Report - Quora Question Papers

Problem Statement

The problem statement is to predict which of the provided pairs of questions contain two questions with the same meaning.

Data

Our task is to build classification models which will classify the target variable(is_duplicate)

id - the id of a training set question pair
qid1, qid2 - unique ids of each question (only available in train.csv)
question1, question2 - the full text of each question
is_duplicate - the target variable, set to 1 if question1 and question2 have essentially the same meaning, and 0 otherwise.

Structure of Data

```
> str(data)
'data.frame':   404290 obs. of  6 variables:
 $ id          : int  0 1 2 3 4 5 6 7 8 9 ...
 $ qid1        : int  1 3 5 7 9 11 13 15 17 19 ...
 $ qid2        : int  2 4 6 8 10 12 14 16 18 20 ...
 $ question1   : chr  "What is the step by step guide to invest in share market in india?" "What is the story of Kohinoor (Koh-i-
the speed of my internet connection while using a VPN?" "Why am I mentally very lonely? How can I solve it?" ...
 $ question2   : chr  "What is the step by step guide to invest in share market?" "What would happen if the Indian government sto
 back?" "How can Internet speed be increased by hacking through DNS?" "Find the remainder when [math]23^{24}[/math] is divided by
 $ is_duplicate: int  0 0 0 0 0 1 0 1 0 0 ...
```

Summary of Data

```
> summary(data)
       id              qid1              qid2          question1        
 Min.   :     0   Min.   :     1   Min.   :     2   Length:404290     
 1st Qu.:101072   1st Qu.: 74438   1st Qu.: 74727   Class :character  
 Median :202145   Median :192182   Median :197052   Mode  :character  
 Mean   :202145   Mean   :217244   Mean   :220956                     
 3rd Qu.:303217   3rd Qu.:346574   3rd Qu.:354693                     
 Max.   :404289   Max.   :537932   Max.   :537933                     
  question2          is_duplicate   
 Length:404290      Min.   :0.0000  
 Class :character   1st Qu.:0.0000  
 Mode  :character   Median :0.0000  
                    Mean   :0.3692  
                    3rd Qu.:1.0000  
                    Max.   :1.0000  
```

Methodology

As per the problem statement we need to analyze text which requires different techniques as compared to numerical or categorical data.
We can approach this problem using various text processing methods which can give the meaning of the text in numerical terms which makes it easy for us to analyze this data.
Stringdist package is extensively used in this project. In which different methods of string comparisons are used.

Let's start

```
> length(unique(data$question1))
[1] 290457
> length(unique(data$question2))
[1] 299175
```

The data contains many repeated questions

Count of Duplicate levels

```
> table(data$is_duplicate)

     0      1
255025 149263
```

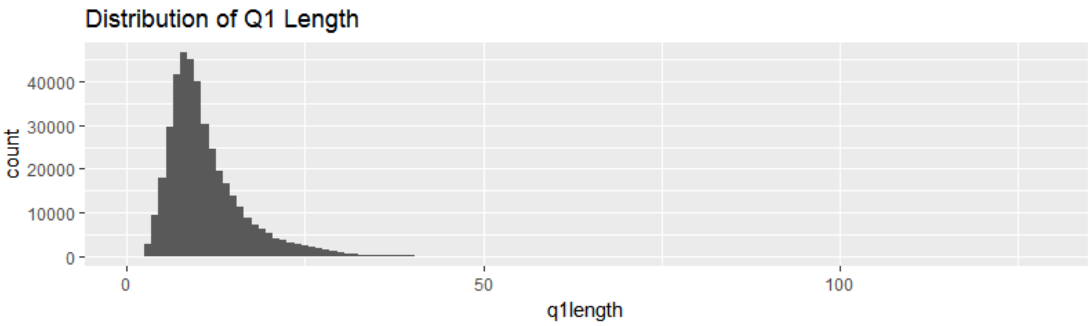
```
> sum(data$is_duplicate==0)/nrow(data)*100
[1] 63.08021
> sum(data$is_duplicate==1)/nrow(data)*100
[1] 36.91979
```

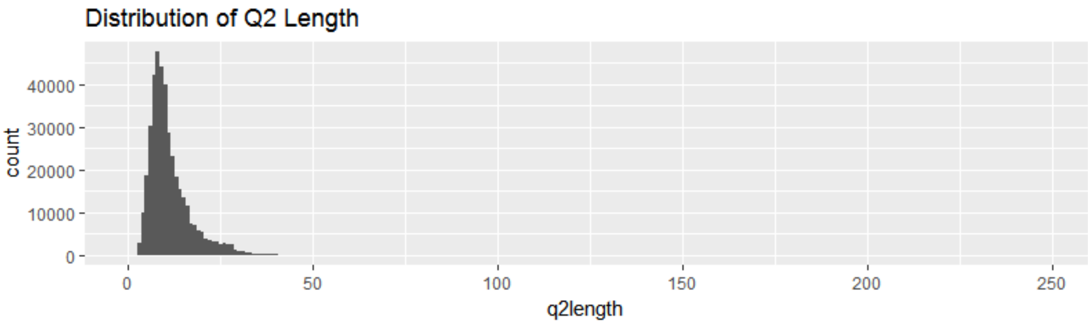
We have 63%of the observation of class 0 in our data.

Distribution of length in question1 & question 2

```
> summary(text_process[,4:5])
    q1length          q2length
 Min.   :  1.00   Min.   :  1.00
 1st Qu.:  7.00   1st Qu.:  7.00
 Median : 10.00   Median : 10.00
 Mean   : 11.13   Mean   : 11.38
 3rd Qu.: 13.00   3rd Qu.: 13.00
 Max.   :128.00   Max.   :247.00
```

From the summary we see that the average length is around 7-13 but some question show out of the ordinary lengths which can be seen in below box plot

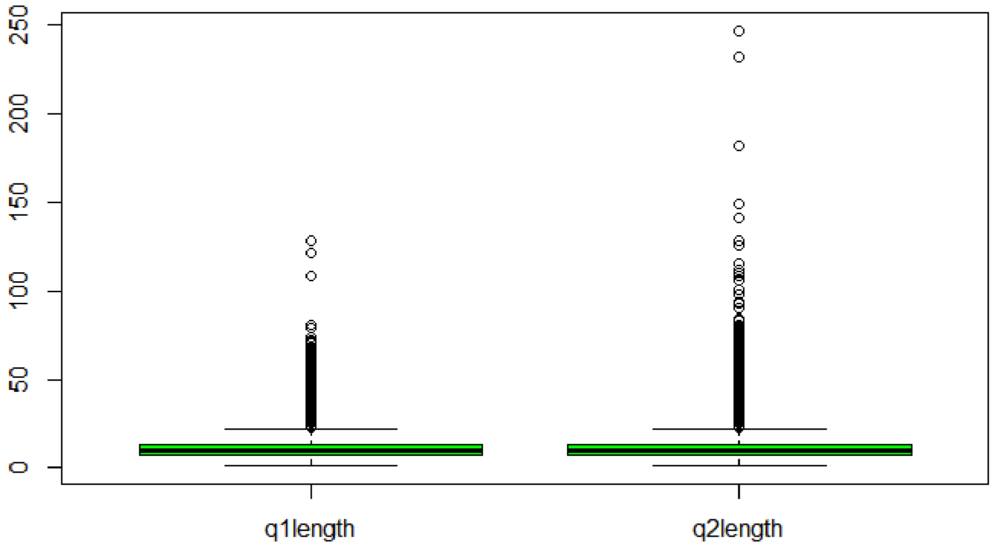




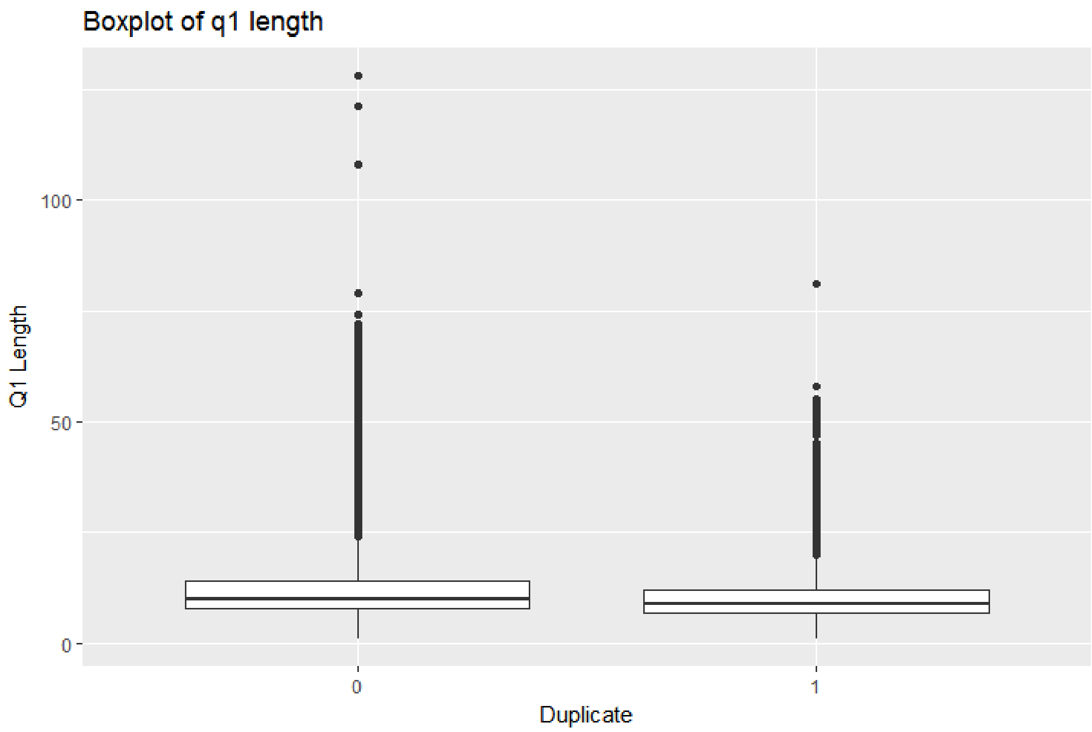

```
> sum(text_process$q1length>=7 & text_process$q1length<=13)/nrow(text_process)*100
[1] 61.27626
> sum(text_process$q1length<7)/nrow(text_process)*100
[1] 14.83474
> sum(text_process$q1length>13)/nrow(text_process)*100
[1] 23.889
>
> sum(text_process$q2length>=7 & text_process$q2length<=13)/nrow(text_process)*100
[1] 60.26259
> sum(text_process$q2length<7)/nrow(text_process)*100
[1] 15.23249
> sum(text_process$q2length>13)/nrow(text_process)*100
[1] 24.50491
```

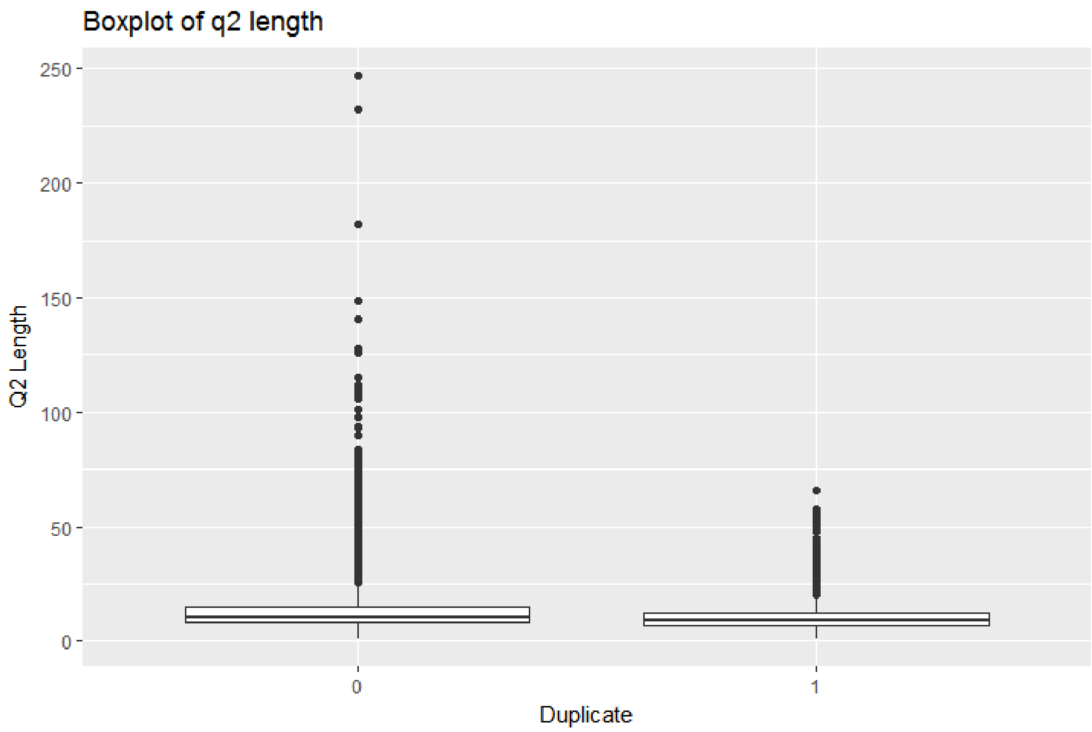
Around 60% question length falls between 7 & 13.

Outlier Analysis on text length for two levels of Duplicate

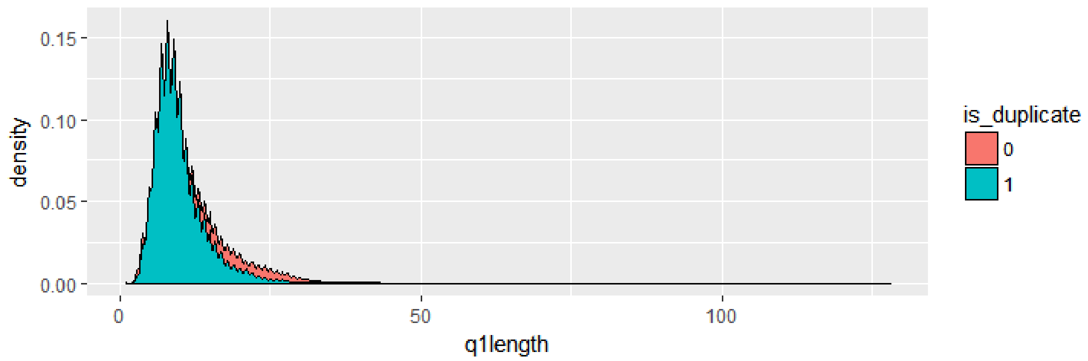


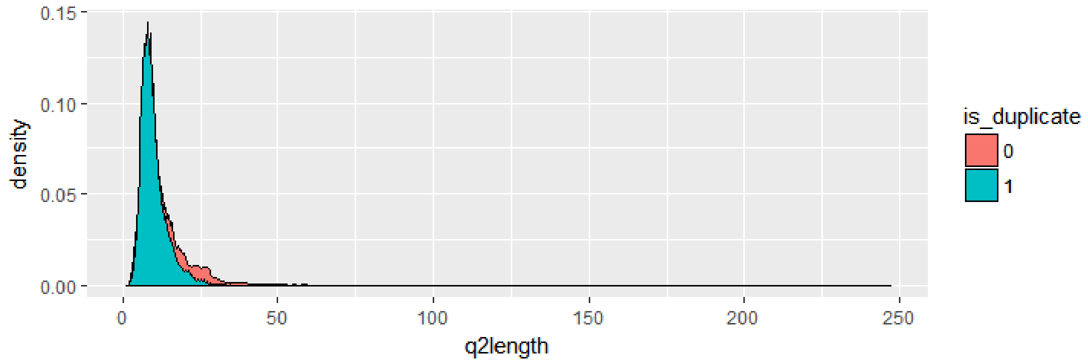
Q1 & Q2 Box Plot for each level of Duplicate



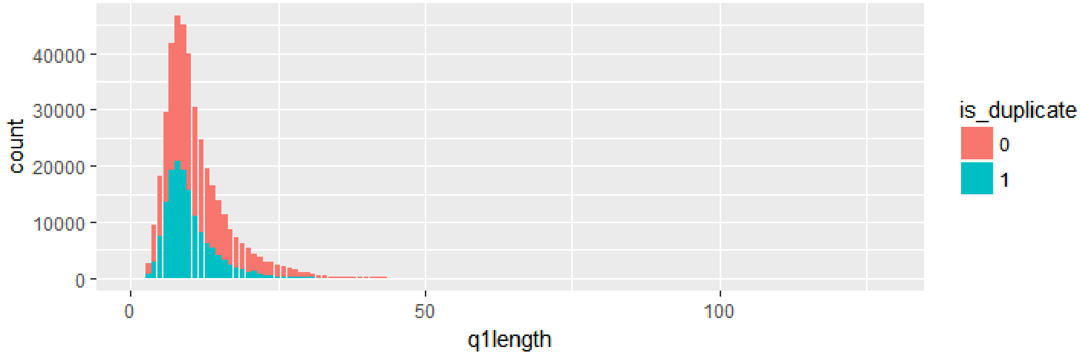


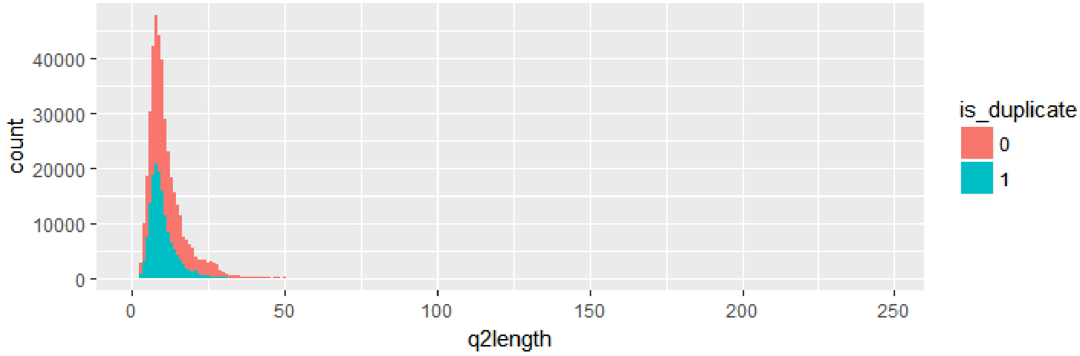
Density Plot for Q1 & Q2 Length

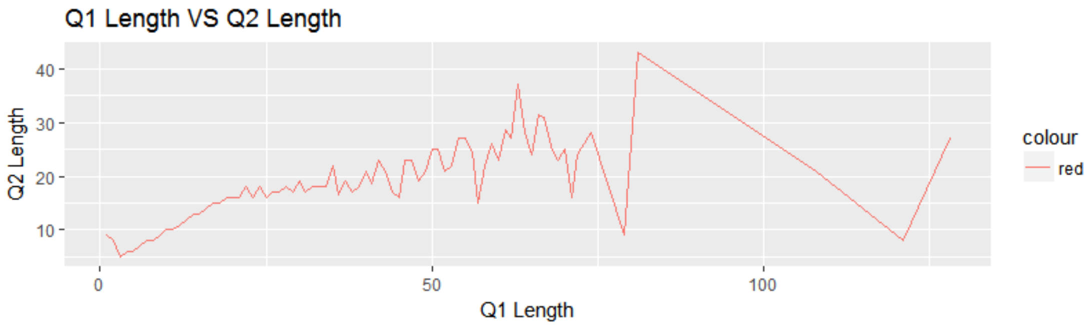


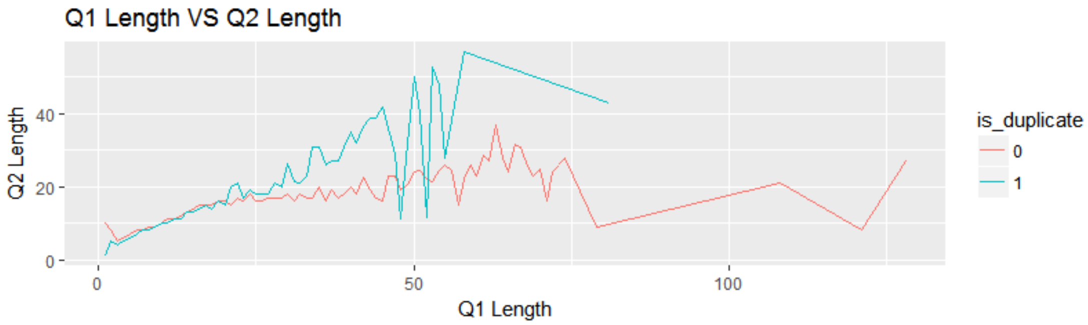


Let’s see it with bar plot









This shows as the length increases difference in duplicity becomes apparent.

Pre-processing

Data pre-processing is done through **regular expressions** all punctuation marks are removed, text is converted to their base forms, text is converted to unitext code . Numbers are not removed from the data as they may hold significance.

Let’s do our work on text analysis with stringdist package which provides us many methods to work on text.

1. **Q-gram**
   It is a subsequence of q consecutive characters of a string. If x (y) is the vector of counts of q-gram occurrences in a (b), the q-gram distance is given by the sum over the absolute differences |xi – yi|.

2. **Jaro Distance**
   The Jaro distance (method='jw', p=0), is a number between 0 (exact match) and 1 (completely dissimilar) measuring dissimilarity between strings. It is defined to be 0 when both strings have length 0, and 1 when there are no character matches between a and b.

3. **Cosine Distance**
   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.

4. **Stringsim**
   It computes pairwise string similarities between elements of character vectors a and b, where the vector with less elements is recycled. The similarity is calculated by first calculating the distance using stringdist, dividing the distance by the maximum possible distance, and substracting the result from 1. This results in a score between 0 and 1, with 1 corresponding to complete similarity and 0 to complete dissimilarity

5. **Longest Common Substring (LCS)**
   The longest string that can be obtained by pairing characters from a and b while keeping the order of characters intact. The lcs-distance is defined as the number of unpaired characters. The distance is equivalent to the edit distance allowing only deletions and insertions, each with weight one.

6. **Full Damerau-Levenshtein distance**
   It is like the optimal string alignment distance except that it allows for multiple edits on substrings.

7. **Jaccard Distance Method**
   It is a statistic used for comparing the similarity and diversity of sample sets.

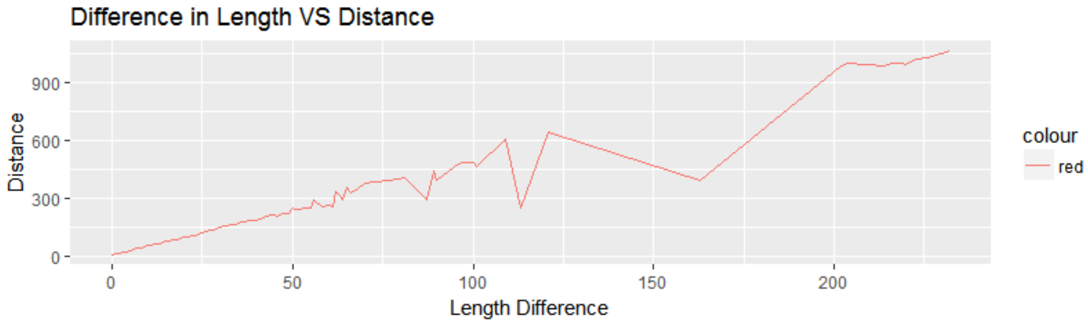
8. **Optimal String Alignment distance (OSA)**
   It counts the number of deletions, insertions and substitutions necessary to turn b into a. allows transposition of adjacent characters. Here, each substring may be edited only once.

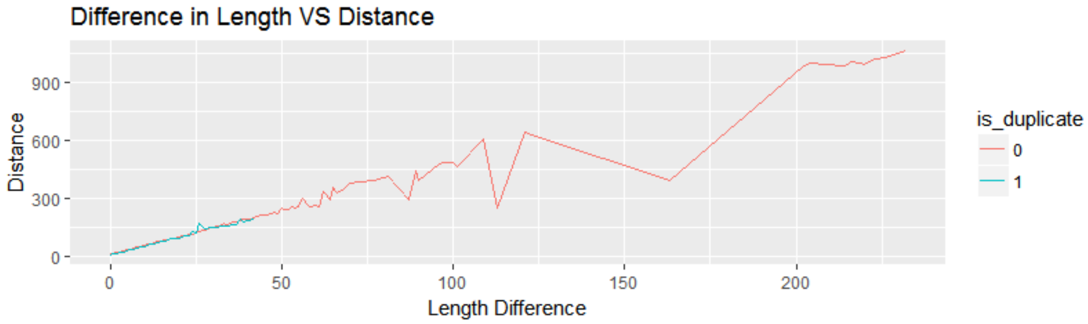
After applying all the methods we have our data which is quantitative and this allows us to perform numerous algorithms.
Let's look at our data now

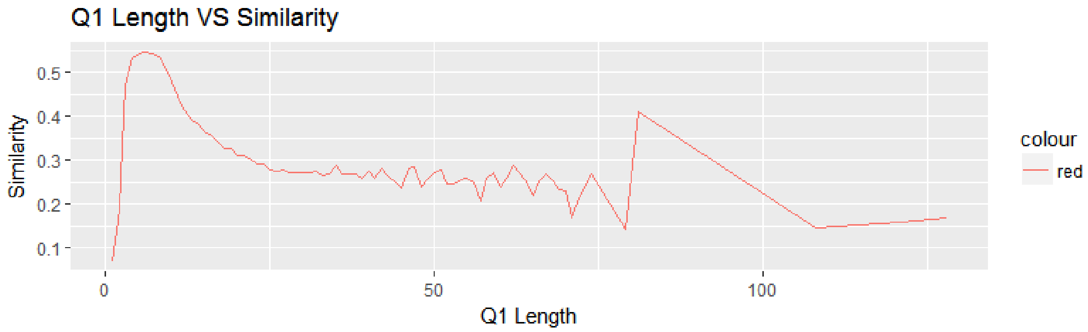
```
> str(text_process)
'data.frame':	404274 obs. of  14 variables:
 $ question1   : chr  "what is the step by step guide to invest in share
market in india" "what is the story of kohinoor koh i noor diamond" "how
can i increase the speed of my internet connection while using a vpn" "wh
y am i mentally very lonely how can i solve it" ...
 $ question2   : chr  "what is the step by step guide to invest in share
market" "what would happen if the indian government stole the kohinoor ko
h i noor diamond back" "how can internet speed be increased by hacking th
rough dns" "find the remainder when math 23 24 math is divided by 24 23"
...
 $ is_duplicate: Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 2 1 1 ...
 $ q1length    : num  14 10 14 11 13 17 4 7 8 9 ...
 $ q2length    : num  12 15 10 13 7 17 11 9 7 9 ...
 $ diff_length : num  2 5 4 2 6 0 7 2 1 0 ...
 $ dist        : num  9 41 30 39 41 15 51 15 5 24 ...
 $ jw_meth     : num  0.0462 0.2791 0.2745 0.3855 0.3406 ...
 $ cosine_meth : num  0.0164 0.0644 0.0486 0.2052 0.0996 ...
 $ simi        : num  0.862 0.506 0.444 0.203 0.301 ...
 $ lcs         : num  9 47 60 69 61 45 59 27 5 50 ...
 $ dl          : num  9 42 40 47 51 38 52 19 4 46 ...
 $ jaccard     : num  0 0.348 0.261 0.409 0.333 ...
 $ osa         : num  9 42 40 47 51 38 52 19 4 46 ...
```

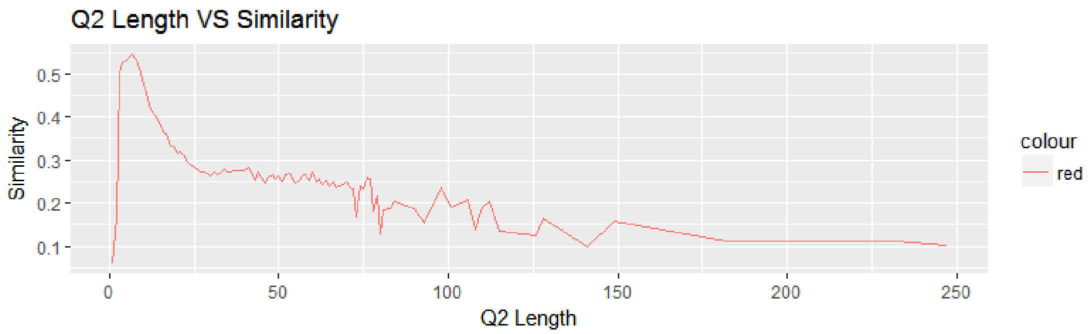
```
> summary(text_process)
  question1           question2          is_duplicate    q1length     
 Length:404274      Length:404274      0:255011      Min.   :  1.00  
 Class :character   Class :character   1:149263      1st Qu.:  7.00  
 Mode  :character   Mode  :character                 Median : 10.00  
                                                     Mean   : 11.13  
                                                     3rd Qu.: 13.00  
                                                     Max.   :128.00  
    q2length       diff_length          dist            jw_meth      
 Min.   :  1.00   Min.   :  0.00   Min.   :   0.00   Min.   :0.0000  
 1st Qu.:  7.00   1st Qu.:  1.00   1st Qu.:  14.00   1st Qu.:0.2063  
 Median : 10.00   Median :  2.00   Median :  23.00   Median :0.2779  
 Mean   : 11.38   Mean   :  3.78   Mean   :  29.22   Mean   :0.2657  
 3rd Qu.: 13.00   3rd Qu.:  5.00   3rd Qu.:  37.00   3rd Qu.:0.3342  
 Max.   :247.00   Max.   :232.00   Max.   :1068.00   Max.   :1.0000  
  cosine_meth           simi             lcs         
 Min.   :0.00000   Min.   :0.0000   Min.   :   0.00  
 1st Qu.:0.03814   1st Qu.:0.3000   1st Qu.:  19.00  
 Median :0.06571   Median :0.4500   Median :  37.00  
 Mean   :0.07792   Mean   :0.4904   Mean   :  47.42  
 3rd Qu.:0.10359   3rd Qu.:0.6552   3rd Qu.:  65.00  
 Max.   :1.00000   Max.   :1.0000   Max.   :1076.00  
       dl             jaccard             osa         
 Min.   :   0.00   Min.   :0.0000   Min.   :   0.00  
 1st Qu.:  15.00   1st Qu.:0.1053   1st Qu.:  15.00  
 Median :  30.00   Median :0.1875   Median :  30.00  
 Mean   :  37.79   Mean   :0.1943   Mean   :  37.81  
 3rd Qu.:  51.00   3rd Qu.:0.2727   3rd Qu.:  51.00  
 Max.   :1071.00   Max.   :1.0000   Max.   :1071.00  
```

diff_length is created by subtracting the length of Question1 & Question2 with absolute values(non negative)

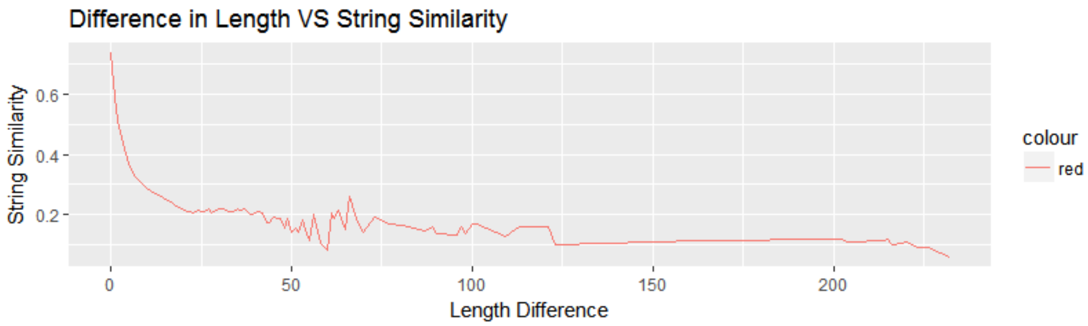


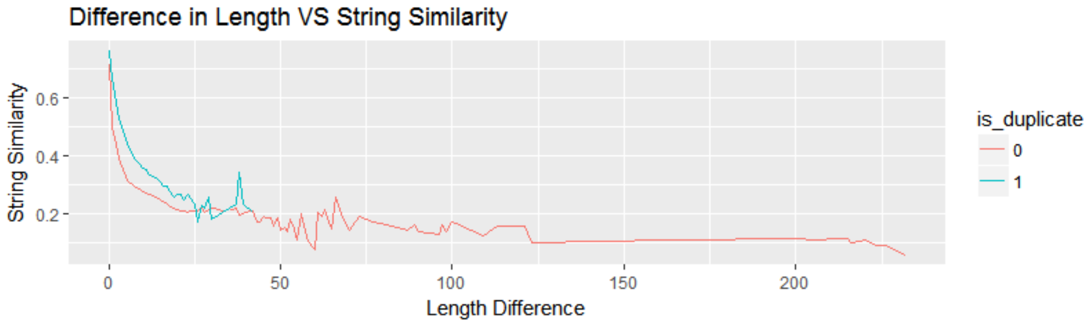






Here it shows that as the length of questions increases their similarity decreases. The same trend is also shown in difference in length(diff_length) below



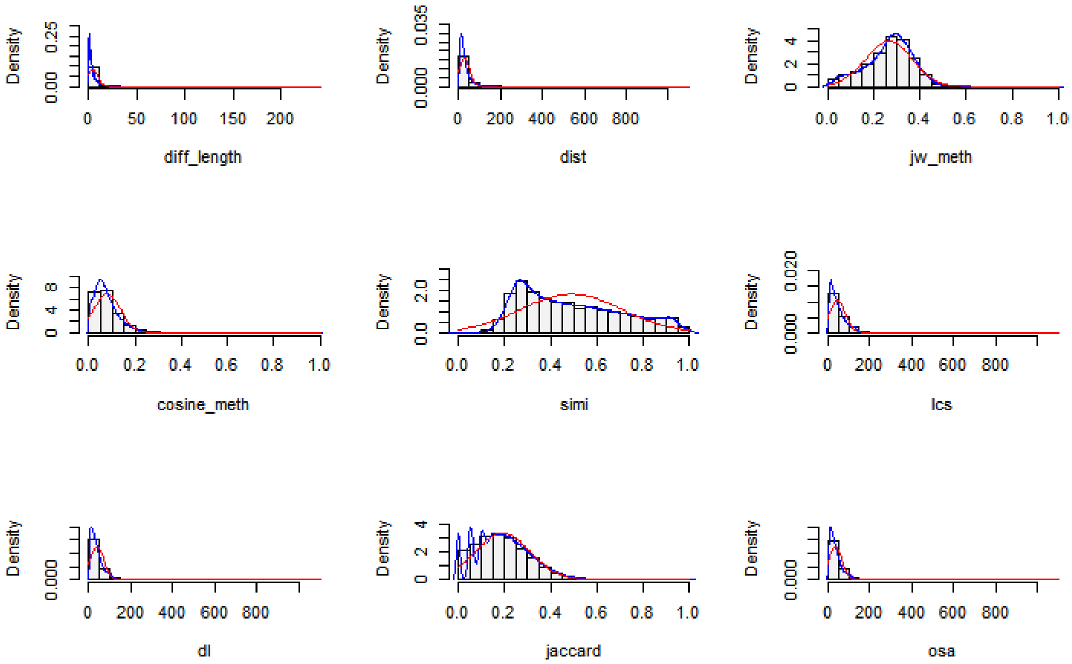


<u>This shows as the difference in string length increases string similarity decreases.</u>

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

    0     1 
28172  4677 
> table(text_process$is_duplicate[which(text_process$diff_length>1)])

     0      1 
168682  78389 
> sum(text_process$is_duplicate==0 & text_process$diff_length>10)/nrow(text_process)*100
[1] 7.457813
> sum(text_process$is_duplicate==1 & text_process$diff_length>10)/nrow(text_process)*100
[1] 0.7724959
```

<u>This gives us a clue that as the difference in string length increases more than 10 there is a high chance that they are not same. We can approve it after applying a model.</u>

A combined plot for all numeric data



Here dist, dl and osa showing the same distribution. So we will check for multicolinearity and remove unnecessary variables.

Checking for Multicollinearity

```
> vif(as.data.frame(t_matrix))
     Variables          VIF
1     q1length     2.001873
2     q2length     2.952932
3  diff_length     6.362810
4         dist    31.394486
5      jw_meth     5.945232
6  cosine_meth     4.057238
7         simi    10.331273
8          lcs    57.094787
9           dl 35008.220338
10     jaccard     3.667862
11         osa 35136.567778
> vifcor(t_matrix[,1:11], th=0.95)
2 variables from the 11 input variables have collinearity problem:

osa dl

After excluding the collinear variables, the linear correlation coefficients ranges between:
min correlation ( jaccard ~ q2length ):  -0.001888916
max correlation ( lcs ~ dist ):  0.9118671

---------- VIFs of the remained variables --------
    Variables       VIF
1    q1length  2.085014
2    q2length  2.689762
3 diff_length  5.066233
4        dist 18.135242
5     jw_meth  6.183475
6 cosine_meth  3.358180
7        simi  8.496280
8         lcs 18.322263
9     jaccard  3.448290
```

From this we see that dl & osa show the maximum collinearity so they are not fed to the model.

Word Cloud

Question 1 Word Cloud

Question 2 Word Cloud

A decision tree **C5.0** is used for classification with accuracy 71%

```
> confusionMatrix(xtab)
Confusion Matrix and Statistics

        predicted
observed      0      1
       0 100140  28881
       1  28750  46511

               Accuracy : 0.7179
                 95% CI : (0.7159, 0.7198)
    No Information Rate : 0.6309
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.394
 Mcnemar's Test P-Value : 0.5881

            Sensitivity : 0.7769
            Specificity : 0.6169
         Pos Pred Value : 0.7762
         Neg Pred Value : 0.6180
             Prevalence : 0.6309
         Detection Rate : 0.4902
   Detection Prevalence : 0.6316
      Balanced Accuracy : 0.6969

       'Positive' Class : 0
```

Another decision tree **“rpart”** is used for classification with accuracy 68%

```
> confusionMatrix(xtab)
Confusion Matrix and Statistics

        predicted
observed      0      1
       0 107073  21948
       1  42105  33156

               Accuracy : 0.6864
                 95% CI : (0.6844, 0.6885)
    No Information Rate : 0.7303
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2864
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7178
            Specificity : 0.6017
         Pos Pred Value : 0.8299
         Neg Pred Value : 0.4405
             Prevalence : 0.7303
         Detection Rate : 0.5241
   Detection Prevalence : 0.6316
      Balanced Accuracy : 0.6597

       'Positive' Class : 0
```

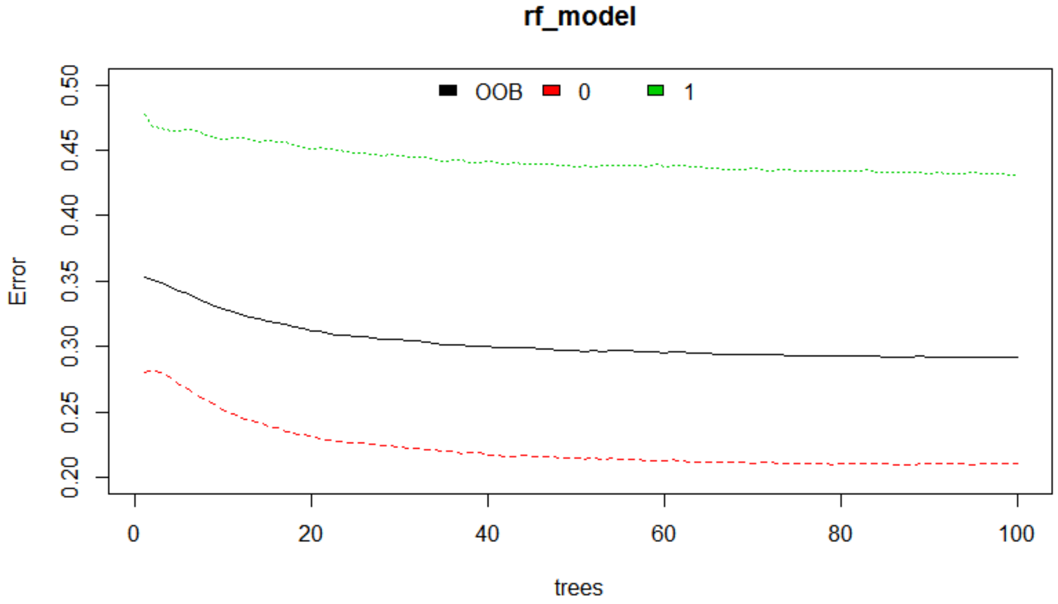
Random Forest Model

A Random Forest model is used to predict **"is_duplicate"** on test data with 85% accuracy.

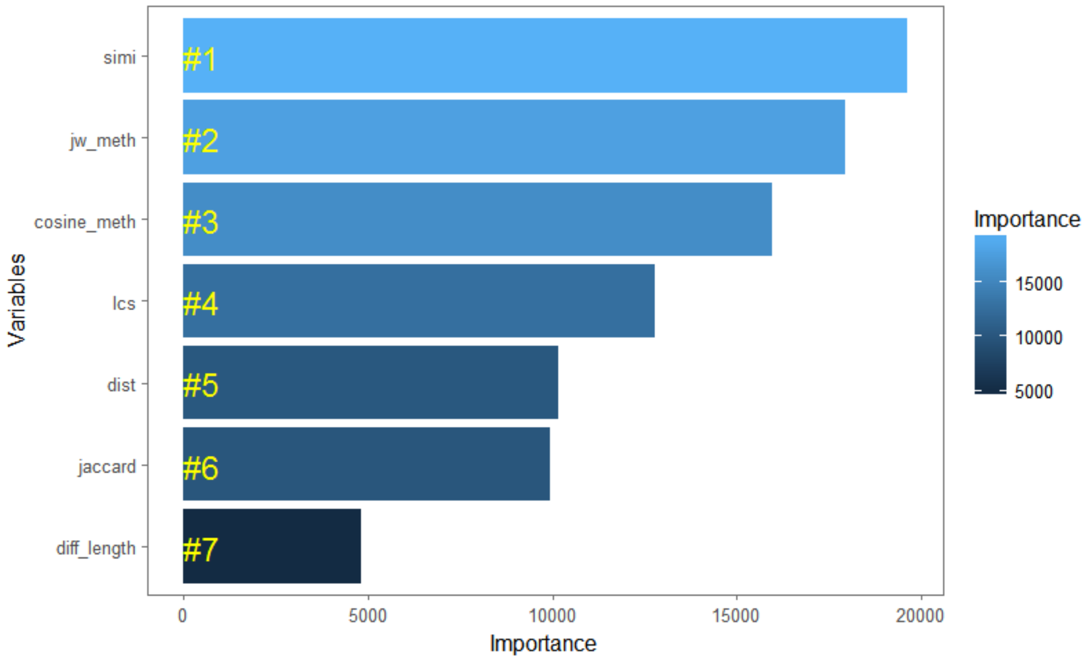
```
> confusionMatrix(xtab)
Confusion Matrix and Statistics

        predicted
observed      0      1
       0 115095  13926
       1  16401  58860

               Accuracy : 0.8515
                 95% CI : (0.85, 0.8531)
    No Information Rate : 0.6437
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6788
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8753
            Specificity : 0.8087
         Pos Pred Value : 0.8921
         Neg Pred Value : 0.7821
             Prevalence : 0.6437
         Detection Rate : 0.5634
   Detection Prevalence : 0.6316
      Balanced Accuracy : 0.8420

       'Positive' Class : 0


> rf_model

Call:
 randomForest(formula = factor(is_duplicate) ~ diff_length + dist + 
jw_meth + cosine_meth + simi + lcs + jaccard, data = train, 
importance = TRUE, ntree = 100)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 2

        OOB estimate of  error rate: 29.31%
Confusion matrix:
      0     1 class.error
0 98981 26979   0.2141870
1 31642 42398   0.4273636
```

It is noted that as sample is increased or if tree is increased there is decrease in error of class 1.
It means as the model trains on more data it is better able to predict for class 1

Error Plot



Variable Importance



The above variable importance plot shows that the methods stringsim(simi), Jaro distance(jw_meth), Cosine distance(cosine_meth) are important predictors to the model.

Now we will look at the predicted and actual values from RF Model

test_model variable is created which contains the observations in which observed class is same as predicted class.

```
> table(test_model$is_duplicate)

     0      1 
114598  59393 
> 
> sum(test_model$is_duplicate==0 & test_model$diff_length>10)/nrow(test_model)*100
[1] 8.720566
> 
> sum(test_model$is_duplicate==1 & test_model$diff_length>10)/nrow(test_model)*100
[1] 0.5483042
```

The above code shows that what we have earlier seen in the train data is correctly predicted in test data that as the **difference in string length increases more than 10 there is a high chance that they are not same.**

Model Evaluation

Among the decision tree models Random Forest gave us the maximum accuracy. But still predicting for string similarity is not easy.

KNN Model

KNN is used with K=1 with 96% accuracy

```
> Conf_matrix

pred      0      1
   0 125917   4392
   1   2669  71305
>
> accuracy = sum(diag(Conf_matrix))/nrow(test)
> accuracy
[1] 0.9654352
```

K=3 with 95% accuracy

```
> Conf_matrix

pred      0      1
   0 126019   7670
   1   2485  68110
>
> accuracy = sum(diag(Conf_matrix))/nrow(test)
> accuracy
[1] 0.9502898
```

K=5 with 95% accuracy

```
> Conf_matrix

pred      0      1
   0 126280   8958
   1   2224  66822
>
> accuracy = sum(diag(Conf_matrix))/nrow(test)
> accuracy
[1] 0.9452625
```

Model Evaluation

K-Nearest neighbour classification is a general technique to learn classification based on instance and do not have to develop an abstract model from the training data set. However the classification process could be very expensive because it needs to compute the similarly values individually between the test and training examples. Choosing a proper K value is important to the performance of K-Nearest Neighbour classifier. If k value is too large the nearest neighbour classifier may misclassify the test instance because its list of nearest neighbours may include data points that are located far away from its neighbourhood. On the other hand, if k value is too small, then the nearest neighbour classifier may be susceptible to over fitting because of noise in the training data set.