



# Quora Question Pairs

01.06.2018

---

Samuel Baafi Boakye | Udacity  
Machine Learning Capstone Proposal

## Domain Background

There was a time when finding answers to questions was a bit difficult and normally practised through emails but now we have social media websites, so questions can be asked by anyone and seen by anyone and also can be answered by anyone and also seen by anyone. This improves the quality of information available to users of such social media platforms. This draws to mind a website like Quora.

Quora is a place to gain and share knowledge about anything. It's a platform to ask questions and connect with people who contribute unique insights and quality answers. This empowers people to learn from each other and to better understand the world. Myriads of people visit Quora to either ask questions or answer them. There is nothing wrong with having similar answers to a question but it could be a problem to have similar questions being asked. This will make it difficult to find quality answers to questions.

I've always been interested in working text data. It is amazing how some algorithms are able to make text data meaningful mathematically. Natural language processing is very important since most of the information in the world isn't hidden in numbers but in text or words. Here are some [solutions](#) provided to this provided by fellow kagglers.

## Problem statement

As stated above, having duplicated questions can only make it difficult for users to find the answers they need. Some questions may have longer sentences, others may be shorter but could mean the same thing. Clearly to manually root out duplicated questions will be difficult. What Quora does to solve this problem is to use an ensemble machine learning algorithm called Random Forest which basically combines a number of Decision Trees to build a stronger model. Questions are sentences and sentences are made up of words. Now how can you tell if question A is the same as question B? You may look at the length of the questions but some questions may be dissimilar in length but similar in meaning. So you could look at the uniqueness of words in each question using the Term Frequency - Inverse Document Frequency (TF-IDF) to achieve this mathematically. This could help differentiate between questions. Word2Vec algorithm could also be used to display words that share common contexts in a sentence are located in close proximity to another in a space ([wiki](#)).

It is my goal to apply natural language processing (NLP) algorithms like the ones stated above in conjunction with machine learning algorithms to build a model that is able to predict if a question is duplicated or not.

## Datasets and Inputs

[Datasets](#) from the Kaggle [Quora Question Pairs](#) competition will be used for this project. The dataset has a train file in csv format which will be used to model a machine learning algorithm to solve the problem. It also includes a test file also in csv format to be used to test how well the model works on unseen data. The question pairs in the train dataset file were obtained from genuine questions asked by users of the Quora website but the questions pairs in the test dataset file were computer-generated.

The train dataset file is made up of six columns and over 400000 rows. It has an `id` column which contains the indexes of each question pair. It also has the `qid1` and `qid2` columns containing the question ids for each question in a pair respectively. Each question in a pair is contained in the `question1` and `question2` columns. The last column which is the target or label column named `is\_duplicate`, holds binary values of 1s and 0s with 1 showing that a question pair is similar in meaning and 0 showing that a question pair is dissimilar in meaning.

The labels of the `is\_duplicate` column was set by human experts. So the label values are clearly subjective as the true meaning of questions on Quora can never be known accurately. It could contain wrong labels but it is assumed the labels represent a reasonable consensus. There would be a lot of noise in the labelling of the question pairs but it is assumed to be accurate yet not 100% accurate.

## Solution Statement

This problem can be solved with natural language processing techniques and there are a number of approaches. The most important columns in this project are `question1` and `question2` and these will be given much focus. Both columns contain questions and these questions may or may not have similar meanings. In order to computationally prove a question is similar to another question or not, the dataset will go through the text preprocessing stage and the feature engineering stage.

Text preprocessing puts emphasis correcting some errors with the following;

- Noise removal to remove unnecessary whitespace and spelling error correction.
- Contractions to remove certain punctuations. For instance "isn't" becomes "is not".
- Tokenization to break each question into smaller pieces or words.

- Normalizing the questions will be the first order of business as we will need to put all the words in a question on the same playing field. So a number like '3' will become 'three'. Whether all words must be in the same case or not.
  - Lemmatization will be applied to words which would return certain words to their dictionary base forms. For instance, better will become good.
  - Stop words are very common words that generally attaches little meaning to a sentence hence they will be removed from each question pair. For example the word "the" is a stop word.
  - Remove non-ascii words

### Feature Engineering


- Text mining features like the length of questions and the difference of length will be part of the feature engineering process. Number of capital letter and question marks will also used. The Term Frequency - inverse Document Frequency (TF-IDF) will also be used to determine the uniqueness of words in each question.
- Feature engineering practices which leverages deep learning methods will be used. They are embedding features. An example is the Word2Vec algorithm which will be used to create word embeddings. This algorithm will take as input a question to produce a vector space in several dimensions with each unique word in the sentence being assigned a vector space. The words are positioned in a vector space such that words that share common contexts are located in close proximity. The main idea here is that words that are similar appear in similar context of the words around them and this could help distinguish between similar questions and dissimilar questions. From a number of research papers the Word2Vec algorithm proved very useful in modelling a solution for the problem.

## Benchmark Model

There are a number of models shown [here](#) with their source embeddings and accuracy score. The last model used spacy to extract features in conjunction with TD-IDF features and a siamese deep neural network with the help of the global vectors for word representations (GloVe 6B tokens, 300D). This outputted an accuracy score of 0.79. I'm looking forward to creating a model that is able to achieve an accuracy score between 75% - 79% or better than the benchmark model above since I will be using a similar model. I would be satisfied with an evaluation metric below 0.4.

## Evaluation Metrics

The log loss metric will be used for evaluation since our model output is the probability of a binary outcome. This measures the accuracy of a classifier by computing the uncertainty of



a model to the true labels. According to this [article](#), log Loss heavily penalises classifiers that are confident about an incorrect classification. For example, if for a particular observation, the classifier assigns a very small probability to the correct class then the corresponding contribution to the Log Loss will be very large indeed. Naturally this is going to have a significant impact on the overall Log Loss for the classifier. The bottom line is that it's better to be somewhat wrong than emphatically wrong. Of course it's always better to be completely right, but that is seldom achievable in practice! Hence the smaller the log loss or uncertainty, the better the model.

## Project Design

I plan on getting to know the dataset a lot better by visualizing the number of duplicate questions and non-duplicate questions. The shape of the dataset will help me know the number of rows and how large the dataset is.

The next step is what is called data preprocessing. I will go through the question1 and question2 columns to check the questions for any spelling mistakes and whitespace which I consider as noise. I will also replace contractions with expressions. Each question will then be tokenized, that is broken into smaller pieces of words and further normalized. Under normalization, stopwords will be removed, words will be returned to their base dictionary forms called lemmatization, non-ascii words will be removed and numbers in their symbolic form will be changed to words. Lastly all words will be converted to lowercase.

Now the data will be ready for the feature engineering stage. A number of features will be extracted like the length of questions, the difference in length and others. Most importantly Term Frequency - inverse Document Frequency (TF-IDF) will be used to extract more features from the data. Afterwards, the word2vec algorithm will be used with the help of the Global Vectors for Word Representations (GloVe 6B tokens, 300D) to extract more features like the cosine distance, euclidean distance and others from the data.

The extracted features will be used to model an algorithm using LogisticRegression, Xgboost since Kaggle competitions work very well with it and the RandomForest ensemble algorithm and the results will be computed to show which algorithm performed better.

In the end a deep learning model will be developed to see if it generates better results compared to the regular supervised learning models.

## Reference

<https://www.kaggle.com/c/quora-question-pairs>