Quora question Pair similarity

# **Introduction**

In this project we will be dealing with the task of pairing up the duplicate questions from Quora. More formally, the followings are our problem statements.

* 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.

**Note-** we are talking about the semantic similarity of the questions.

## **Project Overview:**

# Quora is a question and answers platform and builds around a community of users to share knowledge and express their opinion and expertise on a variety of topics.

# Quora is an emerging site for quality content, launched in 2009 and it has multiple unique topics and domain experts as its user so that the user gets the proper/desired information from the experts in the field.

# With the growing repository of the knowledge base, there is a need for Quora to preserve the trust of the users, maintain the content quality, by discarding the junk, duplicate information.

# Quora has overcome this challenge by organizing the data effectively by using a modern Data science approach to eliminate question duplication.

**Type of Machine Learning Problem:**

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

**Problem Statement:**

* Given a pair of questions, the goal is to predict whether they are semantically similar or not. This is a binary classification problem, where the system should output a similarity score ranging from 0 (not similar) to 1 (highly similar).
* Identify which questions asked on Quora are duplicates of questions that have already been asked.

**Business Objectives and Constraints:**

* The cost of a mis-classification can be very high.
* You would want a probability of a pair of questions to be duplicates so that you can choose any threshold of choice.
* No strict latency concerns.
* Interpretability is partially important.

# **Dataset Description:**

The data is in a csv file named “Train.csv.”

* Train.csv contains 5 columns: qid1, qid2, question1, question2, is\_duplicate.
* Number of rows in Train.csv = 404,290.
* Number of Non-Duplicate data points – 255027 (Class: 0).
* Number of Duplicate data points – 149263 (Class: 1).
* We have 63.08% of non-duplicate pairs and 36.92% duplicate pairs.

‘qid1’ and ‘qid2’ are the ids of the respective questions.

* ‘question1’ and ‘queston2’ are the question bodies themselves and ‘is\_duplicate’ is the target label which is 0 for non-similar questions and 1 for similar questions.
* ‘question1, question2,’ are the x labels and ‘is\_duplicate’ is the y labels for Machine learning problem.

# **Performance metrics:**

Metrics we are going to use in this project. They are

**Confusion Matrix**: Confusion matrix will provide us several metrics like TPR, FPR, TNR, FNR, Precision and recall.

TP – True Positive FP – False Positive TN – True Negative FN – False Negative

**Classification Report**: It is a performance evaluation metric in machine learning which is used to show the precision, recall, F1 Score, and support score of your trained classification model.

We will not use the accuracy metric because there is huge imbalance in the dataset. Confusion matrix and Classification Report are the best metrics used for these data.

**Phase-1:**

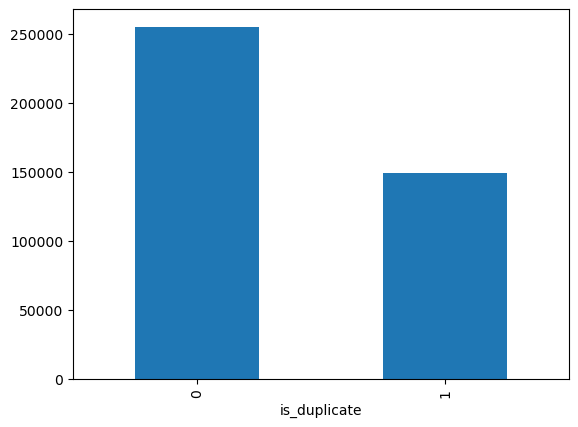
**Understanding the data:**

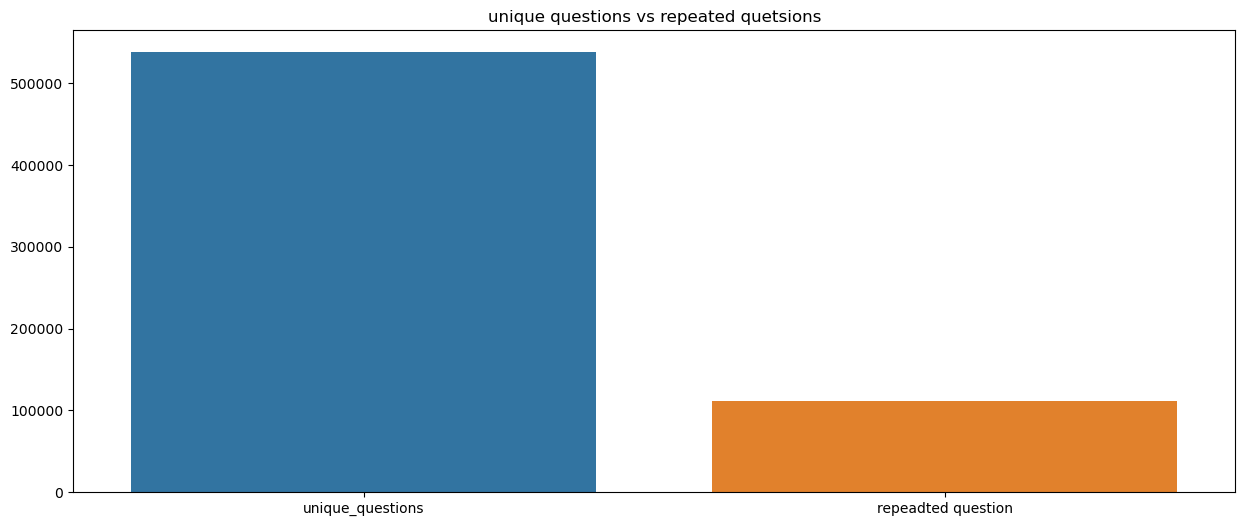
* Data understanding involves accessing the data and exploring the data is the most important like loading the file, Inspecting the data, checking the shape of the data, checking the info of the data that means datatype of the columns has assigned correctly and finally describing the data to understand the mean, median, count etc.,

**Exploratory Data Analysis:**

* These are the Data Cleaning steps have been performed on the data

1. Checking the duplicates.
2. Checking Null values.
3. Checking corrupted data.
4. Did some visualizations on the data Identifying the imbalance using the is\_duplicate variable.



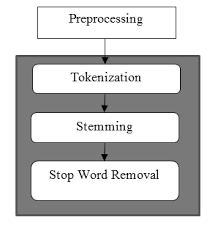


1. Identifying theTotal number of Unique Questions are: 537933
2. Number of unique questions that appear more than one time: 111780 (20.77953945937505%)
3. Max number of times a single question is repeated: 157.

# **Pre-processing of Text:**

Data cleaning and data pre-processing are one of the most important steps in this process. In the data pre-processing stage, we clean up each row’s data.

* Convert text to lower case.
* Removing HTML tags.
* Removing punctuations.
* Expanding Contractions or de-contract words.
* Removing special characters and digits.
* Removing Stop words.
* Performing Stemming and Lemmatization.
* Change abbreviations to its original terms.
* Replace certain numerical values with strings (E.g.: 1,000,000 with 1m)



**Token**: You get a token by splitting *sentence a space*

**Stop Word:** stop words as per NLTK.

**Word**: A token that is not a stop word.

As the above text preprocessing applied and up to these then we have built the bag of words and TF-IDF models.

* Performed the train-test split, converting the text to vectors using the Bag of words and TF-IDF and then built the models on the data.
* Performed the predictive analysis with the model have been trained by sending the test data.
* Evaluation the predicted data using the metrics how accurate they are performing.

**Note: These have been done when we are performing the Word2vec and Glove.**

## **Part-of-speech tagging:**

* Part-of-speech tagging (**POS tagging**) is the task of tagging a word in a text with its part of speech.
* A part of speech is a category of words with similar grammatical properties. Common English parts of speech are noun, verb, adjective, adverb, pronoun, preposition, conjunction, etc.

We have used the pretrained word2vec model which is **word2vec-google-news-300.**

This is downloaded pretrained word2vec from the genism. Downloader and the model size is 1.6gb.

Using these pretrained we have list of words these genism model converts each word in the sentence to a 300-dim vector space.

After generating the vectors, we have extracted the distances feature using these vectors.

**Advanced Feature Extraction (NLP, Fuzzy Features and Distances):**

* **Fuzz ratio:** This ratio uses a simple technique which involves calculating the edit distance (Levenshtein distance) between two strings.
* **Fuzz partial ratio:** The partial ratio helps us to perform substring matching. This takes the shortest string and compares it with all the substrings of the same length.
* **Token sort ratio:** The strings are tokenized and pre-processed by converting to lower case and getting rid of punctuation. The strings are then sorted alphabetically and joined together. Post this, the Levenshtein distance similarity ratio is calculated between the strings.
* **Token set ratio:** Token set ratio performs a set operation that takes out the common tokens instead of just tokenizing the strings, sorting, and then pasting the tokens back together. Extra or same repeated words do not matter.
* **Fuzz token sort ratio:** Fuzzy Wuzzy token sort ratio raw raw score is a measure of the string’s similarity as an int in the range [0, 100].
* **Fuzz token set ratio:** The output of the token sort ratio comes to be 85 while that of the token set ratio comes to be 100 as the token set ratio doesn't take into account the repeated words.

**Distances**:

* **cosine distance:** The measure computes the cosine of the angle between vectors question1 vector and question2 vector.
* **city block distance:** It is also called as Manhattan Distance and it calculates the distance between two real-valued vectors.
* **Euclidean distance:**  Euclidean distance between two points in the Euclidean space is defined as the length of the line segment between two points.
* **Minkowski distance**: It is a metric in a normed vector space which can be considered as a generalization of both the Euclidean distance and the Manhattan distance.
* **Jaccard distance:** Jaccard distance measures the dissimilarity between data sets and is obtained by subtracting the Jaccard similarity coefficient from 1.
* **Canberra distance:** Canberra distance is a weighted version of the Manhattan distance.

**Train and Test split:**

* We use the train and test by randomly splitting in the ratio of 80:20 we choose as we have sufficient points to work with.

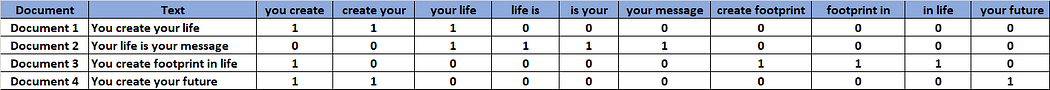
# **Numerical Representation of Text Data:**

**Bag of Words (Bow):**

A bag-of-words Model is a quite simple and flexible technique to represent a text as a numerical vector.

* A vocabulary of known words.
* The measure of the presence of known words.
* Bag of word’s can be used for both Categorical and Text Data.
* Either 1 or 0 to just check whether the word occurs or not, this implementation of Bow’s is called Binary Bag of Words.

To Represent a text as a vector, we can write the number of occurrences of the two consecutive words against the corresponding tokens:



* Bag of Words just creates a set of vectors containing the count of word occurrences in the document.
* Bag of Words vectors are easy to interpret.

**Disadvantages:**

* Curse of Dimensionality problem (Vocabulary size) because it develops the sparse matrix.
* Numerical representation is not captured by the bag of words.
* Semantic Meaning or similarity is not captured.
* These do not capture the sequence order of the words.

**Term Frequency-Inverse Document Frequency (TF-IDF):**

* Term Frequency-Inverse Document Frequency “a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus”.
* This is another technique to convert a text into a numerical vector.

**Term Frequency (TF):**

* Term Frequency of a word is a measure of the frequency of a word in a Document.

*TF is defined by the formula:* A screenshot of a computer

Description automatically generated with low confidence

**Inverse Document Frequency (IDF):**

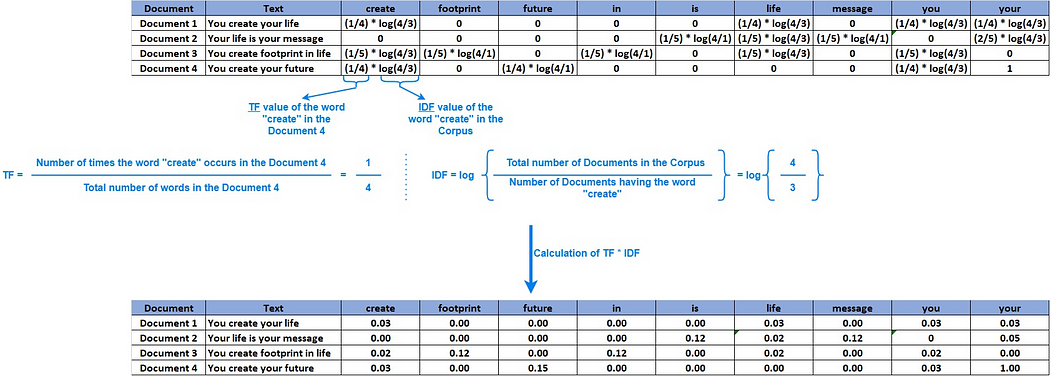
* IDF is a measure of how important a word is.

IDF is defined by the formula:

A picture containing text, font, line, screenshot

Description automatically generated

* In Bow’s, we created vectors by filling the dimensions with the number of occurrences of a word.
* In TF-IDF, we create the vectors by multiplying the Term Frequency and Inverse Document Frequency of a word. Please see the below table using TF-IDF values of the words:



* In **TF-IDF**, more importance is given to the words that are frequent in a Document and also, to the rare words in a Corpus.
* We can even use only the **Inverse Document Frequency** alone to generate vectors depending on the problem.

**Advantages:**

1. TF-IDF contains information on the more important words and the less important ones as well and it will retrieve the information on the data.
2. TF-IDF usually performs better in machine learning models.

**Disadvantages:**

* Curse of Dimensionality problem (Vocabulary size) because it develops the sparse matrix.
* Semantic similarity or Numerical representation is not captured by the TF-IDF.
* These do not capture the sequence order of the words.

**Word2Vec:**

Word2Vec or Word to Vector, also denoted by W2V, is a state-of-the-art (SOTA) technique to convert a word into a vector by considering the semantic meaning of the words, unlike Bow’s and TF-IDF.

* With Word2Vec, a word is transformed into a numerical dense vector (not sparse vector as retrieved in Bow and TF-IDF) with d-dimensions typically 50, 100, 200, 300, etc.
* If two words are semantically similar, then the vectors of these words are closer geometrically.
* If two words are semantically similar, then their vectors will be closer.W2V retains/satisfies relationships between words.

*Semantic relationship in Word2Vec*

A diagram of a verb tense

Description automatically generated with low confidence

We can use any pre-trained Word2Vec Model directly. For example, we can use Google’s Word2Vec Model trained on **Wikipedia data**, which contains 300-dimensional vectors for 100 billion words.

We could convert a Document to a vector in two ways:

**Average Word2Vec:**

* In Average Word2Vec, we take the average Word2Vec values of all the words in a Document to get the vector for a Document.
* It is a simple way to leverage the Word2Vec concept to build vectors for texts.
* Average Word2Vec works fairly well but is not perfect.

Calculating the distances fuzzy wuzzy and cosine, city block or Manhattan, Euclidean etc.,

**Advantages:**

* Able of capturing relationships between different words including their syntactic & semantic relationships
* The size of the embedding vector is small & flexible and size of vocabulary is small.

# **Disadvantages:**

* Word2Vec cannot handle out-of-vocabulary words well.
* It assigns a random vector representation for OOV words.
* It relies on local information of language words.
* The semantic representation of a word relies only on its neighbours & can prove suboptimal
* Parameters for training on new languages cannot be shared.
* If you want to train word2vec in a new language, you have to start from scratch.

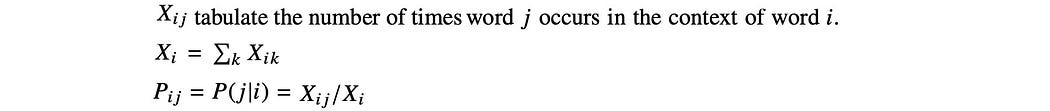
### **Building the Word2Vec model using Genism:**

To create the word embeddings, you can use the following respective:

Using word2vecVector size = 300 and min count =1

**Glove (Global Vectors):**

Glove is another word embedding method. But it uses a different mechanism and equations to create the embedding matrix.

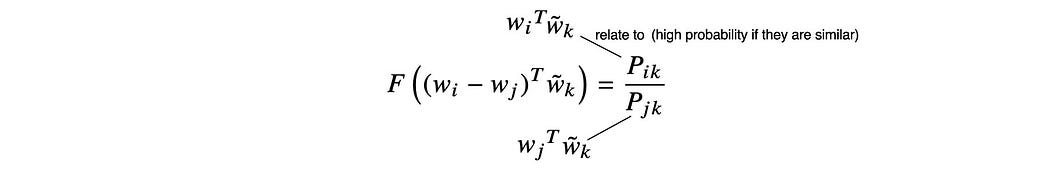


* And the ratio of co-occurrence probabilities as:

A picture containing text, font, line, receipt

Description automatically generated

* Now, we want to develop a model for F given some desirable behaviour we want for the embedding vector w.
* Linearity is important in the word embedding concept.
* So, if a system is trained on this principle, we should expect that F can be reformulated as:



**Model Training:**

Training a model simply means learning the given data entire and finding the best weights or good values for all the weights and the bias from labeled.

* **Feature Combination:** The extracted features are combined to form a feature vector representing each question pair and the calculated the distances.
* Performed the train test split and after performing it dividing into X\_train, X\_test, y\_train and y\_test.
* Using Standardscaler we rescaled the values of train and test data.
* Model Selection: Several machine learning models, such as logistic regression, can be trained on the feature vectors.

**Run these algorithms for Word2vec:**

1. Logistic regression
2. AdaBoostClassifier
3. XGBClassifier
4. GaussianNB
5. Gradient Boosting Decision Tree Classifier
6. Decision Tree classifier
7. Stacking Classifier
8. Hyperparameter for Decision tree and Ridge classifier.

**Run these algorithms for Glove:**

* Logistic regression
* AdaBoostClassifier
* XGBClassifier
* GaussianNB
* Gradient Boosting Decision Tree Classifier
* Decision Tree classifier
* Hyperparameter for Decision tree and Ridge classifier.
* Stacking Classifier

**Hyperparameter Tuning**: The model's hyperparameters are optimized using techniques like grid search or random search to improve performance.

We have performed on some of the models like Decision tree, Ridge classifier, gradient boosting decision tree.

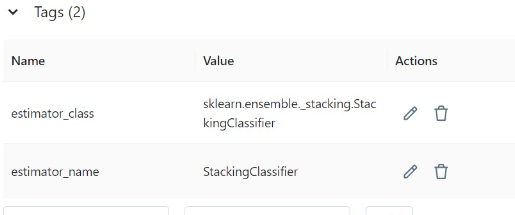
**Model Evaluation:**

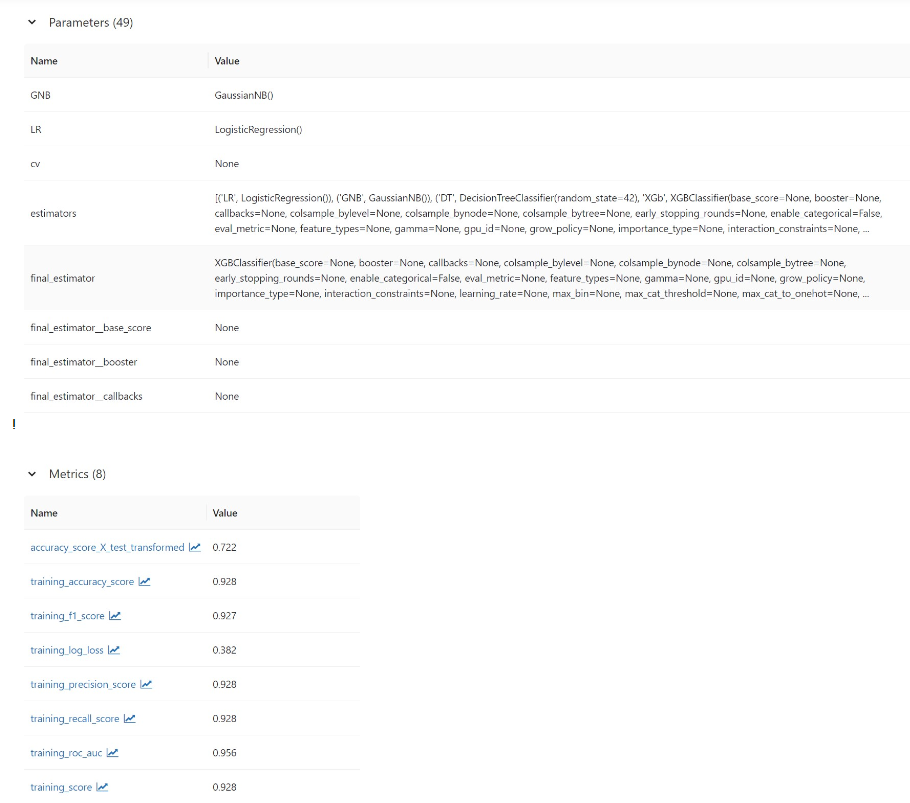
Model evaluation is the process that uses some metrics which help us to analyze the performance of the model.  The trained model is evaluated using appropriate evaluation metrics such as confusion matrix and Classification report that contains accuracy, precision, recall, and F1 score. The model's performance is measured on the train set and test set that are used to select the best model for deployment

**Mlops:**

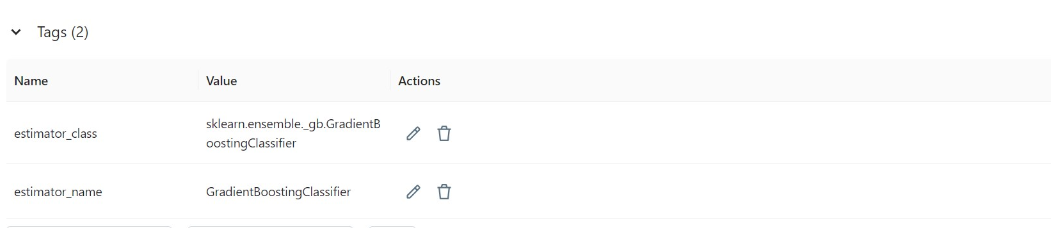
Once the best model is selected, it can be deployed to a production environment where it can be integrated with the Quora platform. The system can provide real-time predictions on the similarity of question pairs, helping users and moderators identify potential duplicates.

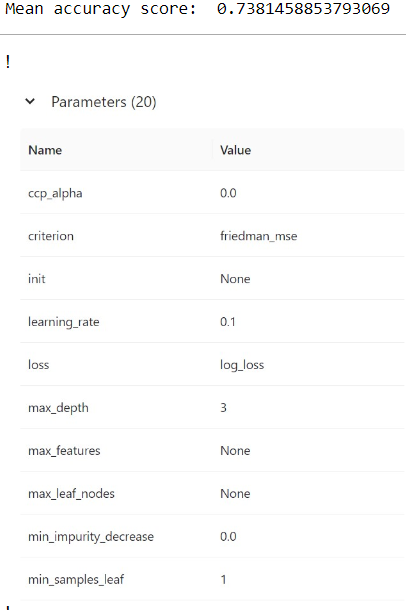
1. **Stacking Classifier:(Word2vec)**

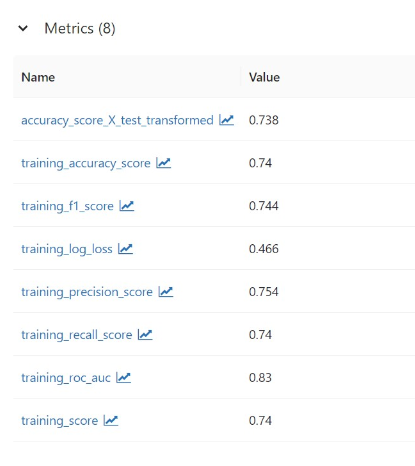
****

****

1. **Gradient Boosting Decision Tree Classifier:**

****





**Challenges:**

It is crucial to monitor the performance of the deployed system over time. Regular evaluations can be performed using a holdout dataset or by collecting user feedback to ensure the system's accuracy and reliability.

**Conclusion:**

The Quora Question Pair Similarity project provides an effective solution for determining the similarity between question pairs on the Quora platform. By accurately identifying similar questions, the system enhances user experience and improves the quality of information available to Quora users.

**Word2vec Results:**

* **Observations:**

1) Xgboosting performed well on train and test data with accuracy score for train is 75% and test is 74%.

2) Gradient boosting classifier with accuracy score on train data 73% and test data 73%.

**Glove Results:**

* **Observation:** among all the machine learning models, Gradient boost classifier gave better accuracy score for both train and test data i.e., 74%
* With bag of words preprocessing technique obtained accuracy score of 76% for logistic model
* Using TFIDF preprocessing technique obtained the accuracy score of 75% for logistic model.
* Problem with above two techniques is curse of dimensionality and semantic similarity. So, to overcome these two problems and to get better results we opted Word2vec and glove.
* Using word2vec preprocessing technique, XGBoost gave better results with accuracy score 75%
* Using Glove technique, Gradient Boost classifier gave better results with accuracy score 74%.

Please note that this is a general outline of a project documentation for the Quora Question Pair Similarity task. The specific implementation details and algorithms may vary based on the preferences and choices made during the project development.