Quora Question Pair Similarity
ISM 6136 – Data Mining Project

Team 5

Sai Kiran Batchu
Subash Chandra Biswal
Sukumar Vinnakota
Kushal Reddy Chavva
Naresh Goud Aakula

GitHub: sbiswal14/BAIS: Masters Projects (github.com)

Text Analytics:

Text analytics is the process of translating large volumes of unstructured text or text that does not have a predefined format to gain insights and patterns that create a business value. It utilizes statistical, linguistic and machine learning techniques like sentiment analysis, similarities, named entity recognition, topic modelling, event extraction, and term frequency in the process.

Using text analytics businesses are able to derive valuable insights from all forms of unstructured data like social media posts, customer feedback, emails, chats, reviews, etc, and improve customer satisfaction, product defects, and assess brand reputation among other things.

Problem Statement:

Text Analytics is being used by many businesses like Google, Quora, Amazon, etc., to tackle problems like identifying the search-related results, identifying similar questions, recommending products based on the consumer behaviour using their search history, social media activity etc., to name a few. Our problem is focused on identifying similar Questions on the Quora platform. Quora is a Q&A website where people post, search and follow questions related to their interests.

Quora has over 300 million users each month posting questions on over 400,000 subjects. It is no surprise that most of the questions have similar words or contexts or meanings and the user has to navigate more questions of similarity for a better answer. Even writers cannot answer the same questions again and again. Identifying similar questions, which is the subject of our research, can reduce the problem by allowing the platform to give the best results for the user.

Following metrics are used to measure the success of the predictions.

- **Log Loss** :- $\log P(y \mid p) = -(y\log(p)+(1-y)\log(1-p))$

  y = true label
  p = predicted probability of a pair being duplicate

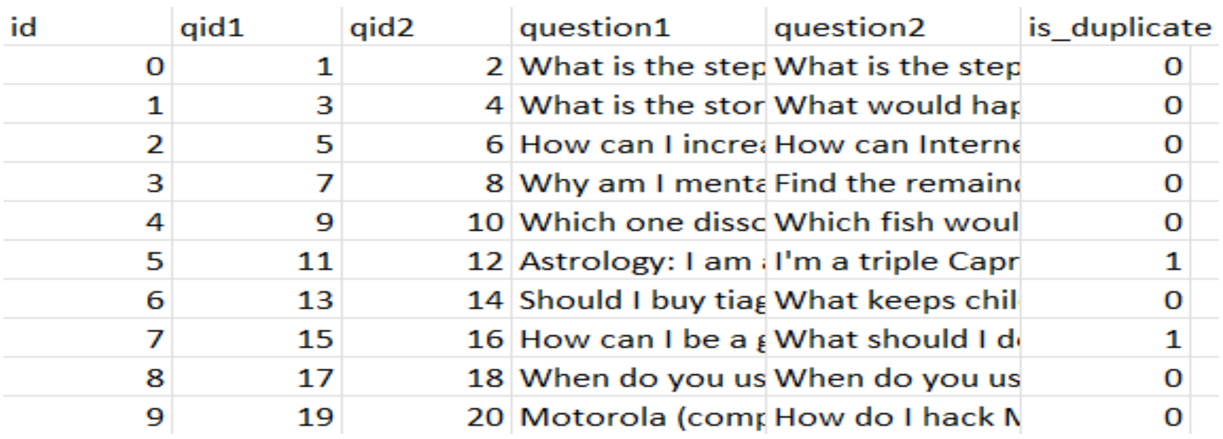
- **Accuracy**

Data and Data Characteristics:

The dataset used for the problem is taken from Kaggle. It consists of 404,290 rows and 6 columns. The description of each column is as follows

- Id: unique Id for each instance
- qid1: unique Id for question1 of the question pair
- qid2: unique Id for question2 of the question pair
- question1: the context of question1
- question2: the context of question2
- is_duplicate: the target variable, set to 1 if question1 and question2 have essentially the same meaning, and 0 otherwise.

The dataset has been split into ***95% and 5% for training and testing*** respectively.



id	qid1	qid2	question1	question2	is_duplicate
0	1	2	What is the step	What is the step	0
1	3	4	What is the stor	What would hap	0
2	5	6	How can I increa	How can Interne	0
3	7	8	Why am I menta	Find the remain	0
4	9	10	Which one dissc	Which fish woul	0
5	11	12	Astrology: I am	I'm a triple Capr	1
6	13	14	Should I buy tiag	What keeps chil	0
7	15	16	How can I be a	What should I d	1
8	17	18	When do you us	When do you us	0
9	19	20	Motorola (comp	How do I hack N	0

Figure 1: First 10 instances of the dataset

Exploratory Data Analysis:

The dataset consists of 63.08% (255,027) instances with class variable 0 and 36.91% (149,263) instances with class variable 1. The maximum length of question1 is 623 characters and the question2 is 1169 characters.

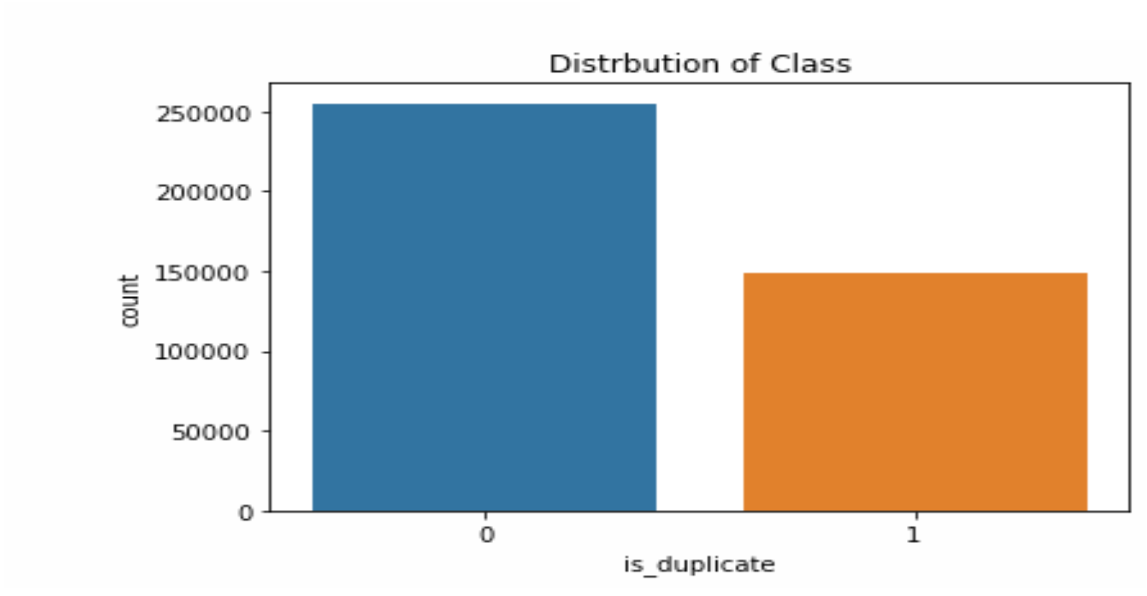


Figure 2: Count plot of the class variable of the data

The histogram of the character length distribution shows that there is not much difference between the distribution of question1 and question2.

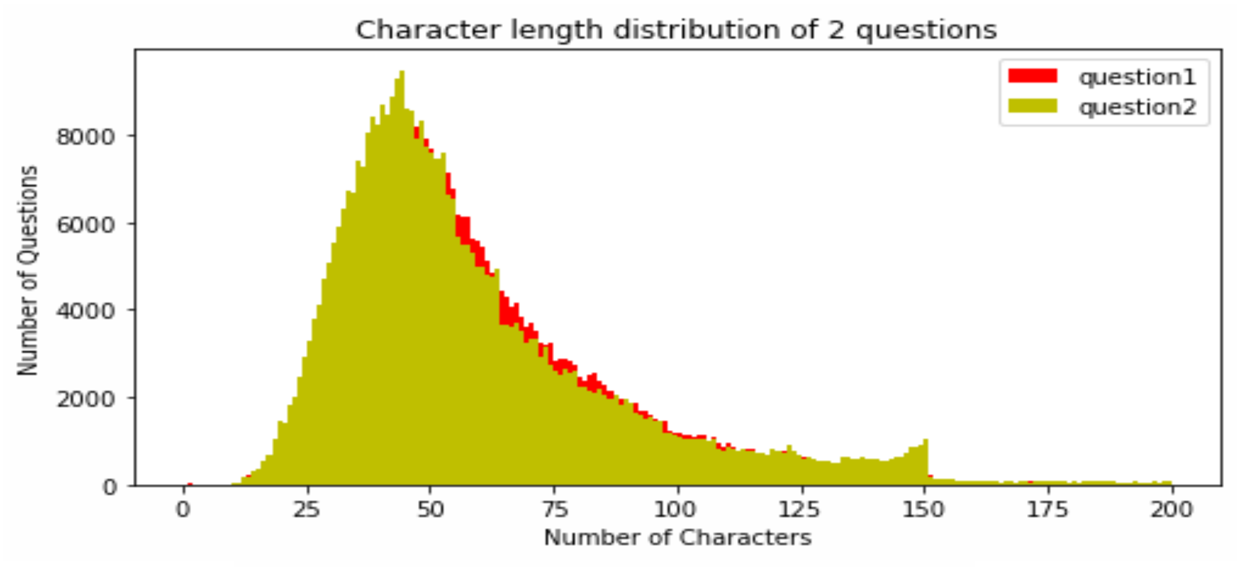


Figure 3: Character Length Distribution of Question1 and Question2

Data Cleaning:

Cleaning and Modelling are done using python. Data is cleaned using the below techniques.

1. **Lowercase**:

   Question1 and Question2 of all the instances are converted into lowercase using the *lower* function of the Python string object.

2. **Tokenize:**

   Tokenization is the process of converting a sentence into individual words called tokens. Using the python Natural Language Tool Kit (NLTK) library's *word_tokenize* function all the instances of question1 and question2 are tokenized.

3. **Removing Stop Words:**

   Stop words are the most common words in text data. They are words that shouldn't be part of the content. For example. I, me, myself, while, until, during, should, could etc., There are 40 predefined stop words in python's NLTK library.

Data is cleaned using the *stopwords.words("english")* function of NLTK library's corpus package.

4. **Lemmatization:**

   Lemmatization aims to remove the influential endings and returns to the dictionary form of a word known as the **lemma**. For example, the lemma of the word Saw would come out as See. Lemmatization is done using python's NLTK library.

Data is cleaned using the *WordNetLemmatizer* function of the NLTK library.

5. **Stemming:**

   Stemming is the crude heuristic process of chopping off the ends of words. For example, stemmed caresses word would be caress. Stemming is done using python's NLTK Library.

   Data is cleaned using the *PorterStemmer* function of the NLTK library's stem package.

Feature Extraction:

The following features are used for modelling.

1. **Cosine Similarity:**

   Question1 and Question2 are converted to vectors of token/term counts using Scikit-learn's CountVectorizer function. The vector is made up of counts of repeated words and n-grams of 2 to 6.

   Cosine similarity is used to find the similarity between two vectors. Using the count vectorized vectors generated above, Cosine similarity is found and used as a feature in the modelling.

2. **Difference in Character Length:**

   The difference in character length of question1 and question2 is the other feature used in modelling.

After the features are extracted from the data then it is Split into 95% training and 5% testing data using the *train_test_split* function of Scikit-learn's model selection package.

Modelling:

Data modelling is done using the models Logistic Regression, Random Forest and XGBoost.

1. **Logistic Regression:**

   Although the name regression Logistic Regression is a classification model which predicts the probability of a discrete outcome given the independent variables. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on. Logistic Regression uses the Logit function to predict the probabilities for the class variable.

   Scikit-learn's linear_model package has a LogisticRegression function which is used in training the model with default parameters. The model is first trained on the training set and then tested using the testing set.

2. **Random Forest:**

   Random Forest is a classification algorithm which is built upon decision trees using the bagging ensemble technique of machine learning. In a Random Forest, multiple decision trees (number specified by n_estimators) are built up to a certain depth(number specified by max_depth). These trees' outcome of predicting a certain class variable given the independent variables is averaged and classified for each instance.

   Scikit-learn's ensemble package has a RandomForestClassifier function which is used in training the model with the hyperparameters

   n_estimators = 700
   max_depth = 8

   GridSearch and 10 fold cross validation is the technique used in hyperparameter tuning. For this GridSearchCV and StratifiedKFold functions of the scikit-learn library are used.

3. **XGBoost:**

   eXtreme Gradient Boosting also known as XGBoost is a machine learning algorithm built on the ensemble technique of Boosting. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. Boosting is a process in which the model is built correcting previous iteration errors in each new iteration of a decision tree. The model makes its predictions using the final tree.

In XGBoost multiple decision trees (number specified by n_estimators) are built up to a certain depth(number specified by max_depth) correcting the error of each iteration using the learning rate.

XGBClassifier of python's XGBoost library is used in training the model with the hyperparameters

n_estimators = 500
max_depth = 4

GridSearch and 10 fold cross validation is the technique used in hyperparameter tuning. For this GridSearchCV and StratifiedKFold functions of the scikit-learn library are used.

Results:

The table of Model type, Log Loss and Accuracy obtained for each model is as below.

Model Type	Log Loss	Accuracy
Logistic Regression	**0.5715**	**0.6571**
Random Forest	**0.5422**	**0.6792**
XGBoost	**0.5404**	**0.6791**

Table 1: Performance of the models on Log Loss and Accuracy metrics

Low log loss value and more accuracy value for the model are desired. The log loss values are 0.5715, 0.5422, 0.5404 and Accuracy values are 0.6571, 0.6792, and 0.6791 for Logistic Regression, Random Forest and XGBoost respectively.

Logistic regression performed the least compared to the other 2 models. Random Forest and XGBoost performed almost similar with the Random Forest model having a minuscule edge over XGBoost.

The Random Forest Model has a prediction accuracy of 67.92% i.e. there is a 67.92% chance that the model's predicted class value is correct and the Log loss value of 0.5422 depicts how close we are to the actual class value. These values give us the similarity between question1 and the question2 of the dataset.

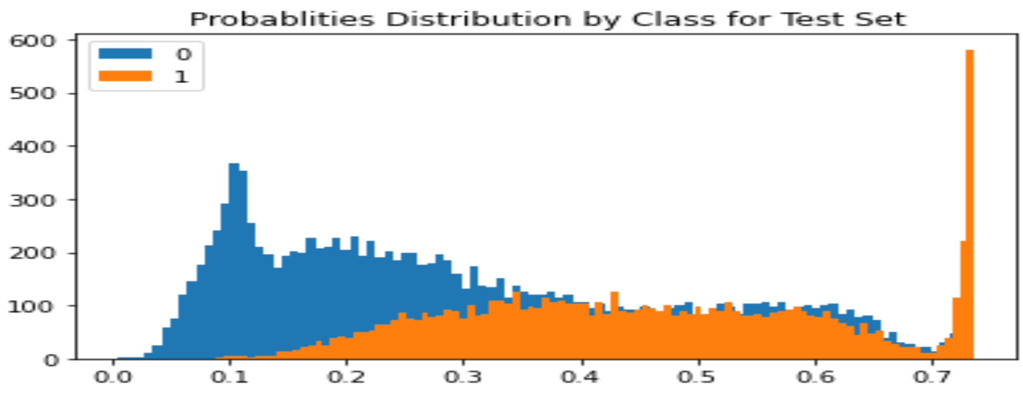


Figure 4: Probabilities distribution by class for test set in Logistic Regression

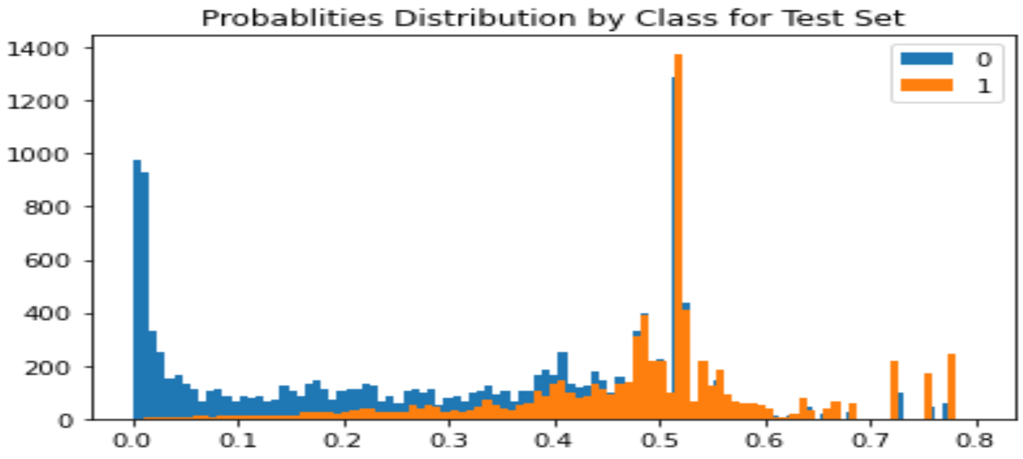


Figure 5: Probabilities distribution by class for test set in Random Forest

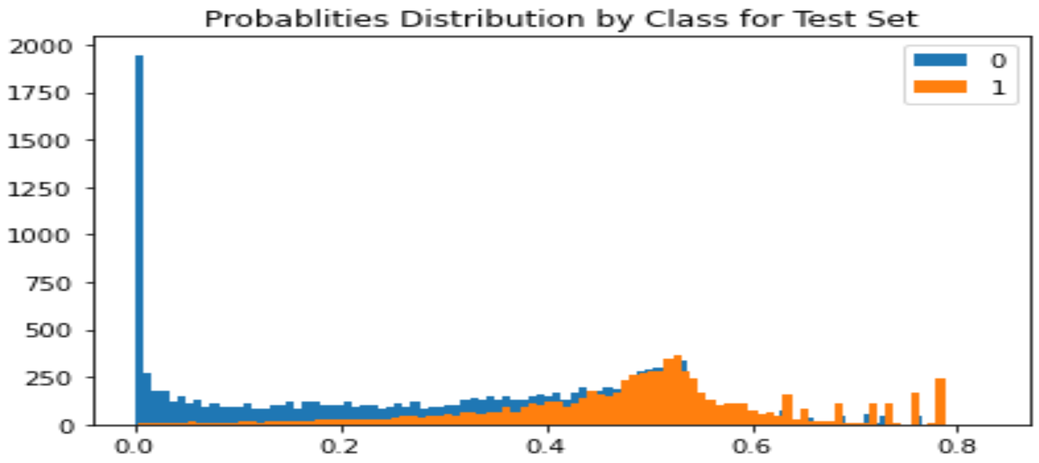


Figure 6: Probabilities distribution by class for test set in XGBoost

The probabilities distribution by dependent variable on the testing set for each of the models shown above depicts that there is a lot of variability in the distribution of the Logistic Regression model which means that the model is not performing as good as Random Forest or XGBoost.

The Random Forest is performing the best in identifying the similarities between each question pair. This model can be used by Quora to eradicate the problem of duplicate questions to a greater extent.

Future Scope:

Deep Learning techniques like Sentence embeddings based on Neural Network Architecture, Word embeddings and Long Short-Term Memory based Neural Network Architecture and Convolutional Neural Networks would probably give better results in predicting similar questions and can be utilised on the Quora platform.