NLP Based Question & Answers for E-commerce

Submitted By: -
Ashish Gore

Guided By: -
Kartik
Parth Sagar

1. **Pre-Introduction**

Project was done in

- **Operating System**: - Windows10
- **Processor**: - Intel core i7
- **RAM**: -16 GB
- **System Type**: -64-bit operating System, x64-based processor
- **Software Used**:- R Version 3.6.1 (architecture x86_64).

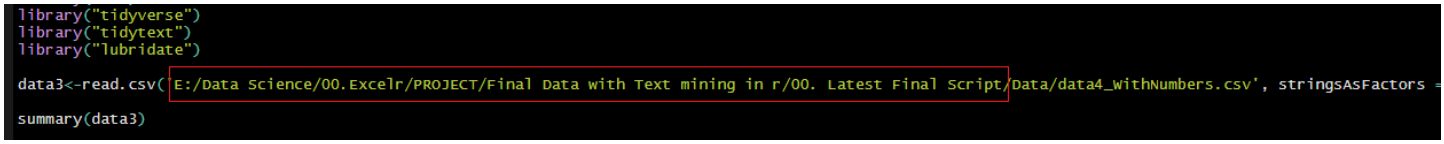
Please install the required library if not already installed in R as shown below

```
install.packages(c("XML", "sqldf", "dplyr", "tm", "tidyverse", "tidytext", "textclean", "
qdapRegex", "hunspell", "textstem", "DataCombine", "stringr", "sentimentr", "magrittr", "
NLP", "lubridate", "caret", "rlist", "shiny", "DT", "ggplot2", "wordcloud", "reshape2", "
reshape2", "ggraph", "ggforce", "igraph", "corrplot", "PerformanceAnalytics"))
```

Sequence of Code file to run

- 01.EDA.R
- 02.EDA.R
- 03.Visualization.R
- 04.Function for Text Analysis.R
- 05.Model.R
- 06.Shiny.R

Please change file path present in starting line of code to the path where you would be storing our folders. Please modify only highlighted path shown below only on files 01.EDA.r, 02.EDA.r, 05.Model.r:



2. Business Objective

In any website or in ecommerce business there are very few FAQ or limited user review. All the questions cannot be shown in website as website will look clumsy. Due to limited exposure of the questions, there is a possibility that user might not find question and the answer He/ She is searching for or the question is not present or never have been asked.

Considering different scenarios, we are proposing a model which will take user question and will bring out relevant top five question and answer from Question Bank.

Our model ensures that

i. User will get relevant answer for the question he/ she is searching for
ii. Website will not be clumsy
iii. Since all the process is happening at server side and only top 5 result is achieved, website will not be clumsy.

3. Project Architecture

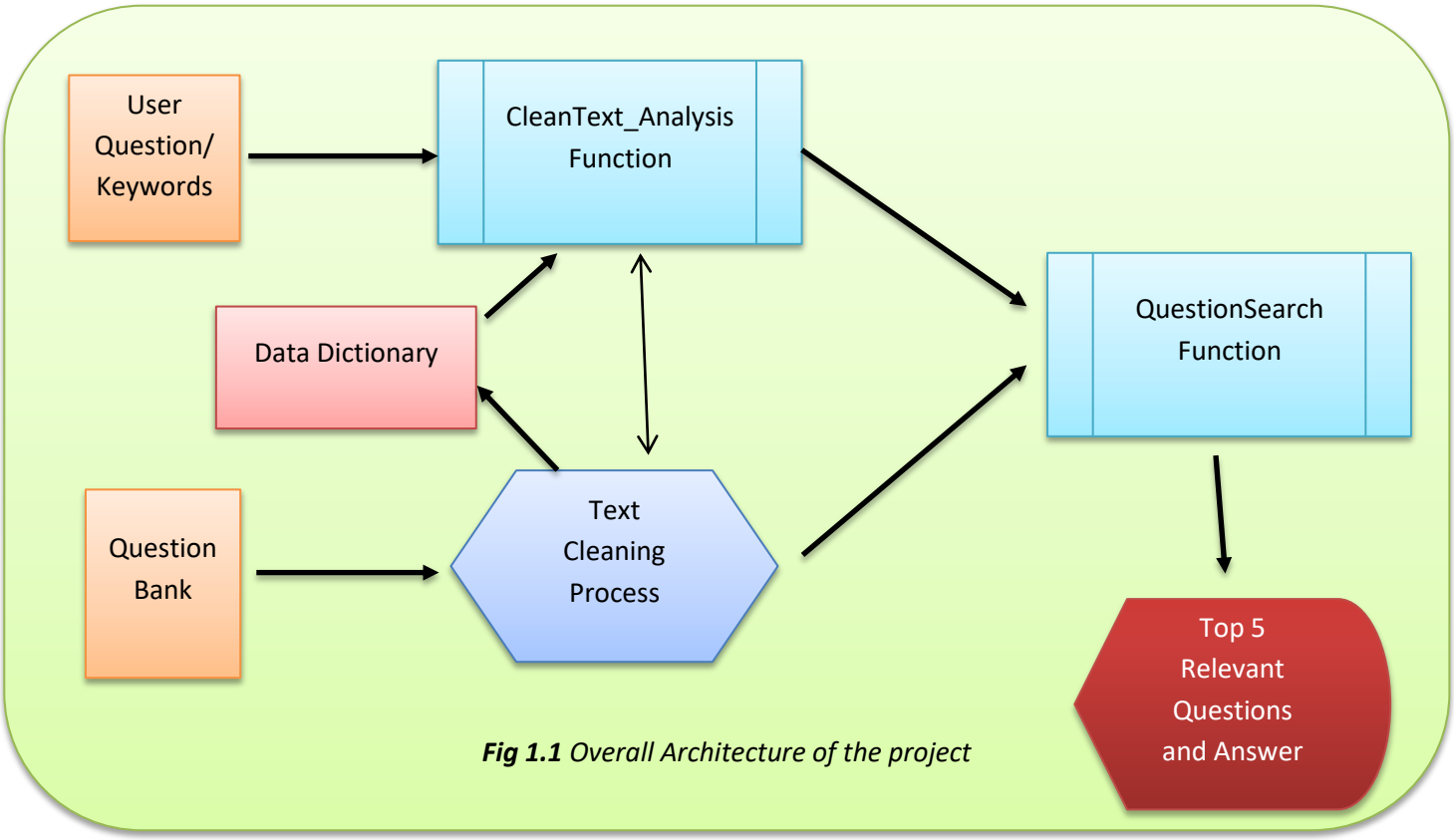


Fig 1.1 *Overall Architecture of the project*

3.1 Input Layer

User Question and Question bank both are considered as input.

- **User question/Keywords** is the text or question that user will provide as an input.
- **Question bank** means the existing question and answer set that is already present. Later User question will be compared with this question bank to find relevant Question and answer.

3.2 Preprocessing

Before considering taking text as input, text needs to be cleaned. Process of cleaning text includes:

- Removal of Stop words (common words occur in a sentence)
- Elimination of URLs (with https or ftp [absolute URLs] and broken URLs (without https or ftp]).
- Exclusion of emails.
- Deletion unwanted words
- Word spell check and correction
- Word stemming

3.3 Model

Model is a function "**QuestionSearch**" which computes cosine similarity between user Questions/Keywords and each Question bank to find the similar questions.

3.4 Final Output

Final Output is the relevant question and answer shown based on the user input/keywords. Top five results will be shown with decreasing order of cosine similarity.

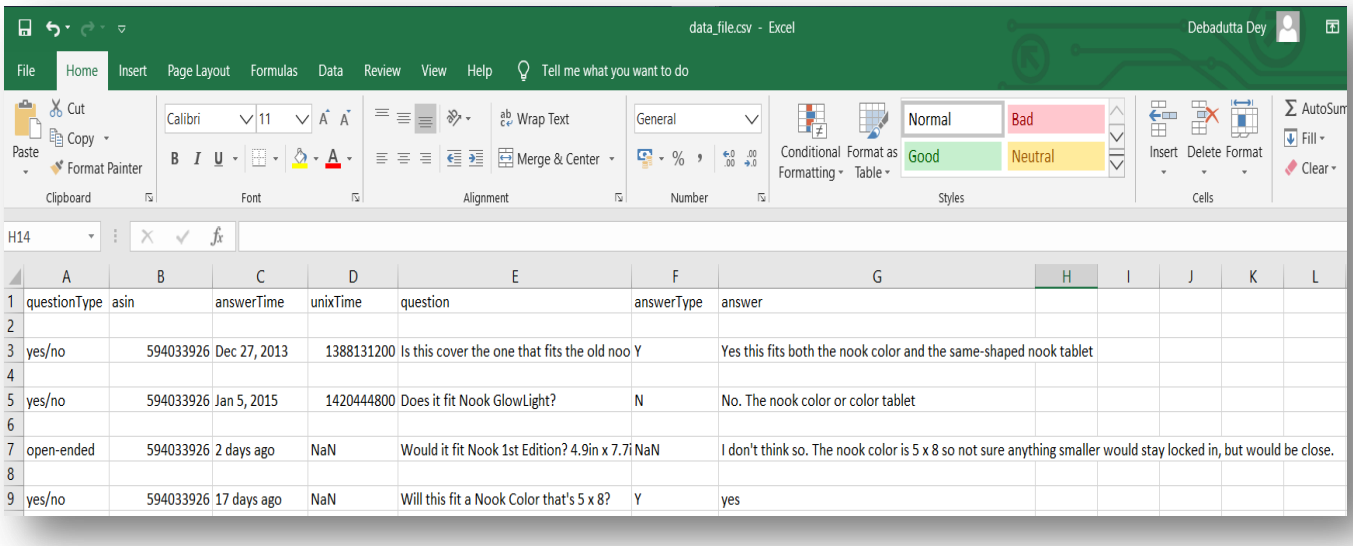
4. Question Bank Data

Data was provided in **NDJson** format.


```
'questionType': 'yes/no', 'asin': '0594033926', 'answerTime':
'Dec 27, 2013', 'unixTime': 1388131200, 'question': 'Is this
cover the one that fits the old nook color? Which I believe is
8x5.', 'answerType': 'Y', 'answer': 'Yes this fits both the
nook color and the same-shaped nook tablet'}
{'questionType': 'yes/no', 'asin': '0594033926', 'answerTime':
'Jan 5, 2015', 'unixTime': 1420444800, 'question': 'Does it
fit Nook GlowLight?', 'answerType': 'N', 'answer': 'No. The
nook color or color tablet'}
{'answer': "I don't think so. The nook color is 5 x 8 so not
sure anything smaller would stay locked in, but would be
close.", 'asin': '0594033926', 'answerTime': '2 days ago',
'question': 'Would it fit Nook 1st Edition? 4.9in x 7.7in ?',
'questionType': 'open-ended'}
{'questionType': 'yes/no', 'asin': '0594033926', 'answerTime':
'17 days ago', 'question': "Will this fit a Nook Color that's
5 x 8?", 'answerType': 'Y', 'answer': 'yes'}
{'questionType': 'yes/no', 'asin': '0594033926', 'answerTime':
'Feb 10, 2015', 'unixTime': 1423555200, 'question': 'will this
fit the Samsung Galaxy Tab 4 Nook 10.1', 'answerType': 'N',
'answer': "No, the tab is smaller than the 'color'"}
{'questionType': 'yes/no', 'asin': '0594033926', 'answerTime':
'Jan 30, 2015', 'unixTime': 1422604800, 'question': 'does it
have a flip stand?', 'answerType': 'N', 'answer': 'No, there
is not a flip stand. It has a pocket in the front flap. It is
a very nice cover.'}
```


***Fig 3.1** shows a portion of sample data in qa_electronics.json*

Data was converted into csv format in python and stored as file bearing name "**data_file.csv**". Python code is present file "**qa_Electronics to Data.csv.ipynb**".



	A	B	C	D	E	F	G	H	I	J	K	L
1	questionType	asin	answerTime	unixTime	question	answerType	answer					
2												
3	yes/no	594033926	Dec 27, 2013	1388131200	Is this cover the one that fits the old noo	Y	Yes this fits both the nook color and the same-shaped nook tablet					
4												
5	yes/no	594033926	Jan 5, 2015	1420444800	Does it fit Nook GlowLight?	N	No. The nook color or color tablet					
6												
7	open-ended	594033926	2 days ago	NaN	Would it fit Nook 1st Edition? 4.9in x 7.7i	NaN	I don't think so. The nook color is 5 x 8 so not sure anything smaller would stay locked in, but would be close.					
8												
9	yes/no	594033926	17 days ago	NaN	Will this fit a Nook Color that's 5 x 8?	Y	yes					

***Fig 3.2** shows a portion of sample data in data_file.csv*

Dataset bearing name "**data_file.csv**" was loaded into R global environment.

```
data=read.csv('data_file.csv', stringsAsFactors = FALSE)
```

Checked and converted the data into correct data type.

- Converted **unixtime** from Posixct to GMT.
- Converted **questiontype** to factor datatype
- Converted **asin** to factor datatype
- Converted **answerType** to factor datatype

```
data$unixTime <- as.POSIXct(as.numeric(data$unixTime), origin = '1970-01-01', tz = 'GMT')
data$questionType<-as.factor(data$questionType)
data$asin <-as.factor(data$asin )
data$answerType<-as.factor(data$answerType)
```

Final Data in R

	questionType	asin	answerTime	unixTime	question	answerType	answer
1	yes/no	0594033926	Dec 27, 2013	2013-12-27 08:00:00	Is this cover the one that fits the old nook color? Which I bel...	Y	Yes this fits both the nook color and the same-shaped nook ...
2	yes/no	0594033926	Jan 5, 2015	2015-01-05 08:00:00	Does it fit Nook GlowLight?	N	No. The nook color or color tablet
3	open-ended	0594033926	2 days ago	*NA*	Would it fit Nook 1st Edition? 4.9in x 7.7in ?	Nan	I don't think so. The nook color is 5 x 8 so not sure anything...
4	yes/no	0594033926	17 days ago	*NA*	Will this fit a Nook Color that's 5 x 8?	Y	yes
5	yes/no	0594033926	Feb 10, 2015	2015-02-10 08:00:00	will this fit the Samsung Galaxy Tab 4 Nook 10.1	N	No, the tab is smaller than the 'color'
6	yes/no	0594033926	Jan 30, 2015	2015-01-30 08:00:00	does it have a flip stand?	N	No, there is not a flip stand. It has a pocket in the front flap. ...
7	yes/no	0594033926	Jan 30, 2015	2015-01-30 08:00:00	does this have a flip stand	?	Hi, no it doesn't
8	open-ended	0594033926	Dec 22, 2014	2014-12-22 08:00:00	also fits the HD+?	Nan	It should. They are the same size and the charging port is in ...
9	yes/no	0594033926	Nov 16, 2014	2014-11-16 08:00:00	Does it have 2 positions for the reader? Horizontal/vertical T...	Y	Yes

Columns Details with their datatype

Column Name	Column Description	Column Datatype
QuestionType	It conveys if the question is open ended or yes/No type of question	Character, Factor
Asin	Product id of a product. Uniquely identifies the product	Character, Factor
AnswerTime	It contains date and time. It records the answer date	Character
UnixTime	It contains date and time. It records the answer date in unix format	
Question	Contains Questions	Character
AnswerType	Conveys if answer is yes or no and for open ended question it is blank. Since it was converted from python null values are represented by NaN	Character, Factor
Answer	Contains Answer	Character

5. Exploratory Data Analysis

i) One Hot Encoding

- Column **QuestionType** represents if questiontype is open ended or yes or no. It is factor datatype and hence performed one hot encoding on questionType and created a new column "**QuestionTypeOE**".

```
## One hot Encoding with QuestionType
data1<-sqldf(c("alter table data1 add column questiontypeOE bit","select * from data1"))
data1<- sqldf(c("update data1 set questiontypeOE=1 where questiontype='open-ended'","select * from data1"))
data1<- sqldf(c("update data1 set questiontypeOE=0 where questiontype<>'open-ended'","select * from data1"))
```

ii) Removing similar columns

- **UnixTime** was provided in posixct format. After converting the unixtime to GMT, found that data in AnswerTime is same as data in UnixTime. Hence removed AnswerTime column.

iii) Imputing Null Values by Median Values

- Null Values were present in **UnixTime** Columns.
- Imputing the null values with median values of that particular product (**asin**).

```
data1$unixTime<-as.Date(data1$unixTime , format = "%m/%d/%y") #mdy(data4$unixTime)


data2<-sqldf("select asin,median(unixTime) MedianDate  from data1 group by asin")
data2$MedianDate<- as.Date(data2$MedianDate, origin='1970-01-01')


data1<- merge(data1,data2,by="asin")
```

- After calculating median values by product, still some records have null and had to impute with median value of total dataset.

```
sqldf("select * from data1 where asin in ('B0007XD4LC','B000VNJD1S')")

sqldf("select median(unixTime) from data1")  #16126
as.Date(16126, origin='1970-01-01') #"2014-02-25"

data1<-sqldf(c("update data1 set MedianDate ='2014-02-25' where asin in ('B0007XD4LC','B000VNJD1S') and MedianDate  is null","select * from data1"))

data1<-sqldf(c("update data1 set unixTime=MedianDate where unixTime is null","select * from data1"))

data1<-data1[, !(colnames(data1) %in% c("MedianDate"))]
```

iv) Creating new columns

- Created new columns which contain the Question length, Question Word Count, Answer length, Answer Word Count.

```
data1$QuestionLength<-nchar(data1$question)
data1$AnswerLength<-nchar(data1$answer)

library(stringr)
data1$questionWCount<-str_count(data1$question,'\\w+')
data1$answerWCount<-str_count(data1$answer,'\\w+')
```

- Creating new columns from the sentiment, average sentiment and standard deviation of question and answer column.

```
library(sentimentr)
summary(data1)

questionSentiment<-sentiment_by(data1$question, by = NULL)
answerSentiment<-sentiment_by(data1$answer, by = NULL)

data1$questionAve_Sentiment<-questionSentiment$ave_sentiment
data1$answerAve_Sentiment<-answerSentiment$ave_sentiment

data1$questionsd<-questionSentiment$sd
data1$answersd<-answerSentiment$sd
```

- Replacing the na values with median

```
sqldf("select median(questionsd) from data1") #0.1020621
sqldf("select median(answersd) from data1") #0.1833333

#median of questionsd is 0.1020621. Imputing this with na values.
data1<-sqldf(c("update data1 set questionsd = 0.1020621  where questionsd is null","select * from data1"))

#median of answersd is 0.1833333. Imputing this with na values.
data1<-sqldf(c("update data1 set answersd = 0.1833333  where answersd is null","select * from data1"))
```

- Creating new columns from **bing** sentiment.

```
data2<-data1 %>% unnest_tokens(word, question) %>%
  inner_join(get_sentiments("bing")) %>% # pull out only sentiment words
  count(rownum,sentiment) %>% # count the # of positive & negative words
  spread(sentiment, n, fill = 0) %>% # made data wide rather than narrow
  mutate(sentiment = positive - negative) # # of positive words - # of negative words
colnames(data2)<-c("rownum","negativeQ","positiveQ","sentimentQ")
```

- Na values in negetiveQ, PositiveQ, sentiment is replaced by 0

```
data3<-merge(x = data1, y = data2, by = "rownum", all.x = TRUE)

data3$negativeQ[is.na(data3$negativeQ)] <- 0
data3$positiveQ[is.na(data3$positiveQ)] <- 0
data3$sentimentQ[is.na(data3$sentimentQ)] <- 0
```

- Same steps were performed for answerwords too.

```
data3<-data2 %>% unnest_tokens(word, answer) %>%
  inner_join(get_sentiments("bing")) %>% # pull out only sentiment words
  count(rownum,sentiment) %>% # count the # of positive & negative words
  spread(sentiment, n, fill = 0) %>% # made data wide rather than narrow
  mutate(sentiment = positive - negative) # # of positive words - # of negative words
colnames(data3)<-c("rownum","negativeA","positiveA","sentimentA")


data4<-merge(x = data4, y = data3, by = "rownum", all.x = TRUE)

data4$negativeA[is.na(data4$negativeA)] <- 0
data4$positiveA[is.na(data4$positiveA)] <- 0
data4$sentimentA[is.na(data4$sentimentA)] <- 0
```

- Removing negetiveQ,positiveQ,negetiveA and positiveA as sum of negetiveQ, positiveQ equals SentimentQ and sum of negetiveA and positiveA equals SentimentA

v) Text Analysis

- Loaded stop words from already created stop words list**.**
- Removed "**yes**" and "**No**" from stop words as it needs to be present in text for further analysis.

```
#removing stopwords in dataframe
stopwd <- read.table('stop.txt')
stopwd <-as.character(stopwd$V1)

#stopwd<-as.character(stopwd$stopwords())

stopwd<-gsub('no',' ',stopwd)
stopwd<-gsub('yes',' ',stopwd)
```

- Created 2 new columns **QuestionWords** and **AnswerWords** in which preprocessing of text will be performed. This is done to retain the original question and answer in the data frame.
- Removing of all the URL's, email's, and words starting with @

```
## removing the Urls
data1$questionWords<-replace_url(data1$questionWords)
data1$answerWords<-replace_url(data1$answerWords)

data1$questionWords<-gsub("\\s*[^[:space:]/]+/[^[:space:]/]+","", data1$questionWords)
data1$answerWords<-gsub("\\s*[^[:space:]/]+/[^[:space:]/]+","", data1$answerWords)

##removing email
data1$questionWords<-gsub('\\S+@\\S+', ' ', data1$questionWords)
data1$answerWords<-gsub('\\S+@\\S+', ' ', data1$answerWords)

##removing words starting with @
data1$questionWords<-gsub('@\\S+', ' ', data1$questionWords)
data1$answerWords<-gsub('@\\S+', ' ', data1$answerWords)
```

- Removing the words which contain or end with **.com, .org** and **.eu**

```
##removing words containing .com in between and does not contain http or fpt
data1$questionWords<-gsub('\\S+.com+\\s', '  ', data1$questionWords)
data1$questionWords<-gsub('\\S+.org+\\s', '  ', data1$questionWords)
data1$questionWords<-gsub('\\S+.eu+\\s', '  ', data1$questionWords)

data1$answerWords<-gsub('\\S+.com+\\s', '  ', data1$answerWords)
data1$answerWords<-gsub('\\S+.org+\\s', '  ', data1$answerWords)
data1$answerWords<-gsub('\\S+.eu+\\s', '  ', data1$answerWords)

##removing words ending with .com and does not contain http or fpt
data1$questionWords<-gsub('\\S+\\.com', '  ', data1$questionWords)
data1$questionWords<-gsub('\\S+\\.org', '  ', data1$questionWords)
data1$questionWords<-gsub('\\S+\\.eu', '  ', data1$questionWords)

data1$answerWords<-gsub('\\S+\\.com', '  ', data1$answerWords)
data1$answerWords<-gsub('\\S+\\.org', '  ', data1$answerWords)
data1$answerWords<-gsub('\\S+\\.eu', '  ', data1$answerWords)
```

- Removing any words which come after or before /

```
##removing word/ and does not contain http or fpt
data1$questionWords<-gsub('\\S+/', '  ', data1$questionWords)
data1$answerWords<-gsub('\\S+/', '  ', data1$answerWords)

##removing /word and does not contain http or fpt
data1$questionWords<-gsub('/\\S+', '  ', data1$questionWords)
data1$answerWords<-gsub('/\\S+', '  ', data1$answerWords)
```

- Removing the words who have combination of word&amp, word&word, &word

```
##removing word&amp;word nd does not contain http or fpta
data1$questionWords<-gsub('\\S+&amp;\\S+', '  ', data1$questionWords)
data1$answerWords<-gsub('\\S+&amp;\\S+', '  ', data1$answerWords)

#replace_html(data1$questionWords,FALSE)

##removing word&amp; and does not contain http or fpt
data1$questionWords<-str_replace_all(data1$questionWords, "&amp;", " ")
data1$answerWords<-str_replace_all(data1$answerWords, "&amp;", " ")

##removing word&word
data1$questionWords<-gsub('\\S+&\\S+', '  ', data1$questionWords)
data1$answerWords<-gsub('\\S+&\\S+', '  ', data1$answerWords)

##removing &word;
data1$questionWords<-gsub('&\\S+;', '  ', data1$questionWords)
data1$answerWords<-gsub('&\\S+;', '  ', data1$answerWords)
```

- Removing any combination of character and number as a word. It is generally used in serial number or model name.

```
##removing combination of character and number as a word. It is genearlly serial number or model name.
data1$questionWords<-gsub('[a-z]+[0-9]+\\S+', ' ', data1$questionWords)
data1$answerWords<-gsub('[a-z]+[0-9]+\\S+', ' ', data1$answerWords)

##removing word-number
data1$questionWords<-gsub('\\S+\\-+[0-9]+', ' ', data1$questionWords)
data1$answerWords<-gsub('\\S+\\-+[0-9]+', ' ', data1$answerWords)

##removing number-word
data1$questionWords<-gsub('[0-9]+-\\S+', ' ', data1$questionWords)
data1$answerWords<-gsub('[0-9]+-\\S+', ' ', data1$answerWords)

##removing word#
data1$questionWords<-gsub('([a-z]+#)', ' ', data1$questionWords)
data1$answerWords<-gsub('([a-z]+#)', ' ', data1$answerWords)
```

- Removing the stop words, numbers and punctuations.

```
## Removing stop words
data1$questionWords <- removeWords(data1$questionWords, stopwd)
data1$answerWords <- removeWords(data1$answerWords, stopwd)

#Removing numbers
data1$questionWords <- removeNumbers(data1$questionWords)
data1$answerWords <- removeNumbers(data1$answerWords)

#Removing punctuation mark
data1$questionWords<-removePunctuation(data1$questionWords)
data1$answerWords<-removePunctuation(data1$answerWords)
```

- Removing single letter words, whitespace and certain word which were not removed by the stop words.

```
#remove single letter words
data1$questionWords <- rm_nchar_words(data1$questionWords, "1,1")
data1$answerWords <- rm_nchar_words(data1$answerWords, "1,1")

#removing whitespace
data1$questiDonWords <- stripWhitespace(data1$questionWords)
data1$answerWords <- stripWhitespace(data1$answerWords)

# replacing certain words
data1$questionWords<-str_replace_all(data1$questionWords, " wi fi ", " wifi ")
data1$questionWords<-str_replace_all(data1$questionWords, " aca ", " ")
data1$questionWords<-str_replace_all(data1$questionWords, " & ", "and")
data1$questionWords<-str_replace_all(data1$questionWords, " aaaaarge ", " ")

data1$answerWords<-str_replace_all(data1$answerWords, " wi fi ", " wifi ")
data1$answerWords<-str_replace_all(data1$answerWords, " aca ", " ")
data1$answerWords<-str_replace_all(data1$answerWords, " & ", "and")
data1$answerWords<-str_replace_all(data1$answerWords, " aaaaarge ", " ")
```

- Word stemming using the "**textstem**" package

```
## word stemming
library(textstem)
data1$questionWords<-textstem::lemmatize_strings(data1$questionWords)
data1$answerWords<-textstem::lemmatize_strings(data1$answerWords)
```

- Created a word dictionary which contains wrong word and correct words using **hunspell** package.
  - Collecting list of words with incorrect spelling

```
# vector of words to replace
wrong <- unlist(hunspell(data1$questionWords))
```

  - Creating a data frame of incorrect word with its first suggested word as correct word.

```
# vector of the first suggested words
correct <- data.frame(wrong,sapply(wrong, function(x) hunspell_suggest(x)[[1]][1]))
colnames(correct)<-c("wrong","correct")
```

  - Repeating the same for the answer column

```
# vector of words to replace
wrongA <- unlist(hunspell(data1$answerWords))

# vector of the first suggested words
correctA <- data.frame(wrongA,sapply(wrongA, function(x) hunspell_suggest(x)[[1]][1]))
colnames(correctA)<-c("wrong","correct")
```

  - Below mentioned words need not be replaced as either they are name of company or name of the product and also combining the dictionary formed from question and answer into 1 dictionary.

```
correct2<-sqldf("select distinct wrong,correct from correct where wrong<>lower(correct)
                and length(lower(correct))>3
                and wrong not in ('korea','vesa','tv','lcd','visio','tvs','ereader','gmail','froyo','bootable','roms',
                'hotspots','ereaders','decrypt','calibri','nikons','quora','sony','peru','photoplaceonlinecom','iphones',
                'jitter','httplandelcom','teleconverter','macbooks','airbooks','ipads','innerfidelitycom','flickr','modi',
                'phillips','mids','wlens','stationsis','lowepro','unshielded','wi','eq','fi','talkabout','talkabouts','hulu',
                'ebooks','ebook','epub','hd','vhs','ethernet','mh','youtube','xbox','motorola','asus','acer','ipad','xp','cd',
                'pc','usb','vizio','wifi','gxt','magnavox','vx','macbook','airbook','gb','aaabu','aac','aas','Aastra','aaxa',
                'abel','abesofmaine','abf','abit','absorbers','abu')

                 union
                select distinct wrong,correct from correctA where wrong<>lower(correct)
                and length(lower(correct))>3
                and wrong not in ('korea','vesa','tv','lcd','visio','tvs','ereader','gmail','froyo','bootable','roms','hotspots',
                'ereaders','decrypt','calibri','nikons','quora','sony','peru','photoplaceonlinecom','iphones','jitter','httplandelcom',
                'teleconverter','macbooks','airbooks','ipads','innerfidelitycom','flickr','modi','phillips','mids','wlens','stationsis',
                'lowepro','unshielded','wi','eq','fi','talkabout','talkabouts','hulu','ebooks','ebook','epub','hd','vhs','ethernet','mh',
                'youtube','xbox','motorola','asus','acer','ipad','xp','cd','pc','usb','vizio','wifi','gxt','magnavox','vx','macbook',
                'airbook','gb','aaabu','aac','aas','Aastra','aaxa','abel','abesofmaine','abf','abit','absorbers','abu')

                ")
```

  - Calculating the weight between two words by **jarowinkler** distance. It shows how much two words are close. If distance =1 it means wrong and correct word are same word.

```
library(RecordLinkage)

correct2<-mutate(correct2, weight = jarowinkler(correct,wrong))

summary(correct2)
```

- Manually correcting the incorrect words as they were provided with incorrect words and also increasing their weight to 91.

```
correct2<-sqldf(c("update correct2 set correct='sony' where wrong='soni'","select * from correct2"))
correct2<-sqldf(c("update correct2 set correct='netflix' where wrong='netfilx'","select * from correct2"))
correct2<-sqldf(c("update correct2 set correct='of the' where wrong='ofnthe'","select * from correct2"))
correct2<-sqldf(c("update correct2 set correct='cosine' where wrong='cosina'","select * from correct2"))
correct2<-sqldf(c("update correct2 set correct=' ' where wrong='arrrrgh'","select * from correct2"))


correct2<-sqldf(c("update correct2 set correct=' ' where wrong='aaaaarge'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='aaaaarge'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='arrrrgh'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='nivida'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='nvida'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='nvidea'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='nvidida'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='nvida'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='ofnthe'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='hewlettpackward'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='itsawrap'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='abccbs'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='blueraydisc'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='boseenjoy'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='bsdvms'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='daysony'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='cosina'","select * from correct2"))

correct2<-sqldf(c("update correct2 set weight=91 where wrong='dlplink'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='samsund'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='samsungs'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='soni'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='netfilx'","select * from correct2"))
correct2<-sqldf(c("update correct2 set weight=91 where wrong='arrrrgh'","select * from correct2"))
```

- Creating a cutoff value (**0.971428571428571**) to keep most significant incorrect and correct words in data dictionary and final dictionary is created and stored in correct3.csv.

```
## final table of wrong and correct words
correct3<-sqldf("select * from correct2 where weight>=0.971428571428571")

correct3$correct<-tolower(correct3$correct)
```

- Replacing the wrong words with correct words created in data dictionary for questionWords and answerwords using the **Findreplace** function in **DataCombine** package.

```
library(DataCombine)
## Correcting the incorrect word
data1<-FindReplace(data1, Var="questionWords", correct3, from = "wrong", to = "correct" ,
                   exact = FALSE, vector = FALSE)


data1<-FindReplace(data1, Var="answerWords", correct3, from = "wrong", to = "correct" ,
                   exact = FALSE, vector = FALSE)
```

- This whole text analysis is converted into a function (**CleanText_Analysis**) except for creating data dictionary. Data dictionary already created is used in this function.

rownum	asin	unixTime	questiontypeOE	question	answerType	answer	questionWords
1	0594033926	2013-12-27	0	Is this cover the one that fits the old nook color? Which I bel...	Y	Yes this fits both the nook color and the same-shaped nook ...	cover fit nook color
2	0594033926	2015-01-05	0	Does it fit Nook GlowLight?	N	No. The nook color or color tablet	fit nook glow light
3	0594033926	2014-12-29	1	Would it fit Nook 1st Edition? 4.9in x 7.7in ?	NaN	I don't think so. The nook color is 5 x 8 so not sure anything...	fit nook st edition in in
4	0594033926	2014-12-29	0	Will this fit a Nook Color that's 5 x 8?	Y	yes	fit nook color
5	0594033926	2015-02-10	0	will this fit the Samsung Galaxy Tab 4 Nook 10.1	N	No, the tab is smaller than the 'color'	fit hamsung galaxy tab nook

answerWords	QuestionLength	AnswerLength	questionWCount	answerWCount	questionAve_Sentiment	answerAve_Sentiment	questionsd	answersd	sentimentQ
yes fit nook color shape nook ttablet	75	65	16	13	0.00000000	0.2218800785	0.000000000	0.183333300	0
no nook color color ttablet	27	34	5	7	0.17888544	0.0000000000	0.102062100	0.000000000	0
nook color not small stay lock close	46	112	11	25	0.08911020	-0.3038635721	0.115470054	0.393750000	0
yes	40	3	11	1	0.14142136	0.8000000000	0.102062100	0.183333300	0
no tab small color	48	39	11	8	0.14142136	0.0000000000	0.102062100	0.183333300	0

vi) Finding the value of ? in AnswerType column

- All the next steps of processing the dataset will be done in “DATA3”.
- Changing the Unix time format into YYYY/MM/DD, factorizing QuestionTypeOE, AnswerType and Asin.

```
data3$unixTime<-as.Date(data3$unixTime , format = "%Y-%m-%d")
data3$questiontypeOE<-as.factor(data3$questiontypeOE)
data3$answerType<-as.factor(data3$answerType)
data3$asin<-as.factor(data3$asin)
```

- Divided the dataset into 2 parts
  - **Data220** containing answertype where “**?**” value is **not present**
  - **Data2201** containing answertype where “**?**” value is **present**

```
## Diving the data into dataset without answertype=?
data220<-sqldf("select rownum, asin,unixTime,questiontypeOE,question,answer,questionWords,answerWords,QuestionLength, AnswerLength,
                    questionWCount,answerWCount,questionAve_Sentiment,answerAve_Sentiment,questionsd,answersd,
                    negativeQ,positiveQ,sentimentQ,negativeA, positiveA,sentimentA,answerType
            from data3 where answerType<>'?'
             order by rownum")

## Diving the data into dataset with answertype=?
data2201<-sqldf("select rownum, asin,unixTime,questiontypeOE,question,answer,questionWords,answerWords,QuestionLength, AnswerLength,
                    questionWCount,answerWCount,questionAve_Sentiment,answerAve_Sentiment,questionsd,answersd,
                    negativeQ,positiveQ,sentimentQ,negativeA, positiveA,sentimentA,answerType
            from data3 where answerType='?'
              order by rownum")

data220$answerType<-as.factor(as.character(data220$answerType))
data2201$answerType<-as.factor(as.character(data2201$answerType))
```

- Now creating a new dataset with only the columns that are needed.

```
data221<-sqldf("select unixTime,questiontypeOE,QuestionLength, AnswerLength,
                    questionWCount,answerWCount,questionAve_Sentiment,answerAve_Sentiment,questionsd,answersd,
                   negativeQ,positiveQ,sentimentQ,negativeA, positiveA,sentimentA,answerType
               from data220")
```

- Changing the variables of answer type : NO to N, YES to Y, NaN to NotApp.

```
data221<-sqldf(c("update data221 set answerType='No' where answerType='N'","select * from data221"))
data221<-sqldf(c("update data221 set answerType='Yes' where answerType='Y'","select * from data221"))
data221<-sqldf(c("update data221 set answerType='NotApp' where answerType='NaN'","select * from data221"))

data221$answerType<-as.factor(as.character(data221$answerType))
```

- Creating a normalize function and normalizing all the columns.

```
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

data221$QuestionLength<-normalize(data221$QuestionLength)
data221$AnswerLength<-normalize(data221$AnswerLength)
data221$questionWCount<-normalize(data221$questionWCount)
data221$answerWCount<-normalize(data221$answerWCount)

data221$questionAve_Sentiment<-normalize(data221$questionAve_Sentiment)
data221$answerAve_Sentiment<-normalize(data221$answerAve_Sentiment)
data221$questionsd <-normalize(data221$questionsd )
data221$answersd<-normalize(data221$answersd)

data221$negativeQ<-normalize(data221$negativeQ)
data221$positiveQ<-normalize(data221$positiveQ)
data221$sentimentQ <-normalize(data221$sentimentQ )
data221$negativeA<-normalize(data221$negativeA)
data221$positiveA<-normalize(data221$positiveA)
data221$sentimentA<-normalize(data221$sentimentA)
```

- Checking the distribution in original and partitioned data

```
#Checking distibution in organl data and partitioned data
prop.table(table(data220$answerType)) * 100
```

- Now we will do sampling using the Caret package.
- Dividing the data into train and test datasets on basis of AnswerType.

```
seed<-1321

library(caret)
set.seed(seed)

indexes <- createDataPartition(data221$answerType, times = 1,
                               p = 0.8, list = FALSE)

train <- data221[indexes,]
test <- data221[-indexes,]
```

- Here we will use all the different types of sampling: Normal, Down, Up and Smote and select the one which has the best output.

```
ctrl <- trainControl(method = "cv",number = 5,
                     summaryFunction = multiClassSummary,
                     classProbs = TRUE
                     )

model_rf1 <- caret::train(answerType ~ .,
                          data = train,
                          method = "rf",
                          metric="ROC",
                          trControl = ctrl
                          )

ctrl$sampling <- "down"
model_rf_down <- caret::train(answerType ~ .,
                              data = train,
                              method = "rf",
                              metric="ROC",
                              trControl = ctrl)

ctrl$sampling <- "up"
model_rf_up <- caret::train(answerType ~ .,
                            data = train,
                            method = "rf",
                            metric="ROC",
                            trControl = ctrl)
```

```
ctrl$sampling <- "smote"
model_rf_smote <- caret::train(answerType ~ .,
                               data = train,
                               method = "rf",
                               metric="ROC",
                               trControl = ctrl)
```

- Preparing a confusion matrix of all the models

```
caret::confusionMatrix(predict(model_rf1,test[1:(length(test)-1)]), test$answerType)
caret::confusionMatrix(predict(model_rf_down,test[1:(length(test)-1)]),test$answerType)
caret::confusionMatrix(predict(model_rf_up,test[1:(length(test)-1)]),test$answerType)
caret::confusionMatrix(predict(model_rf_smote,test[1:(length(test)-1)]),test$answerType)
```

- Printing all the values on which to select the model.

```
print(model_rf1)
print(model_rf_down)
print(model_rf_up)
print(model_rf_smote)
```

- Smote was selected because it had the most balanced accuracy (Comparison of data sampling.xlsx)

Smote

Reference			
Prediction	No	NotApp	Yes
No	1047	0	649
NotApp	0	8953	0
Yes	157	0	2552

Overall Statistics

Accuracy	0.9397
95% CI	(0.9355, 0.9436)
No Information Rate	0.6702
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.877
Mcnemar's Test P-Value	NA

Statistics by Class:

	Class: No	Class: NotApp	Class: Yes
Sensitivity	0.86960	1.0000	0.7973
Specificity	0.94660	1.0000	0.9845
Pos Pred Value	0.61733	1.0000	0.9420
Neg Pred Value	0.98654	1.0000	0.9391
Prevalence	0.09013	0.6702	0.2396
Detection Rate	0.07838	0.6702	0.1910
Detection Prevalence	0.12697	0.6702	0.2028
Balanced Accuracy	0.90810	1.0000	0.8909

Normal

Reference			
Prediction	No	NotApp	Yes
No	833	0	149
NotApp	0	8953	0
Yes	371	0	3052

Overall Statistics

Accuracy	0.9611
95% CI	(0.9577, 0.9643)
No Information Rate	0.6702
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.9194
Mcnemar's Test P-Value	NA

Statistics by Class:

	Class: No	Class: NotApp	Class: Yes
Sensitivity	0.69186	1.0000	0.9535
Specificity	0.98774	1.0000	0.9635
Pos Pred Value	0.84827	1.0000	0.8916
Neg Pred Value	0.97002	1.0000	0.9850
Prevalence	0.09013	0.6702	0.2396
Detection Rate	0.06236	0.6702	0.2285
Detection Prevalence	0.07351	0.6702	0.2563
Balanced Accuracy	0.83980	1.0000	0.9585

Down

Reference			
Prediction	No	NotApp	Yes
No	984	0	462
NotApp	0	8953	0
Yes	220	0	2739

Overall Statistics

Accuracy	0.9489
95% CI	(0.9451, 0.9526)
No Information Rate	0.6702
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.8954
Mcnemar's Test P-Value	NA

Statistics by Class:

	Class: No	Class: NotApp	Class: Yes
Sensitivity	0.81728	1.0000	0.8557
Specificity	0.96199	1.0000	0.9783
Pos Pred Value	0.68050	1.0000	0.9257
Neg Pred Value	0.98153	1.0000	0.9556
Prevalence	0.09013	0.6702	0.2396
Detection Rate	0.07366	0.6702	0.2050
Detection Prevalence	0.10825	0.6702	0.2215
Balanced Accuracy	0.88963	1.0000	0.9170

up

Reference			
Prediction	No	NotApp	Yes
No	829	0	167
NotApp	0	8953	0
Yes	375	0	3034

Overall Statistics

Accuracy	0.9594
95% CI	(0.9559, 0.9627)
No Information Rate	0.6702
P-Value [Acc > NIR]	< 2.2e-16
Kappa	0.916
Mcnemar's Test P-Value	NA

Statistics by Class:

	Class: No	Class: NotApp	Class: Yes
Sensitivity	0.68854	1.0000	0.9478
Specificity	0.98626	1.0000	0.9631
Pos Pred Value	0.83233	1.0000	0.8900
Neg Pred Value	0.96967	1.0000	0.9832
Prevalence	0.09013	0.6702	0.2396
Detection Rate	0.06206	0.6702	0.2271
Detection Prevalence	0.07456	0.6702	0.2552
Balanced Accuracy	0.83740	1.0000	0.9555

Fig 4.1 *shows the output of different data sampling*

- Now time to select which model is accurate for the classification of AnswerType.
- Different models like GLMNET, SVM RADIAL, KNN, NAÏVE BAYES, CART, C5.0, BAGGED CART and RANDOM FOREST.

```
ctrl <- trainControl(method="repeatedcv", number=10, repeats=3,
                     summaryFunction = multiClassSummary,
                     classProbs=TRUE,
                     savePredictions = TRUE,
                     sampling <- "smote"
                    )

metric <- "accuracy"
seed<-1234


# GLMNET
set.seed(seed)
fit.glmnet <- caret::train(answerType~., data=train, method="glmnet", metric=metric, trControl=ctrl)
# SVM Radial
set.seed(seed)
fit.svmRadial <- caret::train(answerType~., data=train, method="svmRadial", metric=metric, trControl=ctrl, fit=FALSE)
# kNN
set.seed(seed)
fit.knn <- caret::train(answerType~., data=train, method="knn", metric=metric, trControl=ctrl)
# Naive Bayes
set.seed(seed)
fit.nb <- caret::train(answerType~., data=train, method="nb", metric=metric, trControl=ctrl)
# CART
set.seed(seed)
fit.cart <- caret::train(answerType~., data=train, method="rpart", metric=metric, trControl=ctrl)
# C5.0
set.seed(seed)
fit.c50 <- caret::train(answerType~., data=train, method="C5.0", metric=metric, trControl=ctrl)
# Bagged CART
set.seed(seed)
fit.treebag <- caret::train(answerType~., data=train, method="treebag", metric=metric, trControl=ctrl)
# Random Forest
set.seed(seed)
fit.rf <- caret::train(answerType~., data=train, method="rf", metric=metric, trControl=ctrl)
```

- Storing the results

```
results <- resamples(list( glmnet=fit.glmnet,
                           svm=fit.svmRadial, knn=fit.knn, nb=fit.nb, cart=fit.cart, c50=fit.c50,
                           bagging=fit.treebag, rf=fit.rf))
# Table comparison
summary(results)
```

- Model RANDOM FOREST was selected.

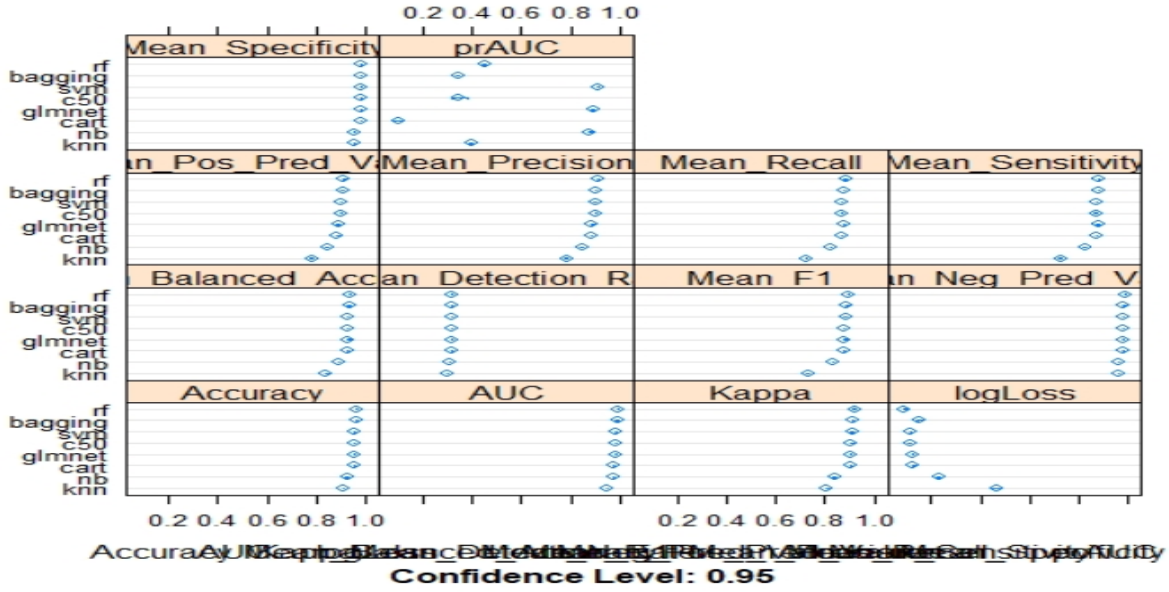


Fig 4.2 *shows comparison of different methods*

- Variable importance of Random forest

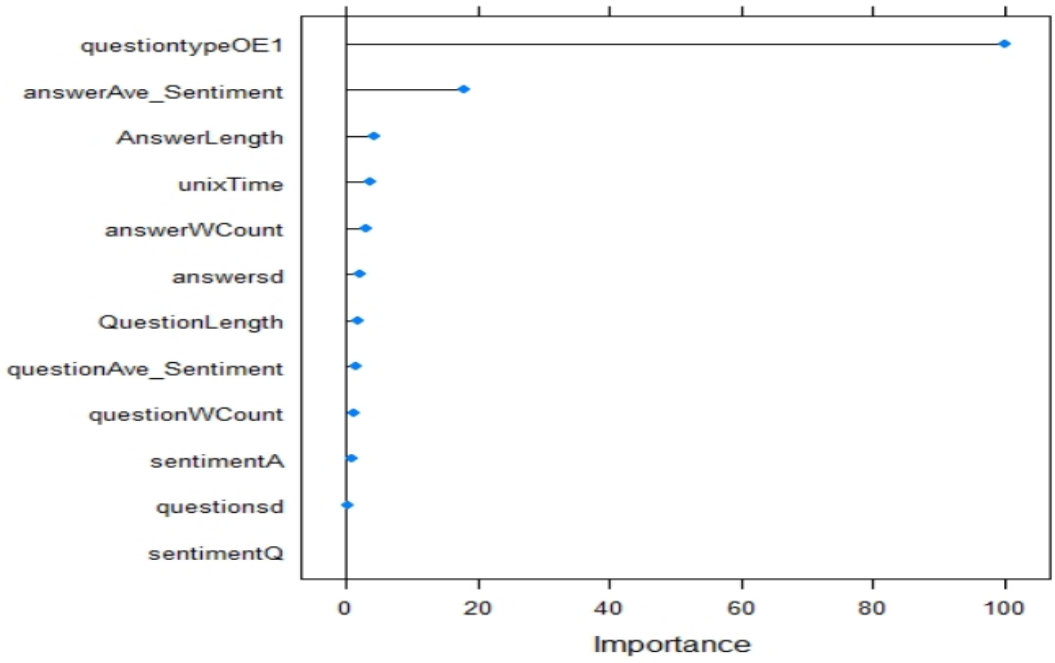


Fig 4.3 *shows the columns and their importance*

- Got the accuracy of **0.9611** or **96.11%**

- Now implementing these models on the test dataset

```
data2202<-data2201


data2202$questionAve_Sentiment<-normalize(data2202$questionAve_Sentiment)
data2202$answerAve_Sentiment<-normalize(data2202$answerAve_Sentiment)
data2202$questionsd <-normalize(data2202$questionsd )
data2202$answersd<-normalize(data2202$answersd)

pred<-predict(model_rf2,data2202)

data2201$answerType<-pred



FinalData1<- sqldf("select rownum, unixTime,questiontypeOE,question,answer,questionWords,answerWords,QuestionLength, AnswerLength,
                        questionWCount,answerWCount,questionAve_Sentiment,answerAve_Sentiment,questionsd,answersd,
                        negativeQ,positiveQ,sentimentQ,negativeA, positiveA,sentimentA,answerType
                   from data220
                   union
                   select rownum, unixTime,questiontypeOE,question,answer,questionWords,answerWords,QuestionLength, AnswerLength,
                        questionWCount,answerWCount,questionAve_Sentiment,answerAve_Sentiment,questionsd,answersd,
                        negativeQ,positiveQ,sentimentQ,negativeA, positiveA,sentimentA,answerType
                   from data2201
                   ")
FinalData<-FinalData1
FinalData<-sqldf(c("update FinalData set answerType='No' where answerType='N'","select * from FinalData"))
FinalData<-sqldf(c("update FinalData set answerType='Yes' where answerType='Y'","select * from FinalData"))
FinalData<-sqldf(c("update FinalData set answerType='NotApp' where answerType='NaN'","select * from FinalData"))

summary(FinalData)
```

- The FINAL DATA looks like this: FinalData.csv
- Here are some visualization of the dataset

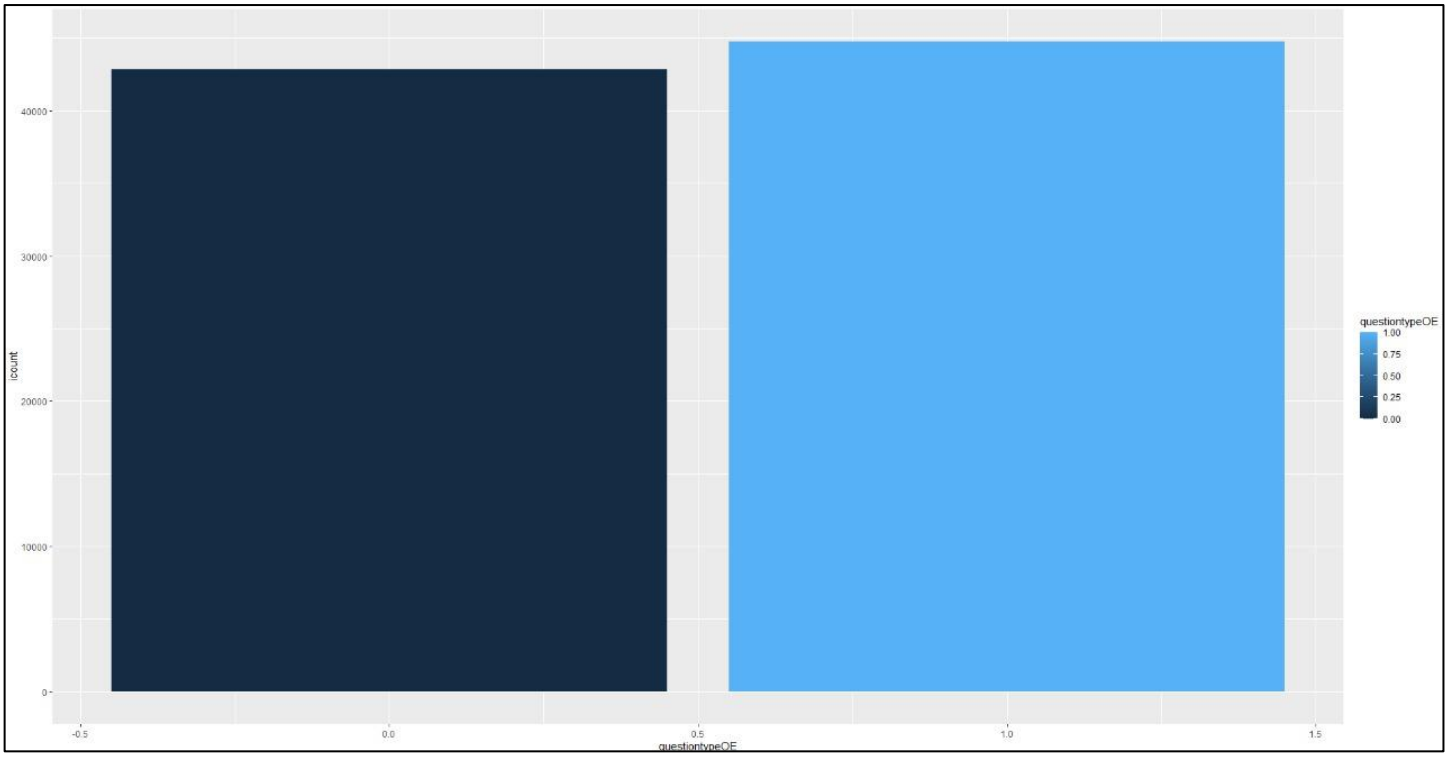


Fig 4.4 *shows the columns and their importance*

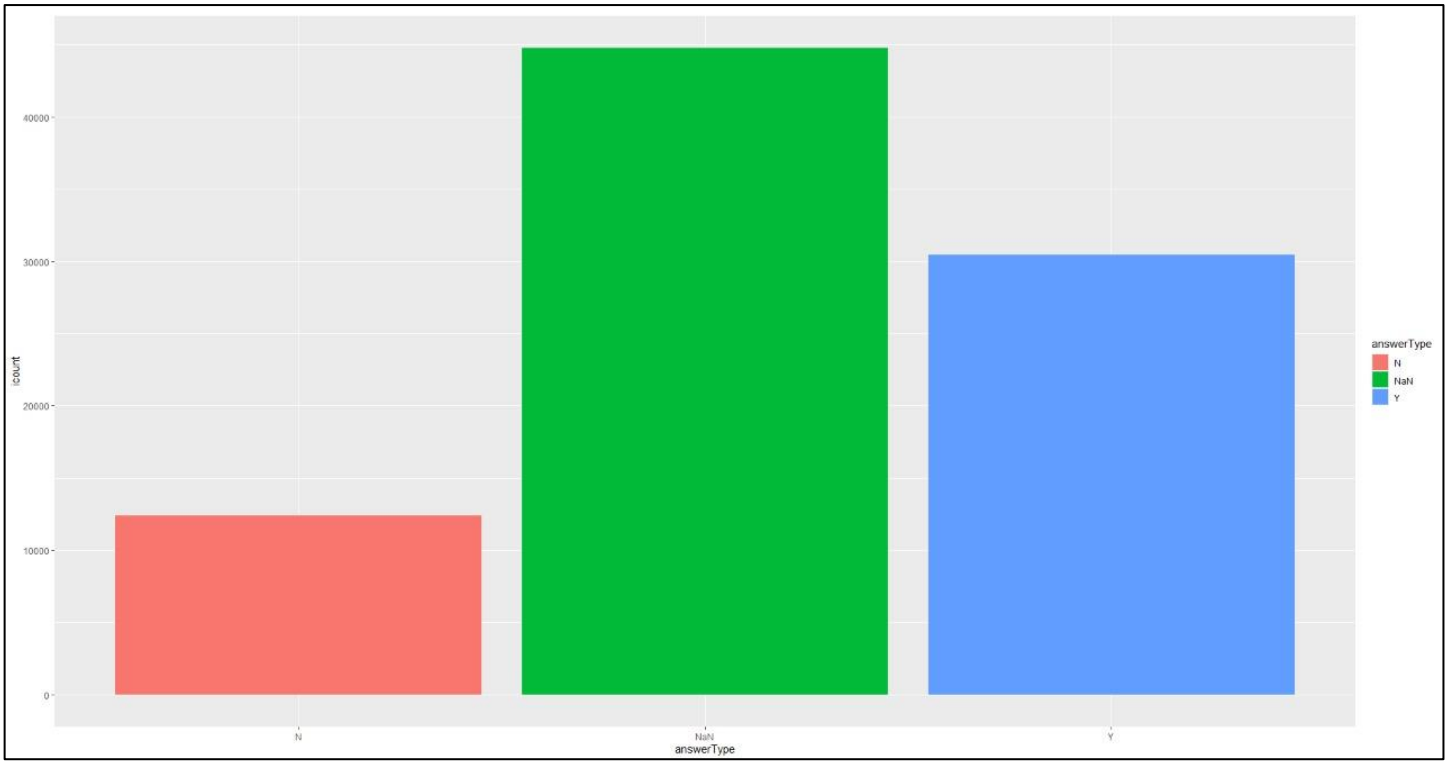


Fig 4.5 *shows the histogram of the column answertype*

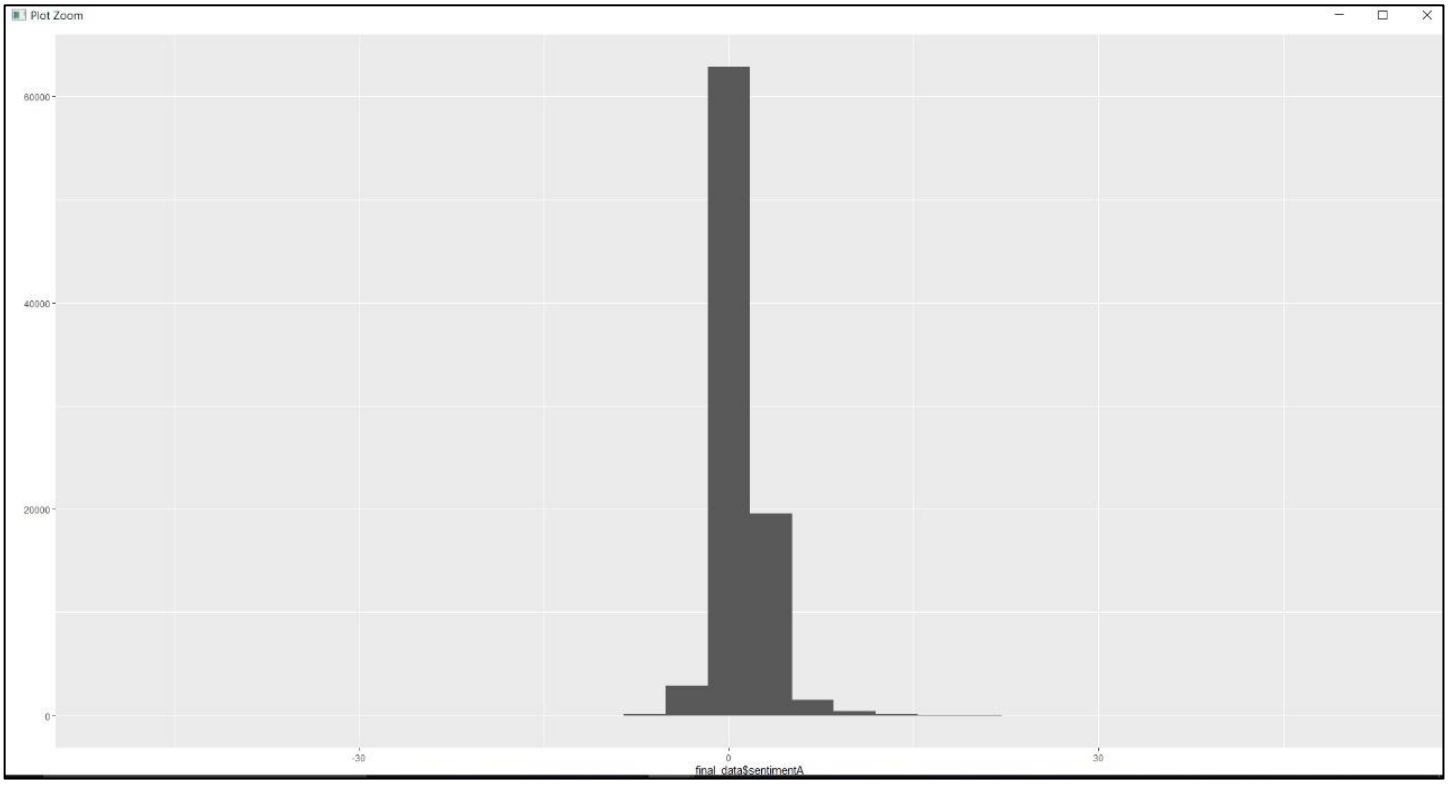


Fig 4.6 *shows the histogram of the column SENTIMENTA*

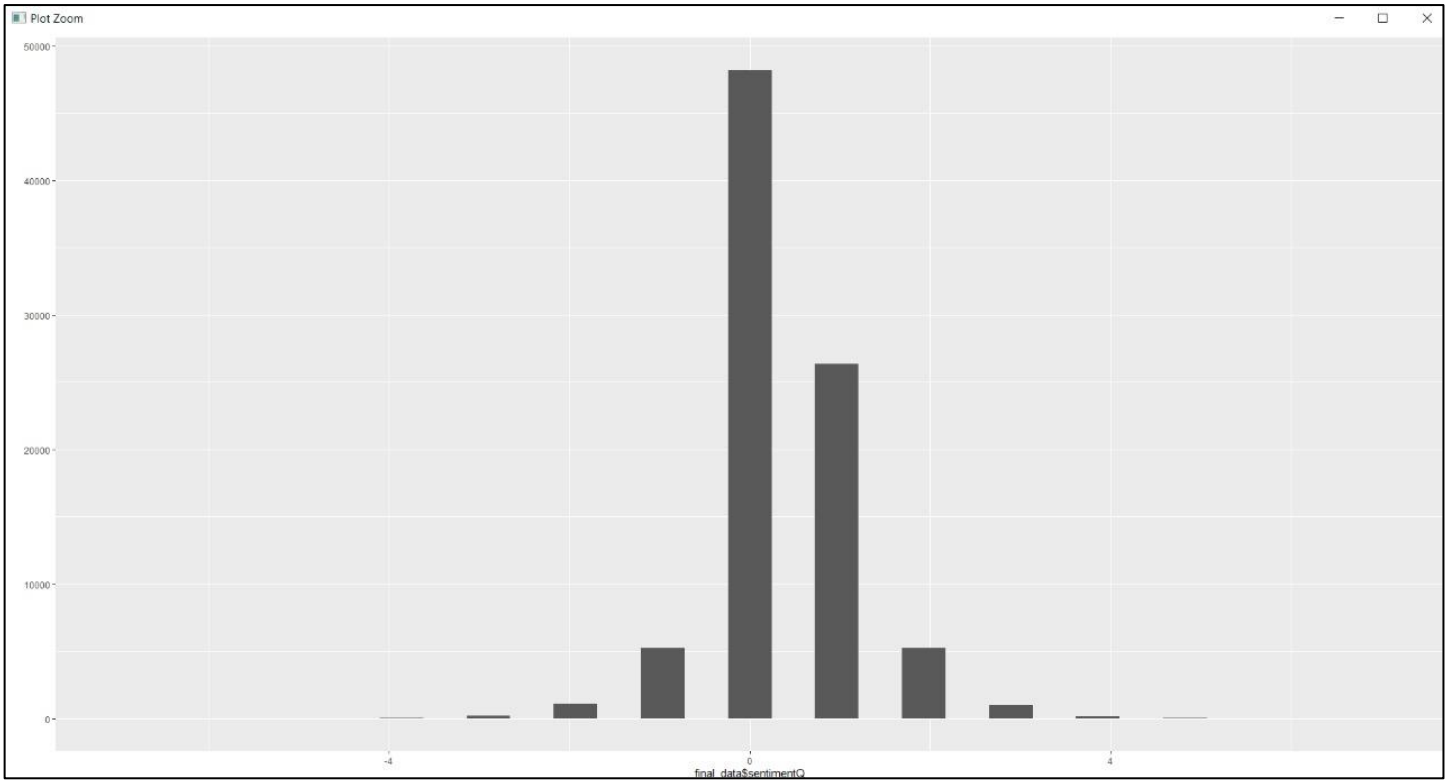


Fig 4.7 *shows the histogram of the column SENTIMENTQ*

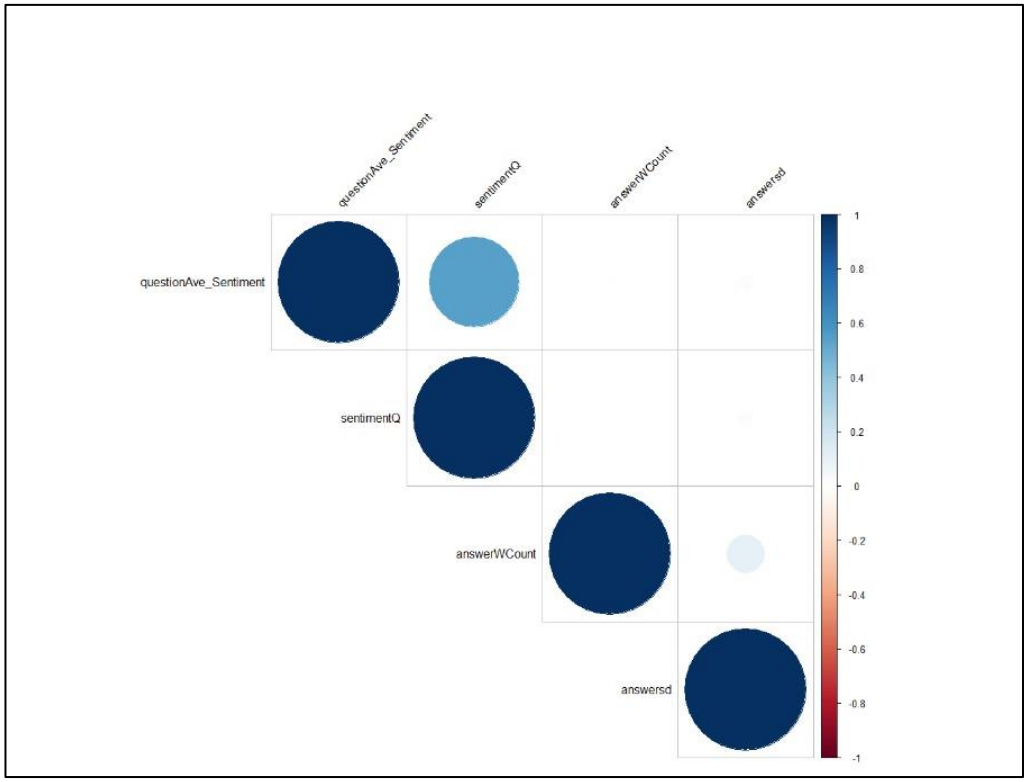


Fig 4.8 *shows the sentimental analysis of question and answers*

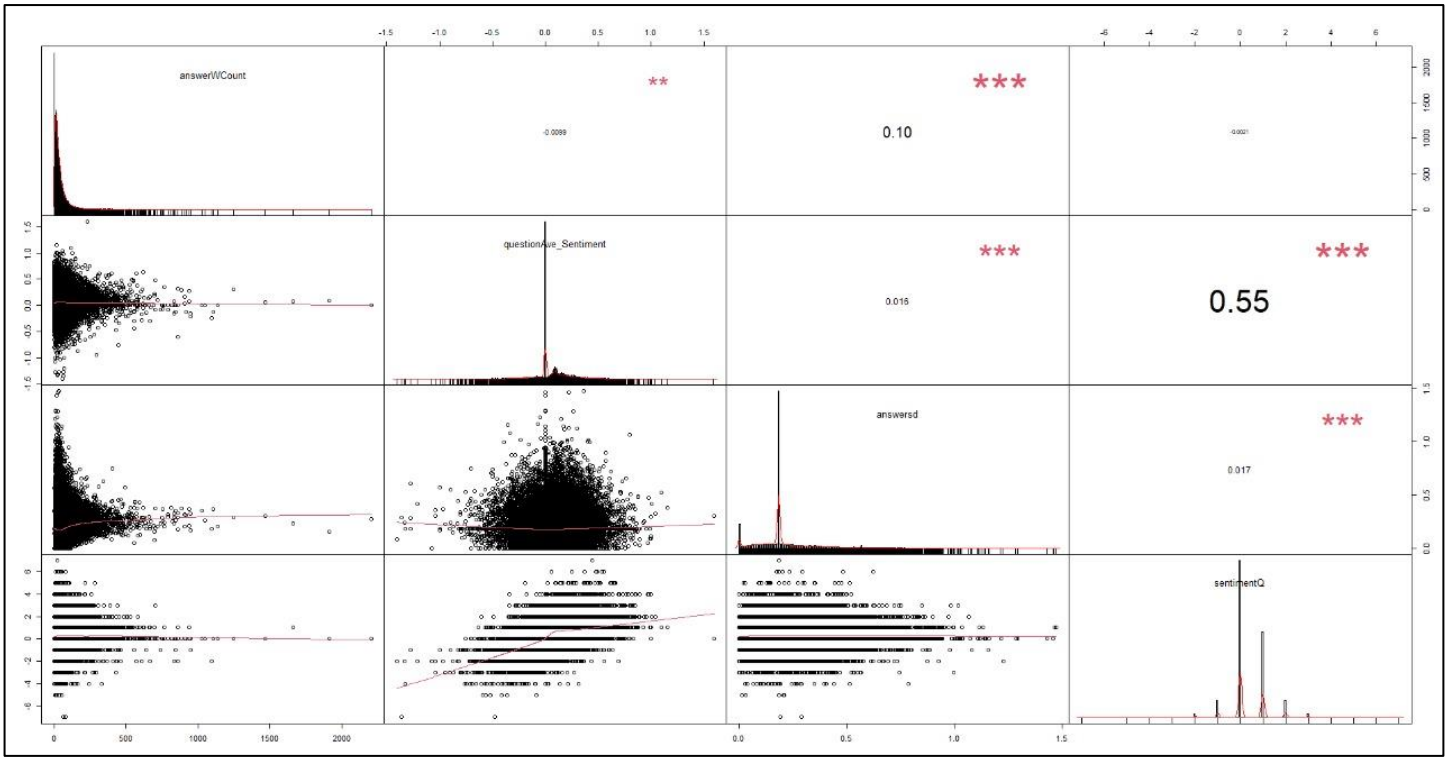


Fig 4.9 *shows co-relation plot of all numeric columns*

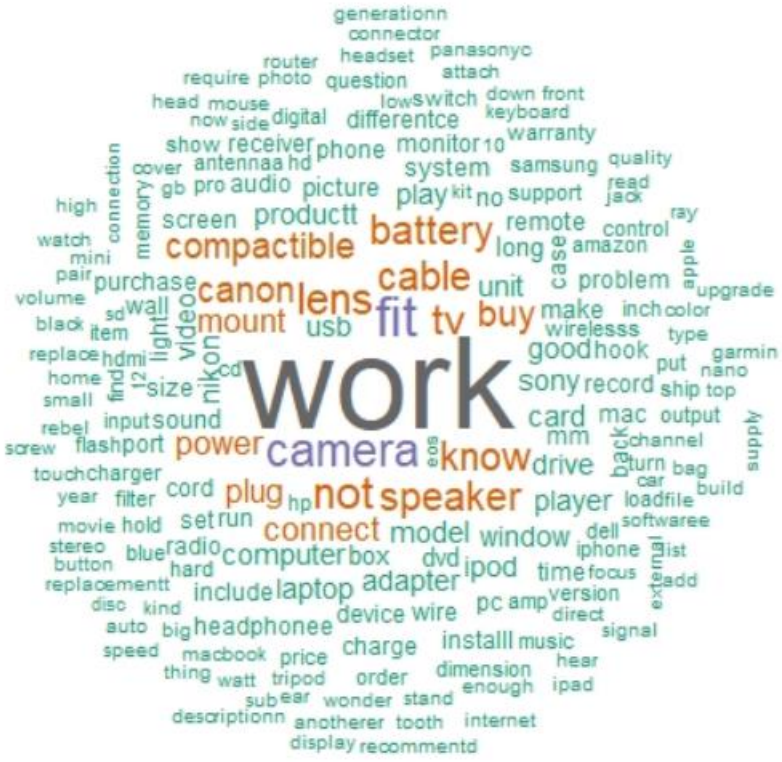


Fig 4.10 *shows a word cloud of the question column*

6. Model Building

- Loading the necessary libraries and the Final Data.

```
library("XML")
library("sqldf")
library("dplyr")
library("tm")
library("tidyverse")
library("tidytext")
library("hunspell")

FinalData=read.csv('FinalData.csv', stringsAsFactors = FALSE)

str(FinalData)
summary(FinalData)

#FinalData$rownum<-as.numeric(FinalData$rownum)

FinalData$unixTime<-as.Date(FinalData$unixTime , format = "%Y-%m-%d")
#stopwords_en <- tibble("word" = stopwords("en", source  = "smart"))

FinalData <- FinalData %>%
  mutate(rowIndex=as.numeric(row.names(.)))

str(FinalData)
```

- Loading stop words to correct user input and removing yes and no from the list

```
## Loading stopwords to correct user input
stopwd <- read.table("stop.txt")
stopwd <-as.character(stopwd$V1)

stopwd<-gsub('no',' ',stopwd)
stopwd<-gsub('yes',' ',stopwd)
```

- Loading correct spellcheck for user input

```
## Loading correct spell check for user input
correct3=read.csv('correct3.csv', stringsAsFactors = FALSE)
```

- First the model will be explained individually and then the screenshot of the whole code is inserted
- The first thing to do is convert the Question Word column into list and get its length

```
QuestionList <- as.list(FinalData$questionWords)
 QuestionLength <- length(QuestionList)
```

- As the model is created in the form of a function, it is necessary to define the function which is going to be based on the USER QUESTION

```
QuestionSearch <- function(UserQuestion)
```

- The first thing the function is going to do is do a text analysis of the user question and do a spell check of the same using the CleanText_analysis function that was created above and the correct3 dataset as well

```
UserQuestion<-CleanText_Analysis(UserQuestion,correct3)
```

- Then it will combine the user question and question list into one called as total questions
- Then the function will create a corpus and remove the white space and stem the document

```
QuestionCorpus <- VCorpus(TotalQuestions) %>%
  tm_map(stemDocument) %>%
  tm_map(stripWhitespace)
```

- Now it will create a term document matrix of the given corpus

```
term.doc.matrix.stm <- TermDocumentMatrix(QuestionCorpus,
                                          control=list(
                                            weighting=function(x) weightSMART(x,spec="ltc"),
                                            wordLengths=c(1,Inf)))
```

- The next part of the code will first get converted into data frame by the tidy function and get grouped by the number of documents as per the term document matrix.
- The mutate function is used to create a new variable from a data set called vtrLen and it

```
TotalQuestions <- VectorSource(c(QuestionList, UserQuestion))
```

will select terms based on most occurring terms. This all process will get stored in a variable called term.doc.matrix.

```
term.doc.matrix <- tidy(term.doc.matrix.stm) %>%
  group_by(document) %>%
  mutate(vtrLen=sqrt(sum(count^2))) %>%
  mutate(count=count/vtrLen) %>%
  ungroup() %>%
  select(term:count)
```

- The next code will take the term.doc.matrix mutate it, and filter it

```
docMatrix <- term.doc.matrix %>%
  mutate(document=as.numeric(document)) %>%
  filter(document<QuestionLength+1)
QuestionMatrix <- term.doc.matrix %>%
  mutate(document=as.numeric(document)) %>%
  filter(document>=QuestionLength+1)
```

- The output will be in QuestionMatrix will now find terms of question and answers based on cosine similarity and return the top 5 question and answer in Worsearch

```
Wordsearch <- docMatrix %>%
  inner_join(QuestionMatrix,by=c("term"="term"),
             suffix=c(".doc",".query")) %>%
  mutate(termScore=round(count.doc*count.query,4)) %>%
  group_by(document.query,document.doc) %>%
  summarise(Score=sum(termScore)) %>%
  filter(row_number(desc(Score))<=5) %>%
  arrange(desc(Score))%>%
  left_join(FinalData,by=c("document.doc"="rowIndex")) %>%
  ungroup() %>%
  select(question,answer) %>%
  data.frame()


return(Wordsearch)
```

- So, the whole code/model looks something like this

```
QuestionSearch <- function(UserQuestion) {

  UserQuestion<-CleanText_Analysis(UserQuestion,correct3)

 TotalQuestions <- VectorSource(c(QuestionList, UserQuestion))


 QuestionCorpus <- VCorpus(TotalQuestions) %>%
   tm_map(stemDocument) %>%
   tm_map(stripWhitespace)


 term.doc.matrix.stm <- TermDocumentMatrix(QuestionCorpus,
                                           control=list(
                                             weighting=function(x) weightSMART(x,spec="ltc"),
                                             wordLengths=c(1,Inf)))


 # QuestionSearch <- function(UserQuestion) {
 term.doc.matrix <- tidy(term.doc.matrix.stm) %>%
   group_by(document) %>%
   mutate(vtrLen=sqrt(sum(count^2))) %>%
   mutate(count=count/vtrLen) %>%
   ungroup() %>%
   select(term:count)

 docMatrix <- term.doc.matrix %>%
   mutate(document=as.numeric(document)) %>%
   filter(document<QuestionLength+1)
 QuestionMatrix <- term.doc.matrix %>%
   mutate(document=as.numeric(document)) %>%
   filter(document>=QuestionLength+1)




 Wordsearch <- docMatrix %>%
   inner_join(QuestionMatrix,by=c("term"="term"),
              suffix=c(".doc",".query")) %>%
   mutate(termScore=round(count.doc*count.query,4)) %>%
   group_by(document.query,document.doc) %>%
   summarise(Score=sum(termScore)) %>%
   filter(row_number(desc(Score))<=5) %>%
   arrange(desc(Score))%>%
   left_join(FinalData,by=c("document.doc"="rowIndex")) %>%
   ungroup() %>%
   select(question,answer) %>%
   data.frame()


 return(Wordsearch)

}
```

- The output of the function is

```
QuestionSearch("sony")
```

```
[1] "Question: Sony MDR-7506"
[1] "Answer: i do not know what the question is, but, the straps worked well enough to hold my manfrotto tripod and manfrotto ball head to the bottom of my manfrotto camera back pack, which the items that it was holding was about 5 l
bs. i do hope this answers your questions, if not, then do please feel free to contact me again."
[1] "Question: Anyone tried this with the Sony HX50V?"
[1] "Answer: I use it with the Sony RX100 and it fits perfectly. If the specs are similar, it should work well. The RX100 doesn't have room to spare, though--it's a comfortable but snug fit. Hope this helps!"
[1] "Question: need help with my sony dw790: Is there something off here?"
[1] "Answer: Yup I HAD that problem as well when I was using RCA inputs but when I changed to Digital Optical, I found that the volume was way too loud and had to turn it down a lot just to fit viewing need. Well, the way I have my s
etup with the same HTIB(Home Theater in Box) is: PS3&amp;gt;Reciever via Digital Optical for Audio PS3&amp;gt;TV via Component(Plan to upgrade to HDMI once the cable comes) TV&amp;gt;Reciever via RCA(Red and White plugs) For yours specifically C
astellanos, I think we need a bit more info on the TV and the Toshiba, like what kind of Inputs and Outputs are on the back of both products or if not that way, give a link to see information on those items. Hopefully we can get this
 straighten out as with others in the same boat."
[1] "Question: A30 w/ a Sony KDL40V2500?"
[1] "Answer: TheA-30 is definitely a better bet. Just received one 2 weeks ago and no regrets at all. Even upconverts regular dvd's beautifully."
[1] "Question: Has anyone tried this on a Sony a99?"
[1] "Answer: No"
>
```

- We have now built the model successfully.

7. Model Deployment

- The model is going to be deployed in R Shiny
- The libraries needed for deployment are

```
library(shiny)
library(rlist)
library(DT)
library(shinyWidgets)
library(shinythemes)
```

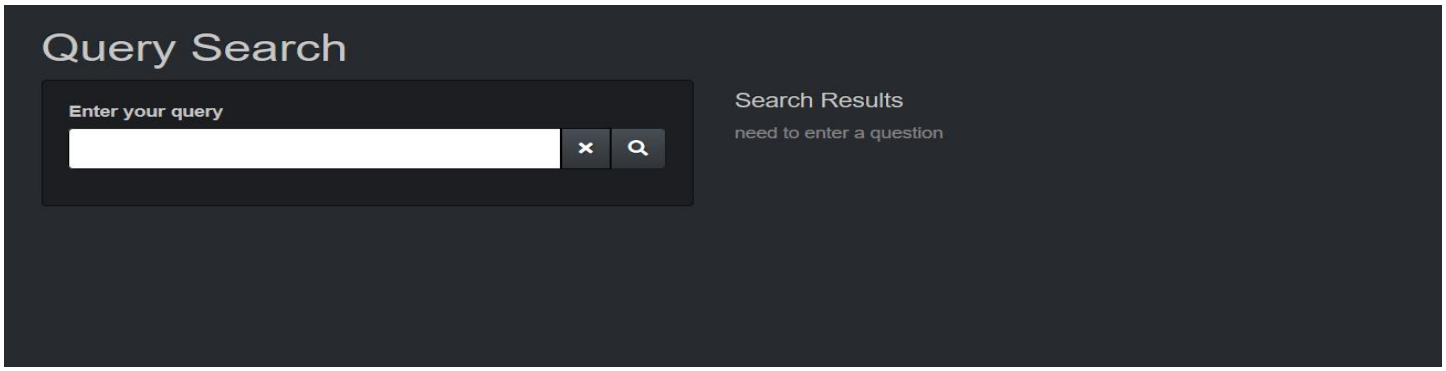
- The UI (User Interface) codes are

```
ui<- fluidPage(theme = shinytheme("slate"),

  pageWithSidebar(
    headerPanel("Query Search"),
    sidebarPanel(
      searchInput("query",label ="Enter your query",btnSearch = icon("search"),btnReset = icon("remove"),width = "450px")

    ),
    mainPanel(
      h4("Search Results"),
      #verbatimTextOutput("results")
      #textOutput("selected_var")
      dataTableOutput("results")
    )
  )
)
```

- The UI will look like this



- Shiny codes are

```
server<-function(input, output) {

  x<-reactive({
                  y<-QuestionSearch(input$query)
                  if(nrow(y)!=0){

                                  op<-list()
                                  for (i in 1:nrow(y)){
                                    op<-list.append(op,paste0(y[i,1]))
                                    op<-list.append(op,paste0(y[i,2]))
                                    i=i+1
                                  }
                                  op<-as.character(op)

                                  iter=length(op)
                                  output <- matrix(ncol=2, nrow = iter)


                                  for (i in 1:iter){
                                    if((i %% 2) == 0) {

                                      output[i,1]<- paste("Answer",i/2)
                                      output[i,2]<- paste(op[i])

                                    }
                                    else {
                                      if(round(i/3)!=0){
                                        if(i>=7){
                                          output[i,1]<- paste("Question",round(i/3)+2)
                                          output[i,2]<- paste(op[i])
                                        }
                                        else{
                                          output[i,1]<- paste("Question",round(i/3)+1)
                                          output[i,2]<- paste(op[i])
                                        }

                                      }
                                      else{
                                        output[i,1]<- paste("Question",i)
                                        output[i,2]<- paste(op[i])
                                      }
                                    }

                                    i=i+1

                                  }
                              }
                          else{
                            output <- matrix(ncol=2, nrow = 1)
                            output[1,1]<- paste(".")
                            output[1,2]<- paste("No Result Found")
                          }
                          output<-data.frame(output)
                      #    datatable(head(output), rownames = FALSE)

                          return(output)

              })

   output$results<-renderDataTable({
                                    validate(
                                             need(input$query, 'need to enter a question')
                                            )

                                    datatable(x(),rownames = FALSE,colnames = "",
                                              options = list(dom = 't',autoWidth = TRUE,
                                                             columnDefs = list(list(width = '90%', targets = 1)))
                                            )
```

```
}

runApp(
  launch.browser= T,shinyApp(ui = ui, server = server)
  )
```

- The output looks like

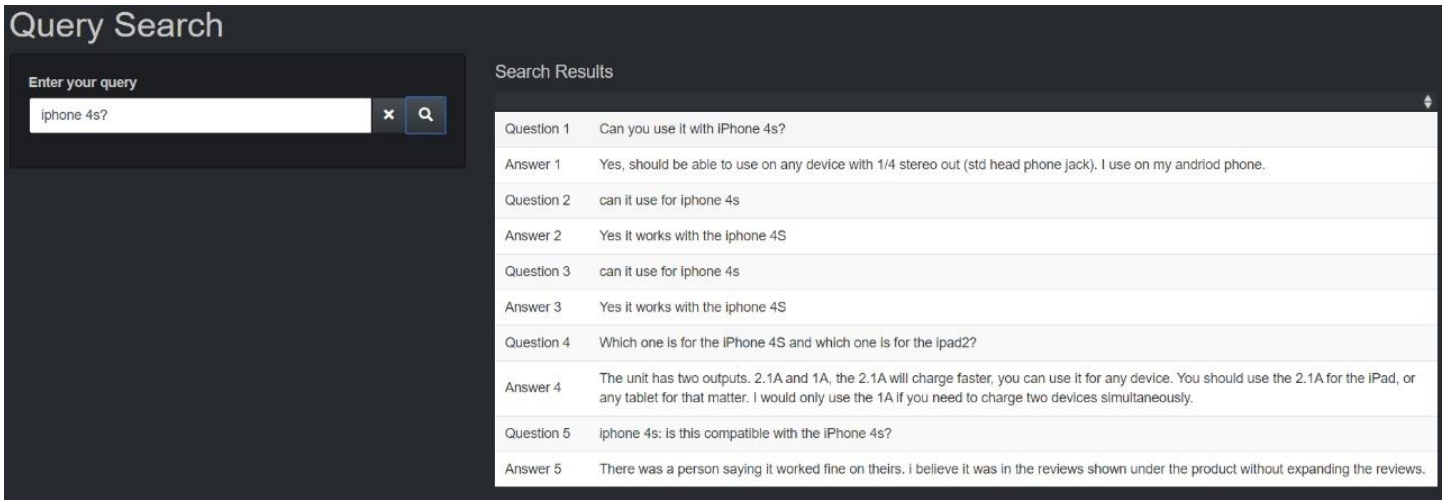


Fig 7.1 *shows the output of putting in a single keyword*

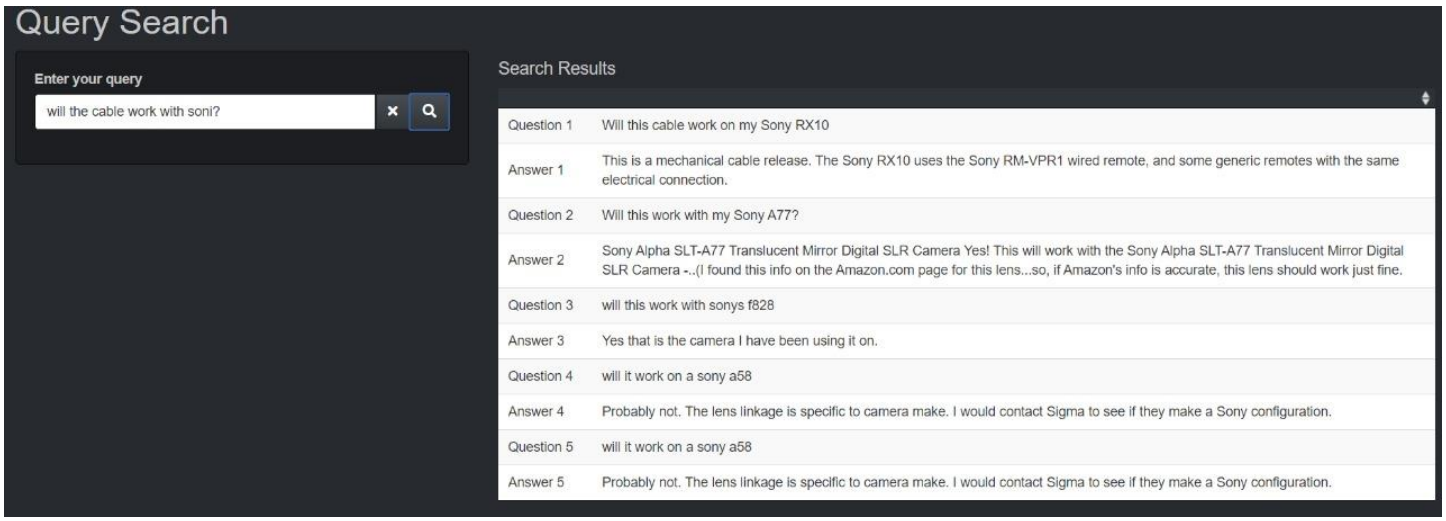


Fig 7.2 *shows an incorrect word in input getting corrected and then giving appropriate ouput*

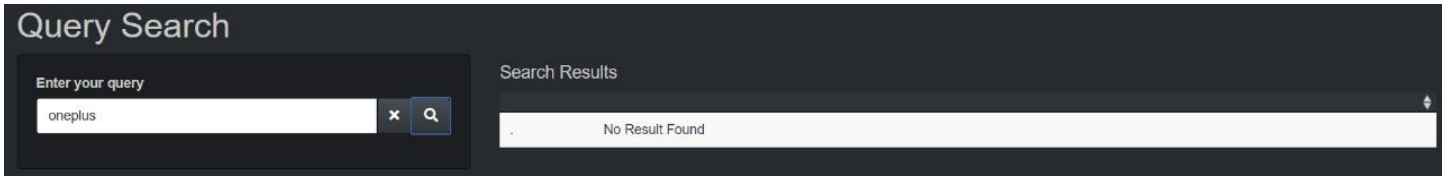


Fig 7.3 *shows if there is no related word in the dataset then it will show NO RESULTS FOUND*

Thank you