Machine Learning Engineer Nanodegree

Capstone Project
Quora Question Pairs

Brett Clubb

February 25, 2017

Introduction

Project Overview

Natural Language Processing (NLP) is an area of active research and development in which a computer analyzes a set of text for the purpose of achieving human-like language processing for a range of tasks or applications. A common goal of NLP systems is to represent the true meaning and intent of an inquiry.[1]

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.

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. Quora values canonical questions because they provide a better experience to active seekers and writers, and offer more value to both of these groups in the long term. [2]

---

[1] "Natural Language Processing - SURFACE - Syracuse University." https://surface.syr.edu/cgi/viewcontent.cgi?article=1043&context=istpub. Accessed 25 Feb. 2018.
[2] "Quora Question Pairs | Kaggle." https://www.kaggle.com/c/quora-question-pairs. Accessed 25 Feb. 2018.

Problem Statement

This project aims to solve a binary classification problem to predict whether or not a provided pair of questions have the same or similar meaning through the use of natural language processing. By determining whether or not question pairs are duplicates, we can improve the experience in finding high quality answers for Quora writers, seekers, and readers by providing a single page for questions with a similar intent. [3]

Metrics

As defined by Quora in the Kaggle competition[4], I will evaluate the model on the log loss between the predicted values and the ground truth, found as the human-labeled value in the is_duplicate column. For each ID in the test set, there should be a prediction on the probability that the questions are duplicates (a number between 0 and 1). The goal of this model will be to minimize this value, since log loss increases as the predicted probability diverges from the actual label.[5]

Log Loss is defined mathematically as $-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \log p_{ij}$, where N is the number of samples or instances, M is the number of possible labels, $y_{ij}y_{ij}$ is a binary indicator of whether or not label j is the correct classification for instance i, and $p_{ij}p_{ij}$ is the model probability of assigning label j to instance i. A perfect classifier would have a Log Loss of precisely zero. Less ideal classifiers have progressively larger values of Log Loss.[6]

[3] "Quora Question Pairs | Kaggle." https://www.kaggle.com/c/quora-question-pairs. Accessed 25 Feb. 2018.
[4] "Quora Question Pairs | Kaggle." https://www.kaggle.com/c/quora-question-pairs#evaluation. Accessed 25 Feb. 2018.
[5] "Log Loss - Deep Learning Course Wiki - Fast.ai." 16 Feb. 2017, http://wiki.fast.ai/index.php/Log_Loss. Accessed 25 Feb. 2018.
[6] "Making Sense of Logarithmic Loss." 14 Dec. 2015, http://www.exegetic.biz/blog/2015/12/making-sense-logarithmic-loss/. Accessed 25 Feb. 2018.

Analysis

Data Exploration

The Quora dataset provides a set of question pairs and indicates whether or not the questions are requesting the same set of information. The data is provided by Quora via Kaggle's competition website.

Data Fields (Inputs):

- id - the id of a training set question pair
- qid1, qid2 - unique ids of each question (only available in train.csv)
- question1, question2 - the full text of each question
- is_duplicate - the target variable, set to 1 if question1 and question2 have essentially the same meaning, and 0 otherwise.

The dataset is split into two files, one for training and one for testing.

Training Data

- Question Pairs: 404,290
- Unique Questions: 537,933
- Known Duplicates: 149,263
- Ratio of Duplicate Pairs: 36.92%

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

Testing Data

- Question Pairs: 2,345,796

	test_id	question1	question2
0	0	How does the Surface Pro himself 4 compare wit...	Why did Microsoft choose core m3 and not core ...
1	1	Should I have a hair transplant at age 24? How...	How much cost does hair transplant require?
2	2	What but is the best way to send money from Ch...	What you send money to China?
3	3	Which food not emulsifiers?	What foods fibre?
4	4	How "aberystwyth" start reading?	How their can I start reading?

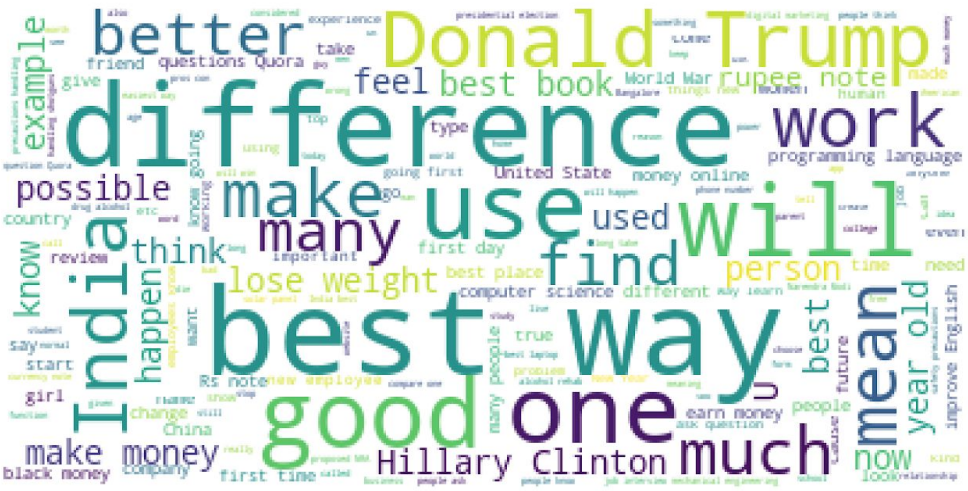
After a quick look at the data, it is clear that there are almost six times the amount of question pairs in the testing dataset than pairs in the training dataset. In the training dataset, 149,263 of the 404,290 question pairs (36.92%) are marked as duplicate questions. This seems like a fair ratio we might see in a larger, real-world setting, since we could argue that most Quora visitors would likely search for a question before asking a new question of their own.

I also noticed that the testing dataset only contains a test_id to identify the question pair and the text of each question. The finished project will need to predict the is_duplicate value for each question pair.

Exploratory Visualization

I followed Adil's Notebook[7] for guidance to make my own word cloud in an effort to help visualize what types of questions we will be working with.

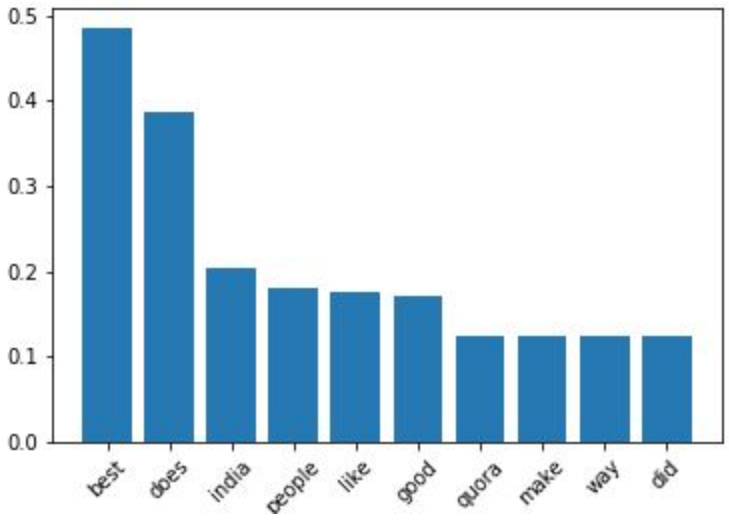
[7] "Word Cloud with Python | Kaggle." 18 Jan. 2017, https://www.kaggle.com/adiljadoon/word-cloud-with-python. Accessed 26 Feb. 2018.



From the word cloud generated from the training dataset, we quickly see several repeating topics that can help us draw some larger conclusions about the dataset.

- **“Good”, “better”, “best”, “difference”:** Many questions in our dataset are trying to compare two things, often in search of the “best” option.
- **“Donald Trump”, “Hillary Clinton”:** Due to the frequent occurrence of the last U.S election’s candidates, it’s safe to assume this data was likely collected around the time of the 2016 U.S. election.
- **“India”, “China”, “rupee”:** It appears many of Quora’s users asking questions are based in the East.

The findings from the word cloud are consistent with the results seen in the bar chart I generated to display the top ten most common words.



After gaining an understanding of the content of the questions at a high level, I decided to take a look at more detailed data - the number of words in each question and the number of characters in each question. This will give me an idea of the size of the texts we'll be comparing and a chance to see if there are any trends in the dataset. More details of this process can be found in the Data Preprocessing section.

Algorithms and Techniques

Because this is a binary classification problem, there are several methods available in supervised learning. I will implement the following algorithms:

- Decision Trees - creates a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.[8]
- K Nearest Neighbors - a chosen point is assigned a label based on the average of its *k* nearest neighbors, where *k* is an integer specified by the user.[9]

[8] "1.10. Decision Trees — scikit-learn 0.19.1 documentation." http://scikit-learn.org/stable/modules/tree.html. Accessed 16 May. 2018.
[9] "1.6. Nearest Neighbors — scikit-learn 0.19.1 documentation." http://scikit-learn.org/stable/modules/neighbors.html. Accessed 16 May. 2018.

- Logistic Regression - linear model for classification which returns the probability of a binary outcome.[10]
- Multinomial Naive Bayes - applies Bayes' theorem with the naive assumption of independence between every pair of features.[11] This particular classifier is often used in text classification when discrete features, like word counts, are given.[12]
- Support Vector Machines - a model representation of the questions as points in space, mapped so that the duplicate and non-duplicate question pairs are divided by a clear gap that is as wide as possible. New questions are then mapped into this same space to form a prediction based on which side of the gap they fall.[13] Since questions further away from the line are more likely to be correctly labeled, a probability is returned based on the distance a question point is from the line.

I will use cross-validation to determine which model performs best and fine-tune its parameters. The cross-validation process will be discussed in detail in the Data Preprocessing section below.

Benchmark

Quora has noted that they currently use a Random Forest model to identify duplicate questions.[14] Because the goal of the competition is to improve on the current implementation, I will utilize a Random Forest Classifier from sklearn to fit a Random Forest model to use as a benchmark.

[10] "sklearn.linear_model.LogisticRegression — scikit-learn 0.19.1 ...." http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. Accessed 16 May. 2018.
[11] "1.9. Naive Bayes — scikit-learn 0.19.1 documentation." http://scikit-learn.org/stable/modules/naive_bayes.html. Accessed 16 May. 2018.
[12] "sklearn.naive_bayes.MultinomialNB — scikit-learn 0.19.1 documentation." http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html. Accessed 21 May. 2018.
[13] "Support vector machine - Wikipedia." https://en.wikipedia.org/wiki/Support_vector_machine. Accessed 16 May. 2018.
[14] "Quora Question Pairs | Kaggle." https://www.kaggle.com/c/quora-question-pairs. Accessed 28 Feb. 2018.

For a better sense of how my models are performing overall, I will also submit them to the Kaggle competition to check the leaderboard score. Because Kaggle scores use negative log loss, a lower score indicates a better-performing model.

Results:
Benchmark Accuracy = 0.713
Benchmark Kaggle Score = 1.437

Methodology

Data Preprocessing

First, I have created a few functions to pass the given data through. These functions will help me collect features from the data. From this first pass, I will collect the character counts of each question, the word counts of each question, and the number of words the questions both posses.

Next, I will calculate TF-IDF scores using TfidfVectorizer to gain an understanding of how often words appear across all questions and which words are most important to determine meaning. This transformation outputs a normalized vector for each question, and in place of each word is its TF-IDF score. I will then use cosine similarity to compare the angle between the Q1 and Q2 vectors to determine a final feature. We expect an exact match of Q1 and Q2 to return a cosine similarity of 1 and questions that are less similar to approach 0.

After gathering all six of the features I'd like to train on, I will split the data into training and test sets. This is a common practice in machine learning to allow for cross-validation. This methodology allows us to take a portion of what we know, the labeled training data, and use that portion to train our models based on the provided label which indicates whether or not the questions are duplicates. We then take the generated model and test, or cross-validate, it against the unused portion of the labeled training data. This allows us to test our model's accuracy by having it first guess whether or not the questions are

duplicates and then cross-validate that with the provided label to determine if the model was correct.

Implementation

I started with implementing a basic Decision Tree Classifier by fitting the normalized training data to a model.

Next, I implemented the K-Nearest Neighbors Classifier. I used Grid Search Cross Validation to find the best parameters for this algorithm. Grid Search allows us to pass in a grid of parameter options we would like to try out. The algorithm is then run multiple times against the various parameters, comparing their accuracy and outputting the best performing parameters. The best parameters for K-Nearest Neighbors were {'n_neighbors': 19, 'weights': 'uniform'}. I used these parameters to fit a model and generate a prediction on the test set of data for Kaggle submission.

I then implemented the Logistic Regression algorithm, again utilizing Grid Search to find that the best parameters were {'penalty': 'l2', 'C': 1000}, and generated the predictions for on the test questions.

Next, I used Grid Search to find the best parameters, {'kernel': 'sigmoid', 'C': 1}, to fit a Support Vector Machine model.

I then implemented a simple Multinomial Naive Bayes model, this time using the unaltered (not normalized) data.

For each of the above implementations, I used cross-validation to fine tune the model's accuracy before using the model to predict whether or not the unlabeled questions from the test set were duplicates. I then output the question id and my prediction to a CSV file for submission to Kaggle which gave me the negative log loss score for each of my models.

After comparing all of the negative log loss scores, it is clear that the Support Vector Machine generated the best model.

Refinement

Because the Support Vector Machine algorithm had the best performing model, I chose to focus on improving its performance. At first, I tried adding more parameters to the GridSearch. This proved that adding each additional parameter to the grid exponentially increased the run time for fitting the model. After 96 hours with no results, I decided to try another approach. It became clear to me that support vector machines do not scale well for large datasets.

In the end, I settled on using a Bagging Classifier on top of a linear Support Vector Classifier (SVC) after finding a similar approach shown on StackOverflow[15]. The Bagging Classifier fits base classifiers, in our case SVCs, on random subsets of the original dataset and then averages the individual predictions to form a final prediction.[16] With the bagging method, the samples are drawn and replaced with fresh samples. This approach allows the algorithm to run in a fraction of the time, typically under 5 minutes, while maintaining comparable accuracy and negative log loss scores.

Results

Model Evaluation and Validation

My final model was fit by running multiple linear SVCs, using samples selected by the bagging method, and taking the average performance to generate a final model. For the SVC parameters, I used a linear kernel, C = 100, max iterations = 10,000, and class weight = 'auto'. I also set the probability parameter to true so that the output would be a probability, as expected by the Kaggle scorer.

[15] "scikit learn - Making SVM run faster in python - Stack Overflow." 15 Aug. 2015, https://stackoverflow.com/questions/31681373/making-svm-run-faster-in-python/32025662. Accessed 3 Jun. 2018.
[16] "sklearn.ensemble.BaggingClassifier — scikit-learn 0.19.1 ...." http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html. Accessed 3 Jun. 2018.

For the Bagging Classifier, I set bootstrap to false to ensure that each sample is only selected once. I set the number of estimators to 1,000 and set the number of max samples to be 1/n_estimators. I set n_jobs to -1, which tells the algorithm to use all available processors when run. When run, I recommend that the computer is mostly idle other than this task. I also wrapped the bagging classifier with the OneVsRestClassifier, which returns a binary output for classification.

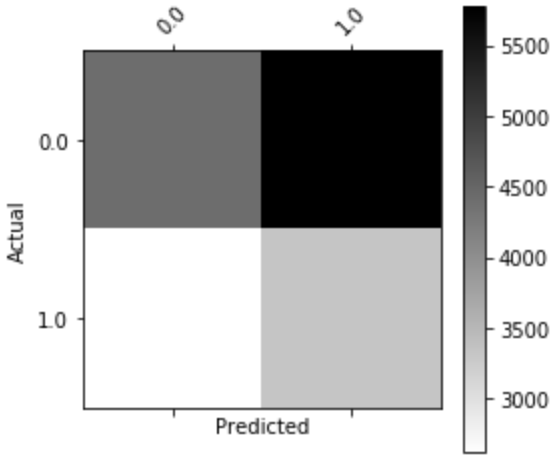
Justification

This refined model yielded an accuracy of 0.633 and a Kaggle score of 0.545, which is the best log loss score any of the models received. Compared to the benchmark model, which yielded an accuracy of 0.713 and a Kaggle score of 1.437, we see the accuracy is in the same ballpark but slightly below the benchmark model's performance. When it comes to Kaggle scores, however, this model has a much better score. Because the negative log loss is used for the Kaggle score, this tells us our model may be more confident in its predicted label than the benchmark model, even though it is not always accurate.

Conclusion

Free-Form Visualization

In order to get a better understanding of my model's predicted values, I chose to create a confusion matrix. The matrix shows the actual value compared to the predicted value. A zero indicates the questions are not duplicates, and a one indicates the questions have similar meaning. In this matrix, the darker squares indicate a higher number of predicted values.



Predicted Actual	0.0	1.0
0.0	4428	5776
1.0	2619	3335

Based on these results, we can see that there were a lot of false positives in our predictions. Of over 16,000 pairs of questions from the X_test subset of the training data, we predicted over 5,500 to be duplicate questions despite the fact that they were actually labeled as not duplicates. The next darkest squares indicate that the largest amount of our questions, over 7,700, were correctly predicted.

Ideally, this model could be improved by bringing down the prediction rate of false positives to achieve better performance.

Reflection

I chose this project because it allowed me to tackle new challenges. This was my first project using natural language processing techniques, and much of my initial research was focused on learning how to transform the question pairs into numbers which the

algorithms could understand. Through plenty of trial and error, I was able to extract some basic features from the data -- character counts, word counts and shared words.

Through my research, I read a lot about term frequency-inverse document frequency (tf-idf) as a way to extract more features from the data. I struggled in trying to determine how I could utilize the transformed vectors meaningfully alongside my already collected features. I concluded that I would implement a function to compare to the vector representations of question 1 and question 2 and measure the cosine angle between the two points. A smaller angle returns a number close to 1 indicating the vectors are similar, and larger angles indicate less similar vectors. This allowed me to extract a weighted value of similarity between the two questions based on their tf-idf vector representations.

After fitting the various models, it was clear that the SVC was the best performing. I spent a lot of time researching ways to improve the model, but ultimately found that SVCs do not scale to large problems well. I did stumble upon the bagging classifier trick which did optimize the training time greatly and allowed for some refinement.

Improvement

Overall, I think if I made any improvements compared to the benchmark model, they were slight at best. After seeing other Kaggle submissions for the Quora Questions competition, it is clear that using neural networks can provide much better accuracy with log loss scores less than 0.2.

In order to make any substantial improvements to my model, I would consider instead using a Siamese network for deep learning. Siamese networks have two or more identical sub-networks that are often used to perform similarity tasks[17]. I think I would first need access to better computing hardware in order to explore training a network like this.

[17] "How to predict Quora Question Pairs using Siamese ... - Medium." 7 Jun. 2017, https://medium.com/mlreview/implementing-malstm-on-kaggles-quora-question-pairs-competition-8b31b0b16a07. Accessed 3 Jun. 2018.