Quora Question Pairs

Error404BrainNotFound
Sai Akshara - IMT2016022
Soumya Kariveda - IMT2016053
Lakshmi Manonmaie - IMT2016057

1 Problem Statement

In Quora, 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.So our problem is to predict the probability that the given a question pair is having the same meaning.

2 Data Analysis

Data set consists of approximately 403k data points in test set and 1000 data points in the train set.
For every train data point id, qid1, qid2, question1, question2, is_duplicate columns are given.
For every test data point id, question1, question2 columns are given.

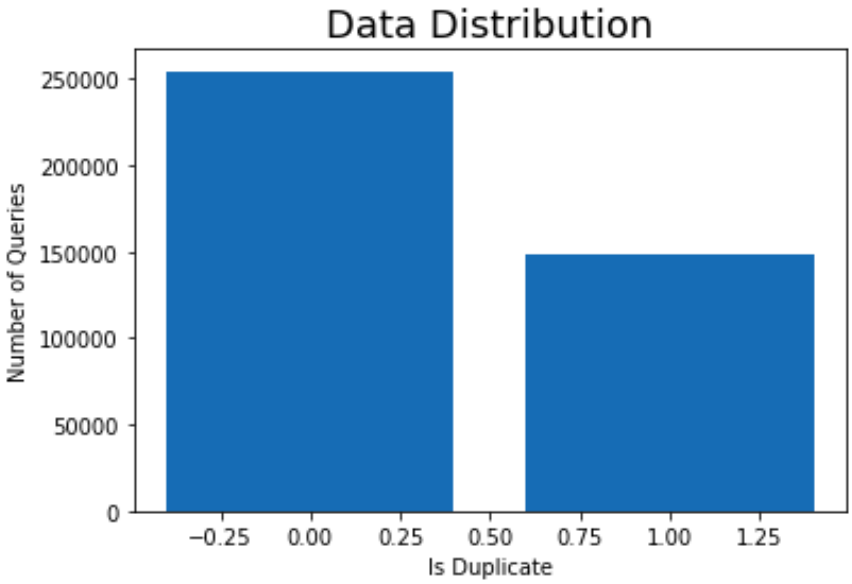
Description of columns:

id: Every question pair is given with a unique id
qid1,qid2: unique ids of each question
question1,question2: the full text of each question
is_duplicate: the target variable, set to 1 if question1 and question2 have same meaning, and 0 otherwise.

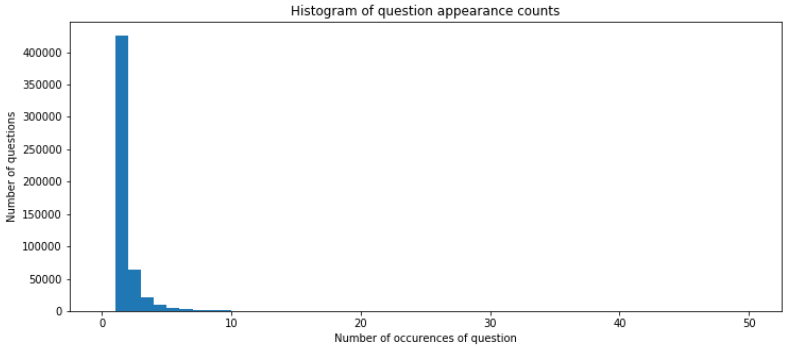
In the given data we are having 254k negative and 148k positive data points(positive as in the question pair given is of same meaning).

Visualization

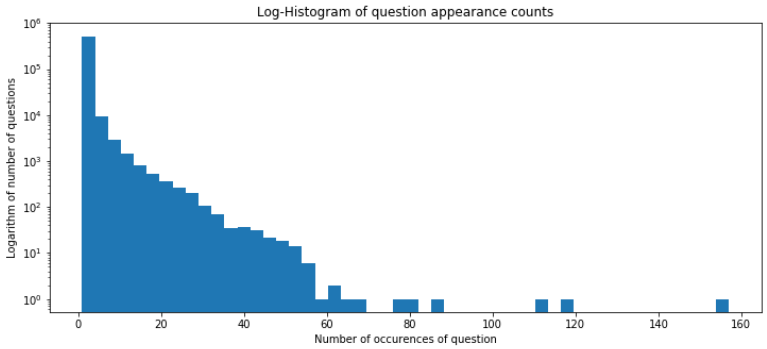
Data distribution



There are many questions that occur multiple times, this can be found using the given qid's.Total number of questions in the training data are 536k, this tells that there are some questions that occur multiple times.Maximum number of times a question appeared in data set is 157.The distribution of number of questions vs number of occurrences of the question is as follows



Since the number of questions with high frequency, to get a better idea of distribution, the following plot is log of number of questions vs frequencies



3 Approach Taken

3.1 Data Pre-Processing

Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and lacking in certain behaviors or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues.

The Pre-processing techniques we thought of are:

- **Filling NULL values:** While loading the train and test data from the CSV files we filled the empty questions with "no" instead of removing them as it's always good to fill the empty data than dropping them.
- **Dropping inappropriate rows:** We observed that there are few rows in the train data set, for which the label are not 0 or 1, it is given as some string which dosen't make sense, so we had to drop rows with inappropriate labels.
- **Handling case-sensitivity:** To make the model easily understand the questions it's better to make all the words to lowercase. To do so, we applied lower() method to data frame.
- **Handling punctuation:** During text processing the punctuation is not of much importance, so we excluded it.
- **Removing stop words:** One of the major forms of pre-processing is to filter out useless data. A stop word is a commonly used word (such as "the", "a", "an", "in") that a search engine has to ignore, so removed the stop words.

- **Lemmatize vs Stemming:** Stemming algorithms work by cutting off the end or the beginning of the word, taking into account a list of common prefixes and suffixes that can be found in an inflected word. This indiscriminate cutting can be successful in some occasions, but not always, and this approach may have some limitations. On the other hand, lemmatization takes the morphological analysis of the words into consideration. To do so, it is necessary to have detailed dictionaries which the algorithm can look through to link the form back to its lemma. So we decided upon using lemmatize over stemming.

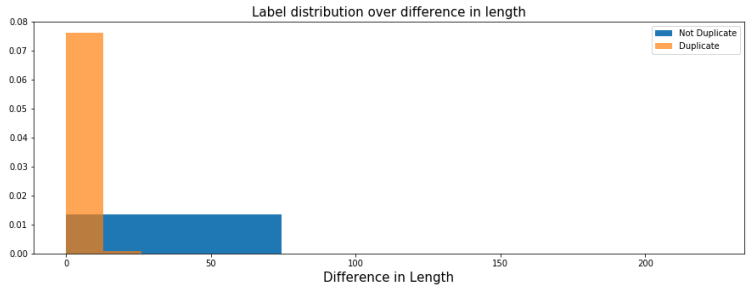
But the data before pre-processing showed better results than that after, probably this is due to over fitting of the model. So we considered removing, Lemmatization and removing stop words in pre-processing for good results.

3.2 Feature Engineering

- TF-IDF(Term Frequency-Inverse Document Frequency): It is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.
  TF(w) = doc.count(w)/total words in corpus
  IDF(w) = log(total number of documents/number of documents containing word w)
  Initially we tried creating our own dictionary of total words in corpus to frequency. Total words have been taken from all the words of questions. TF has been calculated using an inbuilt function and IDF from the dictionary what we have created. Though the results between inbuilt TF-IDF and what we computed was almost the same , but w.r.t time inbuilt was more efficient. So we switched to the inbuilt TF-IDF.

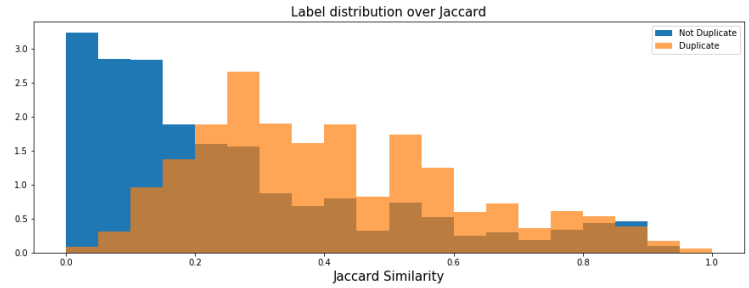
- **Cosine similarity**: It is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The vectors are TF-IDF vectors.

- **Length Difference**: Difference in the length of two questions w.r.t to number of words.



- **Euclidean distance**: 2-norm between TF-IDF vectors.
- **TF-IDF ngram**: Calculating TF-IDF by taking n words at a time.
- **Cosine ngram**: Calculating Cosine distance using TF-IDF ngram vectors.
- **Jaccard**: It's a measure of similarity for the two sets of data.

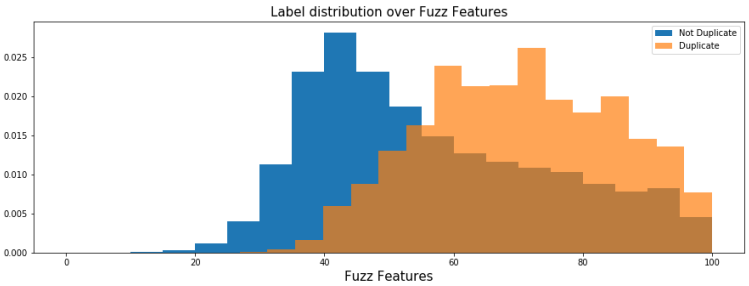
$$J(X,Y) = |X \cap Y|/|X \cup Y|$$



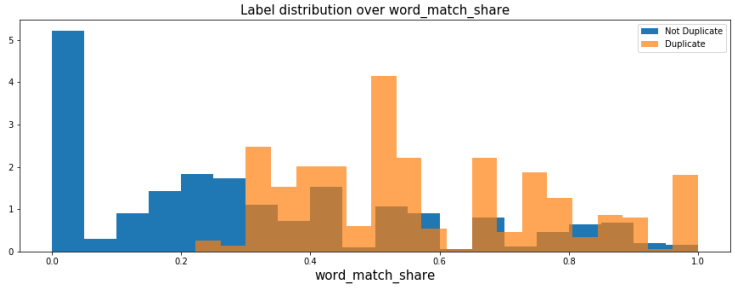
- **Manhattan**: 1-norm distance between TF-IDF vectors.
- **Canberra**:It is like manhattan distance but the variables of the two objects is divided by the sum of the absolute variable values prior to summing.

$$d(x,y) = \Sigma(|(p_i - q_i)|/|p_i| + |q_i|)$$

- **Bray–Curtis dissimilarity**: Is a statistic used to quantify the compositional dissimilarity between two different sites, based on counts at each site.
- **Fuzzy wuzzy**: Calculated by the number of operations taken to convert a question to another question. These operations include remove , insert , replace.



- **Frequencies q1,q2**:A question that is asked often has more chances to be redundant. Feature here is to count the frequency of questions.
- **Match score**: Ratio of common words multiplied by 2 and total number of words in both the questions.



- **TF-IDF match score**: tf-idf values of the words are considered while calculating the match score.
- **Euclidean ngram**: Calculating Euclidean distance using TF-IDF ngram vectors.
- **Diff Index**: We have sort all questions and found the difference in index among the question pair.So basic idea is if the question pair begins with different set of 'IS','WHO','WHERE','WHEN','HOW' etc then it is more likely that the questions are different. Though the idea seems right but score dropped by 0.01.

We have plotted the correlation matrix of features and we removed some features having relatively high correlation. The final features in our project are Match score,Frequencies q1,q2 , Cosine distance ngram,Difference in length,Jaccard similarity,fuzzy wuzzy,Bray-curtis dissimilarity,Euclidean ngram.
Correlation among our final features

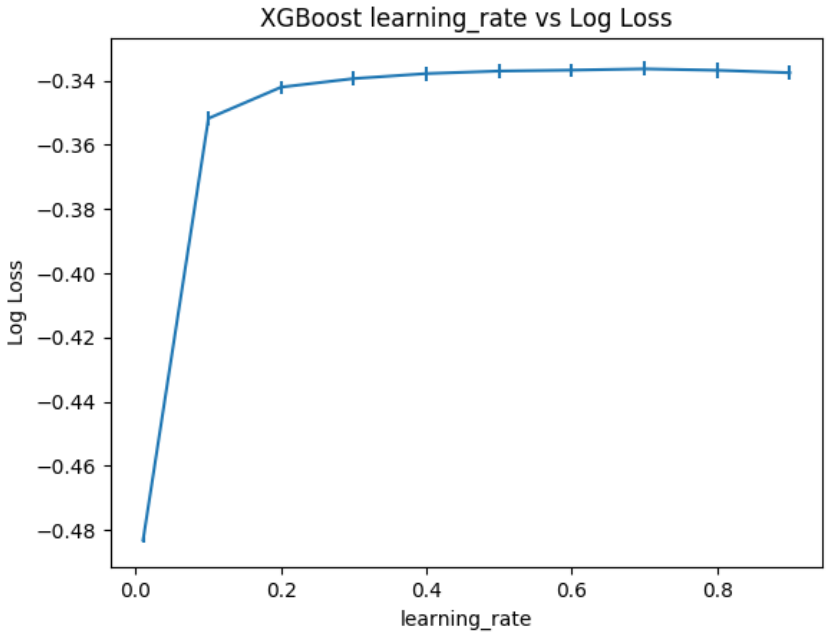
	match_score	q1_freq	q2_freq	cosine_dist_ngram	diff_length	jaccard	fuzz_ratio	bray_curtis_dist	euclidian_dist_ngram
match_score	1	0.104331	0.0682488	0.688059	-0.420395	0.831471	0.792622	-0.711686	-0.620172
q1_freq	0.104331	1	0.628938	0.0366694	-0.0717687	0.0655671	0.086363	-0.0450259	-0.0267432
q2_freq	0.0682488	0.628938	1	0.0310392	-0.0658245	0.0448552	0.0642822	-0.037933	-0.0248461
cosine_dist_ngram	0.688059	0.0366694	0.0310392	1	-0.323463	0.884184	0.799176	-0.991067	-0.976261
diff_length	-0.420395	-0.0717687	-0.0658245	-0.323463	1	-0.412902	-0.546797	0.373458	0.292171
jaccard	0.831471	0.0655671	0.0448552	0.884184	-0.412902	1	0.877746	-0.906848	-0.818988
fuzz_ratio	0.792622	0.086363	0.0642822	0.799176	-0.546797	0.877746	1	-0.836257	-0.734611
bray_curtis_dist	-0.711686	-0.0450259	-0.037933	-0.991067	0.373458	-0.906848	-0.836257	1	0.959988
euclidian_dist_ngram	-0.620172	-0.0267432	-0.0248461	-0.976261	0.292171	-0.818988	-0.734611	0.959988	1

3.3 Model Building

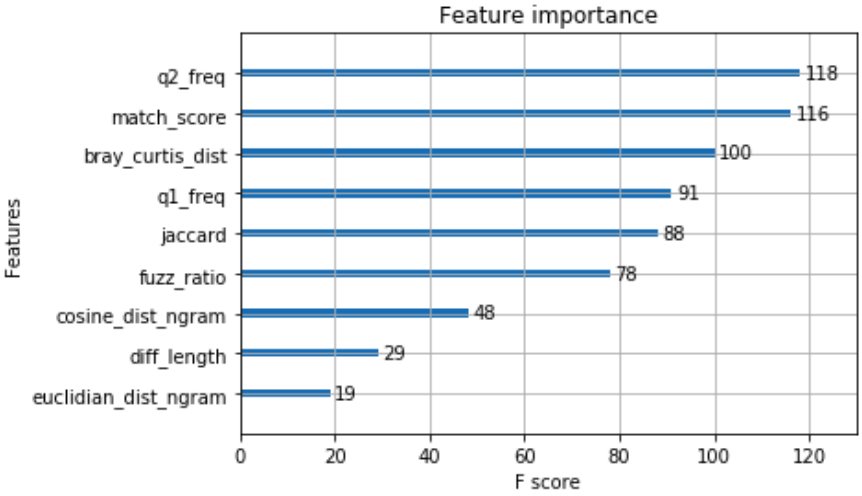
We used logistic regression at the beginning, because that is the basic algorithm for classification.Then we thought to try with XG Boost, which obviously gave better results than logistic regression.Then after doing some more feature engineering continued with XG Boost and then we tried on Random Forests.Log-loss given by the models are:

- XG Boost : 0.28503
- Random Forest : 0.67293

XG boost gave us the better results, we think it is because XG Boost is usually used for anomaly detection(finding that the 2 questions are different) and it is the best model if the train data is unbalanced, because boosting is a step by step approach and tries to correct errors made in previous iteration. So, we thought XG Boost is the best single model that can be used to solve this particular problem, but the cons of XG Boost is it takes a lot of time and also have a high chance of over fitting which is why we had to remove some pre-processing steps which are performed(after observing that we are getting better results removing it,but we couldn't figure out why).We choose the learning rate by doing grid search.The best learning rate turned out to be 0.7. Learning rate of 0.4 and above almost gave same log-loss.
Here is the plot of learning rate vs negative log-loss



The importance(F-score) of the features given by the model are

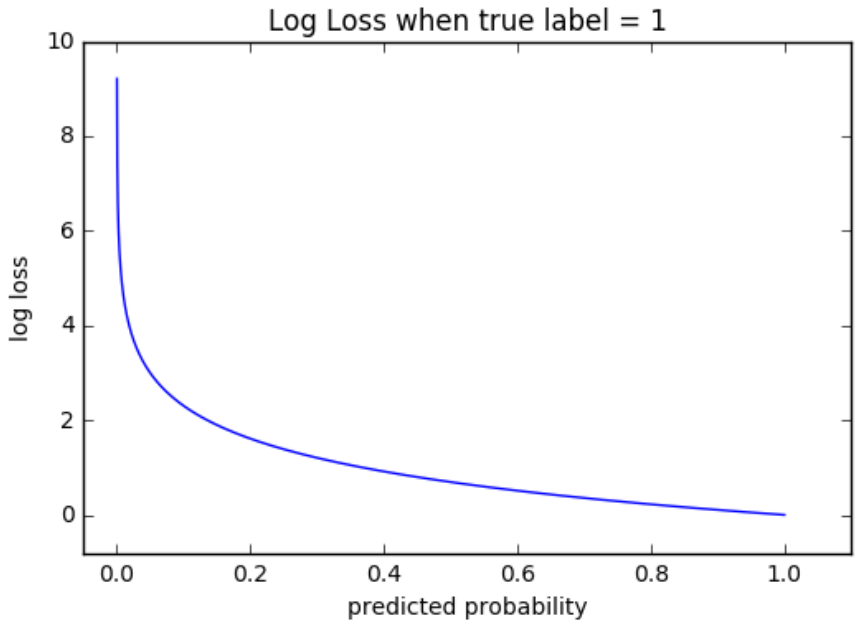


4 Analysis of results

- For pre-processing we filled NULL valued questions with "no" and dropped some columns with some inappropriate labelling.
- We used to_lower() to make the model case-insensitive.Removed stop words in some appropriate features.
- For features we used tfidf match score, match score,q1 frequency, q2 frequency, difference in length with respect to words, jaccard similarity, fuzz ratio.
- We also used tfidf n-gram vectors to compute cosine distance, euclidian distance and bray-curtis distance and the distances computed are added as features.
- We used XG Boost as model with learning rate of 0.7.

5 Metrics used

We are given to use log-loss as a metric for test data, so while doing validation to find best learning rate we used the same metric.Log loss will penalize heavily for incorrect predictions, so this is one of the good evaluation metric.



The graph clearly shows that penalizes exponentially.

Drive link for pickled model:

Drive link with enabled permissions : **Pickled Model Drive Link**