Problem Definition

Over 100 million people visit Quora every month, so it's no surprise that many people ask similarly worded questions. Multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question, and make writers feel they need to answer multiple versions of the same question.

Currently, Quora uses a Random Forest model to identify duplicate questions. We aim to tackle this natural language processing problem by applying advanced techniques to classify whether question pairs are duplicates or not. Doing so will make it easier to find high quality answers to questions resulting in an improved experience for Quora writers, seekers, and readers.

Implementation Overview

1. Dataset:
   Dataset is available on http://qim.fs.quoracdn.net/quora_duplicate_questions.tsv released by Quora itself.
2. Data Visualization
   We spent enough time studying the data, observing the writing style of questions, use of punctuations and numbers so as to ease the process of data cleaning. We observed that some labels in train set look incorrect. Only 37% questions have been marked as duplicate. These observations helped us in solving the problem in better way.
3. Web Scrapping for Answers
   Since answers were not provided in the dataset itself, we had to run python script for scraping answers from quora. We could only download answers for around 1000 answers because of the security settings of there website.
4. Data Cleaning
   Data cleaning is one of the most important steps before running any training models on the dataset to get better results. Clean data makes the model learn better and perform better on test dataset.
5. TF-IDF (unigram & bigram) and Cosine Similarity model
   It was the first model we ran to find the vectorized representation for question pairs. TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. Cosine similarity gave us the measure for the similarity between the 2 vectors. We found cosine similarity between TF-IDF vectorized representation of both the question pairs and the answer pairs.
6. Word2Vec trained with GoogleNewsVector
   GoogleNewsVector includes word vectors for a vocabulary of 3 million words and phrases that they trained on roughly 100 billion words from a Google News dataset. Google's Word2Vec is a deep-learning inspired method that focuses on the meaning of words. Word2Vec attempts to understand meaning and semantic relationships among words. It works in a way that is similar to deep approaches, such as recurrent neural nets or deep neural nets, but is computationally more efficient. We found the word2vec representation for both question pairs and answer pairs and calculated WMDSimilarity between both pair of representations.

7. Classification Algorithms
   This is the last step for identifying the questions as duplicate or not. We used a variety of classification algorithms – Logistic Regression, Random Forest, Support Vector Machine, Naïve Bayes Classifier, Decision Tree, Stochastic Gradient Descent (SGD) learning. Moreover, in the end we used a voting algorithm to chose best possible outcome from the results of 5 classification algorithms.

Innovation

- Multi View Classification Algorithm
  We incorporated multi view classification in our model wherein we considered the questions both ways – Syntactically and Semantically using Word2Vec and TF-IDF vectorizers.
- Including Answers as a Feature
  Till now no one had thought of incorporating answers to the pair of questions given as a judging criterion for the two questions to be similar. We completed this task elegantly using python script and collected answers to approx. 1000 pairs of answers. This increased the accuracy of our model from 74% to 84% (approx.).
- Working Interface
  We also made a working interface of the model for easy use and to depict the working of the model in a better way.

Results

Following are the results obtained on the dataset using different models:

	Train Accuracy	Test Accuracy	F1 Score
Logistic Regression	84.95	81.91	0.805383
Random Forest Classifier	96.77	78.72	0.777145
Support Vector Classifier	85.19	82.98	0.817917
Stochastic Gradient Descent	84.71	85.11	0.836562
Naïve Bayes	85.42	81.91	0.807632
KNN Classifier	89.01	84.04	0.835029

We are also using VotingC Classifier to choose the best possible answers by using all models. It gives weight to all the classifier results and choose the best possible outcome.

We obtained great results when we also included answers as a judging criterion – it was a great idea to be clubbed in the project.

Working Screenshots