Quora Question Pair Similarity

Introduction

The amount of data we produce every day is truly staggering. It is believed that in today's setup, over 2.5 quintillion bytes of data are created every single day and it's only going to grow from there. Does this mean that the information we seek for can be obtained much easier? No, it's certainly not. The retrieval of valuable data from the knowledge base suffers from a much bigger challenge of "Information overload". Information overload is the excessive or redundant information available to someone who aims to complete a task or arrive at a decision. This impedes the decision-making process, resulting in a poor or sometimes no decision being made. Our project aims to tackle this challenge in one of the leading Knowledge sharing platforms- Quora.

The Problem

Quora is a Knowledge sharing platform that allows users to ask and answer questions based on a vast range of topics. It typically receives about a million questions each day, not all of which are original. Several of them have already been asked on Quora and have extensive responses. The purpose of this project is to figure out which of the offered pairs of questions contains two identical questions with the same meaning.

Allowing duplicates would degrade the quality of answers, significantly impacting the experience of the person asking the questions, the person answering the questions, and the person looking for an answer on the internet.

In an ideal world, Quora would utilize some "method" once a question is asked to locate a subset of its existing question database that contains questions that are "similar" to or around the same topic as the new question being asked. Once this subset has been discovered, Quora will use a machine learning algorithm to see if any questions in this subset are duplicates. If yes, it will alert the questioner and direct them to it; otherwise, it will ask the question.

Our role in this project is to work on the machine learning component, which entails pairing the new question with all of the other questions in the subset and using machine learning to see if any of the pairs are duplicates.

Example:

Suppose this already exists in the knowledge base:

'Astrology: I am a Capricorn Sun Cap moon and cap rising...what does that say about me?'

And a user is attempting to post this question:

'I'm a triple Capricorn (Sun, Moon and ascendant in Capricorn) What does this say about me?'

As we can see that, both the questions are identical but paraphrased differently, the machine learning component is expected to identify that this is a duplicate. Then, Quora will alert this user and direct them to the existing question.

Dataset:

The train and test data sets are available on the Kaggle competition website. **(https://www.kaggle.com/c/quora-question-pairs/data?select=test.csv.zip)**
The train set comprises 6 variables and 404,290 question pairs (rows) (columns). The row ID, question 1, question 2, question 1 ID, question 2 ID, and the class label, which is 0 for duplicate pairs and 1 for non-duplicate pairs, are the columns. Each question is identified by its question IDs (qids). The is_duplicate variable is our target variable that the model predicts, 0 means that the questions are duplicate and 1 means that the questions are not duplicate. Because the test set contains over a million question pairs and the actual outputs are not available for the users, we decided to focus just on the train set, on which I will execute a 70–30 train–test split.

Baselines and Evaluation metrics:

For the Baseline model, I decided to go with the 'most_frequent' strategy of the DummyClassifier. DummyClassifier is a classifier that makes predictions using simple rules. The strategy 'most_frequent' always predicts the most frequent label in the training set.
I have performed a 70-30 train test split on the training data and fit the training data onto the model. After this I tested the accuracy of the model on the test data and it comes upto 63.08%.

For the Evaluation metrics, we are going to use the classification report that gives the precision, recall, F1 score and accuracy of the model. We will use these values to compare between the proposed models.

For the dummy model, the classification report is as follows:

Experiments and Results

Loading Data And Getting Basics Stats:

We need to first load the data and analyze the basic structure of the data.

Let's examine the head of the data containing 5 rows.

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

As we can see, the data has 5 columns:
Id: It represents the id of the question pair question1 and question2.
Qid1: It contains an id associated with the 1st question.
qid2: It contains an id associated with the 2nd question.
question1: it gives the content of the first question.
question2: It gives the content of the 2nd question.
Is_duplicate: this is our target variable that tells whether the 2 questions are duplicate

We can describe the data using info() as follows:

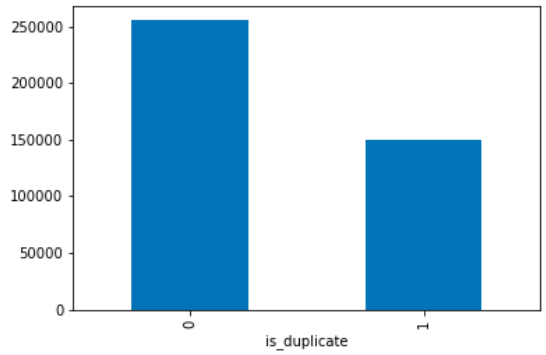
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 404290 entries, 0 to 404289
Data columns (total 6 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   id            404290 non-null  int64 
 1   qid1          404290 non-null  int64 
 2   qid2          404290 non-null  int64 
 3   question1     404289 non-null  object
 4   question2     404288 non-null  object
 5   is_duplicate  404290 non-null  int64 
dtypes: int64(4), object(2)
memory usage: 18.5+ MB
```

Here we can see that we have some missing value in question1 (1 Question) and question2 (2 Question)

Distribution of data points among output classes

Let's plot the data based on their value in is_duplicate.

```
<AxesSubplot:xlabel='is_duplicate'>
```



As we can see, there are about 150000 question pairs that are classified as duplicate and about 250000 question pairs that are classified as not duplicate.

Percentage of Similar question and Non-Similar Question:

We determine the non similar questions by extracting the number of rows with is_duplicate=0 and the number of similar questions by extracting the number of rows with is_duplicate=1
We divide these numbers by the total number of rows to find the percentage of each.

```
-->> Question pairs are not Similar (is_duplicate = 0):
     63.08%
-->> Question pairs are Similar (is_duplicate = 1):
     36.92%
```

Let's determine the questions that are repeated multiple times.
We see that the maximum number of times a single question is repeated is 157 with qid: 2559.

```
Total number of  Unique Questions are: 537933

Number of unique questions that appear more than one time: 111780 (20.78%)

Max number of times a single question is repeated: 157

2559      157
30782     120
4044      111
2561       88
14376      79
          ...
212185      1
210136      1
214230      1
220373      1
2047        1
Length: 537933, dtype: int64
```

Here we can see questions with qid [2559 ,30182 4044] are some the question that are repeated multiple times

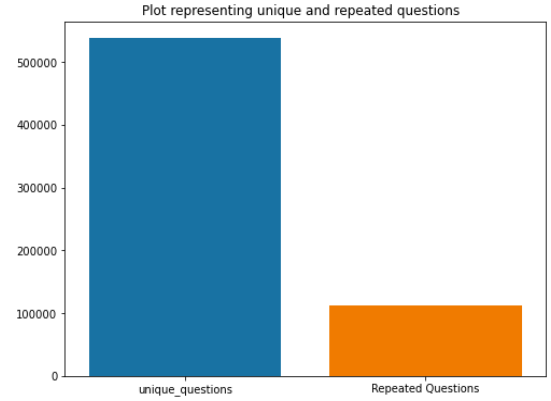
Lets have look at those questions one by one

```
['What are the best ways to lose weight?']
==================================================

['How is borderline personality disorder (BPD) treated?']
==================================================

['How can I lose weight quickly?']
```

Let us look at a plot representing unique and repeated questions:



As we can see, there are about 120000 repeated questions and 5400 unique questions.

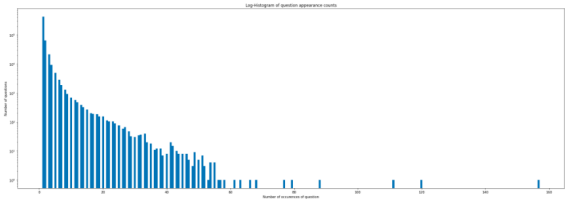
Let's check whether there are any repeated pairs of questions,

```
Number of duplicate questions :  0
```

As we can see, there are no pairs of duplicated questions.

Plotting Questions based on their frequency

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



Checking and removing null values:

```
df[df.isnull().any(1)] # Checking if any value is null in our dataset
```

	id	qid1	qid2	question1	question2	is_duplicate
105780	105780	174363	174364	How can I develop android app?	NaN	0
201841	201841	303951	174364	How can I create an Android app?	NaN	0
363362	363362	493340	493341	NaN	My Chinese name is Haichao Yu. What English na...	0

As we can see, there are only 3 rows with null values, hence we can completely remove these rows.

```
df.dropna(inplace=True)  # Droping Null Value
df[df.isnull().any(1)]
```

id	qid1	qid2	question1	question2	is_duplicate

Basic Feature Extraction

We add the following 3 basic features to our data set:

1. Q1_Len: length of Question 1
2. Q2_Len: length of Question 2
3. Q1_Words: Number of Words in Question 1
4. Q2_Words: Number of Words in Question 2
5. common_Word: We are converting both Question 1 and Question 2 to set (and also converting them to lower so that every word has the same format) and finding their intersection so that we can get common words. Then we are simply finding the length of those common words.
6. word_Total: Converting them to lower form then removing extra spaces and them removing the repeated words by converting them to sets. Then finding Length of Both the questions and adding them to find total words in both.
7. Shared_Word: Here we are finding the total number of shared words and dividing by the total number of words [(A intersection B)/A+B].

The resulting dataset with the additional columns for these features will then look as follows:

	id	qid1	qid2	question1	question2	is_duplicate	Q1_Len	Q2_Len	Q1_Words	Q2_Words	common_Word	word_Total	Shared_Word
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0	66	57	14	12	10	23	0.434783
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0	51	88	8	13	4	20	0.200000
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0	73	59	14	10	4	24	0.166667
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0	50	65	11	9	0	19	0.000000
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0	76	39	13	7	2	20	0.100000

Analysing our extracted features

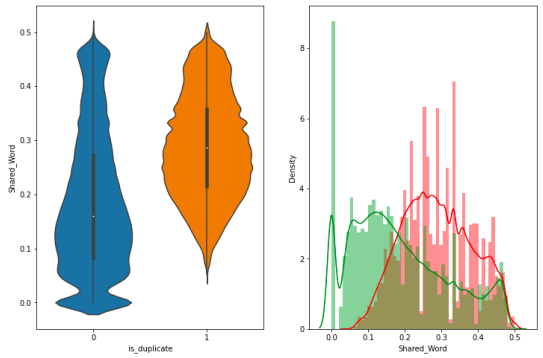
```
Minimum length of the questions in question1 :  1
Minimum length of the questions in question2 :  1
Number of Questions with minimum length [question1] : 69
Number of Questions with minimum length [question2] : 25
Maximum length of the questions in question1 :  125
Maximum length of the questions in question2 :  237
Number of Questions with minimum length [question1] : 1
Number of Questions with minimum length [question2] : 13
Maximum number of Common word :  41
Maximum number of Shared Word :  0.5
```

Analyzing Shared Word

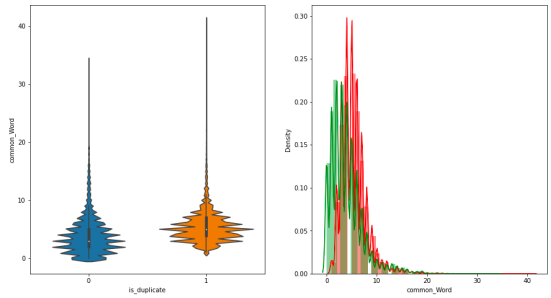
Let us plot a violin plot with x axis corresponding to shared words and y axis corresponding to is_duplicate.

Let us plot a dist plot of Shared_Word grouped by the value of is_duplicate.

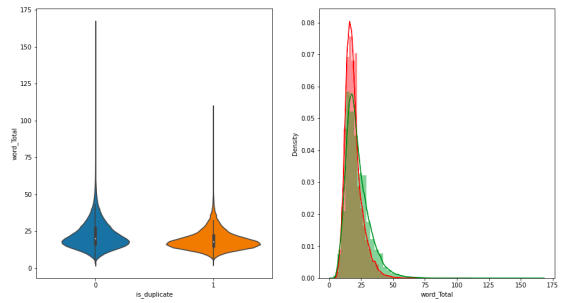
The resulting plot are as follows:



Let's repeat the same task for common_Word:



For word_Total, the corresponding plots are as follows:



Text pre-processing

This step involves Creating our very own stopword list after removing some stop words like how,whom ,not , etc that may be useful to differentiate between questions.

We create our custom stopwords, consisting of the following words:

```
from nltk.stem import SnowballStemmer
snow=nltk.stem.SnowballStemmer('english')

# Creating custom stopwords

stopwords= set(['the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're',\
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't",\
            'won', "won't", 'wouldn', "wouldn't"])
```

We define a simple function to perform stemming while removing StopWords

```
def removeStopWord(word):
  token=word.split(" ")  ## coverting string to token (list of word) \\ like ["this","is","token"]
  removestop=[snow.stem(x) for x in token if x not in stopwords]  ##removing stopwords and also doing Stemming
  removed=" ".join(removestop)  ##joing back the list into sentence
  return removed
```

We convert the string into a token, then we remove the stopwords while also performing stemming, finally joining the list back into our original sentence.
We also replace common words like 1000 to 1k or 1m and many others and remove special characters.

After preprocessing the data, we look at how pre-processing changed our question text.

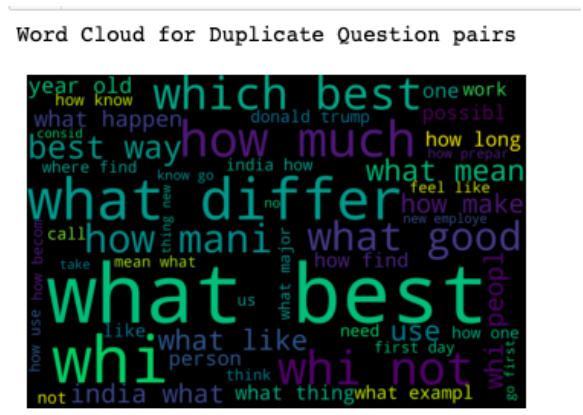
Advance Feature Engineering Using NLP and Fuzzy Features

We use the following features to better describe our database:

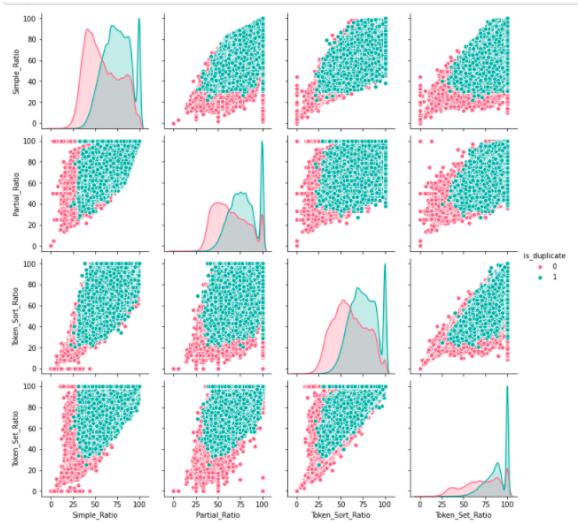
1. Simple Ratio : Measurement of edit distance (Minimum number of edits required to convert one sentence to other)
2. Partial Ratio : How much accurately a part of sentence match to other sentence ("Chennai Super Kings", "Super Kings")
3. Token Sort Ratio : Tokenizing the string in question, sorting the tokens alphabetically, and then joining them back into a string

4. Token Set Ratio : Tokenize both strings,split the tokens into two groups of intersection and remainder. We use those sets to build up a comparison string.
5. Last_Word : Checks if last word is same in both Q1 and Q2
6. First_Word : Checks if First word is same in both Q1 and Q2
7. Length_diff : Finds the length difference between Q1 and Q2
8. StopWord_Ratio : Number of stopwords in both Questions
9. Token_Ratio : Number of tokens in both Questions
10. Longest_Substr_ratio : Ratio of the Longest Substring that is found in between Q1 and Q2

WordCloud and Pairplot for our database



To analyse the pairwise relationships between 'Simple_Ratio', 'Partial_Ratio', 'Token_Sort_Ratio', 'Token_Set_Ratio', 'is_duplicate', we create a pairplot between them:



Featurizing text data

We use the TfidfVectorizer and CountVectorizer to vectorize our dataset.

We perform this by merging questions of both Q1 and Q2 to a single list in which the first 404287 index will be of question 1 and then the rest of question 2. We then convert our text to a matrix of TF-IDF features. After this we Map out feature_names with their respective tf-idf score (dict key:word and value:tf-idf score)

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.

The final dataset has the following columns:

- ID
- Is_duplicate

- Q1_len
- Q2_len
- Q1_words
- Q2_words
- Common_word
- word_Total
- Shared_word
- Simple_ratio
- Partial_ratio
- Token_sort_ratio
- Token_Set_ratio
- Last_word
- First_word
- StopWord_ratio
- Token_ratio
- Long_substr_ratio

Once the final dataset with all our desired features is obtained, we move on to train them on Machine Learning Models.

Machine Learning Models

The aforementioned data is fed, trained and predicted using the three models, namely, Logistic Regression, Support Vector Machine and XGBoost.

Logistic Regression

Logistic regression is a statistical analysis approach for predicting a data value based on previous data set observations. The method enables a machine learning application to classify incoming data using an algorithm based on historical data.
A logistic regression model analyzes the relationship between one or more existing independent variables to predict a dependent data variable. A logistic regression, for example, could be used to predict whether a political candidate will win or lose an election, or if a high school student would be accepted to a specific institution. Multiple input criteria can be taken into account by the analytical model that results. It assesses new cases on their likelihood of falling into a certain outcome group based on historical data about previous outcomes involving the same input criteria.

Linear Support Vector Machine

Linear SVM is the newest ultra-fast machine learning (data mining) approach for handling multiclass classification problems from extremely huge data sets. It uses a unique proprietary version of a cutting plane algorithm for creating a linear support vector machine. LinearSVM is a linearly scalable method, which means it generates an SVM model that scales linearly with the size of the training data set in terms of CPU time. When compared to other known SVM models, it obviously outperforms them when high accuracy is necessary. We'd be grateful if you could share the results of LinearSVM on your data sets with us.

XGBoost

XGBoost is a distributed gradient boosting toolkit that has been tuned for efficiency, flexibility, and portability. It uses the Gradient Boosting framework to create machine learning algorithms. XGBoost is a parallel tree boosting (also known as GBDT, GBM) algorithm that solves a variety of data science issues quickly and accurately. The same algorithm may tackle problems with billions of examples in a distributed environment (Hadoop, SGE, MPI).

Deep Neural Networks

Artificial neural networks (ANNs) and simulated neural networks (SNNs) are a subset of machine learning that are at the heart of deep learning methods. Their name and structure are derived from the human brain, and they resemble the way biological neurons communicate with one another.

The **tensorflow** library gives us easy to go functions to build neural networks in just a couple of lines. We have built two models, one for each question. Both of the LSTM networks are then merged to build a single model. The model is being optimized using the Adam Optimizer. Adam optimizer involves a combination of two gradient descent methodologies: Momentum: This algorithm is used to accelerate the gradient descent algorithm by taking into consideration the 'exponentially weighted average' of the gradients. Using averages makes the algorithm converge towards the minima in a faster pace. Binary Crossentropy is used as a loss metric. The model is then fit and trained on the training set and evaluated using the test set. The model summary and the training process of each epoch is shown below.

```
Epoch 1/10
356/356 [==============================] - 67s 154ms/step
accuracy: 0.7632
Epoch 2/10
356/356 [==============================] - 53s 150ms/step
accuracy: 0.7768
Epoch 3/10
356/356 [==============================] - 53s 150ms/step
accuracy: 0.7852
Epoch 4/10
356/356 [==============================] - 53s 150ms/step
accuracy: 0.7918
Epoch 5/10
356/356 [==============================] - 53s 150ms/step
accuracy: 0.7930
Epoch 6/10
356/356 [==============================] - 53s 150ms/step
accuracy: 0.7914
Epoch 7/10
356/356 [==============================] - 53s 150ms/step
accuracy: 0.7903
Epoch 8/10
356/356 [==============================] - 53s 150ms/step
accuracy: 0.7866
Epoch 9/10
356/356 [==============================] - 53s 150ms/step
accuracy: 0.7872
Epoch 10/10
356/356 [==============================] - 53s 150ms/step
accuracy: 0.7929
```

The accuracy is coming upto 79.29% which is the highest of all the methods that we had tried.