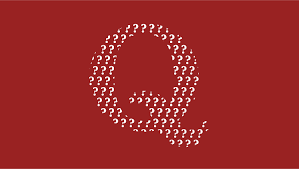
A Project Report on

**Quora Question Pairs**



Submitted by :

**Abhishek Darekar**

## **TABLE OF CONTENTS**

**CHAPTER TITLE PAGE NO.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  |
| 01 | | Introduction | 3 |
|  | 1.1 | Problem Decription | 3 |
|  | 1.2 | Problem Statement | 3 |
|  | 1.3 | Business Objectives and Constraints |  |
| 02 | | Mapping the real world problem to an ML problem | 4 |
|  | 2.1 | Data Overview | 4 |
|  | 2.2 | Data type & Missing Values | 5 |
| 03 | | EDA | 6 |
| 04 | | Basic Feature Extraction (before cleaning) | 8 |
|  |  |  |  |
| 05 | | Advanced Feature Extraction. | 10 |
|  | 5.1 | Preprocessing | 10 |
| 5.2 | | Advanced Feature Explained | 11 |
| 5.3 | | Analysis of extracted features | 12 |
|  |  | DONE TILL SLIDE Two | 12 |
| 06 | | Featurizing text data with tfidf weighted word-vectors | 14 |
| 07 | | Modelling | 16 |
|  | 10.1 |  | 17 |
|  | 10.2 |  | 19 |
|  | 10.3 |  | 21 |
|  | 10.4 |  | 23 |
| 11 | | Finalize the Model | 25 |
|  | 11.1 | Model Evaluation | 25 |
|  | 11.2 | Model Selection | 26 |
| 12 | | Save The Model | 27 |
| 13 | | Apply model on Test Data. | 28 |
| 14 | | Reference | 39 |
| 15 | | Appendix | 30 |

Chapter 1 : Introduction

### 1.1 Problem Description

Quora is a place to gain and share knowledge—about anything. It’s a platform to ask questions and connect with people who contribute unique insights and quality answers. This empowers people to learn from each other and to better understand the world.

Over 100 million people visit Quora every month, so it's no surprise that many people ask similarly worded questions. Multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question, and make writers feel they need to answer multiple versions of the same question. Quora values canonical questions because they provide a better experience to active seekers and writers, and offer more value to both of these groups in the long term.

> Credits: Kaggle.

### 1.2 Problem Statement

- Identify which questions asked on Quora are duplicates of questions that have already been asked.

- This could be useful to instantly provide answers to questions that have already been answered.

- We are tasked with predicting whether a pair of questions are duplicates or not.

### 1.3 Business Objectives and Constraints

1. The cost of a mis-classification can be very high. (Q1 & Q2 are not Duplicates but we declare them as Duplicates. so the new question will show wrong answers i.e answers of the old question which we declared duplicate.)

2. You would want a PROBABILITY Score of a pair of questions to be duplicates so that you can choose any THRESHOLD of choice.

3. No strict latency concerns.

4. Interpretability is partially IMPORTANT. (Understand why Q1 and Q2 have been declared Duplicate or not).

### 2. Mapping the real world problem to an ML problem

- It is a binary classification problem, for a given pair of questions we need to predict if they are duplicate or not.

- Metric(s):

- \* log-loss : https://www.kaggle.com/wiki/LogarithmicLoss

- \* Binary Confusion Matrix

- Our Primary Metric will be Log-loss , along with that we will use Confusion matrix as well

### 2.1 Data Overview

--Data will be in a file Train.csv

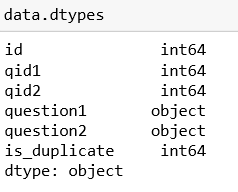
- Train.csv contains 5 columns : qid1, qid2, question1, question2, is\_duplicate

- Size of Train.csv - 60MB

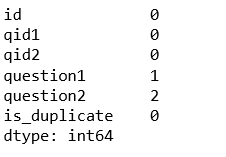
- Number of rows in Train.csv = 404,290

### 2.2 Data type & Missing Values

Data types :

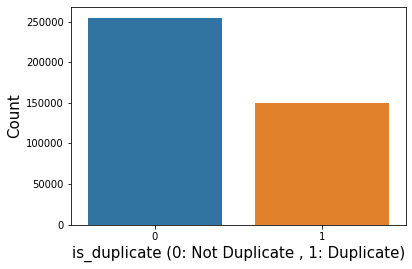


Missing values :



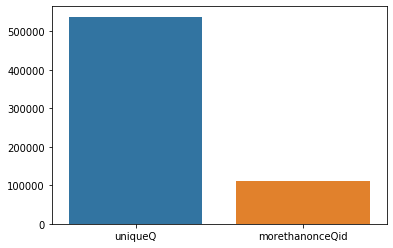
Very Few Missing Columns so we can just Delete.

### 3. EDA

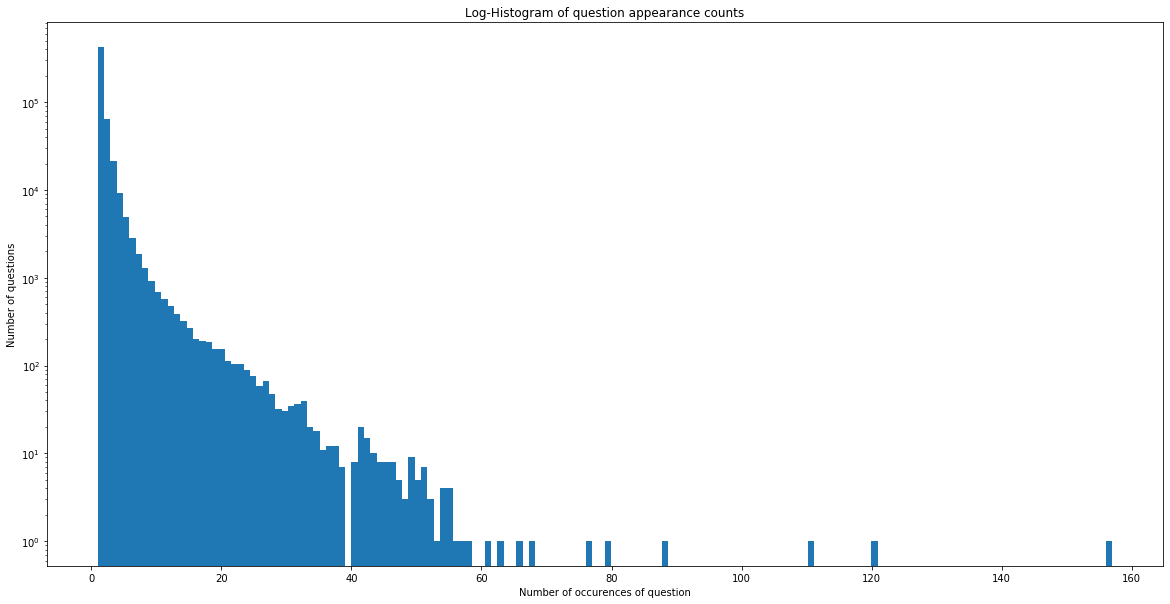
* Counts for our Target Variable: is\_duplicate
* Some Useful Information :

1. The total Number of pairs are :404287
2. The % of Unsimilar pairs : 63.08
3. The % of Duplicate pairs : 36.92
4. Total Question id are : 808574
5. Unique Questions are : 537929
6. Questions that Occur more than once 111778 (20.779322178205675):

* Distribution Of Unique Question id’s and Question id’s asked more than once



* **Number of occurrences of each question**



* Maximum number of times a single question is repeated: 157

### 4. Basic Feature Extraction (before cleaning)

Let us now construct a few features like:

- **freq\_qid1** = Frequency of qid1's

- **freq\_qid2** = Frequency of qid2's

- **q1len** = Length of q1

- **q2len** = Length of q2

- **q1\_n\_words** = Number of words in Question 1

- **q2\_n\_words** = Number of words in Question 2

- **word\_Common** = (Number of common unique words in Question 1 and Question 2)

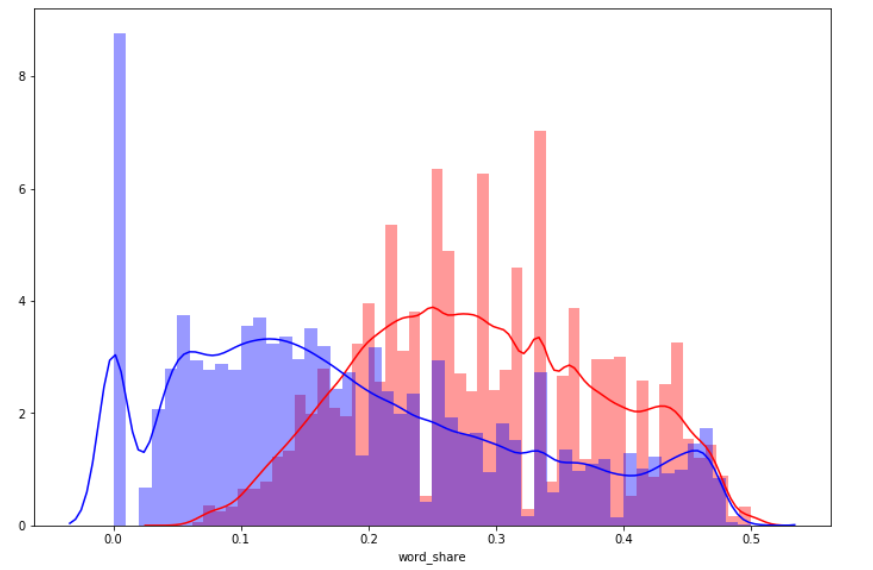
- **word\_Total** =(Total num of words in Question 1 + Total num of words in Question 2)

- **word\_share** = (word\_common)/(word\_Total)

- **freq\_q1+freq\_q2** = sum total of frequency of qid1 and qid2

- **freq\_q1-freq\_q2** = absolute difference of frequency of qid1 and qid2

* **Observation :**
* word\_share, freq\_qid1, freq\_q1+q2, freq\_qid2,word\_Common Have Good Correlation with is\_duplicate.
* **Visualise the word\_share Feature** :



* The distributions for word\_share have some overlap on the far right-hand side, i.e., there are quite a lot of questions with high word similarity
* The average word share and Common no. of words of qid1 and qid2 is more when they are duplicate(Similar)

### 5. Advance Feature Extraction

### 5.1 Preprocesing

* Removing html tags
* Removing Punctuations
* Performing stemming
* Removing Stopwords
* Expanding contractions ex :replace("€", " euro ").replace("'ll", " will") etc
* # Convert to String and make it Lower Case Replacing ex" 1000 with 1K

### 5.2 Advanced Feature Explained

* -**cwc\_min** : Ratio of common\_word\_count to min lenghth of word count of Q1 and Q2 :cwc\_min = common\_word\_count / (min(len(q1\_words), len(q2\_words))
* **cwc\_max** : Ratio of common\_word\_count to max lenghth of word count of Q1 and Q2 :cwc\_max = common\_word\_count / (max(len(q1\_words), len(q2\_words))
* csc\_min : Ratio of common\_stop\_count to min lenghth of stop count of Q1 and Q2 :csc\_min = common\_stop\_count / (min(len(q1\_stops), len(q2\_stops))
* csc\_max : Ratio of common\_stop\_count to max lenghth of stop count of Q1 and Q2 :csc\_max = common\_stop\_count / (max(len(q1\_stops), len(q2\_stops))
* ctc\_min : Ratio of common\_token\_count to min lenghth of token count of Q1 and Q2 :ctc\_min = common\_token\_count / (min(len(q1\_tokens), len(q2\_tokens))
* ctc\_max : Ratio of common\_token\_count to max lenghth of token count of Q1 and Q2 :ctc\_max = common\_token\_count / (max(len(q1\_tokens), len(q2\_tokens))
* - last\_word\_eq : Check if First word of both questions is equal or not : last\_word\_eq = int(q1\_tokens[-1] == q2\_tokens[-1])
* first\_word\_eq : Check if First word of both questions is equal or not :first\_word\_eq = int(q1\_tokens[0] == q2\_tokens[0])
* abs\_len\_diff : Abs. length difference : abs\_len\_diff = abs(len(q1\_tokens) - len(q2\_tokens))
* mean\_len : Average Token Length of both Questions : mean\_len = (len(q1\_tokens) + len(q2\_tokens))/2
* **longest\_substr\_ratio** : Ratio of length longest common substring to min lenghth of token count of Q1 and Q2longest\_substr\_ratio = len(longest common substring) / (min(len(q1\_tokens), len(q2\_tokens))
* **FuzzyWuzzy: Fuzzy String Matching in Python**

<https://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/>

* There are techniques to measure Similarity between 2 sentences.
* We can Measure the Similarity using the term **Edit Distance.**
* Edit distance is the measure of how many Add,Delete,shift operations are needed to convert sentence A such that it is like sentence B.
* -**fuzz\_ratio** : Will return good score if the whole sentence A and B are similar.
  + **fuzz\_partial\_ratio** : Looks at Partial/Sub strings in the sentence, If it match for the Partial substring is Perfect, we will get a good score.
  + **token\_sort\_ratio** : Takes a sentence, break it into tokens and Sort them and then compares and returns score.
  + **token\_set\_ratio** : ex :

s1 = "mariners vs angels"

s2 = "los angeles angels of anaheim at seattle mariners"

Sort the :

t1 = "angels mariners vs"

t2 = "anaheim angeles angels los mariners of seattle vs"

t0 = [SORTED\_INTERSECTION]

t1 = [SORTED\_INTERSECTION] + [SORTED\_REST\_OF\_STRING1]

t2 = [SORTED\_INTERSECTION] + [SORTED\_REST\_OF\_STRING2]

t0 = "angels mariners"

t1 = "angels mariners vs"

t2 = "angels mariners anaheim angeles at los of seattle"

fuzz.ratio(t0, t1) ⇒ 90

fuzz.ratio(t0, t2) ⇒ 46

fuzz.ratio(t1, t2) ⇒ 50

fuzz.token\_set\_ratio("mariners vs angels", "los angeles angels of anaheim at seattle

### 5.3 Analysis of extracted features

- Creating Word Cloud of Duplicates and Non-Duplicates Question pairs

- We can observe the most frequent occuring words

* Total number of words in duplicate pair questions : 16109886
* Total number of words in non duplicate pair questions : 3335825
* Word Clouds generated from duplicate pair question's text :



* Word Clouds generated from non duplicate pair question's text



- We can clearly see there are certain words that occur most frequenlty in both Duplicate and Non-Duplicate sentences.

- There are also some words that occur in both Duplicate and Non-Duplicate sentences.

* **token\_set\_ratio,token\_sort\_ratio,fuzz\_ratio,fuzz\_partial\_ratio,ctc\_min,ctc\_max,last\_word\_eq Have decebt correlation i.e Above 30**
* **Pair plot** of features ['ctc\_min', 'cwc\_min', 'csc\_min', 'token\_sort\_ratio'] :

#ctc\_min (Histogram): Classes are well separated

#cwc\_min (Histogram): Classes are well separated at the edges

#csc\_min (Histogram): NOt Separable

#token\_sort\_ratio (Histogram) : Separable but both classes are very close

##csc\_min'&ctc\_min (Pairplots): look like having some value

#cwc\_min & token\_sort\_ratio Pairplots): look like having some value

#csc\_min token\_sort\_ratio Pairplots): look like having some value

* **Distribution** of the token\_sort\_ratio :

#In the pdf there is some overlap, but there are separable points as well.

#class 1 data tends to have larger Token sorty ratio

#violen : They dont overlap, so it can be of value

### 6. Featurizing text data

### 6. Featurizing text data with tfidf weighted word-vectors

* But instead of W2V we have used GLOVE.
* After we find TF-IDF scores, we convert each question to a weighted average of word2vec vectors by these scores.
* Here we use a pre-trained GLOVE model which comes free with "Spacy". <https://spacy.io/usage/vectors-similarity>.
* It is trained on Wikipedia and therefore, it is stronger in terms of word semantics.
* Instead of W2V we will be using GLOVE, it is similar to W2V i.e given a word it returns a vector.
* But it is stronger as it is trained on Wikipedia data.

**The Shape of Final Data is :**

Number of features in nlp dataframe : 17

Number of features in preprocessed dataframe : 12

Number of features in question1 w2v dataframe : 384

Number of features in question2 w2v dataframe : 384

Number of features in **final dataframe** : 794

### 7.Modeling

* We Will be trying 3 Models here :

1. Logistic Regression with hyperparameter tuning.
2. Logistic Regression with hyperparameter tuning.
3. XGBOOST.

* Here We will be using **Stochastic Gradient Descent** to implement Logistic and SVM :
* **Gradient Descent** is a very generic optimization algorithm capable of finding optimal solutions to a wide range of problems. The general idea of Gradient Descent is to tweak parameters iteratively in order to minimize a cost function.

**-** Suppose you are lost in the mountains in a dense fog; you can only feel the slope of the ground below your feet. A good strategy to get to the bottom of the valley quickly is to go downhill in the direction of the steepest slope. This is exactly what Gradient Descent does: it measures the local gradient of the error function with regards to the parameter vector θ, and it goes in the direction of descending gradient. Once the gradient is zero, you have reached a minimum!

**-** Concretely, you start by filling θ with random values (this is called random initialization), and then you improve it gradually, taking one baby step at a time, each step attempting to decrease the cost function (e.g., the MSE), until the algorithm converges to a minimum

**Training a model means searching for a combination of model parameters that minimizes a cost function (over the training set). It is a search in the model’s parameter space: the more parameters a model has, the more dimensions this space has, and the harder the search is: searching for a needle in a 300-dimensional haystack is much trickier than in three dimensions. Fortunately, since the cost function is convex in the case of Linear Regression, the needle is simply at the bottom of the bowl.**

* **Stochastic Gradient Descent** The main problem with Batch Gradient Descent is the fact that it uses the whole training set to compute the gradients at every step, which makes it very slow when the training set is large. At the opposite extreme, Stochastic Gradient Descent just picks a random instance in the training set at every step and computes the gradients based only on that single instance. Obviously this makes the algorithm much faster since it has very little data to manipulate at every iteration. It also makes it possible to train on huge training sets, since only one instance needs to be in memory at each iteration (SGD can be implemented as an out-of-core algorithm.)

### 7.1 Prepare the Data (We are taking 50K points only)

1. Convert all data types to numeric (All out Features are already numeric, But this is just a good practice to follow.
2. Random train test split( 70:30) :

Number of data points in train data : (35000, 794)

Number of data points in test data : (15000, 794)

### 7.2 Building a random model (Finding worst-case log-loss)

Why ?

- Because our Key Parameter is Log loss.

- Range of logloss is (0 to Infinity).

Working **:**

1. We give x as input and it returns a random value as y.

2. This is a random model (The Most Dumb Model).

3. So we Calculate the Log-Loss for this, we get the Worst Case Log-loss value

4. Now example this gives Log-loss of 0.88, Any of our Decent models should give a Log-Loss significantly higher than this.

5. Closer it is to Zero it is better

Closer it is to 0.88 it is Random

greater than 0.88 means it is worst than Random Model

|  |  |
| --- | --- |
|  |  |
| Log loss on Test Data using Random Model | 0.8856168693464945 |
|  |  |

### 7.2 Logistic Regression with hyperparameter tuning

* Trying Logistic because our Data has high number of dimensions
* SGD with loss="Log" Means a Logistic regression.
* And as we learned, Wherever, we use Log-loss we need to Calibrate the model. Therefore we will use: **CalibratedClassifierCV()** to calibrate the model
* Train and Test loss being similar shows there is no Over-Fit, if train loss is very small and test loss no that small, its overfit.
* Consider Precision and Recall too.

Observation :

- Log loss for Logistic with alpha 0.1 is: 0.4553985519406958 , As compared to Log loss for Random model : 0.8893385806507957

- This is a good sign as we Have reduced the error by almost 50%, Altough our goal is to Reach 0.

- The train log loss is: 0.47184264623543154 , The test log loss is: 0.4809830449608359 : These Values being close means there is no Overfit

- Precision for both classes is around 75 which is decent. Recall for Class 2 is very bad.

|  |  |
| --- | --- |
| value of best alpha | 0.1 |
| The train log loss is: | 0.4396422788207989 |
| The test log loss is: | 0.45797144971971526 |
| Precision | Both classes is around 75 |
| Recall | Class 2 is very bad |

### 7.3 Logistic Regression with hyperparameter tuning

* SGD with loss="hinge" Means a SVM.
* And as we learned, Wherever we use Log-loss we need to Caliberate the model.Therefore we will use : CalibratedClassifierCV() to caliberate the model
* Train and TEst loss being siimilar shows there is no Over-Fit , if train loss is very small and test loss no that small, its overfit.
* Consider Precision And Recall too.

**Observation** :

- Log loss for SVM with alpha 0.01 is: 0.5135127143773024 , As compared to Log loss for Random model : 0.8893385806507957

- This is a good sign as we Have reduced the error But we got a much better Performance using Logistic Model, Although our goal is to Reach 0.

- The train log loss is: 0.6164904929480097 , The test log loss is: 0.6133784573048636 : \*\*These Values being close means there is no Overfit\*\*

- Precision for both classes is around 75 which is decent. Recall for Class 2 is very bad.

|  |  |
| --- | --- |
| value of best alpha | 0.01 |
| The train log loss is: |  |
| The test log loss is: |  |
| Precision | Both classes is around 75 |
| Recall | Class 2 is very bad |

### 7.4 XGBOOST

Observation :

- Log loss for XGB with alpha 0.01 is: 0.35786879575108177 , As compared to Log loss for Random model : 0.8893385806507957

- This is a good sign as we Have Achieved The Best log-loss SO far Using XGB and the score is Good.

- The train log loss is: 0.357869 , The test log loss is: 0.35786879575108177 : \*\*These Values being close means there is no Overfit\*\*