



A PROJECT REPORT ON **QUORA QUESTION PAIRS**

[SEMANTIC ANALYSIS OF QUESTION PAIRS]

Date: 13 May 2017

PRESENTED BY : HEMANT KUMAR SAIN
SUBMITTED TO: EDWISOR.COM

ACKNOWLEDGEMENT

Project development is not an easy task. It requires corporation and help of various people. It always happens that word run out when we are really thankful and sincerely want to inspire my feelings of gratitude towards the one when helped in the completion of the project.

I would like to give my sincere thank to whole Edwisor team, for Giving me an opportunity to learn under them and guided me throughout the project gained comprehensive knowledge of various aspects of how to approach an analytical problem and analyze it. Without their knowledge i would have never been able to complete my project.

I would also like to convey my sincere thank to Mr. Muquayyar Ahmed, Ayush Choudhary and Papori Goswami for their valuable comments and suggestion that has helped in completing my course and Project as well.

A special thanks to all my fellows for helping me out on discussion board with all my queries from the first day of my learning to throughout the project.

I also obliged to whole Edwisor team and all members for their valuable support throughout.

.

Hemant Kumar Sain

CONTENT

Chapter 1: Introduction of Text Mining

- 1.1 What is Text Mining
- 1.2 How does Data Analytics Works
- 1.3 Technology Required

Chapter 2: Analysis of Quora Question Pairs

- 2.1 Problem Statement

Chapter 3: Data Preparation

- 3.1 Data Collection
- 3.2 Data Cleaning
- 3.3 Exploratory Data Analysis

Chapter 4: Model Building

- 4.1 Brief of Random Forest
- 4.2 Confusion Matrix
- 4.3 Variable Importance

Conclusion

INTRODUCTION OF TEXT MINING

1.1 Text Mining

Text mining can help an organization derive potentially valuable business insights from text-based content such as word documents, email and postings on social media streams like Facebook, Twitter and LinkedIn. Mining unstructured data with natural language processing (NLP), statistical modeling and machine learning techniques can be challenging, however, because natural language text is often inconsistent. It contains ambiguities caused by inconsistent syntax and semantics, including slang, language specific to vertical industries and age groups, double entendres and sarcasm.

Text analytics software can help by transposing words and phrases in unstructured data into numerical values which can then be linked with structured data in a database and analyzed with traditional data mining techniques. With an iterative approach, an organization can successfully use text analytics to gain insight into content-specific values such as sentiment, emotion, intensity and relevance. Because text analytics technology is still considered to be an emerging technology, however, results and depth of analysis can vary wildly from vendor to vendor.

EXAMPLE:

1 – Risk management

No matter the industry, Insufficient risk analysis is often a leading cause of failure. This is especially true in the financial industry where adoption of Risk Management Software based on text mining technology can dramatically increase the ability to mitigate risk, enabling complete management of thousands of sources and petabytes of text documents, and

providing the ability to link together information and be able to access the right information at the right time.

2 – Knowledge management

Not being able to find important information quickly is always a challenge when managing large volumes of text documents just ask anyone in the healthcare industry.

Here, organizations are challenged with a tremendous amount of information decades of research in genomics and molecular techniques, for example, as well as volumes of clinical patient data that could potentially be useful for their largest profit center: new product development. Here, knowledge management software based on text mining offer a clear and reliable solution for the “info-glut” problem.

3 – Cybercrime prevention

The anonymous nature of the internet and the many communication features operated through it contribute to the increased risk of internet-based crimes. Today, text mining intelligence and anti-crime applications are making internet crime prevention easier for any enterprise and law enforcement or intelligence agencies.

4 – Customer care service

Text mining, as well as natural language processing are frequent applications for customer care. Today, text analytics software is frequently adopted to improve customer experience using different sources of valuable information such as surveys, trouble tickets, and customer call notes to improve the quality, effectiveness and speed in resolving problems. Text analysis is used to provide a rapid, automated response to the customer, dramatically reducing their reliance on call center operators to solve problems.

5 – Fraud detection through claims investigation

Text analytics is a tremendously effective technology in any domain where the majority of information is collected as text. Insurance companies are taking advantage of text mining technologies by combining the results of text analysis with structured data to prevent

frauds and swiftly process claims.

6 – Contextual Advertising

Digital advertising is a moderately new and growing field of application for text analytics. Here, companies such as Admantx have made text mining the core engine for contextual retargeting with great success. Compared to the traditional cookie-based approach, contextual advertising provides better accuracy, completely preserves the user's privacy.

7 – Business intelligence

This process is used by large companies to uphold and support decision making. Here, text mining really makes the difference, enabling the analyst to quickly jump at the answer even when analyzing petabytes of internal and open source data. Applications such as the Cogito Intelligence Platform (link to CIP) are able to monitor thousands of sources and analyze large data volumes to extract from them only the relevant content.

8 – Content enrichment

While it's true that working with text content still requires a bit of human effort, text analytics techniques make a significant difference when it comes to being able to more effectively manage large volumes of information. Text mining techniques enrich content, providing a scalable layer to tag, organize and summarize the available content that makes it suitable for a variety of purposes.

9 – Spam filtering

E-mail is an effective, fast and reasonably cheap way to communicate, but it comes with a dark side: spam. Today, spam is a major issue for internet service providers, increasing their costs for service management and hardware\software updating; for users, spam is an entry point for viruses and impacts productivity. Text mining techniques can be implemented to improve the effectiveness of statistical-based filtering methods.

10 – Social media data analysis

Today, social media is one of the most prolific sources of unstructured data; organizations have taken notice. Social media is increasingly being recognized as a valuable source of market and customer intelligence, and companies are using it to analyze or predict customer needs and understand the perception of their brand. In both needs Text analytics can address both by analyzing large volumes of unstructured data, extracting opinions, emotions and sentiment and their relations with brands and products.

1.2 How does data mining work?

While large-scale information technology has been evolving separate transaction and analytical systems, data mining provides the link between the two. Data mining software analyzes relationships and patterns in stored transaction data based on open-ended user queries. Several types of analytical software are available: statistical, machine learning, and neural networks. Generally, any of four types of relationships are sought

Classes: Stored data is used to locate data in predetermined groups. For example, a restaurant chain could mine customer purchase data to determine when customers visit and what they typically order. This information could be used to increase traffic by having daily specials.

- **Clusters:** Data items are grouped according to logical relationships or consumer preferences. For example, data can be mined to identify market segments or consumer affinities.
- **Associations:** Data can be mined to identify associations. The beer-diaper example is an example of associative mining.

- **Sequential patterns:** Data is mined to anticipate behavior patterns and trends. For example, an outdoor equipment retailer could predict the likelihood of a backpack being purchased based on a consumer's purchase of sleeping bags and hiking shoes.

Data mining consists of five major elements:

- Extract, transform, and load transaction data onto the data warehouse system.
- Store and manage the data in a multidimensional database system.
- Provide data access to business analysts and information technology professionals.
- Analyze the data by application software.
- Present the data in a useful format, such as a graph or table.

Different levels of analysis are available:

- **Artificial neural networks:** Non-linear predictive models that learn through training and resemble biological neural networks in structure.
- **Genetic algorithms:** Optimization techniques that use processes such as genetic combination, mutation, and natural selection in a design based on the concepts of natural evolution.
- **Decision trees:** Tree-shaped structures that represent sets of decisions. These decisions generate rules for the classification of a dataset. Specific decision tree methods include Classification and Regression Trees (CART) and Chi Square Automatic Interaction Detection (CHAID) . CART and CHAID are decision tree techniques used for classification of a dataset. They provide a set of rules that you can apply to a new (unclassified) dataset to predict which records will have a given outcome. CART segments a dataset by creating 2-way splits while CHAID segments using chi square tests to create multi-way splits. CART typically requires less data preparation than CHAID.

- **Nearest neighbor method:** A technique that classifies each record in a dataset based on a combination of the classes of the k record(s) most similar to it in a historical dataset (where $k \geq 1$). Sometimes called the k -nearest neighbor technique.
- **Rule induction:** The extraction of useful if-then rules from data based on statistical significance.
- **Data visualization:** The visual interpretation of complex relationships in multidimensional data. Graphics tools are used to illustrate data relationships.

1.3. What technological infrastructure is required?

Today, data mining applications are available on all size systems for mainframe, client/server, and PC platforms. System prices range from several thousand dollars for the smallest applications up to \$1 million a terabyte for the largest. Enterprise-wide applications generally range in size from 10 gigabytes to over 11 terabytes. has the capacity to deliver applications exceeding 100 terabytes. There are two critical technological drivers:

- **Size of the database:** the more data being processed and maintained, the more powerful the system required.
- **Query complexity:** the more complex the queries and the greater the number of queries being processed, the more powerful the system required.

Relational database storage and management technology is adequate for many data mining applications less than 50 gigabytes. However, this infrastructure needs to be significantly enhanced to support larger applications. Some vendors have added extensive indexing capabilities to improve query performance. Others use new hardware architectures such as Massively Parallel Processors (MPP) to achieve order-of-magnitude improvements in query time. For example, MPP systems from NCR link hundreds of high-speed Pentium processors to achieve performance levels exceeding those of the largest supercomputers

ANALYSIS OF QUORA QUESTION PAIRS

2.1: Problem Defination:

The goal of this project is to predict which of the provided pairs of questions contain two questions with the same meaning. The ground truth is the set of labels that have been supplied by human experts. The ground truth labels are inherently subjective, as the true meaning of sentences can never be known with certainty. Human labeling is also a 'noisy' process, and reasonable people will disagree.

As a result, the ground truth labels on this dataset should be taken to be 'informed' but not 100% accurate, and may include incorrect labeling. The labels, on the whole, to represent a reasonable consensus, but this may often not be true on a case by case basis for individual items in the dataset.

CHAPTER-3

DATA PREPARATION

Data preparation (or data preprocessing) in this context means manipulation of data into a form suitable for further analysis and processing. It is a process that involves many different tasks and which cannot be fully automated.

There are few steps explained below which are used to prepare the data for model building.

1. Data Collection
2. Data Cleaning
3. Exploratory Data Analysis

3.1. Data Collection:

Data is given in two sets i.e Test.CSV and Train.CSV

All the operations supposed to perform on both datasets.

To import that data sets we have used following R code.

```
data = read.csv("train.csv", header = T , stringsAsFactors = F, nrow = 100000)
```

By running the above code an output is given by the Rstudio which is shown in Fig.1

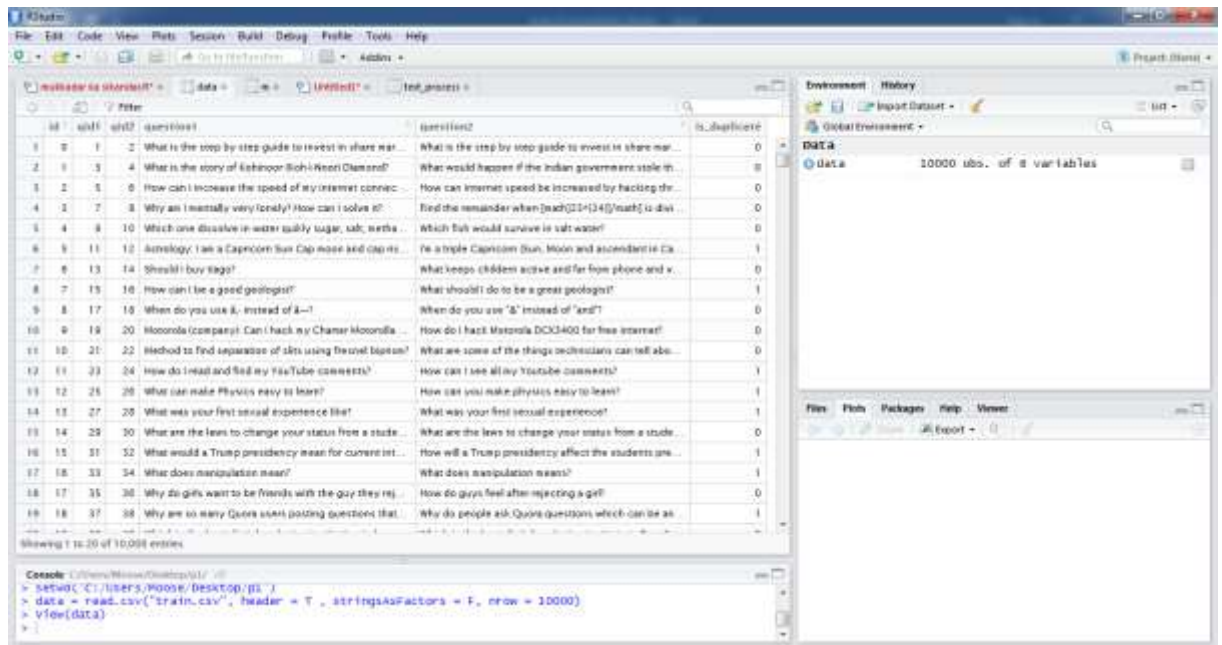


Fig.1 Data Collection

3.2. Data Cleaning

To perform this task all the library which to be used in our whole process is initialized and after that basic data cleaning technique is applied to remove the noise present in data and to remove all the unnecessary elements i.g all the major misspelled words which is present in data are replaced by correct ones, basic missing values are imputed, punctuation marks and extra spaces are removed by using following R code.

```
library(stringr)
```

```
library(psych)
```

```
library(tm)
```

```
library(randomForest)
```

```
library('dplyr')
```

```
library("caret")
```

```
library(ggplot2)
```

```
library(ggthemes)
```

```
library(wordcloud)
```

```
library(wordcloud2)
```

```
library(stringdist)
```

```
library(slam)
```

```
library(class)
```

```
library(caret)
```

```
library(gridExtra)
```

```
library(stringi)
```

```
library("randomForest")
```

```
library(caret)
```

```
library(clusterSim)
```

```
#Calculating different string methods without preprocessing
```

```
preprocess = function(all_text){
```

```
#Converting all words to lower case
```

```
all_text = data.frame(tolower(as.matrix(all_text)) , stringsAsFactors = FALSE)
```

```
all_text = data.frame(apply(all_text,2 , function(y)
```

```
stri_trans_general(y , "latin-ascii")) , stringsAsFactors = FALSE)
```

```
all_text = data.frame(apply(all_text,2 , function(y)
```

```
gsub("'ve", " have ", y)),stringsAsFactors = FALSE)
```

```
all_text = data.frame(apply(all_text,2 , function(y)
```

```
gsub("'s", " is ", y)),stringsAsFactors = FALSE)
```

```
all_text = data.frame(apply(all_text,2 , function(y)
```

```
gsub("'can't", " cannot ", y)),stringsAsFactors = FALSE)
```

```
all_text = data.frame(apply(all_text,2 , function(y)
```

```
gsub("'hadn't", " had not ", y)),stringsAsFactors = FALSE)
```

```
all_text = data.frame(apply(all_text,2 , function(y)
```

```
gsub("'i'm", " i am ", y)),stringsAsFactors = FALSE)
```

```
all_text = data.frame(apply(all_text,2 , function(y)
```

```
gsub("'re", " are ", y)),stringsAsFactors = FALSE)
```

```
all_text = data.frame(apply(all_text,2 , function(y)
gsub("'d", " would ", y)),stringsAsFactors = FALSE)
```

```
all_text = data.frame(apply(all_text,2 , function(y)
gsub("'ll", " will ", y)), stringsAsFactors = FALSE)
```

#Removing punctuation marks

```
all_text = data.frame( apply(all_text, 2, function(y)
gsub("[[:punct:]]", " ", y, perl = T)))
```

#Delete extra spaces

```
all_text = data.frame(lapply(all_text, function(y)
gsub('^ *|(?<= ) | *$', "", y, perl = TRUE))), stringsAsFactors = FALSE)

return(all_text)

}
```

#apply preprocessing associated with function “preprocess” and store the value in variable named ”text_process”

```
text_process = preprocess(data[,4:5])
```

```
text_process$is_duplicate = data$is_duplicate
```

```
text_process = data.frame(apply(text_process, 2, function(x) gsub("'^$|^ $", NA, x)))
```

```
sum(is.na(text_process))
```

#Removing incomplete observations

```
text_process = text_process[complete.cases(text_process),]
```

```
str(text_process)
```

#Converting variables to their respective classes

```
text_process$question1 = as.character(text_process$question1)
```

```
text_process$question2 = as.character(text_process$question2)
```

#various distance variable created and stored in a matrix named “t_matrix”

```
t_matrix = ques_matrix(text_process[,1:2])
```

#Combining the performed methods to rest of the data

```
text_process = cbind(text_process,t_matrix)
```


3.3. Exploratory Data Analysis

Exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods.

In this case we have taken several steps like visualizing the Wordcloud, line plots, graph plots and observation of data. which served basic understanding of data.

All the steps is explained by step by step using following R code.

```
length(unique(data$question1))      #unique observation in question1
```

```
output : 86995
```

```
length(unique(data$question2))      #unique observation in question2
```

```
output : 87221
```

```
table(data$is_duplicate)
```

```
output : 0      1  
        62746 37254
```

```
summary(text_process[,1:3])
```

```
output : question1      question2      is_duplicate  
        Length:99995    Length:99995    0:62741  
        Class :character Class :character 1:37254  
        Mode  :character Mode  :character
```

```
sum(data$is_duplicate==0)/nrow(data)*100  # % of data contains 0
```

```
output : 62.74
```

sum(data\$is_duplicate==1)/nrow(data)*100 # % of data contains 1

output : 37.254

sum(text_process\$q1length>=7 & text_process\$q1length<=13)/nrow(text_process)*100

output : 61.43

sum(text_process\$q1length<7)/nrow(text_process)*100

output : 14.87

sum(text_process\$q1length>13)/nrow(text_process)*100

output : 23.69

sum(text_process\$q2length>=7 & text_process\$q2length<=13)/nrow(text_process)*100

output : 60.21

sum(text_process\$q2length<7)/nrow(text_process)*100

output : 15.42

sum(text_process\$q2length>13)/nrow(text_process)*100

output : 24.36

```
table(text_process$is_duplicate[which(text_process$q2length>20)])
```

```
output :  0    1  
         6921 1205
```

```
table(text_process$is_duplicate[which(text_process$diff_length>1)])
```

```
output :  0    1  
         41584 19544
```

```
sum(text_process$is_duplicate==0 &  
text_process$diff_length>10)/nrow(text_process)*100
```

```
output :  7.43
```

```
sum(text_process$is_duplicate==1 &  
text_process$diff_length>10)/nrow(text_process)*100
```

```
output :  0.817
```

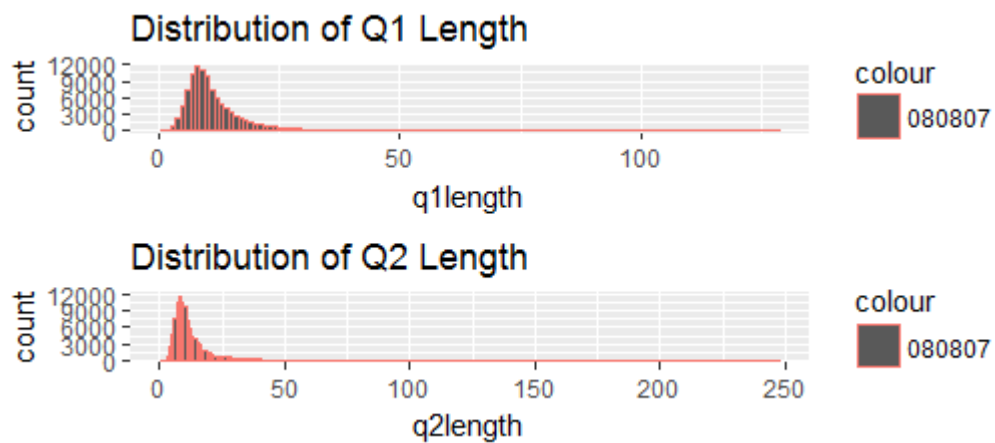
#distribution of Q1 & Q2 length

```
h1 = qplot(q1length, data = text_process, geom = "histogram", binwidth = 1, main =  
"Distribution of Q1 Length", color="080807")
```

```
h2 = qplot(q2length, data = text_process, geom = "histogram", binwidth = 1, main =  
"Distribution of Q2 Length",color="080807")
```

```
grid.arrange(h1,h2)
```

Output :



Q1 Length Boxplot

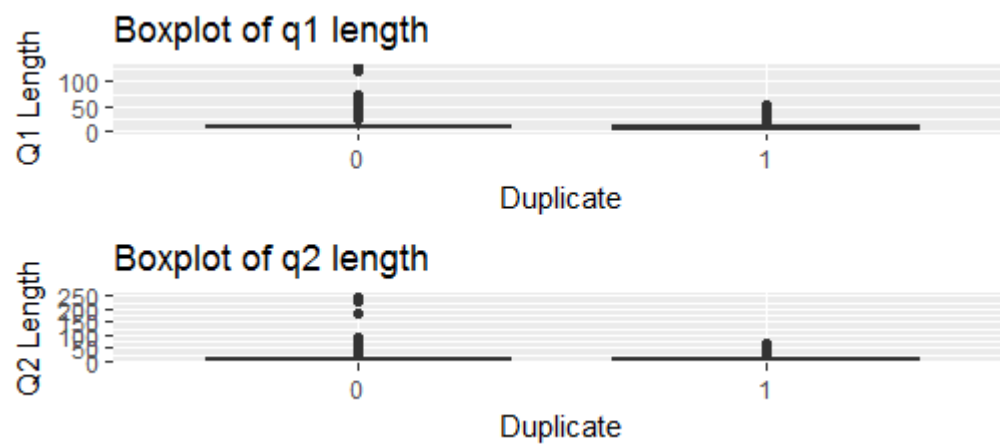
```
q1_l = qplot(is_duplicate,q1length, data = text_process, geom = "boxplot",  
             main = "Boxplot of q1 length", xlab = "Duplicate", ylab = "Q1 Length")
```

Q2 Length Boxplot

```
q2_l = qplot(is_duplicate,q2length, data = text_process, geom = "boxplot",  
             main = "Boxplot of q2 length", xlab = "Duplicate", ylab = "Q2 Length")
```

```
grid.arrange(q1_l,q2_l)
```

Output:



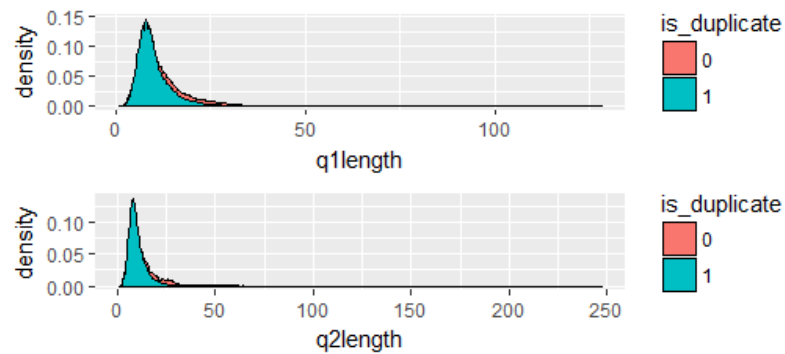
#Density plot of q1 and q2 length

```
d1=qplot(q1length, data = text_process, geom = "density", fill = is_duplicate)
```

```
d2=qplot(q2length, data = text_process, geom = "density", fill = is_duplicate)
```

```
grid.arrange(d1,d2)
```

Output:



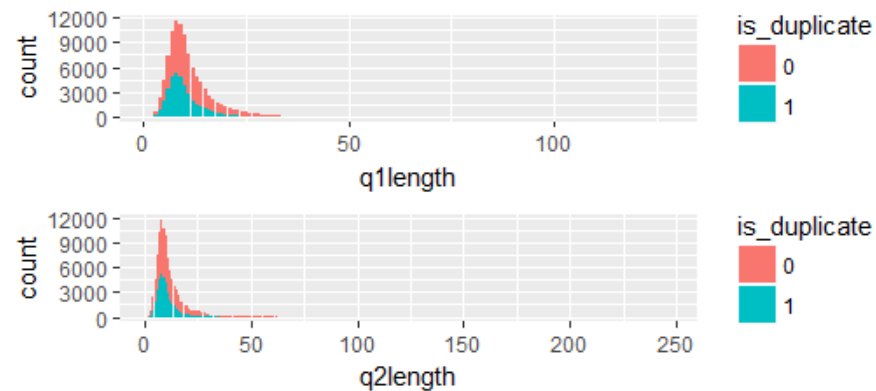
#Barplot of q1 and q2 length

```
b1=qplot(q1length, data = text_process, geom = "bar", fill = is_duplicate)
```

```
b2=qplot(q2length, data = text_process, geom = "bar", fill = is_duplicate)
```

```
grid.arrange(b1,b2)
```

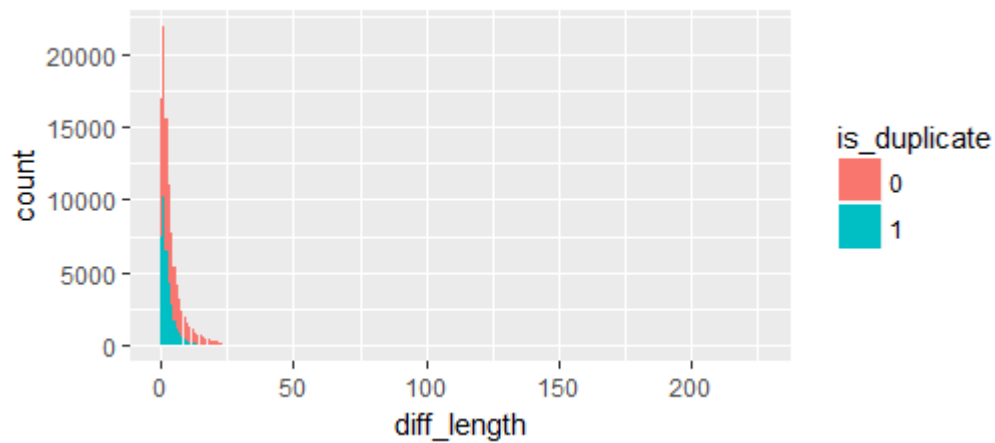
Output:



q1 and q2 length difference plot

```
qplot(diff_length, data = text_process, geom = "bar", fill = is_duplicate)
```

Output :



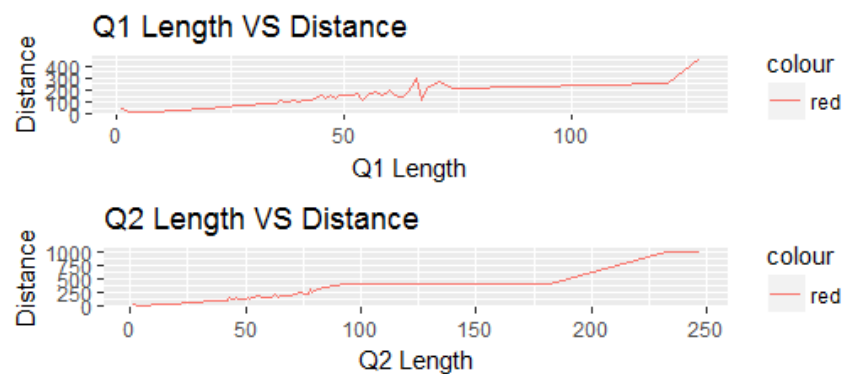
line plot of q1 and q2 length vs column dist

```
l1=line(text_process,"q1length","dist","Q1 Length VS Distance","Q1 Length","Distance")
```

```
l2=line(text_process,"q2length","dist","Q2 Length VS Distance","Q2 Length","Distance")
```

```
grid.arrange(l1,l2)
```

Output:



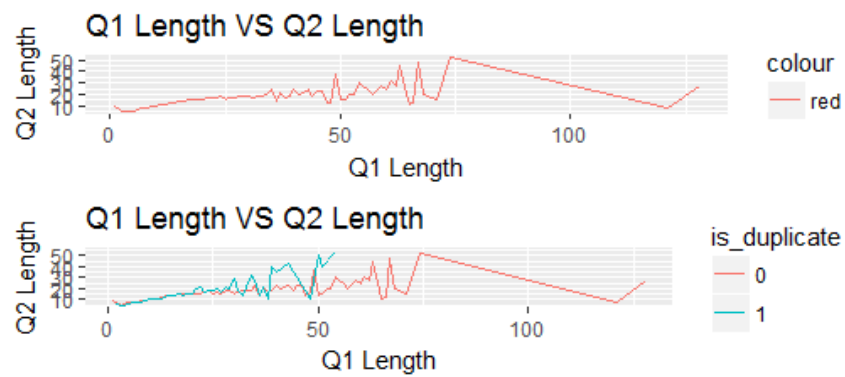
q1 and q2 length line plot

```
l3=line(text_process,"q1length","q2length","Q1 Length VS Q2 Length","Q1 Length","Q2 Length")
```

```
l4=line2(text_process,"q1length","q2length","Q1 Length VS Q2 Length","Q1 Length","Q2 Length")
```

```
grid.arrange(l3,l4)
```

Output:



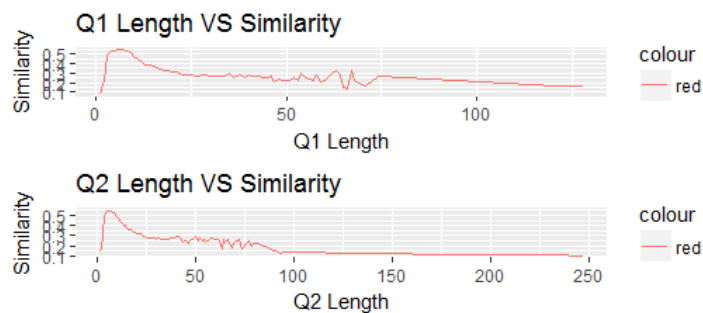
q1 and q2 length similarity line plot

```
l5=line(text_process,"q1length","simi","Q1 Length VS Similarity","Q1 Length","Similarity")
```

```
l6=line(text_process,"q2length","simi","Q2 Length VS Similarity","Q2 Length","Similarity")
```

```
grid.arrange(l5,l6)
```

Output:



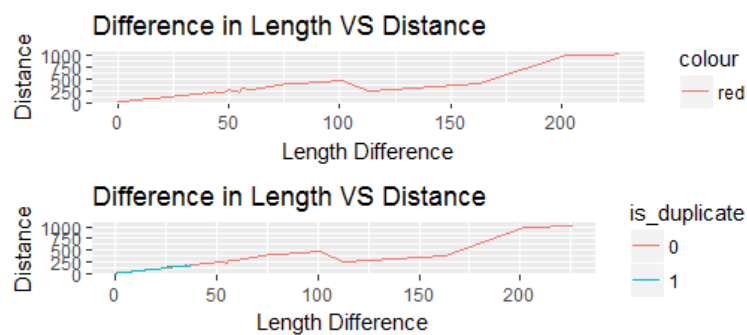
line plot of q1 and q2 length difference from column "dist"

```
l7=line(text_process,"diff_length","dist","Difference in Length VS Distance","Length  
Difference","Distance")
```

```
l8=line2(text_process,"diff_length","dist","Difference in Length VS Distance","Length  
Difference","Distance")
```

```
grid.arrange(l7,l8)
```

Output:



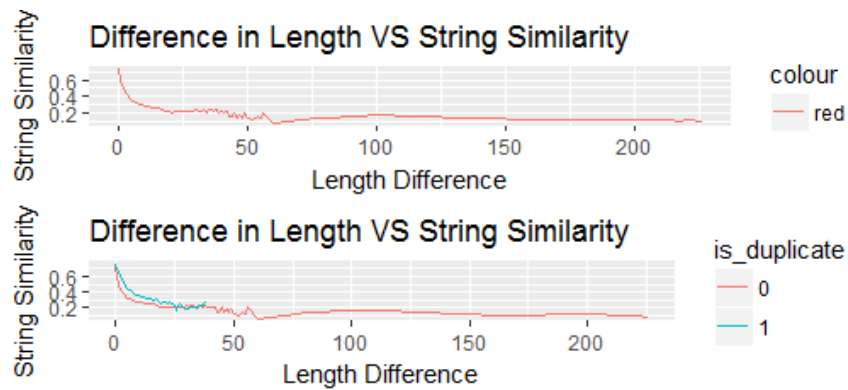
Difference in Length VS String Similarity

```
l9=line(text_process,"diff_length","simi","Difference in Length VS String Similarity","Length  
Difference","String Similarity")
```

```
l10=line2(text_process,"diff_length","simi","Difference in Length VS String  
Similarity","Length Difference","String Similarity")
```

```
grid.arrange(l9,l10)
```

Output:



#wordcloud for frequent terms

```
corpus = Corpus(VectorSource(text))
```

#Remove Stopwords and our predefined words

```
corpus = tm_map(corpus, removeWords, c('i','its','it','us','use','want',  
                                     'added','used','using','will','yes','say',  
                                     'can','take','one', stopwords('english'))))
```

#remove unnecessary spaces

```
corpus = tm_map(corpus, stripWhitespace)
```

#Word cloud

```
w = wordcloud(corpus, max.words = 80, scale=c(6, 1), colors=brewer.pal(8, "Dark2"))
```

Output:



CHAPTER-4

MODEL BUILDING

In this phase processed data is used to build model, in this case Random Forest machine learning algorithm is used to build classification model because train data is numeric and continues.

Preprocessed data has normalized before applying RF algorithm on the top of it so that range of each variable can be equalized.

Internal mechanism of Random Forest can be explained as follow

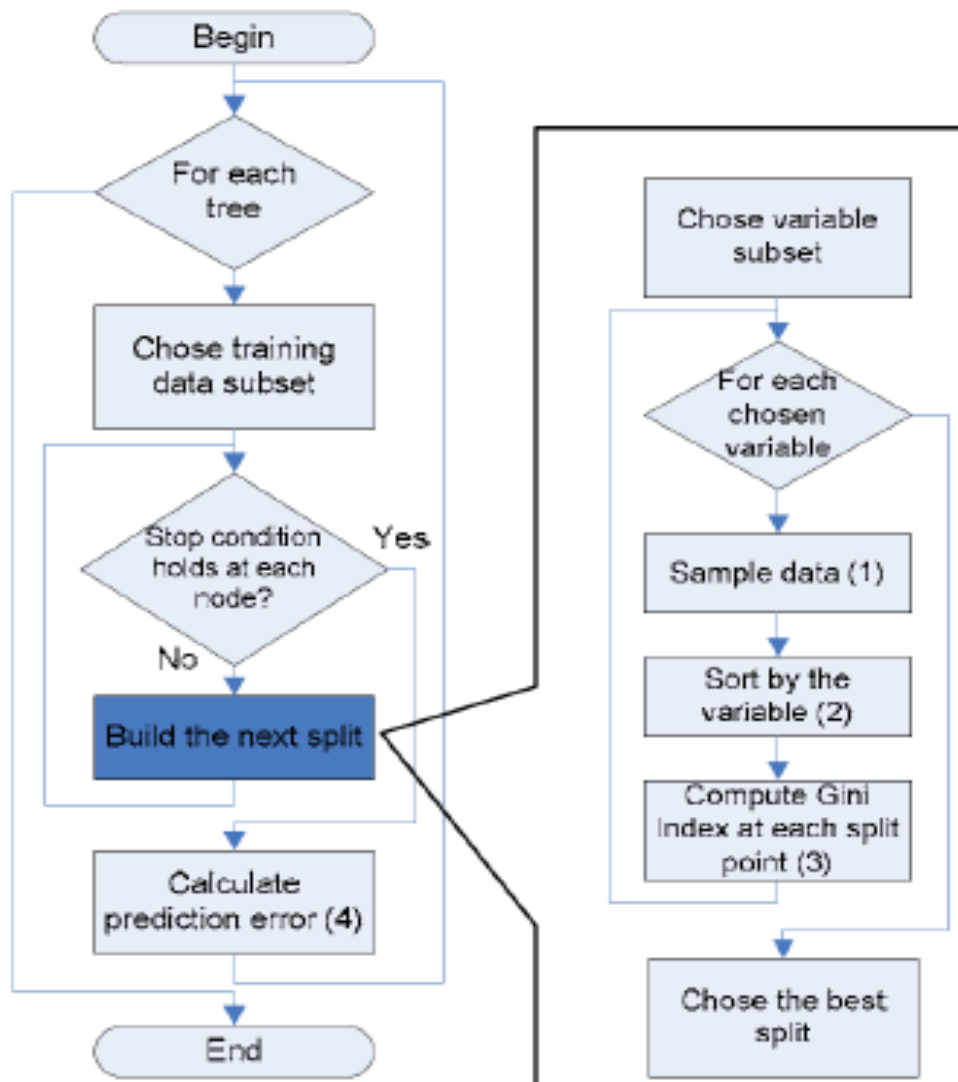
4.1: Random Forest-

- Random forest is an ensemble that consists of many decision trees
- The method combines Breiman's "bagging" idea and the random selection of features
- Outputs the class that is the mode of the class's output by individual trees.
- Mean for regression
- Can be used for classification and regression

Algorithm:

1. Each tree (CART) is constructed using the following steps:
 - a. Let the number of training cases be N , and the number of variables in the classifier be M .
 - b. We are told the number m of input variables to be used to determine the decision at a node of the tree; m should be much less than M .
 - c. $m = \sqrt{M}$
 - d. Choose a training set for this tree by choosing 66% data with replacement from all N available training cases (i.e. take a bootstrap sample). Use the rest of the cases to estimate the error of the tree, by predicting their classes.
 - e. For each tree grown, 33-34% of samples are not selected in bootstrap, called out of bootstrap (OOB) samples.
 - f. Error from OOB is called out of bag error which we feed to next DT

- g. For each node of the tree, randomly choose m variables on which to base the decision at that node. Calculate the best split based on these m variables in the training set using GINI Index.
 - h. Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).
- 2. Number of decision trees depend on error rate
- 3. Build trees until the error no longer decreases.
- 4. For prediction a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the average vote of all trees is reported as random forest prediction.
- 5. Output of the algorithm could be rules, number or probabilities depend on the requirement.
- 6. Input variables could be continuous or categorical.



Random Forest Algorithm Flow chart

For model building whole train data is again divided into two parts as train and test 70% and 30% respectively of train data.

RF model can be build using following R code output can be found as.

```
var1=text_process

var1 = data.Normalization(var1[,4:14], type = "n4", normalization = "column")

var1$is_duplicate = text_process$is_duplicate

train = var1[[sample(nrow(var1), 70000,replace = F),]]

test = var1[!(1:nrow(var1)) %in% as.numeric(row.names(train)), ]

fit_classify = randomForest(is_duplicate ~ ., train, importance = TRUE, ntree = 300)

fit_classify

pred = predict(fit_classify, test[,12])

xtab = table(observed = test[,12], predicted = pred)

confusionMatrix (xtab)
```

```
> fit_classify

Call:
randomForest(formula = is_duplicate ~ ., data = train, importance = TRUE,      ntree = 300)
      Type of random forest: classification
      Number of trees: 300
No. of variables tried at each split: 3

      OOB estimate of  error rate: 28.5%
Confusion matrix:
      0      1 class.error
0 34517  9438  0.2147196
1 10512 15533  0.4036091
> |
```

Random Forest Model

4.2: Calculate Error Matrix:

Confusion matrix is calculated as error matrix which state that how well model is performing.

as we can see Accuracy is shown below 90% means 90% of data is correctly classified in our test data.

confusionMatrix(xtab)

Output:

```
Console C:/Users/Moose/Desktop/p1/ ↗
> confusionMatrix(xtab)
Confusion Matrix and Statistics

      predicted
observed    0    1
      0 17449 1333
      1  1430 9786

      Accuracy : 0.9079
      95% CI : (0.9046, 0.9111)
      No Information Rate : 0.6293
      P-Value [Acc > NIR] : <2e-16

      Kappa : 0.8029
      Mcnemar's Test P-Value : 0.0678

      Sensitivity : 0.9243
      Specificity : 0.8801
      Pos Pred Value : 0.9290
      Neg Pred Value : 0.8725
      Prevalence : 0.6293
      Detection Rate : 0.5817
      Detection Prevalence : 0.6261
      Balanced Accuracy : 0.9022

      'Positive' Class : 0

> |
```

Confusion Matrix

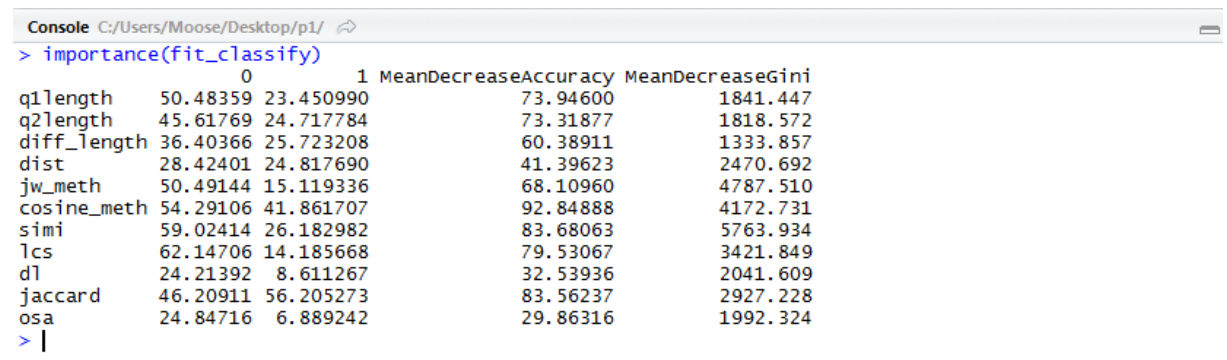
4.3: Variable Importance:

Variable importance also determined by RF algorithm though it is not much reliable.

Importance can be obtained by using following R code

Importance(fit_classify)

Output:



	0	1	MeanDecreaseAccuracy	MeanDecreaseGini
q1length	50.48359	23.450990	73.94600	1841.447
q2length	45.61769	24.717784	73.31877	1818.572
diff_length	36.40366	25.723208	60.38911	1333.857
dist	28.42401	24.817690	41.39623	2470.692
jw_meth	50.49144	15.119336	68.10960	4787.510
cosine_meth	54.29106	41.861707	92.84888	4172.731
simi	59.02414	26.182982	83.68063	5763.934
lcs	62.14706	14.185668	79.53067	3421.849
dl	24.21392	8.611267	32.53936	2041.609
jaccard	46.20911	56.205273	83.56237	2927.228
osa	24.84716	6.889242	29.86316	1992.324

CHAPTER-5

MODEL DEPLOYMENT

In this phase model can be applied on desired dataset and classified values can be predicted.

In this case test data set which is provided having 2500k of observation so we used only 100k of sample due to constrain of computational resources.

All the preprocessing steps are applied to this test data set as before, and finally applied RF model on the top of it Using R code and classified output values obtained as.

#importing and Mining for test data set

```
data1 = read.csv("test.csv", header = T , stringsAsFactors = F, nrow = 100000)

text_process = preprocess(data1[,2:3])

text_process = data.frame(apply(text_process, 2, function(x) gsub("^$|^ $", NA, x)))

sum(is.na(text_process))
```

#Removing incomplete observations

```
text_process = text_process[complete.cases(text_process),]

str(text_process)
```

#Converting variables to their respective classes

```
text_process$question1 = as.character(text_process$question1)

text_process$question2 = as.character(text_process$question2)

t_matrix = ques_matrix(text_process[,1:2]) #distance matrices
```

#Combining the performed methods to rest of the data

```
text_process = cbind(text_process,t_matrix)
```

#Model Deployment

```
var2 = text_process
```

```
var2 = data.Normalization(var2[,3:13], type = "n4", normalization = "column")
```

```
pred2 = predict(fit_classify, var2)
```

```
output = cbind(text_process,pred2)
```

```
names(output)[14] = "pred_is_duplicate"
```

```
output = subset(output, select = c(question1, question2,pred_is_duplicate))
```

```
write.csv(output, file = "pred_output.csv")
```

```
table(output$pred_is_duplicate)
```

```
output:      0      1  
       78280 21718
```

CONCLUSION

By looking into confusion accuracy of model can be considered about 90%, it means this model is able to predict approximate upto 90% accurate values.

