Group Members

Sudipta Sen - MIT2021031
Trinetra Devkatte - MIT2021096
Chandrakishor Singh - MIT2021117

Machine Learning Task

1. Objective:

Our objective is to solve the problem of facing similar questions for many Q&A sites like stackoverflow, Quora etc. We are attempting to create ML models which can effectively detect whether two questions are similar or not. It will help the moderators of the forum to spend less time on manually tagging similar questions or to redirect the users to similar questions.

2. Introduction:

Duplication of questions is one of the many issues that Q&A platforms like Quora and Stack Overflow confront. The experience of both the questioner and the answerer is ruined when queries are repeated. We can simply display the questioner the answers to the prior question because he or she is asking a repeat question. Furthermore, the answerer is not required to repeat his or her response for essentially the same questions.

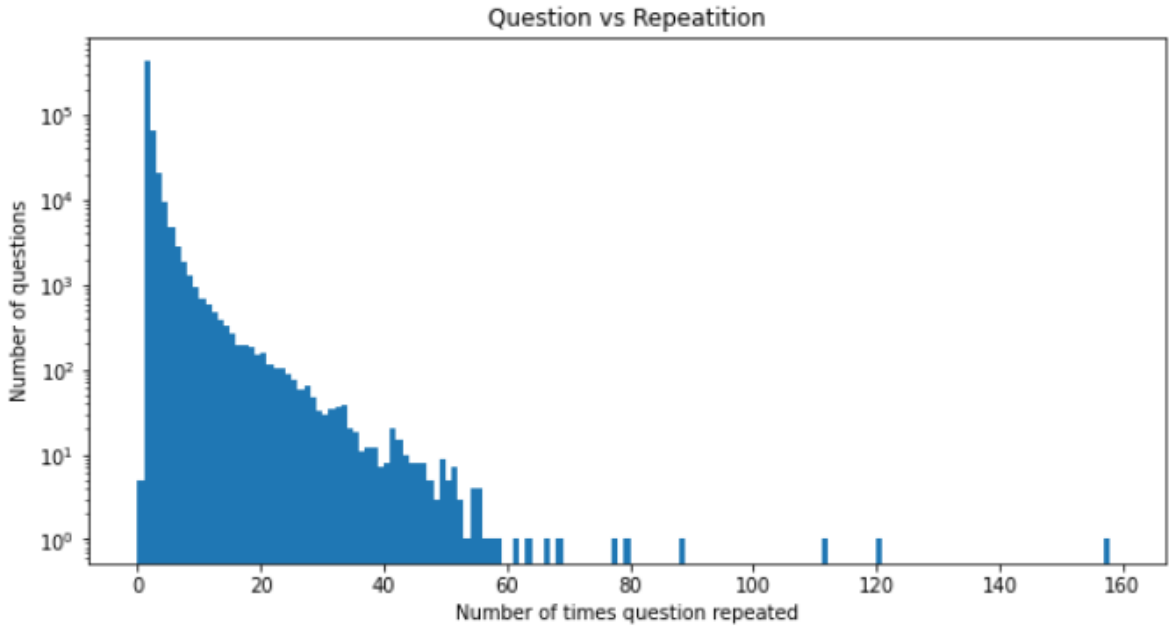
For example, we have a question like "How can I become a Data Scientist?" that has a few responses. Later on, someone else inquires, "What should I do to become a Data Scientist?" Both queries, as we can see, are asking the same thing. Despite the fact that the question wordings change, the aim of both inquiries is the same.

As a result, the answers to both questions will be the same. As a result, we can only present the responses to the first question. As a result, the individual who asks the question will receive immediate responses, and those who have already answered the first question will not have to repeat themselves.

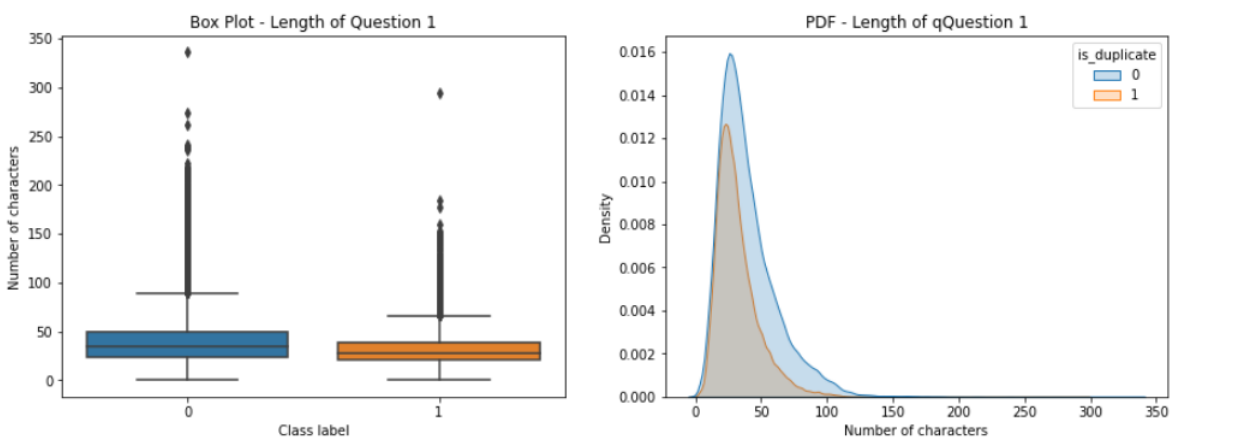
This project seeks to create a system that can detect similar questions on Quora using a combination of machine learning and data mining approaches centered on natural language processing. Natural language processing allows computers to interpret natural language in the same way that people do. NLP uses machine learning algorithms to take real-world information, process it, and try to make sense of it in a way that a computer can understand and interpret it properly, whether the language is spoken or written. For this NLP task, we applied numerous machine learning classification algorithms and framed the challenge as supervised learning.

EDA

1. Questions vs Repetition: Number of times a question is being repeated.



2. Length of Question(Number of characters)



Dataset

1. **Data Collection:** The data came from a previous Kaggle competition called Quora Questions Pair, which was held by Quora. It consists of two files: a train dataset and a test dataset. The train dataset is used to train the dataset, and the test dataset is used to test the NLP model that was built. The training file consists of 6 features id, qid1 , qid2 , question1 , question2 , is_duplicate
2. **Data Cleaning:**Impurities such as punctuation and stopwords were present in input text data. So, as part of the data cleaning process, we lowered the text, decontrated the word, removed urls, numbers, special characters, punctuations, stopwords, and applied Lemetization to the text.
3. **Feature Engineering:** We introduced fuzzy wuzzy features to our data, We've also added other features such as total sentences, number of words, number of unique words in a sentence, number of characters. We discovered that these characteristics make a considerable impact on the model's performance.
4. **Vectorization:**Machines can only understand numbers; they can't deduce meaning from the collections of text in our dataset. So we use basic vectorization techniques to translate text into a format that our computers can understand.
    a. **TF-IDf Vectorization:** Bag of Words converts a document from the corpus into a numeric vector by mapping each document word to a feature vector, The BoW approach is straightforward and effective, but it treats all words similarly and is unable to distinguish between common and uncommon words. Tf-idf solves the BoW Vectorization problem.

$$\text{TF - IDF} = \text{TF} * \text{IDF}$$

$$\text{TF('word')} = \frac{\text{'word' count in document}}{\text{Total 'word' count in a document}}$$

$$\text{IDF('word')} = log\left(\frac{\text{No of total documents}}{\text{No of documents with 'word' in it}}\right)$$

IDF drops as a word appears in more documents.Because the cell value is a multiplication of TF * IDF, more common words are given less weight and rare words are given more weight.

b. **Word2Vec:** When we have two separate terms with comparable meanings, TF-IDf is useless because it treats them differently. Word2Vec is effective in this situation since it uses Cosine similarity, which is dependent on the meaning of words.

$$cos(\theta) = \frac{A * B}{|A| * |B|}$$

So, if the cosine similarity value is around 1, it means the words are similar, if it's around 0, it means the words have no relation, and if it's around -1, it means the words have the opposite meaning.

c. **Avg Word2vec:** We need a huge text corpus that generates a vector for each word. As a result, Average word2vec finds the average of several vectors to create a single vector.
Assume you have a document with w1, w2,..., wn words in it. Because each word has its own vector, we'll transform it to average word2vec and then divide by the number of words in the document.

d. **TF-IDF weighted Word2Vec:** This method was employed in our project. This approach works by first calculating the tfidf value of each word, then multiplying the tfidf value to the word vector of that word, and after adding all the vectors we then divide it by total tfidf value.

e. **Data splitting:** The dataset was divided into two parts: training and testing. The dataset came with separate train and test folders. The train folder was used to create and train the model to recognise patterns, while the test dataset was used to put the model to the test. However, we separated the dataset into train and test datasets to avoid over-fitting. Furthermore, on the test dataset, our model is tested using the f1-score.

Machine Learning Models

A. Logistic Regression Model

This Algorithm is frequently used to determine the likelihood that an instance belongs to a specific class. It is often assumed that if the probability is more than 50%, the instance belongs to that class, which is usually the positive class or target variable. It uses the Gradient Descent Algorithm to function.

$$Z_\theta(x) = \theta^T . X$$

$$h_\theta(x) = 1 / (1 + e^{-Z(x)})$$

B. Naive Bayes

The Naive Bayes algorithm is a probabilistic machine learning algorithm that may be applied to a wide range of classification applications. The term naïve refers to the assumption that the features that make up the model are unrelated to one another. Based on the Bayes theorem, Naive Bayes appears to be a simple yet powerful algorithm.

$$P(Y|X) = P(X|Y) * P(Y) / P(X)$$

P(Y|X) = probability of outcome|evidence
P(X|Y) = probability of likelihood of evidence
P(Y) = prior(probability of each target class)
P(X) = probability of evidence

We calculate the probability of likelihood outcomes for each target class, and the predicted class is the one with the highest probability value.

C. Decision Tree

A decision tree is a visual depiction of a function that converts a vector of attribute values into a single output value, or "decision." A decision tree makes a decision by running a series of tests, beginning at the root and progressing via the relevant branches until it reaches a leaf. Each internal node in the tree represents a test of the value of one of the input attributes; the branches from the node are labeled with the attribute's possible values; and the leaf nodes describe what value the function should return

We have used Entropy as a criterion for splitting , It is given by the formula

$$E(S) = -P_{+}\log(P_{+}) - P_{-}\log(P_{-})$$

In the above Equation ,
E is the entropy
P_{+} represents the probability of class being positive
P_{-} represents the probability of class being negative

The Information gain is given by : IG = E(Y) - E(Y/X)
While splitting Feature with highest IG is used as the root of the tree

D. Random Forest

Random forest is a supervised machine learning algorithm that is commonly used to solve classification and regression problems. It creates decision trees from various samples, using the majority vote for classification and the average for regression. One of the most essential characteristics of the Random Forest Algorithm is that it can handle data sets with both continuous and categorical variables, as in regression and classification. For classification difficulties, it produces superior results.

This algorithm uses the Bagging Technique, in which the primary dataset is partitioned into sub-datasets and trained on various Decision Tree models before making a decision based on the majority vote.

E. Gradient Boosting Algorithm

It is an ensemble learning technique that is used to convert weak learners to strong ones by their previous errors. It can be used in both classification and regression problems. For the regression problem, it uses MSE for cost function and for classification, it uses Log Loss as a cost function.

When we wish to reduce the Bias error, we usually employ the Gradient Boosting Algorithm. The most crucial part of this technique is to identify the best n estimator value, which may be done with GridSearchCV.

F. XGBoost Algorithm

XGBoost is a gradient boosting-based ensemble Machine Learning technique that leverages decision trees. Artificial neural networks outperform all other algorithms or frameworks in forecasting problems involving unstructured data (pictures, text, etc.).

$$n(X) = G_b(X) = \mathbf{\Sigma} gb(X; \Theta_b)$$

Here, gb(X; Θ_b) is simple function of shallow trees and n is the natural parameters of the conditional distribution Y

Results

Below table shows the results of all algorithms which includes accuracy, F1-score, precision, recall, AUC score etc. Based on the results, it is clear that XGBoost performs better than all other algorithms.

Model Name	**Accuracy score**	**Precision score**	**Recall score**	**F1-score**	**Log Loss value**	**AUC score**
Logistic Regression	0.6925	0.686	0.716	0.701	0.556	0.776
Naive Bayes	0.653	0.618	0.812	0.702	8.241	0.699
Decision Tree	0.711	0.652	0.915	0.761	0.880	0.759
Random Forest	0.720	0.672	0.838	0.746	0.526	0.802
Gradient Boosting	0.705	0.698	0.766	0.731	0.566	0.780
XGBoost	0.738	0.706	0.851	0.772	0.525	0.798

References

1. https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/
2. https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP
3.https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d
4. https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP