Fall 2017

KNOWLEDGE DISCOVERY IN DATABASES

FINAL PROJECT REPORT

Group 9:

Alekhya Akkinepally
Venkata Sai Vamsi Krishna Gollapudi
Praneeth Bomma
Prapul Kumar Dasari
Sriram Palaniappan Kirubakaran

1. DATA CLEANING:

The days when one would get data in tabulated spreadsheets are truly behind us. A moment of silence for the data residing in the spreadsheet pockets. Today, more than 80% of the data is unstructured – it is either present in data silos or scattered around the digital archives. Data is being produced as we speak – from every conversation we make in the social media to every content generated from news sources. In order to produce any meaningful actionable insight from data, it is important to know how to work with it in its unstructured form.

One of the first steps in working with text data is to pre-process it. It is an essential step before the data is ready for analysis. Majority of available text data is highly unstructured and noisy in nature – to achieve better insights or to build better algorithms, it is necessary to play with clean data. For example, social media data is highly unstructured – it is an informal communication – typos, bad grammar, presence of unwanted content like URLs, stop word's, Expressions etc. are the usual suspects.

So we have performed basic data preprocessing operations on the given text corpus we have done five basic pre-processing steps which involve stripping the white spaces, converting the text into lower cases, as we know that we are operating on around ten thousand files there will be many stop words like the, that, they etc. which really doesn't help us in achieving our task of finding the surprise element so we will be removing the stop words from our text files. Then we have removed the punctuations in the whole corpus and we will be removing the sparse elements. After carrying these five data pre-processing steps our text data is cleaned and we will make further technical operations on this cleaned data. For making sure that the data is cleaned let's compare the dtm's version of both the corpus before and after cleaning of the data as we can see that the total number of terms are reduced (because of removal of stop words) we can say that the data pre-processing step was done successfully

2. MAIN TASK Problem Statement:

To find out the surprising elements from the given diabetes text corpus.

Languages used: R Language, Python.

Operations performed: Clustering Analysis, PAM, Cosine Similarity, and Word Cloud.

Libraries Used: tm, cluster, factoextra, magrittr, skmeans, topicmodels, tidytext, wordcloud.

Definition of Surprise:

For our entire analysis we are interested in finding the personalized surprise rather than a generalized surprise. A personalized surprise is nothing but a element which we found may be surprising to us and may not be surprising to others, it mainly depends on the background knowledge of the individual related to diabetes.

We have defined surprise elements as the elements which are present in the whole corpus for a minimum number of times and are not related to the diabetes like medicines, diet, treatment etc. or any health related words like cancer, cardiovascular etc.

Libraries Used:

Tm: This library includes the framework for carrying various text mining applications within R, this library includes various functions for cleaning the data and carrying various other operation on text.

Cluster: This library includes various functions for carrying various clustering that is finding groups in the data operation on the corpus like k-means, sk-means etc.

Factoextra: This library contains functions for extracting and visualizing the results of multivariate

Data analyses like PCA (Principle Component Analysis).It contains also functions for simplifying some clustering analysis steps and provides 'ggplot2' - based elegant data visualization.

Magrittr: *The magrittr* is a package with two aims: to decrease development time and to improve readability and maintainability of code. Or even shortr: to make your code short which includes new “pipe”-like operator, %>% with which you may pipe a value forward into an expression or function call.

Skmeans: This is used for performing spherical k-means clustering and also includes several features several methods, including a genetic and a fixed-point algorithm and an interface to the CLUTO vcluster program.

Word Cloud: This is used for building a word cloud with the words in a text document with the frequency determining the size of the word.

Slam: This contains data structures and algorithms for sparse arrays and matrices, based on index arrays and simple triplet representations, respectively.

Topicmodels: Provides an interface for performing Latent Dirichlet Allocation (LDA) models and Correlated Topics Models (CTM).

TidyText: This package is used for performing text mining for word processing and sentiment analysis using 'dplyr', 'ggplot2', and other tidy tools.

3. METHODS USED FOR FINDING SURPRISE ELEMENT:

A: Document Clustering:

Document clustering is the process of grouping or partitioning text documents into meaningful groups. The hypothesis of the clustering algorithm is based on minimizing the distance between objects in a cluster, while keeping the intra-cluster distance at maximum.

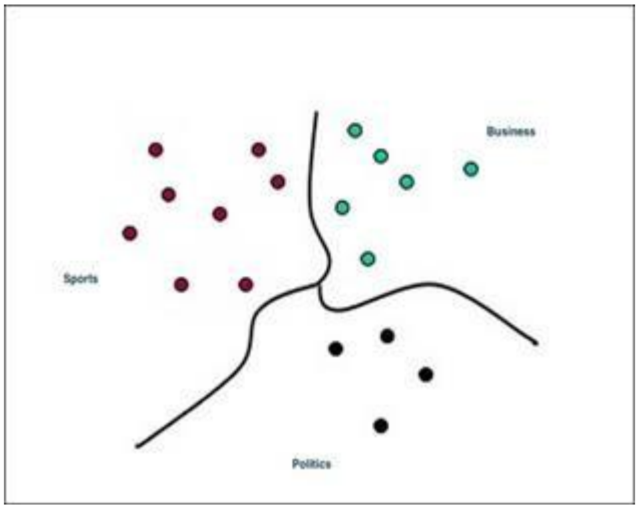


Figure: Example for clustering a group of documents

For example, if we have a collection of news articles and we perform clustering on the collection, we will find that the similar documents are closer to each other and lie in the same cluster.

In the following sections we will be performing various kinds of clustering on the corpus for getting the cluster which is at a wide distance.

K-Means Clustering:

K-means clustering is one of many techniques within unsupervised learning that can be used for text analysis. *Unsupervised* refers to the fact that we're trying to understand the structure of our underlying data, rather than trying to optimize for a specific, pre-labeled criterion (such as creating a predictive model for conversion). Unsupervised learning is a great technique for exploratory analysis in that the analyst enforces few assumptions on the data, so previously unexamined relationships can be determined *then* analyzed; contrast that with pre-defined relationships specified by the analyst (such as *visitors from mobile* or *visitors from social*), then evaluating how various metrics differ across these pre-defined groups.

Without getting too technical, k-means clustering is a method of partitioning data into 'k' subsets, where each data element is assigned to the closest cluster based on the distance of the data element from the center of the cluster. In order to use k-means

clustering with text data, we need to do some text-to-numeric transformation of our text data. Luckily, R provides several packages to simplify the process.
For converting Text to Numeric Data that is into Document Term Matrix, we will be using the simple "tm" library and use the function document term matrix for converting the corpus into DTM. After converting the text to numerical data our major task of clustering comes into play as we are performing k-means clustering as we have to give the k value we don't know the optimum value of k but for getting the optimum value we will be using an elbow methods for getting the optimum k value.
After finding the k- value we will be performing the clustering operations and convert those clusters into a dendrogram. A dendrogram decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram. A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster and the dendrogram is as shown in the diagram below.

INTERPRETATION:

Now from the above figure by using the scale on the left side we can know which clusters are at a large distance from the remaining other clusters. Now we will be grouping those clusters into required number and draw a rectangle on the graph which is as shown.

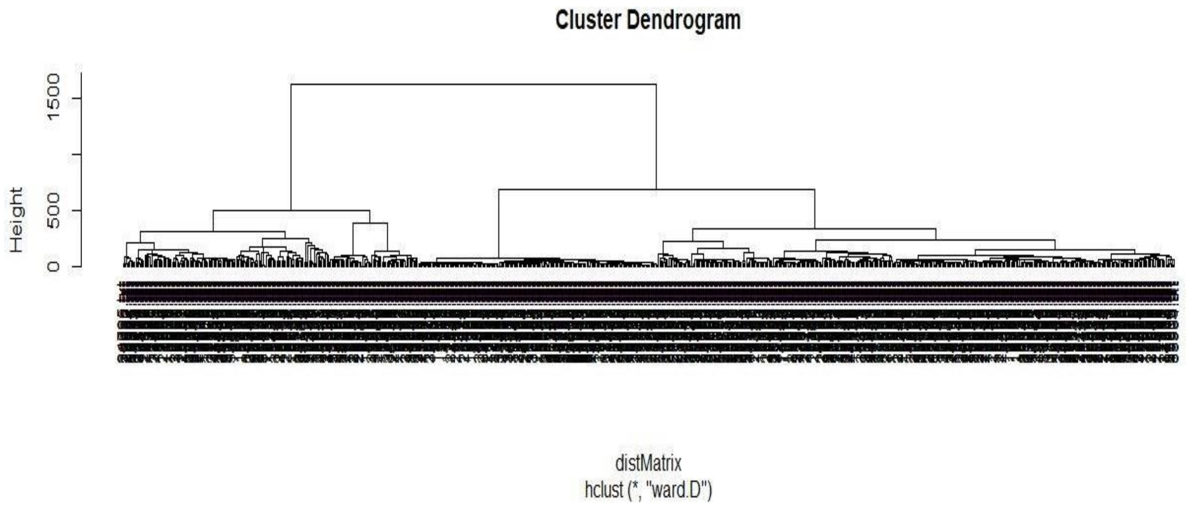


Figure: Hierarchical clustering and dendrogram plotting

Now we can say that the group of documents with larger distance from the remaining other are the documents with some different content than the remaining other documents.

Next Step:

What we actually taught was to consider the group which are away from others and run a topic modeling on that text corpus and extract three topics from the whole corpus of documents and find out the top ten words of each topic and wanted to implement a machine learning algorithm for determining the surprise element. But as we are having an text corpus of around ten thousand documents the dendrogram which we got is very clumsy and we could not know the exact labels from the diagram, we tried various other libraries like dendextent and tried to get the clear labels of the documents but we could not do that so we stopped the further proceeding with this method.

B: The K-Medoid Clustering Method:

Next we haven't deviated much from our main idea of clustering we have used the same clustering technique but to overcome the weakness in k-means clustering like k-means applicability only to objects in a continuous n-dimensional space (k-modes method for categorical data), Need to specify *k*, the *number* of clusters, in advance, Sensitive to noisy data and *outliers* (k-medoids can be applied), Not suitable to discover clusters with *non-convex shapes. We have used the k-modes method* for replacing means of clusters with modes, using categorical dissimilarity measures to do the object assignment using a frequency-based method to update modes of clusters.A mixture of categorical and numerical data: *k-prototype* method.
As we know that the k-means algorithm is sensitive to outliers because an object with an extremely large value may substantially distort the distribution of the data, so to overcome this we will be using k-mediod where this K-Medoids instead of taking the mean value of the object in a cluster as a reference point, medoids can be used, which is the most centrally located object in a cluster and this k-mediod is generally used for the carrying processing on words.
The beginning step in this clustering is to take an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering and this process is iterated till the same mediods occur between two consecutive results for this kind of process for finding the medidods a technique called *PAM* (Partition Around Mediods) works effectively and especially for small data sets, but does not scale well for large data sets (due to the computational complexity).This algorithm mainly has two phases one is BUILD and the other is SWAP.
After following this method for clustering we have plotted the clusters we have given the clusters number depending upon the elbow method and are as shown below:

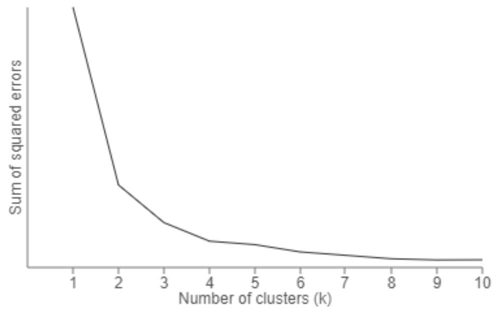


Figure: Graph obtained when using the elbow method

INTERPRETATION:

From the diagram below we can say that we have got the required number of clusters and by using the scales on the x-axis and y-axis we can know which cluster is at a different position when compared to others.

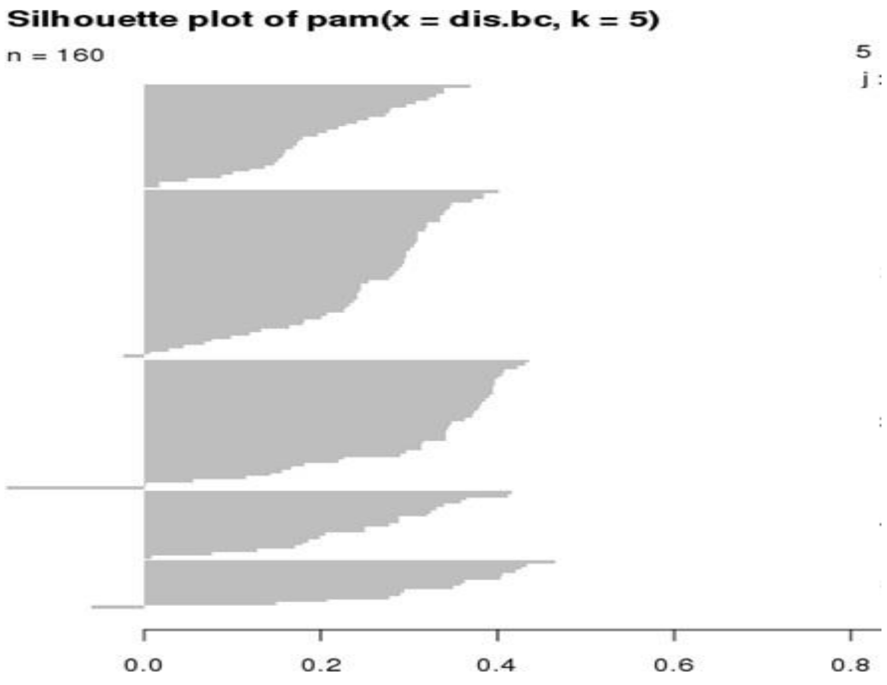


Figure: Clustering using PAM technique

NEXT STEP:

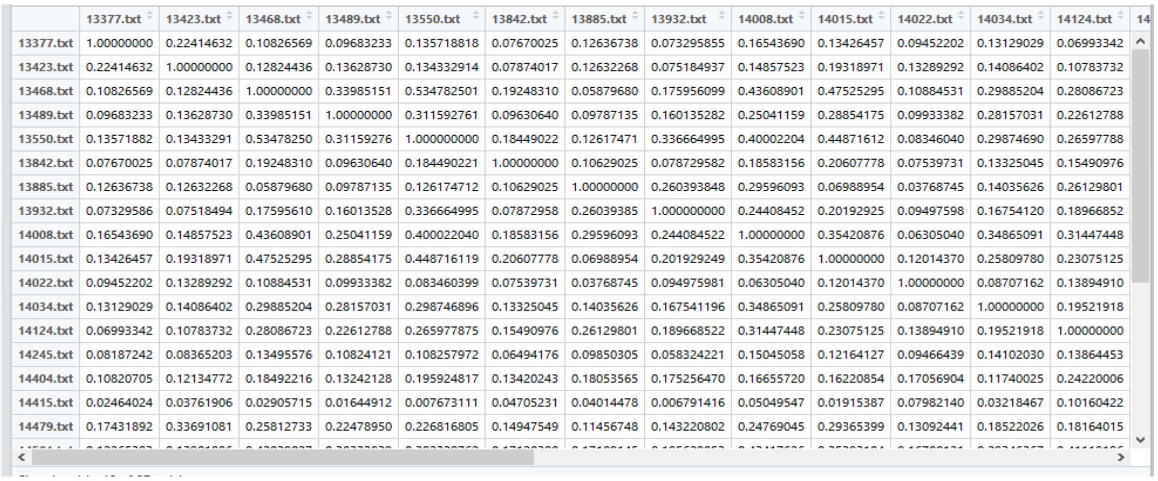
After getting the documents from the cluster we thought of dividing the the whole new corpus into three topics and find the top ten words from each of the topic and move on to machine learning technique for finding the surprise element. But we have struck with the same problem as that of the previous hierarchical clustering that is we could not get the labels as required by us and this is mainly because if large number of documents.

C: COSINE SIMILARITY:

As we are dealing with a surprise element we can find it by finding the similarity between each document to the other and put a threshold and observe the ones which doesn't pass the threshold contains a surprise element or a terms which are not related to diabetes. **Cosine similarity** is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them if the value is near to zero they are not similar and when the value is near to one then they are similar documents. In information retrieval and text mining, each term is notionally assigned a different dimension and a document is characterized by a vector where the value of each dimension corresponds to the number of times that term appears in the document. Cosine similarity then gives a useful measure of how similar two documents are likely to be in terms of their subject matter.

Working with strings:

The first step is to turn our documents into vectors that we can work with. We do this by considering the Term Frequency of unique words in a given string. So for performing this we will be first reading whole corpus and perform preprocessing techniques and then convert the whole into document term matrix and then into term frequency inverse document matrix using the required functions and packages in r. After performing this operation we will be calculating the cosine similarity between each and every documents in the whole corpus and converts the result into a matrix the result will be as shown below.



	13377.txt	13423.txt	13468.txt	13489.txt	13550.txt	13842.txt	13885.txt	13932.txt	14008.txt	14015.txt	14022.txt	14034.txt	14124.txt
13377.txt	1.00000000	0.22414632	0.10826569	0.09683233	0.135718818	0.07670025	0.12636738	0.073295855	0.16543690	0.13426457	0.09452202	0.13129029	0.06993342
13423.txt	0.22414632	1.00000000	0.12824436	0.13628730	0.134332914	0.07874017	0.12632268	0.075184937	0.14857523	0.19318971	0.13289292	0.14086402	0.10783732
13468.txt	0.10826569	0.12824436	1.00000000	0.33985151	0.534782501	0.19248310	0.05879680	0.175956099	0.43608901	0.47525295	0.10884531	0.29885204	0.28086723
13489.txt	0.09683233	0.13628730	0.33985151	1.00000000	0.311592761	0.09630640	0.09787135	0.160135282	0.25041159	0.28854175	0.09933382	0.28157031	0.22612788
13550.txt	0.13571882	0.13433291	0.53478250	0.31159276	1.000000000	0.18449022	0.12617471	0.336664995	0.40002204	0.44871612	0.08346040	0.29874690	0.26597788
13842.txt	0.07670025	0.07874017	0.19248310	0.09630640	0.184490221	1.00000000	0.10629025	0.078729582	0.18583156	0.20607778	0.07539731	0.13325045	0.15490976
13885.txt	0.12636738	0.12632268	0.05879680	0.09787135	0.126174712	0.10629025	1.00000000	0.260393848	0.29596093	0.06988954	0.03768745	0.14035626	0.26129801
13932.txt	0.07329586	0.07518494	0.17595610	0.16013528	0.336664995	0.07872958	0.26039385	1.000000000	0.24408452	0.20192925	0.09497598	0.16754120	0.18966852
14008.txt	0.16543690	0.14857523	0.43608901	0.25041159	0.400022040	0.18583156	0.29596093	0.244084522	1.00000000	0.35420876	0.06305040	0.34865091	0.31447448
14015.txt	0.13426457	0.19318971	0.47525295	0.28854175	0.448716119	0.20607778	0.06988954	0.201929249	0.35420876	1.00000000	0.12014370	0.25809780	0.23075125
14022.txt	0.09452202	0.13289292	0.10884531	0.09933382	0.083460399	0.07539731	0.03768745	0.094975981	0.06305040	0.12014370	1.00000000	0.08707162	0.13894910
14034.txt	0.13129029	0.14086402	0.29885204	0.28157031	0.298746896	0.13325045	0.14035626	0.167541196	0.34865091	0.25809780	0.08707162	1.00000000	0.19521918
14124.txt	0.06993342	0.10783732	0.28086723	0.22612788	0.265977875	0.15490976	0.26129801	0.189668522	0.31447448	0.23075125	0.13894910	0.19521918	1.00000000
14245.txt	0.08187242	0.08365203	0.13495576	0.10824121	0.108257972	0.06494176	0.09850305	0.058324221	0.15045058	0.12164127	0.09466439	0.14102030	0.13864453
14404.txt	0.10820705	0.12134772	0.18492216	0.13242128	0.195924817	0.13420243	0.18053565	0.175256470	0.16655720	0.16220854	0.17056904	0.11740025	0.24220006
14415.txt	0.02464024	0.03761906	0.02905715	0.01644912	0.007673111	0.04705231	0.04014478	0.006791416	0.05049547	0.01915387	0.07982140	0.03218467	0.10160422
14479.txt	0.17431892	0.33691081	0.25812733	0.22478950	0.226816805	0.14947549	0.11456748	0.143220802	0.24769045	0.29365399	0.13092441	0.18522026	0.18164015

Figure: Cosine similarity matrix

INTERPRETATION:

As we calculated the cosine similarity if the value is neat to zero we can say that the particular document is containing a surprise element. But, as we are dealing with around ten thousand documents our matrix will be of the size 10000X10000 so observing the one which doesn’t cross the threshold is bit difficult so we will carry further calculations and get the only documents which don’t cross the threshold from the whole corpus.

DRAWBACKS:

We have reached our goal and found out the documents which didn’t cross the threshold level and read those documents and found few things which are surprising to us. But we got a doubt here how we can fix the threshold. We have randomly detected a threshold but we can’t say whether it is optimum or not, and also in finding cosine similarity we are finding similarity between each document, which really doesn’t answer our question. But if we are having a reference document we can find out the similarity of all the others with this document and set a threshold value and then we will get the real surprising elements because we are having a reference document but as we are not having any reference document we could not say that the documents we found are having the surprise element. So we have moved to a next process which is also in the lines of clustering as the main task.

D: SK-MEANS CLUSTERING:

In classic k-means, we seek to minimize a Euclidean distance between the cluster center and the members of the cluster. The intuition behind this is that the radial distance from the cluster-center to the element location should "have sameness" or "be similar" for all elements of that cluster. In spherical k-means, the idea is to set the center of the cluster such that it makes both uniform and minimal the angle between components. The intuition is like looking at stars - the points should have consistent spacing between each other. That spacing is simpler to quantify as "cosine similarity", but it means there are no "milky-way" galaxies forming large bright swathes across the sky of the data. Think about vectors, the things you graph as arrows with orientation, and fixed length. It can be translated anywhere and be the same vector.

The orientation of the point in the space (its angle from a reference line) can be computed using linear algebra, particularly the dot product. If we move all the data so that their tail is at the same point, we can compare "vectors" by their angle, and group similar ones into a single cluster. You could think of it as a constellation. The stars in a single cluster are close to each other in some sense. Like a real constellation we are not accounting for radial distance. This is accomplished by adjusting them so their lengths are equal by dividing by the norm.

The value of this approach is that it allows us to contrive vectors which otherwise have no length, such as in the tf-idf method, where the vectors are word frequencies in documents. Two "and" words added does not equal a "the". Words are non-continuous and non-numeric. Spherical k-means can be used to cluster based on words.

STEPS INVOLVED:

After performing the data preprocessing operation on the whole corpus and we will be convert the whole corpus into document term matrix, after converting into DTM we will convert it into term frequent inverse document matrix we will be doing this because this helps in giving the value to a word depending upon the count of the word, and the number of document in which it is present. After converting into tf-idm we will be performing the sk-means clustering with the number of clusters determined by the result of the elbow method which we mentioned in the earlier section.

USER DEFINED FUNCTION:

Here are trying to improve the clusters and to understand better what features/words make a particular document fall into a particular cluster, I would like to know what the most distinguishing features for each cluster are so what I mean by this is if we are able to find the most distinguishing elements from each cluster we can easily know the surprising element from the corpus.

So we have written a function which takes three input from the user which are the sk-means clustering output if the given input cluster is not a s-means cluster the function shows an message to provide sk-means cluster as the input, and the second input is the number of words the user wants from the cluster we will be taking this value as ten. And the third and the most important input to be provided is the UNIQUE value this variable can take two values either true or false if the value is true the function return the distinguishing words and when this value is false it returns the most common terms from each cluster.

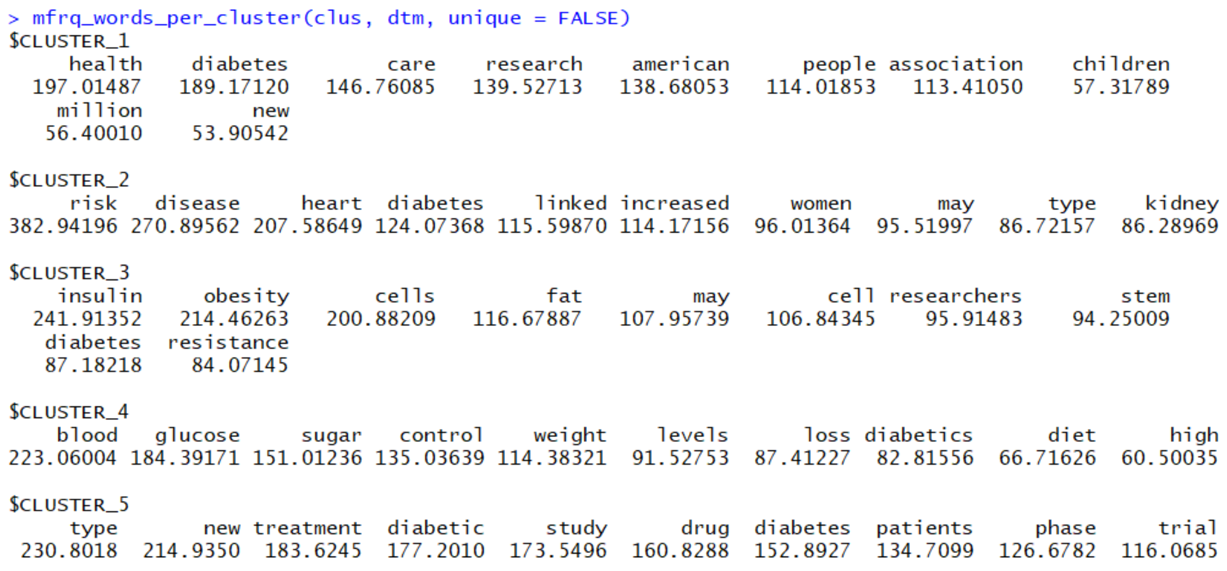
This UNIQUE takes the default value as true because we are mainly concentrated in finding the distinguishing terms. We have actually incorporated this function because as the part of the project we need to find the most likely and the divergent terms so if we have it true we will get the most divergent terms and if false we will be getting the most likely terms the most likely terms are nothing but the terms which are most related to diabetes like insulin, sugar, health etc.

WHAT THE FUNCTION DO:

Once the inputs are provided there are two more functions in the designed function one is for calculating the words count and getting the frequency of the each word count depending on which the importance of the word is known and there is another function which helps in printing out the words from the each cluster. If we write the true for unique there is a if condition which eliminates the words occurring several times.

APPLYING THE INPUTS TO THE DESIGNED FUNCTION:

So we are first interested in finding the most likely words so we are keeping the UNIQUE value false and giving the clusters obtained in the above step mentioned and giving k value as 10 the output is as shown below:



```
> mfrq_words_per_cluster(clus, dtm, unique = FALSE)
$CLUSTER_1
     health    diabetes        care    research    american      people association    children 
  197.01487   189.17120   146.76085   139.52713   138.68053   114.01853   113.41050    57.31789 
    million         new 
   56.40010    53.90542 

$CLUSTER_2
     risk   disease     heart  diabetes    linked increased     women       may      type    kidney 
382.94196 270.89562 207.58649 124.07368 115.59870 114.17156  96.01364  95.51997  86.72157  86.28969 

$CLUSTER_3
    insulin     obesity       cells         fat         may        cell researchers        stem 
  241.91352   214.46263   200.88209   116.67887   107.95739   106.84345    95.91483    94.25009 
   diabetes  resistance 
   87.18218    84.07145 

$CLUSTER_4
    blood   glucose     sugar   control    weight    levels      loss diabetics      diet      high 
223.06004 184.39171 151.01236 135.03639 114.38321  91.52753  87.41227  82.81556  66.71626  60.50035 

$CLUSTER_5
     type       new treatment  diabetic     study      drug  diabetes  patients     phase     trial 
 230.8018  214.9350  183.6245  177.2010  173.5496  160.8288  152.8927  134.7099  126.6782  116.0685 
```

Figure: Most likely words when UNIQUE=FALSE in the user defined function

INTERPRETATION:

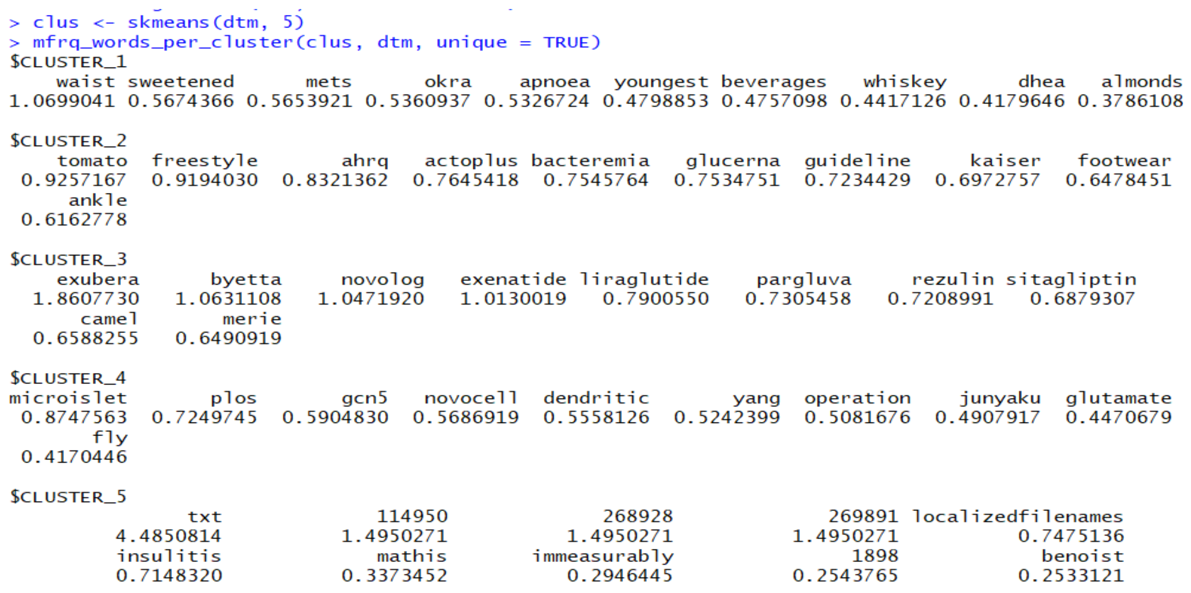
From the above results we have got the top ten words of each cluster and from the words we can say that the words (health, insulin, treatment) obtained are the words known to almost every individual having basic education background.

Most Likely Words:

So the most likely words are health, diabetes, care, research, risk, disease, heart, kidney, insulin, obesity, fat, stem, blood, glucose, sugar, treatment, drug, diet, resistance, patients, study etc.

SURPRISE ELEMENT:

So now we have moved to the actual task that is finding the surprise element so we are keeping unique value as true and giving the clusters obtained in the above step and giving the k value as 10 the output is`



```
> clus <- skmeans(dtm, 5)
> mfrq_words_per_cluster(clus, dtm, unique = TRUE)
$CLUSTER_1
    waist sweetened      mets      okra    apnoea  youngest beverages   whiskey      dhea   almonds 
1.0699041 0.5674366 0.5653921 0.5360937 0.5326724 0.4798853 0.4757098 0.4417126 0.4179646 0.3786108 

$CLUSTER_2
    tomato  freestyle       ahrq   actoplus bacteremia   glucerna  guideline     kaiser   footwear 
 0.9257167  0.9194030  0.8321362  0.7645418  0.7545764  0.7534751  0.7234429  0.6972757  0.6478451 
     ankle 
 0.6162778 

$CLUSTER_3
    exubera      byetta     novolog   exenatide liraglutide    pargluva     rezulin sitagliptin 
  1.8607730   1.0631108   1.0471920   1.0130019   0.7900550   0.7305458   0.7208991   0.6879307 
      camel       merie 
  0.6588255   0.6490919 

$CLUSTER_4
microislet       plos       gcn5   novocell  dendritic       yang  operation    junyaku  glutamate 
 0.8747563  0.7249745  0.5904830  0.5686919  0.5558126  0.5242399  0.5081676  0.4907917  0.4470679 
       fly 
 0.4170446 

$CLUSTER_5
               txt             114950             268928             269891 localizedfilenames 
         4.4850814          1.4950271          1.4950271          1.4950271          0.7475136 
         insulitis             mathis       immeasurably               1898            benoist 
         0.7148320          0.3373452          0.2946445          0.2543765          0.2533121 
```

Figure: Most divergent words when UNIQUE=TRUE in the user defined function

INTERPRETATION:

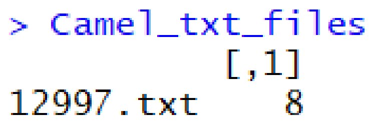
From the above results we have got the distinguishing words of each cluster and from the words we can say that the words like camel (in cluster 3), okra (in cluster 1) etc. are very surprising to us and we have asked some of our friends and they are also surprised with these two words because as we are technical students we don't have any knowledge of how these words are related to diabetes. Though there are other words obtained in the list we have taken these two words only because we don't know any of these words as they are new to us, and those people whom we have asked also told the same thing so we have considered these two words as our main surprise elements.

Most Divergent Words:

So the most likely words are health, diabetes, care, research, risk, disease, heart, kidney, insulin, obesity, fat, stem, blood, glucose, sugar, treatment, drug, diet, resistance, patients, study etc.

NEXT STEP:

Step 1: As we have got the surprise element now we are interested in finding how the terms are related to diabetes. As there are around ten thousand odd document we cannot read all the document so in order to overcome this we have made a few matrix manipulations so that we will get the document's in which these words are present.



```
> Camel_txt_files
          [,1]
12997.txt    8
```

Figure: Document containing the word camel and its count

Step 2: So after getting those text documents we thought off getting the rough details of the content present in the document rather than reading the whole document so we have created a new text corpus which reads these text documents containing these words using the PATTERN function

Step 3: After getting the new text corpus then we will be running the basic data preprocessing techniques for cleaning the data

Step 4: After cleaning the data the data we will be running a word cloud on the text document and this gives the word cloud for all the words in the text documents, but we will be getting all the words so in order to have a better view we will be increasing the frequency so that we will be getting the words which crosses those frequency and the word cloud is as shown:

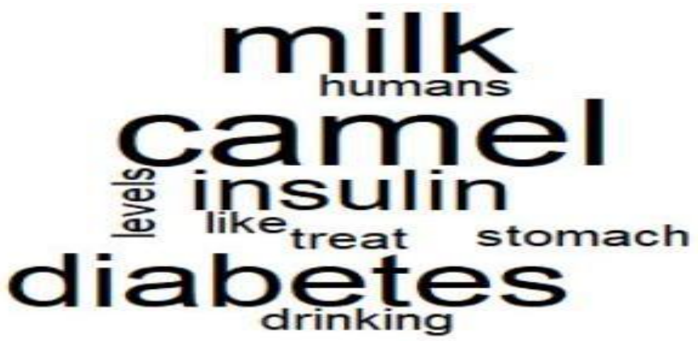
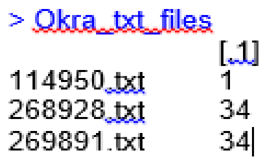


Figure: Word Cloud
for the text documents containing camel word

So from the above word cloud we can say that the following terms (milk, camel, treat, stomach, etc.) are mainly present in the documents so we can relate all together and get the conclusion as there is some relation as **drinking camel milk acts as insulin and treats diabetes.**

Step 5: Now we have read the whole document containing these surprise terms and we found the statement that camel milk reduces the diabetes in humans which is a surprise because we don't know that camel milk reduces diabetes.

We have carried the same operation for the word okra and found that okra was present in three text documents then we repeated the steps 1 to steps 4 and formed the following word cloud by increasing the frequency in the function so can get to the conclusion that there is some relation between okra reduces health risks, okra intake reduces diabetes as we are having many terms we cannot get the exact information from the data so we have increased the frequency so we have got fewer words and got the idea that okra eating reduces diabetes and this may be tested on rats.



```
> Okra_txt_files
              [,1]
114950.txt    1
268928.txt    34
269891.txt    34
```

Figure: Document containing the word okra and its count

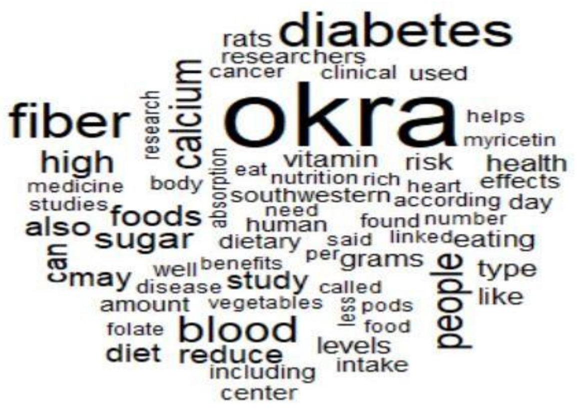


Figure: Word Cloud for the text documents containing okra word with less frequency

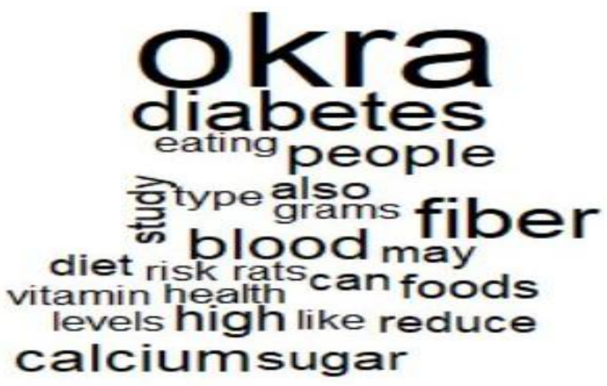


Figure: Word Cloud for the text documents containing okra word with more frequency

Now we have read the three documents and came to conclusion that the following okra reduces diabetes and we have got the same conclusion of what we have got from the above word cloud.

E: TITLE EXTRACTION:

Next we have tried a new technique and wanted to make analysis on the titles of all the ten thousand documents. We have extracted the titles using python and created the new text files with the same name. And we have done topic modeling but we haven't moved any further because of time factor. We can apply the same methods mentioned above to this extracted data and get the surprise element.

4. CONCLUSION:

As our main aim is finding out the surprise element we have made around four approaches for getting this done but of all the implementations we found that the fourth method is better when compared to others methods in getting the surprise element. We can further take this method by applying the machine learning techniques for getting the surprise element from the newly added document.

APPENDIX:

CODE:

```
install.packages("tm")

install.packages("magrittr")

install.packages("factoextra")

install.packages("skmeans")

install.packages("wordcloud")

library(tm)

library(cluster)

library(factoextra)

library(magrittr)

library(skmeans)

library(wordcloud)


require("slam")


setwd("C:/Users/Prapul Kumar/Desktop/KDD/Rdata")


text_corpus<-Corpus(DirSource("diabetes"))

text_corpus <- tm_map(text_corpus, stripWhitespace)

text_corpus <- tm_map(text_corpus, content_transformer(tolower))

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


text_corpus1<-Corpus(DirSource("test"))
```

```
text_corpus1 <- tm_map(text_corpus1, stripWhitespace)

text_corpus1 <- tm_map(text_corpus1, content_transformer(tolower))

text_corpus <- tm_map(text_corpus1, removeWords, stopwords("english"))

#text_corpus <- tm_map(text_corpus, removePunctuation)


dtm <- DocumentTermMatrix(text_corpus)




summary(text_corpus)

inspect(dtm)


dtm <- weightTfIdf(dtm, normalize = TRUE)

inspect(dtm)


#

mfrq_words_per_cluster  <-  function(clus,  dtm,  first  =  10,  unique  =
TRUE){

  if(!any(class(clus) == "skmeans")) return("clus must be an skmeans
object")


  dtm <- as.simple_triplet_matrix(dtm)

  indM  <-  table(names(clus$cluster),  clus$cluster)  ==  1  #  generate
bool matrix


  hfun <- function(ind, dtm){ # help function, summing up words
```

```
    if(is.null(dtm[ind, ]))  dtm[ind, ] else  col_sums(dtm[ind, ])
  }
  frqM <- apply(indM, 2, hfun, dtm = dtm)

  if(unique){
    # eliminate word which occur in several clusters
    frqM <- frqM[rowSums(frqM > 0) == 1, ]
  }
  # export to list, order and take first x elements
  res <- lapply(1:ncol(frqM), function(i, mat, first)
    head(sort(mat[, i], decreasing = TRUE), first),
    mat = frqM, first = first)

  names(res) <- paste0("CLUSTER_", 1:ncol(frqM))
  return(res)
}
#we have to delete a empty file to run this (data preprocessing)
clus <- skmeans(dtm, 5)
mfrq_words_per_cluster(clus, dtm)
mfrq_words_per_cluster(clus, dtm, unique = FALSE)
#
m3<-as.matrix(dtm)
df3<-as.data.frame(m3)
#
```

```
m  <- as.matrix(dtm)

dataframe <-as.data.frame(m)

#m <- m[1:2, 1:3]

distMatrix <- dist(dataframe, method="euclidean")

flatclust <- pam(distMatrix,k=2,metric = "manhattan",medoids = NULL)

plot(flatclust, cex=0.9, hang=-1)

#flatclust1<- as.matrix(flatclust)

class(flatclust)

flatclust

#

install.packages("dendextend")

library(dendextend)

dendoclust <- hclust(distMatrix,method="ward.D")

dd <- as.dendrogram(dendoclust)

labels(dd)

label.dendrogram(dd)

plot(dendoclust, cex=0.9, hang=-1)

rect.hclust(dendoclust,k=25)

install.packages("ggplot2")

library(ggplot2)

m<-as.matrix(dtm)

gc()

#wordcloud of camel document

text_corpus<-Corpus(DirSource("test1"))
```

```
text_corpus <- tm_map(text_corpus, stripWhitespace)

text_corpus <- tm_map(text_corpus, content_transformer(tolower))

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

library(wordcloud)

wordcloud(text_corpus,min.freq = 1.5)

m<-as.matrix(dtm)

memory.limit()

dtm

m5<-m[,"camel"]

m5<-as.matrix(m5)

m6<-m[,"okra"]

m6<-as.matrix(m6)

# Cosine similarity

dtm <- DocumentTermMatrix(text_corpus)

m<-t(m)

ma<-cosine(m)

ma<-as.matrix(ma)
```

Title Extraction Using Python.

```
# -*- coding: utf-8 -*-

"""

Created on Mon Dec  4 08:34:10 2017


@author: prapul

"""
```

```
import os

path = os.getcwd()

path = path + "\\test"

arr = []

for file in next(os.walk(path))[2]:

    arr.append(path+"\\"+file)


for file in arr:

    print(file)

    file1 = open(file,"r", encoding="utf8")

    file2Name = file.replace("test","output")

    file2 = open(file2Name,"w", encoding="utf8")

    for line in file1:

        file2.write(line)

        file2.close()

        break
```