input csv file are available on my system R has a rich set of packages for Natural Language Processing (NLP) and generating plots. The foundational steps involve loading the text csv into an R Corpus, then cleaning and stemming the data before performing analysis. I will demonstrate these steps and analysis like Word Frequency, Word Cloud, Word Association, and various plots and charts.

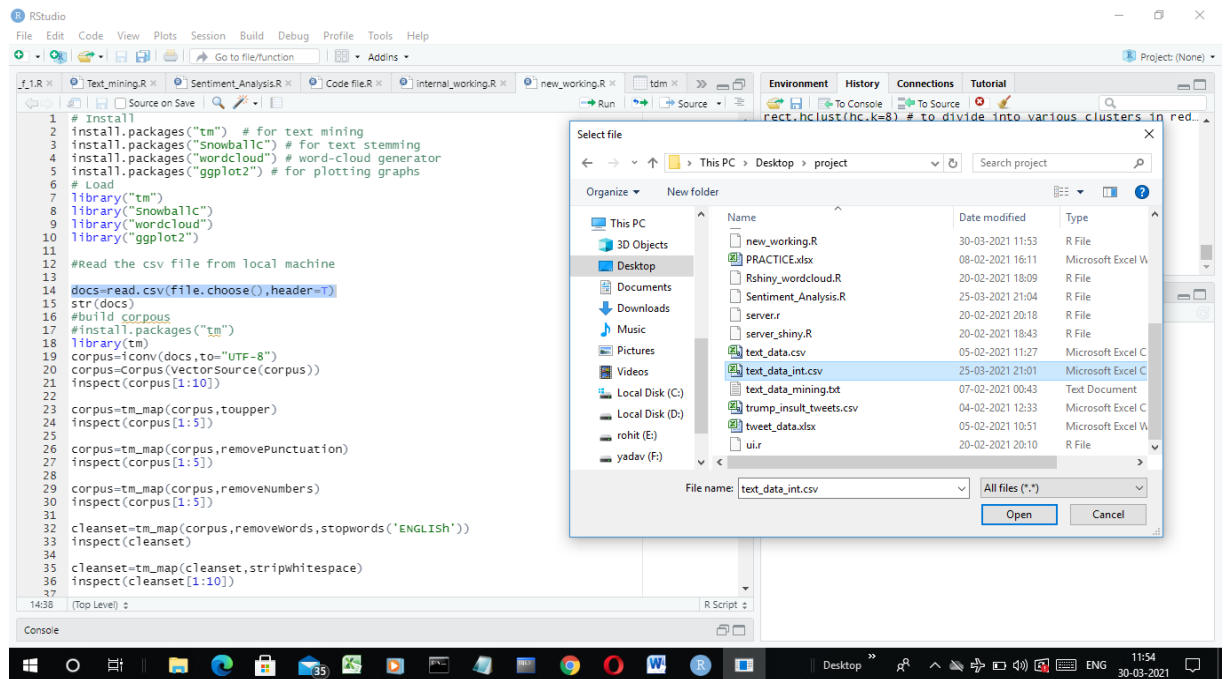
Although the extracted text was not clean and had many impurities like numbers, punctuations, special characters which was then removed using user R

Installing and loading R packages

The following packages are used in the examples in this article:

- **tm** for text mining operations like removing numbers, special characters, punctuations and stop words (Stop words in any language are the most commonly occurring words that have very little value for NLP and should be filtered out. Examples of stop words in English are “the”, “is”, “are”.)
- **snowballc** for stemming, which is the process of reducing words to their base or root form. For example, a stemming algorithm would reduce the words “fishing”, “fished” and “fisher” to the stem “fish”.
- **wordcloud** for generating the word cloud plot.
- **ggplot2** for plotting graphs

- **# Install**
- `install.packages("tm")` # for text mining
- `install.packages("SnowballC")` # for text stemming
- `install.packages("wordcloud")` # word-cloud generator
- `install.packages("ggplot2")` # for plotting graphs
- **# Load**
- `library("tm")`
- `library("SnowballC")`
- `library("wordcloud")`
- `library("ggplot2")`



Cleaning up text-data

Cleaning the text data starts with making transformations like removing special characters from the text. This is done using the `tm_map()` function to replace special characters like `/`, `@` and `|` with a space. The next step is to remove the unnecessary whitespace and convert the text to upper case.

Then remove the *stopwords*. They are the most commonly occurring words in a language and have very little value in terms of gaining useful information. They should be removed before performing further analysis. Examples of stopwords in English are “the, is, at, on”. There is no single universal list of stop words used by all NLP tools. stopwords in the `tm_map()` function supports several languages like English, French, German, Italian, and Spanish. Please note the language names are case sensitive. Next, remove numbers and punctuation.

The last step is text stemming. It is the process of reducing the word to its root form. The stemming process simplifies the word to its common origin. For example, the stemming process reduces the words “fishing”, “fished” and “fisher” to its stem “fish”. Please note stemming uses the *SnowballC* package. (You may want to skip the text stemming step if your users indicate a preference to see the original “unstemmed” words in the word cloud plot)

```
#Replacing "/", "@" and "|" with space
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
TextDoc <- tm_map(TextDoc, toSpace, "/")
TextDoc <- tm_map(TextDoc, toSpace, "@")
TextDoc <- tm_map(TextDoc, toSpace, "\\")

# Convert the text to uppercase
TextDoc <- tm_map(TextDoc, content_transformer(uppercase))

# Remove numbers
TextDoc <- tm_map(TextDoc, removeNumbers)

# Remove english common stopwords
TextDoc <- tm_map(TextDoc, removeWords, stopwords("english"))

# Remove your own stop word
# specify your custom stopwords as a character vector
TextDoc <- tm_map(TextDoc, removeWords, c("and"))

# Remove punctuations
TextDoc <- tm_map(TextDoc, removePunctuation)

# Eliminate extra white spaces
TextDoc <- tm_map(TextDoc, stripWhitespace)

# Text stemming - which reduces words to their root form
TextDoc <- tm_map(TextDoc, stemDocument)
```

Building term document matrix

After cleaning the text data, the next step is to count the occurrence of each word, to identify popular or trending topics. Using the function `TermDocumentMatrix()` from the text mining package, you can build a Document Matrix – a table containing the frequency of words.

Build a term-document matrix

```
TextDoc_dtm <- TermDocumentMatrix(TextDoc)
```

```
dtm_m <- as.matrix(TextDoc_dtm)
```

Sort by decreasing value of frequency

```
dtm_v <- sort(rowSums(dtm_m),decreasing=TRUE)
```

```
dtm_d <- data.frame(word = names(dtm_v),freq=dtm_v)
```

Bar chart for the words frequency less than 10

One could interpret the following from this bar chart:

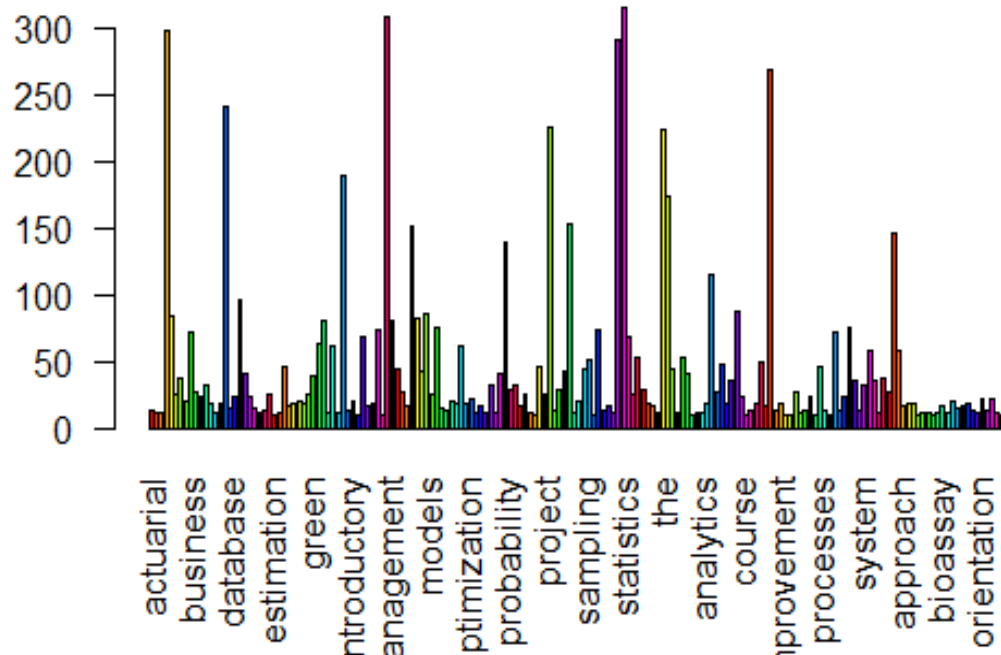
- The most frequently occurring word is “Statistics” and “Management”. Also notice that negative words like “not” don’t feature in the bar chart, which indicates there are no negative prefixes to change the context or meaning of the word “good” (In short, this indicates most responses don’t mention negative phrases like “not good”).

barplot frequent term

```
freq=rowSums(m)
```

```
freq=subset(freq,freq>=10) #words appear more than 10 times
```

```
barplot(freq,las=2,col=rainbow(25))
```



Generate the word cloud

A word cloud is one of the most popular ways to visualize and analyze qualitative data. It's an image composed of keywords found within a body of text, where the size of each word indicates its frequency in that body of text. Use the word frequency data frame (table) created previously to generate the word cloud. In your R script, add the following code and run it to generate the word cloud and display it in the *Plots* section of RStudio.

```
# generate wordcloud

set.seed(1234)
w=sort(rowSums(tdm),decreasing = T)
wordcloud(words = names(w), freq =w, min.freq = 5,
          max.words=200, random.order=FALSE,
          colors=brewer.pal(8, "Dark2"),scale=c(2,0.3))
```

Below is a brief description of the arguments used in the word cloud function:

- **words** – words to be plotted
- **freq** – frequencies of words
- **min.freq** – words whose frequency is at or above this threshold value is plotted (in this case, I have set it to 5)
- **max.words** – the maximum number of words to display on the plot (in the code above, I have set it 100)
- **random.order** – I have set it to FALSE, so the words are plotted in order of decreasing frequency
- **rot.per** – the percentage of words that are displayed as vertical text (with 90-degree rotation). I have set it 0.40 (40 %), please feel free to adjust this setting to suit your preferences
- **colors** – changes word colors going from lowest to highest frequencies

You can see the resulting word cloud



Hierarchical clustering

The first algorithm we'll look at is [hierarchical clustering](#). strategies for hierarchical clustering fall into two types:

Agglomerative: where we start out with each document in its own cluster. The algorithm iteratively merges documents or clusters that are closest to each other until the entire corpus forms a single cluster. Each merge happens at a different (increasing) distance.

Divisive: where we start out with the entire set of documents in a single cluster. At each step the algorithm splits the cluster recursively until each document is in its own cluster. This is basically the inverse of an agglomerative strategy.

The algorithm we'll use is [hclust](#) which does agglomerative hierarchical clustering. Here's a simplified description of how it works:

1. Assign each document to its own (single member) cluster
2. Find the pair of clusters that are closest to each other and merge them.
So you now have one cluster less than before.
3. Compute distances between the new cluster and each of the old clusters.
4. Repeat steps 2 and 3 until you have a single cluster containing all documents.

We'll need to do a few things before running the algorithm. Firstly, we need to convert the DTM into a standard matrix which can be used by [dist](#), the distance computation function in R (the DTM is not stored as a standard matrix). We'll also shorten the document names so that they display nicely in the graph that we will use to display results of *hclust* (the names I have given the documents are just way too long). Here's the relevant code:

```
# Convert dtm to matrix

tdm=TermDocumentMatrix(cleanset, control = list(minWordLength=c(1,Inf)))

t=removeSparseTerms(tdm,sparse = 0.98)

m=as.matrix(t)

write.csv(m,file="tdmmat.csv")
```

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source
Console Terminal Jobs
C:/Users/DELL/Desktop/project/
> m=as.matrix(t)
> m
      DOCS
Terms  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
analysing 2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
analysis  2  7 19  4  3  2  0  1  0  1  0  1  0  0  0  0  0  0
applied   7  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0
aryabhata 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
basic     6  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
bayesian  3  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
big       1  0  1  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
bioequivalence 1  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
biometry  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
biostatistics 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
business  1  0  3  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
callisters 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
cambridge 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
categorical 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
circular  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
classical 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
clean     1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
clinical  1  0  0  0  2  1  0  0  0  0  0  0  0  0  0  0  0  0
combinatorial 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
combinatorics 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
contributions 1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
data      14  4  7  5  1  0  1  0  0  0  0  0  0  0  1  0  0  0

```

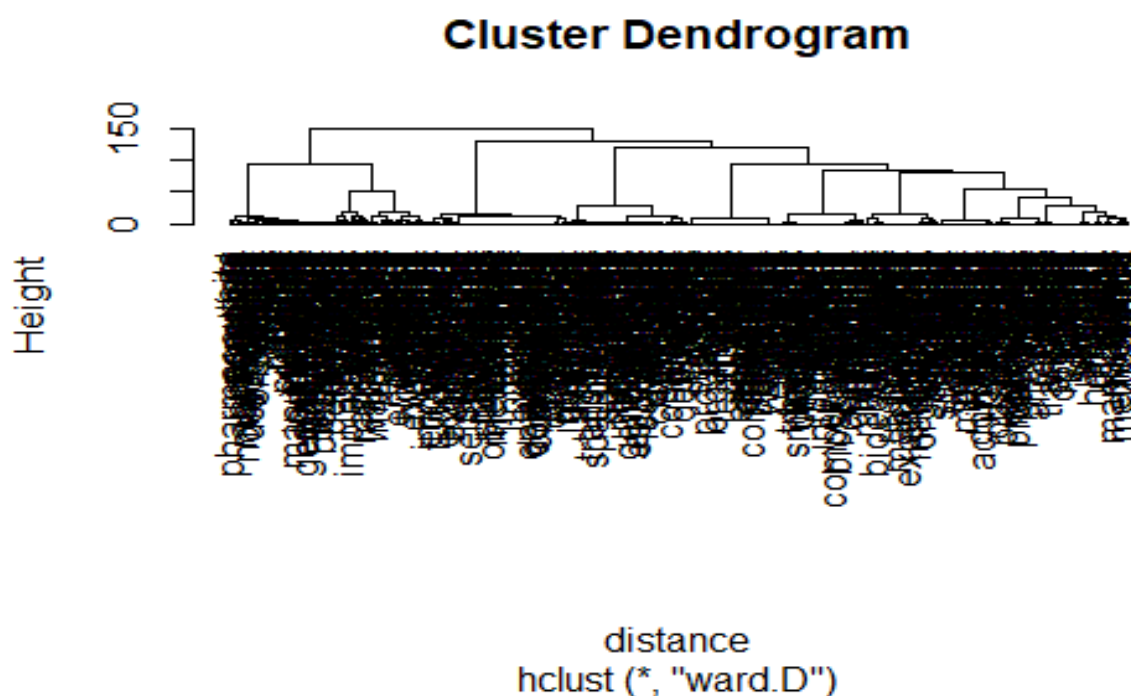
Next we run *hclust*. The algorithm offers several options check out the documentation for details. I use a popular option called [Ward's method](#) – there are others, and I suggest you experiment with them as each of them gives slightly different results making interpretation somewhat tricky (did I mention that clustering is as much an art as a science??). Finally, we visualise the results in a [dendrogram](#).

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
_f_1.R x Text_mining.R x Sentiment_Analysis.R x Code file.R x Internal_working.R x new_working.R x tdm x
129
130 #Hierarchical word clustering using dendrogram
131 distance=dist(scale(m), method = "canberra")
132 print(distance,digits = 4)
133 hc=hclust(distance,method = "ward.D")
134 plot(hc,hang=-1)
135 rect.hclust(hc,k=8) # to divide into various clusters in red rectangles
136 hc$order
137
138
139 hc=hclust(distance,method = "complete")
140 plot(hc,hang=-1)
141 rect.hclust(hc,k=8)

```


Cluster dendrogram



Dendrogram from hierarchical clustering of corpus

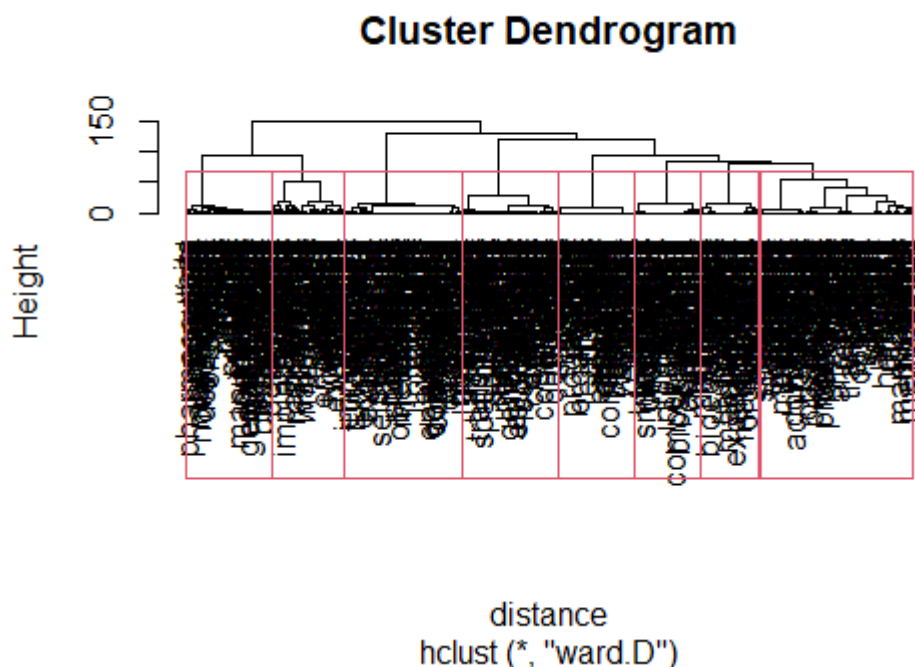
A few words on interpreting dendrograms for hierarchical clusters: as you work your way down the tree in above figure, each branch point you encounter is the distance at which a cluster merge occurred. Clearly, the most well-defined clusters are those that have the largest separation; many closely spaced branch points indicate a lack of dissimilarity (i.e. distance, in this case) between clusters. Based on this, the figure reveals that there are 2 well-defined clusters – the first one consisting of the three documents at the right end of the cluster and the second containing all other documents. We can display the clusters on the graph using the *rect.hclust* function like so:

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
_f_1.R × Text_mining.R × Sentiment_Analysis.R × Code file.R × internal_working.R × new_working.R × tdm ×
Source on Save Run Source
129
130 #Hierarchical word clustering using dendrogram
131 distance=dist(scale(m), method = "canberra")
132 print(distance,digits = 4)
133 hc=hclust(distance,method = "ward.D")
134 plot(hc,hang=-1)
135 rect.hclust(hc,k=8) # to divide into various clusters in red rectangles

```

Output:



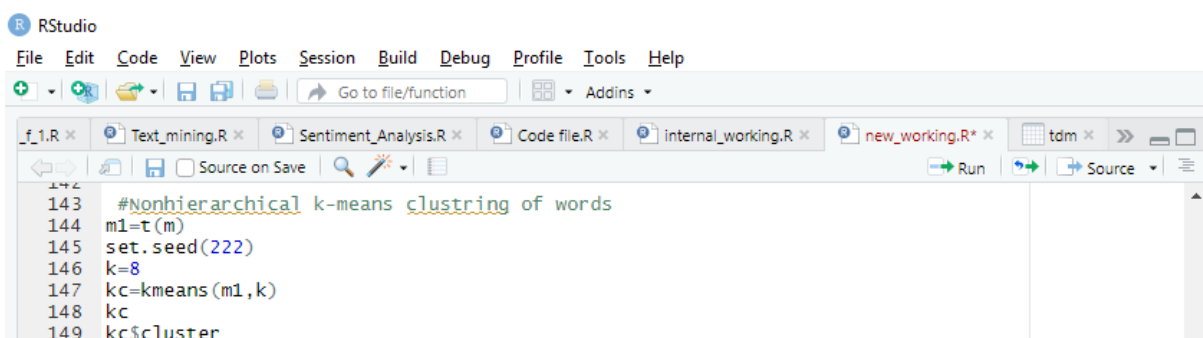
K means clustering

In hierarchical clustering we did not specify the number of clusters upfront. These were determined by looking at the dendrogram *after* the algorithm had done its work. In contrast, our next algorithm – [*K means*](#) – requires us to define the number of clusters upfront (this number being the “k” in the name). The algorithm then generates k document clusters in a way that ensures the within-cluster distances from each cluster member to the [*centroid*](#) (or *geometric mean*) of the cluster is minimised. Here’s a simplified description of the algorithm:

1. Assign the documents randomly to k bins

2. Compute the location of the centroid of each bin.
3. Compute the distance between each document and each centroid
4. Assign each document to the bin corresponding to the centroid closest to it.
5. Stop if no document is moved to a new bin, else go to step 2.

An important limitation of the k means method is that the solution found by the algorithm corresponds to a *local* rather than *global* minimum ([this figure](#) from Wikipedia explains the difference between the two in a nice succinct way). As a consequence it is important to run the algorithm a number of times (each time with a different starting configuration) and then select the result that gives the overall lowest sum of within-cluster distances for all documents. A simple check that a solution is robust is to run the algorithm for an increasing number of initial configurations until the result does not change significantly. That said, this procedure does *not* guarantee a globally optimal solution.



```

143 #Nonhierarchical k-means clustering of words
144 m1=t(m)
145 set.seed(222)
146 k=8
147 kc=kmeans(m1,k)
148 kc
149 kc$cluster

```

Output:

```

Clustering vector:
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
6  3  5  2  7  8  4  1  1  1  1  1  1  1  1  1  1  1

within cluster sum of squares by cluster:
[1] 196.9091  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
(between_SS / total_SS =  96.6 %)

Available components:
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"

```

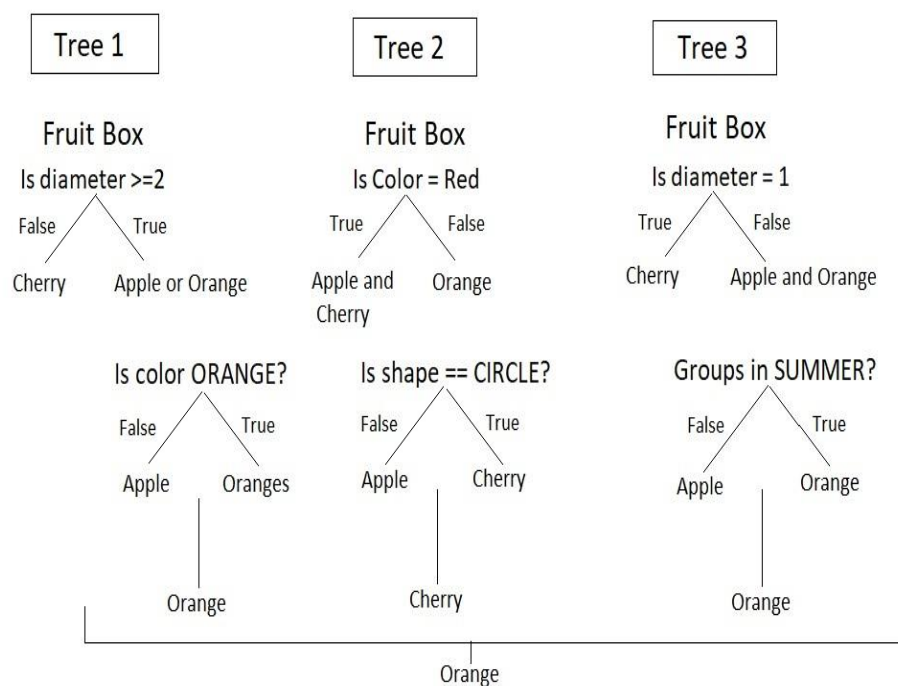
Further on the clustering vector is attached with the library collected data for continuing the further data analysis by creating a classifier model such a Random forest model for the comparison with binary logistic regression model

Random forest takes random samples from the observations, random initial variables (columns) and tries to build a model. Random forest algorithm is as follows:

- Draw a random bootstrap sample of size **n** (randomly choose **n** samples from training data).
- Grow a decision tree from bootstrap sample. At each node of tree, randomly select **d** features.
- Split the node using features (variables) that provide best split according to objective function. For instance, by maximizing the information gain.
- Repeat steps 1 to step 2, **k** times (k is the number of trees you want to create using subset of samples).
- Aggregate the prediction by each tree for a new data point to assign the class label by majority vote i.e pick the group selected by most number of trees and assign new data point to that group.

Example:

Consider a Fruit Box consisting of three fruits Apples, Oranges, and Cherries in training data i.e $n = 3$. We are predicting the fruit which is maximum in number in a fruit box. A random forest model using the training data with a number of trees, $k = 3$.



```
# Installing package
install.packages("caTools")    # For sampling the dataset
install.packages("randomForest") # For implementing random forest
algorithm

# Loading package
library(caTools)
library(randomForest)

# Splitting data in train and test data
split <- sample.data(data, SplitRatio = 0.7)
split

train <- subset(data, split == "TRUE")
test <- subset(data, split == "FALSE")

# Fitting Random Forest to the train dataset
set.seed(120) # Setting seed
classifier_RF = randomForest(x = train[-5],
                             y = train$Title,
                             ntree = 500)

classifier_RF
```

Methods and Terminologies Used

1. TF-IDF

In information retrieval TF-IDF, short for “Term Frequency–Inverse Document Frequency”, is a numerical statistic that is intended to reflect how important a word is to a document in a collection. The TF–IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. TF–IDF is one of the most popular term-weighting schemes today.

TF-IDF consist of two parts TF and IDF. TF measures the occurrences of a word in a given document while IDF measures how rare is the word between the documents. There are various variants of both TF and IDF

$$w_{ij} = tf_{ij} \times \log_2 \frac{N}{n}, \text{ where}$$

w_{ij} = weight of term T_j in document D_i

tf_{ij} = frequency of term T_j in document D_i

N = number of documents in collection

n = number of documents where T_j occurs at least once

Variants of term frequency (TF) weight

weighting scheme	TF weight
binary	0, 1
raw count	$f_{t,d}$
term frequency	$f_{t,d} / \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K) \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Variants of inverse document frequency (IDF) weight

weighting scheme	IDF weight ($n_t = \{d \in D : t \in d\} $)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log \left(1 + \frac{N}{n_t} \right)$
inverse document frequency max	$\log \left(\frac{\max_{\{t' \in d\}} n_{t'}}{1 + n_t} \right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

2. Vector Space model

Vector space model or term vector model is an algebraic model for representing text documents (and any objects, in general) as vectors of identifiers, such as, for example, index terms. It is used in information filtering, information retrieval, indexing and relevancy rankings. Its first use was in the SMART Information Retrieval System.

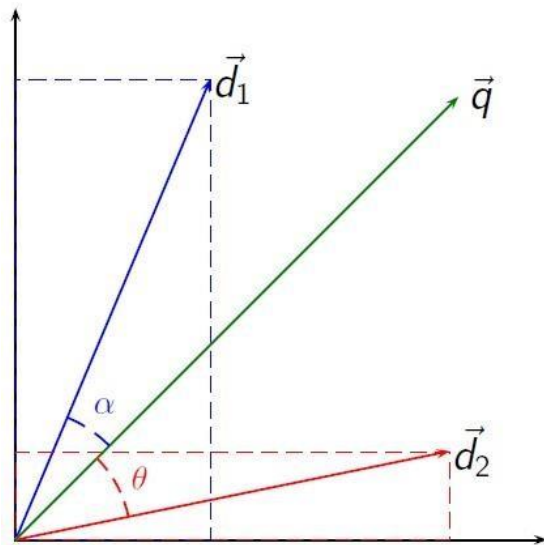
Documents and queries are represented as vectors.

$$d_j = (w_1, w_2, \dots, w_n) \quad ; j = 1, 2, \dots, k \text{ (number of documents)}$$

i.e. every row of TF-IDF can be treated as a vector in a dimension space equal to the number of columns in it. Relevance rankings of documents in a keyword search can be calculated, using the assumptions of document similarities theory, by comparing the deviation of angles between each document vector and the original query vector where the query is represented as a vector with same dimension as the vectors that represent the other documents.

$$\cos\theta = \frac{d_2 \cdot q}{\|d_2\| \|q\|}$$

$$\text{where, } \|q\| = \sqrt{\sum_{i=1}^n q_i^2}$$



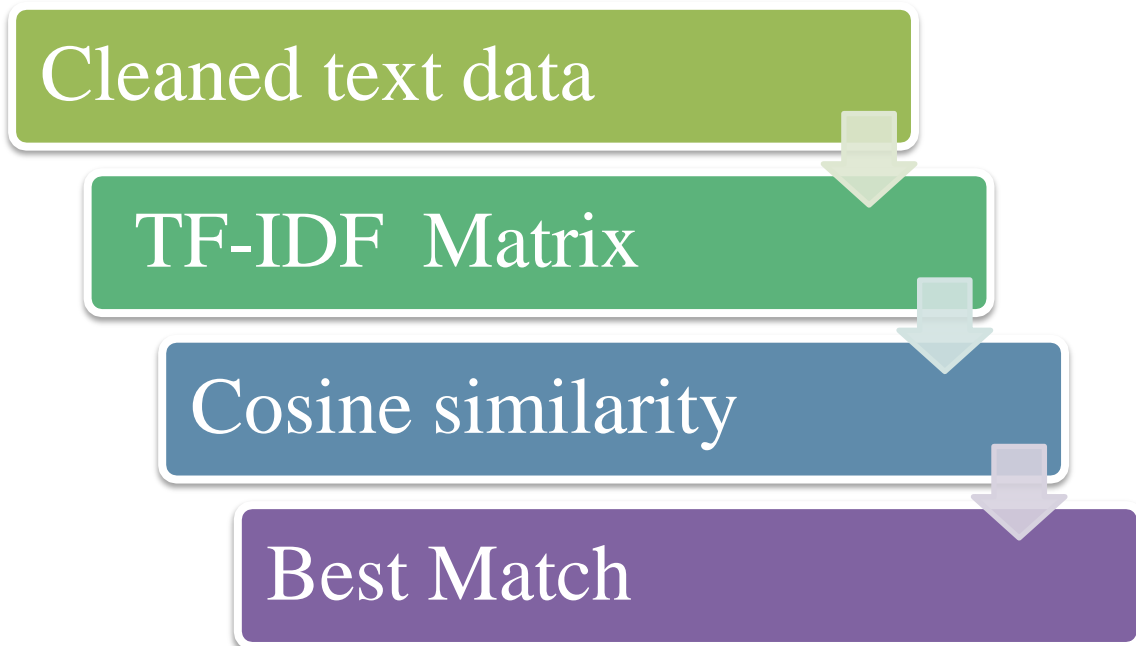
3. Graphical User Interface (GUI)

The graphical user interface (GUI) is a form of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces.

Construction of IR system

The flow chart for construction of IR system is as shown below.

Using the clean data obtained from the initial step TF-IDF matrix was constructed.

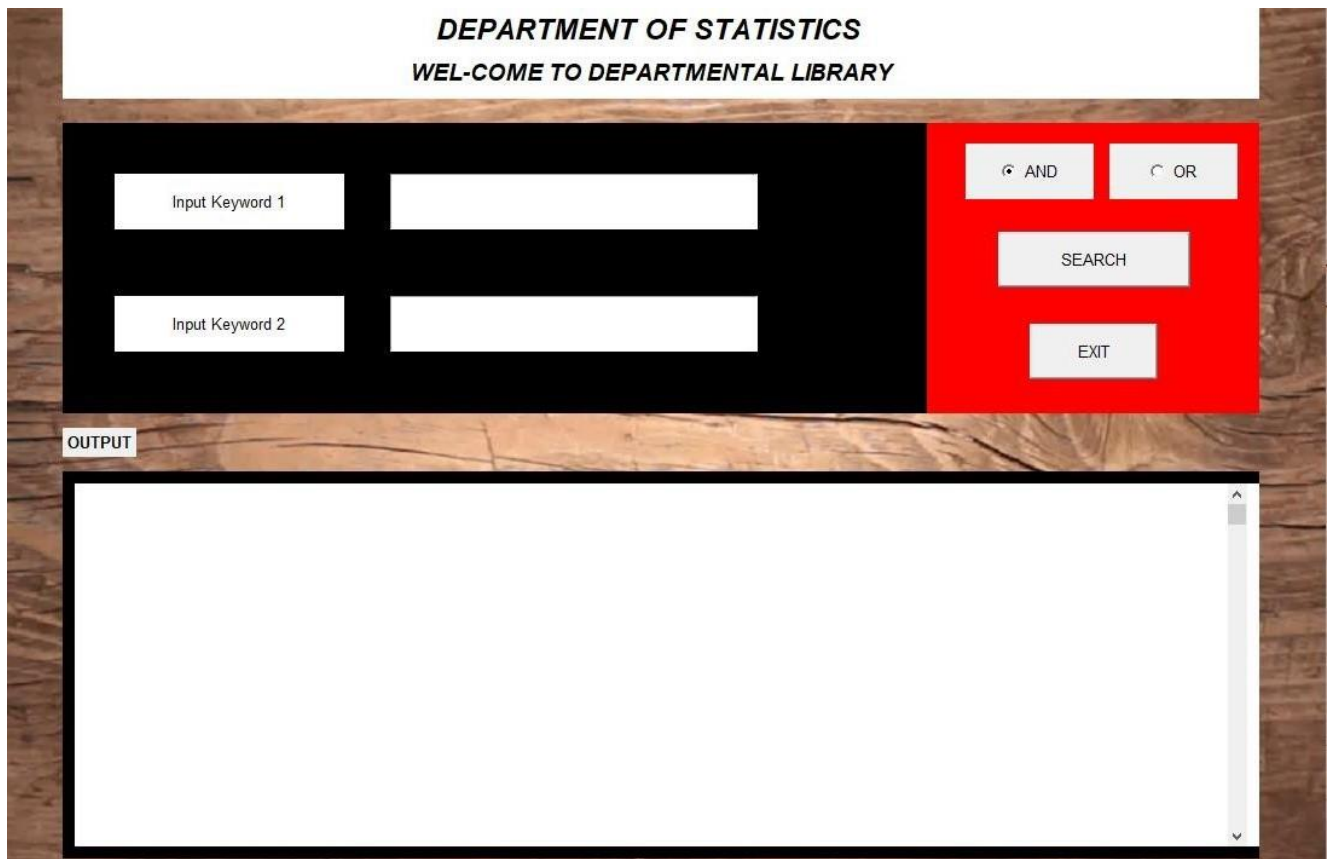


when query is passed the query is also transformed into the form of TF-IDF i.e. in order to determine the best match for the query with the current set of documents cosine similarity is evaluated for query and each of the documents. The values of cosine similarity lie between zero and one, where zero represents no similarity while one representing complete similarity.

As there are $N = 2392$ books we get N values of cosine similarity and more than one document can be relevant to a query thus an appropriate threshold was chosen in order to get all relevant books of an given query and also to avoid errors in the current case a threshold of 0.7 is chosen.

Construction of GUI

The GUI was Created Using tkinter module in python which enables create simple yet compact GUI. Tkinter provide facilities to add Frames, Lables, Canvas, Buttons ,Text Box etc. to the widgets. The design of the GUI is as shown in figure below.



A user can input one or two queries. If the user enters only one query then he get books relivent to that query on the other hand if the user enters two queries the he have to choose either AND or OR. Based on the choice made by the user will get the output. To perform this operation a user defined function is written and binded to the SEARCH button of GUI. The flow of this search function is as shown below.

Flow chart of search function

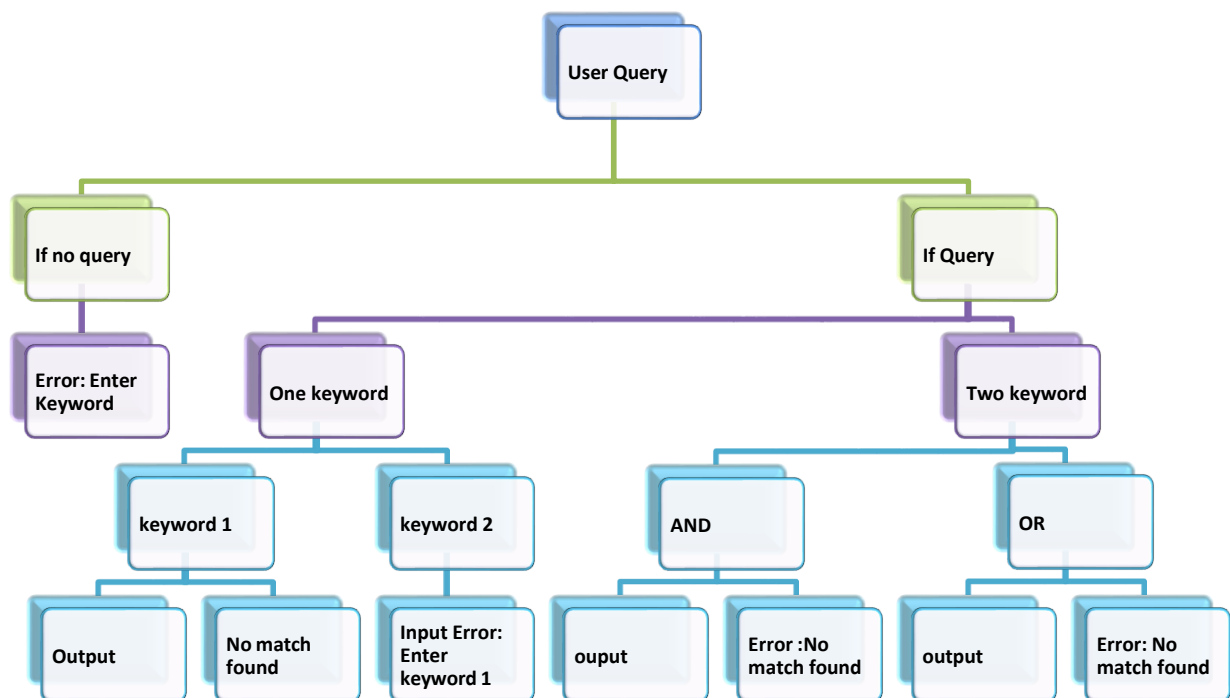


Diagram: Flow chart of search function

Conclusion

This brings us to the end of a long ramble through text-mining, data cleaning, clustering, building classifier model etc. We've explored the two most common methods: *hierarchical* and *k means* clustering. Apart from providing the detailed steps to do clustering, I have attempted to provide an intuitive explanation of how the algorithms work. I hope I have succeeded in doing so. Further on we made an Library IR system to search the books in the library according to the keywords of the books.

Here we created an information retrieval system with the help of GUI in python this ends our project work, also here we concluded a better project work from our side and waiting for yours favourable response.

- **A information retrieval system was created.**
- **GUI for the Information retrieval system was created.**

DEPARTMENT OF STATISTICS
WEL-COME TO DEPARTMENTAL LIBRARY

Input Keyword 1: regression

Input Keyword 2:

Buttons: AND, OR, SEARCH, EXIT

OUTPUT

Book no.	Name of Book	Author
575	Introduction To Linear Regression Analysis	D. C. Montgomery, E. A. Peck
700	Regression Analysis by Example	Samprit Chatterjee
564	Applied Multivariate Statistical Analysis	R. A. Johnson And D. Wichern
619	Data mining methods and models	Daniel. T. Latose
699	Introduction to Applied Statistics	----
521	Statistical Inference	G. Casela, R. L. Berger
684	Survival Analysis	Miller
636	Biostatistics Principles and Practice	A Antonisamy

Reference

- Sanjay K. Dwivedi, Jitendra Nath Singh, Rajesh Gotam
“*Information Retrieval Evaluative Model*” FTICT 2011:
Proceedings of the 2011, International conference on “Future
Trend in Information & Communication Technology,
Ghaziabad, India, Feb -2011

- Jitendra Nath Singh & Sanjay Kumar Dwivedi: “*Analysis of
Vector Space Model in Information Retrieval*”. Proceedings of
the 2012, National Conference on Communication Technologies
& its impact on Next Generation Computing CTNGC 2012

- www.google.com

Appendix

#####Extracted text data in excel sheet #####

EXTERNAL FINAL DATA.csv - Microsoft Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
	SR_NO	SHELF NO	ROW NO	BOOK NO	NO OF AU	1ST AUTH	PUBLICAT	EDITION	BOUND	COLOUR	NO OF PA	TYPE	YEAR	NO. OF CC	REMARK	TITLE OF BOOKS					
1	1	1	1	5 SF/16/13	4	PROF. JAY INTERNAT		1	SOFT	RED	QB.28	STATISTIC	2014								
2	2	1	1	5 SF/16/20	2	CRAIG GYC WILEY		2	SOFT	YELLOW	375	STATISTIC	2012								
3	3	1	1	5 SF/16/21	2	CRAIG GYC WILEY		2	SOFT	YELLOW	375	STATISTIC	2012								
4	4	1	1	5 SF/16/22	2	CRAIG GYC WILEY		2	SOFT	YELLOW	375	STATISTIC	2012								
5	5	1	1	5 SF/16/23	2	CRAIG GYC WILEY		2	SOFT	YELLOW	375	STATISTIC	2012								
6	6	1	1	5 SF/16/24	2	CRAIG GYC WILEY		2	SOFT	YELLOW	375	STATISTIC	2012								
7	7	1	1	5 SF/16/25	2	CRAIG GYC WILEY		2	SOFT	YELLOW	375	STATISTIC	2012								
8	8	1	1	5 SF/16/26	2	CRAIG GYC WILEY		2	SOFT	YELLOW	375	STATISTIC	2012								
9	9	1	1	5 SF/16/27	2	CRAIG GYC WILEY		2	SOFT	YELLOW	375	STATISTIC	2012								
10	10	1	1	5 SF/16/55	2	VIKAS ARC GLOBALA		2	SOFT	GREEN	663	STATISTIC	2015								
11	11	1	1	5 SF/16/28	1	J K SHARA TRINITY		6	SOFT	PURPLE	943	STATISTIC	2012								
12	12	1	1	5 SF/16/29	1	J K SHARA TRINITY		6	SOFT	PURPLE	943	STATISTIC	2012								
13	13	1	1	5 SF/16/30	1	J K SHARA TRINITY		6	SOFT	PURPLE	943	STATISTIC	2012								
14	14	1	1	5 SF/16/18	2	CRAIG GYC WILEY		2	SOFT	YELLOW	375	STATISTIC	2012								
15	15	1	1	5 SF/16/19	2	CRAIG GYC WILEY		2	SOFT	YELLOW	375	STATISTIC	2012								
16	16	1	1	5 SF/16/9	4	PROF. JAY INTERNAT		1	SOFT	RED	9.34	STATISTIC	2014								
17	17	1	1	5 SF/16/11	4	PROF. JAY INTERNAT		1	SOFT	RED	9.34	STATISTIC	2014								
18	18	1	1	5 SF/16/10	4	PROF. JAY INTERNAT		1	SOFT	RED	9.34	STATISTIC	2014								
19	19	1	1	5 SF/16/12	4	PROF. JAY INTERNAT		1	SOFT	RED	9.34	STATISTIC	2014								
20	20	1	1	5 SF/16/15	4	PROF. JAY INTERNAT		1	SOFT	RED	9.34	STATISTIC	2014								
21	21	1	1	5 SF/16/16	4	PROF. JAY INTERNAT		1	SOFT	RED	9.34	STATISTIC	2014								
22	22	1	1	5 SF/16/17	4	PROF. JAY INTERNAT		1	SOFT	RED	9.34	STATISTIC	2014								
23	23	1	1	5 SF/16/1	2	RAMEZ ELI PEARSON		6	SOFT	YELLOW	1214	STATISTIC	2013								
24	24	1	1	5 SF/16/2	2	RAMEZ ELI PEARSON		6	SOFT	YELLOW	1214	STATISTIC	2013								
25	25	1	1	5 SF/16/2	2	RAMEZ ELI PEARSON		6	SOFT	YELLOW	1214	STATISTIC	2013								

Final_text_data.csv - Microsoft Excel

	A	B	C	D	E	F	G	H	I	J	K
58	INTRODUCTION	TO	LINEAR	REGRESSION	ANALYSIS						
59	INTRODUCTION	TO	LINEAR	REGRESSION	ANALYSIS						
60	INTRODUCTION	TO	LINEAR	REGRESSION	ANALYSIS						
61	OPERATIONS	RESEARCH	THEORY	AND	APPLICATIONS						
62	OPERATIONS	RESEARCH	THEORY	AND	APPLICATIONS						
63	OPERATIONS	RESEARCH	THEORY	AND	APPLICATIONS						
64	THE	CERTIFIED	RELIABILITY	ENGINEER	HANDBOOK						
65	INTERNATIONAL	MARKETING	MANAGEMENT								
66	SUPPLY	CHAIN	MANAGEMENT								
67	QUALITY	CONTROL	DEFINE	MEASURE	ANALYZE	IMPROVE	CONTROL				
68	LEAN	PRODUCTION	SIMPLIFIED								
69	HUMAN	RESSOURCE	DEVELOPMENT	THEORY	&	PRACTICE					
70	LEAN	SIX	SIGMA	BUSINESS	TRANSFORMATION	FOR	DUMMIES				
71	THE	CERTIFIED	QUALITY	ENGINEER							
72	APPLIED	BUSINESS	STATISTICS	MAKING	BETTER	BUSINESS	DECISIONS				
73	APPLIED	BUSINESS	STATISTICS	MAKING	BETTER	BUSINESS	DECISIONS				
74	APPLIED	BUSINESS	STATISTICS	MAKING	BETTER	BUSINESS	DECISIONS				
75	APPLIED	BUSINESS	STATISTICS	MAKING	BETTER	BUSINESS	DECISIONS				
76	APPLIED	BUSINESS	STATISTICS	MAKING	BETTER	BUSINESS	DECISIONS				
77	STATISTICAL	QUALITY	CONTROL	A	MODEL	INTRODUCTION					
78	STATISTICAL	QUALITY	CONTROL	A	MODEL	INTRODUCTION					
79	STATISTICAL	QUALITY	CONTROL	A	MODEL	INTRODUCTION					
80	STATISTICAL	QUALITY	CONTROL	A	MODEL	INTRODUCTION					
81	STATISTICAL	QUALITY	CONTROL	A	MODEL	INTRODUCTION					
82	DATA	MINING	AND	DATA	WAREHOUSING						

#####Variables data summary#####

```
docs=read.csv(file.choose(),header=T)
str(docs)
###publication
table(docs$PUBLICATION)
publi.freq=table(docs$PUBLICATION)
publi.freq
publi.freq=publi.freq[order(publi.freq,decreasing = T)]
publi.freq
prop.table(publi.freq)
round(prop.table(publi.freq),2)
barplot(publi.freq)
pie(publi.freq)
#### year
table(docs$YEAR)
year.freq=table(docs$YEAR)
year.freq
year.freq=year.freq[order(year.freq,decreasing = T)]
year.freq
prop.table(year.freq)
round(prop.table(year.freq),2)
barplot(year.freq)
pie(year.freq)
###color
table(docs$COLOUR)
color.freq=table(docs$COLOUR)
color.freq
color.freq=color.freq[order(color.freq,decreasing = T)]
color.freq
prop.table(color.freq)
round(prop.table(color.freq),2)
barplot(color.freq)
pie(color.freq)
#####author
table(docs$X1ST.AUTHOR)
author.freq=table(docs$X1ST.AUTHOR)
author.freq
```

```

author.freq=author.freq[order(author.freq,decreasing = T)]
author.freq
prop.table(author.freq)
round(prop.table(author.freq),2)
barplot(author.freq)
pie(author.freq)
####BOUnd
table(docs$BOUND)
bound.freq=table(docs$BOUND)
bound.freq
bound.freq=bound.freq[order(bound.freq,decreasing = T)]
bound.freq
prop.table(bound.freq)
round(prop.table(bound.freq),2)
barplot(bound.freq)
pie(bound.freq)
####edition
table(docs$EDITION)
edi.freq=table(docs$EDITION)
edi.freq
edi.freq=edi.freq[order(edi.freq,decreasing = T)]
edi.freq
prop.table(edi.freq)
round(prop.table(edi.freq),2)
barplot(edi.freq)
pie(edi.freq)

####type
table(docs$TYPE)
type.freq=table(docs$TYPE)
type.freq
type.freq=type.freq[order(type.freq,decreasing = T)]
type.freq
prop.table(type.freq)
round(prop.table(type.freq),2)
barplot(type.freq)
pie(type.freq)

```



```
#####Codes for cleaning data set in R #####
```

```
getwd() #get current working Directory
```

```
setwd("") #set current working directory to folder
```

```
# Install
```

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

```
install.packages("SnowballC") # for text stemming
```

```
install.packages("wordcloud") # word-cloud generator
```

```
install.packages("ggplot2") # for plotting graphs
```

```
# Load
```

```
library("tm")
```

```
library("SnowballC")
```

```
library("wordcloud")
```

```
library("ggplot2")
```

```
#Read the csv file from local machine
```

```
docs=read.csv(file.choose(),header=T)
```

```
str(docs)
```

```
#build corpous
```

```
#install.packages("tm")

library(tm)

corpus=iconv(docs,to="UTF-8")

corpus=Corpus(VectorSource(corpus))

inspect(corpus[1:10])

corpus=tm_map(corpus,toupper)

inspect(corpus[1:5])

corpus=tm_map(corpus,removePunctuation)

inspect(corpus[1:5])

corpus=tm_map(corpus,removeNumbers)

inspect(corpus[1:5])

cleanset=tm_map(corpus,removeWords,stopwords('ENGLISH'))

inspect(cleanset)

cleanset=tm_map(cleanset,stripWhitespace)
```

```
inspect(cleanset[1:10])
```

```
cleanset=tm_map(cleanset, removeWords,"AND")
```

```
#Term document Matrix
```

```
tdm=TermDocumentMatrix(cleanset)
```

```
tdm
```

```
tdm=as.matrix(tdm)
```

```
View(tdm)
```

```
dtm=DocumentTermMatrix(cleanset)
```

```
(dtm)
```

```
freq=colSums(as.matrix(dtm))
```

```
freq
```

```
ord=order(freq,decreasing = T)
```

```
length(freq)
```

```
freq[head(ord)]
```

```
freq[tail(ord)]
```

```
findFreqTerms(dtm,lowfreq = 5)
```

```
#histogram
```

```
wf=data.frame(term=names(freq),occurrences=freq)
```

```
#install.packages("ggplot2")
```

```
library(ggplot2)
```

```
p=ggplot(subset(wf,freq>10),aes(term,occurrences))
```

```
p=p + geom_bar(stat = "identity")
```

```
p=p + theme(axis.text = element_text(angle = 45,hjust = 1))
```

```
p
```

```
#Wordcloud
```

```
#install.packages("wordcloud")
```

```
library(wordcloud)
```

```
w=sort(rowSums(tdm),decreasing = T)
```

```
set.seed(222)
```

```
wordcloud(words = names(w),
```

```
  freq = w,
```

```
  max.words = 200,
```

```
  random.order = F,
```

```
  min.freq = 5,
```

```
  rot.per = 0.40,
```

```
  colors = brewer.pal(8,'Dark2'),
```

```
  scale = c(2,0.3))
```

```
#wordcloud2 #install.packages("wordcloud2")
```

```
w=data.frame(names(w),w)
```

```
colnames(w)=c('word','freq')
```

```
head(w)
```

```
wordcloud2(w), size=0.5,
```

```
  shape='circle',
```

```
  rotateRatio = 0.5, minSize = 1)
```

Code for data cleaning in python:

```
import
glob

import
nltk

nltk.download('punkt')

def con(a):
    for i in a:
        if
            i==a[
                0]:
                    sen=i
        else:
            sen=sen+'
'+i
    return [sen]

def clean(path):
    path1=['C:\\Users\\DELL\\Documents\\text
files\\'+path][0]
    c=open(path1,encoding="windows-
1252")
    a=c.read()
    b=nltk.word_tokenize(a)
    words = [word for word in b if word.isalpha()]
```

```

f=open(r'C:\Users\DELL\Document\books data set\clean text
files\\'+path,'w',encoding="utf8")
f.write(con(words)[0
])f.close()
s=glob.glob('*.txt')

cd C:\Users\DELL\Documents\text filesfor path in s:
clean(path)

```

Code for IR system:

```

##### importing required modules

#####import pandas as pd

import numpy as np

from sklearn.feature_extraction.text import
TfidfVectorizerfrom sklearn.metrics.pairwise import
cosine_similarity import tkinter as tk

from tkinter import *

from PIL import Image, ImageTk

##### my funtions #####

def match(cs):

    s1=np.sort(cs[cs>0.07])[

    ::-1]t=[]

for i in range(len(s1)):

```

```

for j in range
(cs.shape[1]):if
cs.item(0,j)==s1[i]:
    t.append(
j)return t
#####

def print_output(t):
    output1.delete(0,EN
D)
    output2.delete(0,EN
D)
    output1.insert(END,' *(3)+'Book no.'+' *(25)+'Name of
Book'+ ' *(40-len('Name of Book'))+'\n')
    output2.insert(END,'
'*8+'Author'+'\n')for i in
range(len(t)):
    a=str(data['book'][t[i]])
    b=data['name'][t[i]]
    c=data["Author"][t[i]]
    output1.insert(END,' *(5)+a+' *(15)+b+' *(60-
len(b))+'\n')output2.insert(END,c+'\n')

```



```
#####  
#####
```

```
def search():
```

```
    query1=input1.ge
```

```
    t()
```

```
    query2=input2.ge
```

```
    t()
```

```

if(len(query1)==0 and
    len(query2)==0):
    output1.delete(0,END)
    output2.delete(0,END)
    output1.insert(END, '\t'+ 'Input Error: Please enter
keyword...!!!'+'\n')elif(len(query1)==0 ):
    output1.delete(0,EN
D)
    output2.delete(0,EN
D)
    output1.insert(END, '\t'+ 'Input Error: Enter keyword 1
first..!!!'+'\n')elif(len(query2)==0):
    d=TF.transform([query1])
    cs=cosine_similarity(d[0:1],
TFX)t=match(cs)
    if(len(t)==0):
        output1.delete(0,EN
D)
        output2.delete(0,EN
D)
        output1.insert(END," Sorry no match found...!"
+'\n')else:

```

```
        print_output(t)
else:
    if(var.get()==
        0):#and
        d1=TF.transform([query1])
        d2=TF.transform([query2])
```

```

cs1=np.array(cosine_similarity(d1[0:1],
TFX))

cs2=np.array(cosine_similarity(d2[0:1],
TFX))t1=match(cs1)

t2=match(cs2)

t=list(set(t1) &
set(t2))

if(len(t)==0):

    output1.delete(0,END
D)

    output2.delete(0,END
D)

    output1.insert(END," Sorry no match found...!"
+'\n')else:

    print_output(t)
else:

    #or

    d1=TF.transform([query1
])

    d2=TF.transform([query2
])

    cs1=np.array(cosine_similarity(d1[0:1],

```

```

TFX))
cs2=np.array(cosine_similarity(d2[0:1],
TFX))t1=match(cs1)
t2=match(cs2)
t=list(set(t1) |
set(t2))
if(len(t)==0):
    output1.delete(0,END)
    D)
    output2.delete(0,END)
    D)
    output1.insert(END," Sorry no match
found...!")else:
    print_output(t)

```

```
#####  
##### ###
```

```
def yscroll1(*args):  
  
    if output2.yview() !=  
        output1.yview():  
        output2.yview_moveto(args[0]  
        ) scrollbar.set(*args)
```

```
def yscroll2(*args):  
  
    if output1.yview() !=  
        output1.yview():  
        output1.yview_moveto(args[0]  
        ) scrollbar.set(*args)
```

```
def yview(*args):  
  
    output1.yview(*ar  
    gs)  
    output2.yview(*ar  
    gs)
```

model Building

```
data=pd.read_excel(r"C:\Users\DELL\Documents\books data  
set\clean text files\finaldata.xlsx")
```

```
dfy=data['name']
```

```
dfx=data['conten
```

```
t']
```

```
TF=TfidfVectorizer(min_df=1,stop_words='english',lowercase=True,  
ngram_range=(1,3))
```

```
TFX=TF.fit_transform(dfx)
```

Code for GUI:

```
##### GUI
#####

app1=tk.Tk()

app1.title('Information Retrival System for Departmental Library
Using NLP')HEIGHT = 800

WIDTH = 1100

app1.geometry(str(WIDTH)+'x'+str(HEIGHT)+'+'+str(100)+'+'+str(1
0))app1.resizable(0, 0)

C = tk.Canvas(app1, height=HEIGHT, width=WIDTH)

background_image=
tk.PhotoImage(file=r"C:\Users\DELL\Documents\
wood.png")

background_label = tk.Label(app1,

image=background_image)

background_label.place(x=0, y=0, relwidth=1,

relheight=1) C.pack()

##### title #####

title_frame=tk.Frame(app1, bg='white', bd=3,borderwidth=5)

title_frame.place(relx=0.5, rely=0.005, relwidth=0.90,

relheight=0.15,anchor='n')
```

```
title1=tk.Label(title_frame,text="SARDAR PATEL UNIVERSITY,  
VV NAGAR",bg='white',fg='black',font='TimesNewRoman 22 bold  
italic',justify='center')
```

```
title1.place(relx=0.0,rely=0.21,relheight=0.4, relwidth=1,anchor='w')
```

```
title2=tk.Label(title_frame,text="DEPARTMENT OF  
STATISTICS",bg='white',fg='black',font='TimesNewRoman 18 bold  
italic',justify='center')
```

```
title2.place(relx=0.0,rely=0.53,relheight=0.3, relwidth=0.98,anchor='w')
```

```
title3=tk.Label(title_frame,text="WEL-COME TO  
DEPARTMENTAL  
LIBRARY",bg='white',fg='black',font='TimesNewRoma  
n 14 bold italic',justify='center')
```

```
title3.place(relx=0.0,rely=0.85,relheight=0.4,  
relwidth=0.985,anchor='w')##### Input box
```

```
#####
```

```
frame2=tk.Frame(app1, bd=3,borderwidth=5,bg='black')
```

```
frame2.place(relx=0.75, rely=0.18, relwidth=0.70,  
relheight=0.3,anchor='ne')
```

```
inp_lab1=tk.Label(frame2,text="Input Keyword  
1",bg='white',fg='black',font='TimesNewRoman 10',justify='right')
```

```
inp_lab1.place(relx=0.30,rely=0.16,relwidth=0.25,relheight=0.2,anchor='n  
e')
```

```
inp_lab2=tk.Label(frame2,text="Input Keyword  
2",bg='white',fg='black',font='TimesNewRoman 10',justify='right')
```

```
inp_lab2.place(relx=0.30,rely=0.6,relwidth=0.25,relheight=0.2,ancho  
r='ne') input1=tk.Entry(frame2,font='TimesNewRoman')
```

```
input1.place(relx=0.75,rely=0.16,relwidth=0.4,relheight=0.2,anchor=
'ne') input2=tk.Entry(frame2,font='TimesNewRoman')
input2.place(relx=0.75,rely=0.6,relwidth=0.4,relheight=0.2,anchor='
ne') ##### option box #####
```

```
Out_frame.place(relx=0.5, rely=0.54, relwidth=0.90, relheight=0.4,
anchor='n')
```

```
##### text box #####
```

```
frame3=tk.Frame(app1, bd=3,borderwidth=5,bg='RED')
```

```
frame3.place(relx=0.95, rely=0.18, relwidth=0.25,
```

```
relheight=0.3,anchor='ne')##### ADD RADIO BUTTON
```

```
#####
```

```
var=tk.IntVar()
```

```
rd1=tk.Radiobutton(frame3,text="AND",font='TimesNewRoman
10',variable=var,value=0)
```

```
rd1.place(relx=0.1,rely=0.15,relwidth=0.4,relheight=0.2,anchor='w')
```

```
rd2=tk.Radiobutton(frame3,text="OR",font='TimesNewRoman
10',variable=var,value=1)
```

```
rd2.place(relx=0.95,rely=0.15,relwidth=0.4,relheight=0.2,anchor='e')
```

```
#####adding search and quit button #####
```

```
searchbut=tk.Button(frame3, text='SEARCH',
font='TimesNewRoman 10',command=search)
```

```
searchbut.place(relx=0.5,rely=0.37,relwidth=0.6,relheight=0.2,anchor='n')
```

```
quitbut=tk.Button(frame3, text='EXIT', font='TimesNewRoman
10',command=app1.destroy)
```

```
quitbut.place(relx=0.5,rely=0.7,relwidth=0.4,relheight=0.2,anchor='n')
```

```
##### output window
```

```
#####
```

```
outlab=tk.Label(app1,bd=3,text="OUTPUT",font='TimesNewRoman  
10 bold')outlab.place(relx=0.05, rely=0.51,anchor='w')
```

```
Out_frame = tk.Frame(app1, bg='black', bd=10)
```

```

output1=tk.Listbox(Out_frame,bg='white',fg='black',font='TimesNew
Roman 14',yscrollcommand=yscroll1,relief=FLAT,borderwidth=0,
highlightthickness=0)

output1.place(relx=0.0, rely=0.0, relwidth=0.6, relheight=1, anchor='nw')

output2=tk.Listbox(Out_frame,bg='white',fg='black',font='TimesNewRom
an 14',yscrollcommand=yscroll2,relief=FLAT,borderwidth=0,
highlightthickness=0)

output2.place(relx=0.6, rely=0.0, relwidth=0.5, relheight=1,
anchor='nw')#####

scrollbar =
tk.Scrollbar(Out_frame,command=yview)

scrollbar.pack( side = RIGHT, fill = Y )

#output['yscrollcommand'] = scrollbar.set

#####

#mylist.pack( side = LEFT, fill = BOTH )

#scrollbar.config( command = mylist.yview )


credits=tk.Label(app1,text="created by Rohit
yadav",font='TimesNewRoman 8 italic')

credits.place(relx=0.05, rely=0.97,anchor='w')

app1.mainloop()

```