EE 768 Project Report

Balraj Kumar 150110056
Anku Kumar Choudhary 150110078
Abhishek Gandhi 150110093

PROJECT OVERVIEW

Quora is a well known platform where any user can ask any type of questions and any user can answer any question. So it is obvious that there are millions of questions on Quora. Statistically, each day thousands of new questions are added. There is a huge chance that many of these questions added might already be present there or a question with similar intent may be present there. Users who wish to answer that problem might post it to different question threads. It would have been better if all the questions of same intent were at one place and answer to all those questions were also clubbed together.

PROBLEM STATEMENT

Given a set of question pairs, identify the questions with same intent. If two questions bears the same meaning mark them as same , else they are independent to each other.

DATA

Quora itself has provided a labelled set of question pairs and another set of question pairs with no labels. So basically we have a train set which has five columns - question1ID, question2ID, question1, question2 and is_duplicate (if the pairs have same intent then the value is 1 otherwise it is 0). Then there is a test set which has only pair of questions - question1 and question2. We are supposed to create another column with values 1 if they have same meaning and 0 if they are independent.

DATA ANALYSIS

The competition provides 3 .csv files - training set, testing set, and sample submission. It is interesting to note that the testing set is approximately 5x the size of the training set. The Data section of the competition homepage states the following:
"As an anti-cheating measure, Kaggle has supplemented the test set with computer-generated question pairs. Those rows do not come from Quora, and are not counted in the scoring. All of the questions in the training set are genuine examples from Quora."
The train set contains many repeated questions, about 80% are unique and also few questions as outliers repeating 100+ times. There is about a 2:1 class imbalance in favor of non-duplicate question pairs. As a result, it will make sense to evaluate models on roc auc along with the log loss metric. We will also have to address the class imbalance when we construct our training and validation sets, so they are representative of the whole set.
Test set has autogenerated questions where some of the questions didn’t even make sense. There are only 2-4 missing values which is a great news.

PREPROCESSING

Feature Analysis

This problem can be solved by comparing the words in the pairs. If there are many similar words in them then there would be higher chance that they are of same intent.

1. **Tokeninzing**: Tokenizing raw text data is an important pre-processing step for many NLP methods. It is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens." In the context of actually working through an NLP analysis, this usually translates to converting a string like "My favorite color is blue" to a list or array like ["My", "favorite", "color", "is", "blue"]. Python package Natural Language Toolkit provides useful utilities for tokeninzing texts. We used RegexpTokenizer() which splits a string into substring using regular expressions.

2. **Stopword filtering:** A stop word is a commonly used word (such as "the", "a", "an", "in") that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We would not want these words taking up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to be stop words. NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages.

3. **Stemming:** Stemming is the process of reducing a word into its stem, i.e. its root form. The root form is not necessarily a word by itself, but it can be used to generate words by concatenating the right suffix.For example, the words fish, fishes and fishing all stem into fish, which is a correct word. On the other side, the words study, studies and studying stems into studi, which is not an English word. There is a SnowballStemmer() algorithm in nltk which can do this.

4. **Lemmatizing**: The purpose of Lemmatisation is to group together different inflected forms of a word, called lemma. The process is somehow similar to stemming, as it maps several words into one common root. The output of lemmatisation is a proper word, and basic suffix stripping wouldn't provide the same outcome. For example, a lemmatiser should map gone, going and went into go. In order to achieve its purpose, lemmatisation requires to know about the context of a word, because the process relies on whether the word is a noun, a verb, etc. WordNetLemmatizer() can do this.

5. **tf-idf**: we used it as a weighting factor for information retrieval. The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

Combine all these features in a single pkl file. And then ROC_AUC_SCORE was calculated which seems almost same for all the features.

Training and Building Model

Doc2Vec: Doc2vec modifies the word2vec algorithm to unsupervised learning of continuous representations for larger blocks of text, such as sentences, paragraphs or entire documents. Doc2Vec's learning strategy exploits the idea that the prediction of neighboring words for a given word strongly relies on the document also.

Build a model based on all these features combined.

Train-validate-test-split : divide the given train set into two parts with 80:20 distribution randomly. Train the model on one of the parts and test it on the other and see the accuracy.

Model Justification

The prediction associated with each pair of the questions is whether they are duplicate. Thus, it is intuitive to solve this task using binary classification, with 0 meaning non-duplicate and 1 meaning duplicate. Since the task is to do analysis on the given wording, we need to describe each question using feature vector. We come up with three ways of generating the feature vectors using Tf- Idf score of the word given the dictionary. With the feature vector, we are able to compute the similarity between the pair of two questions. Then, perform classification based on the similarity.

So here we applied XGBoost to train the model. A max_depth=9 and a min_child_weight=1 performed the best at a 0.42318143814217912 log loss, which unfortunately, is still not a very good score. Upon looking at the feature importance, word_share_stemmed_filtered and word_share_lemmatized were identified as least important.

Then GPU training was done which calculated the training log loss and validation logloss.

Finally prepare the test data using all the important features and run it on the trained model.

Conclusions

Since non-duplicate instances are more than duplicate ones, if we guess all is duplicate are 0, we have 63% accuracy. Thus, we use 63% as our baseline prediction accuracy. XGBoost gave a better accuracy