Yelp Score Predictor

Lucas Saechao

218794239

CSC 180 - Intelligent Systems

Project 1

February 26, 2021

Predicting a Business' Star Rating Based On Yelp Review

Abstract

The following report describes the motivation, challenges, and methods used in order to train a neural network model to predict a business' Yelp rating. This model is trained using the Yelp academic dataset, a subset of data originally used for the Yelp Dataset Challenge, which would provide students an opportunity to conduct a research or analysis using Yelp's data, and share particular insights and discoveries*. The dataset contains greater than 5,000,000 reviews, conducted for over 150,000 businesses, and is a comprehensive corpus, appropriate for the purpose of this project.

Problem Statement

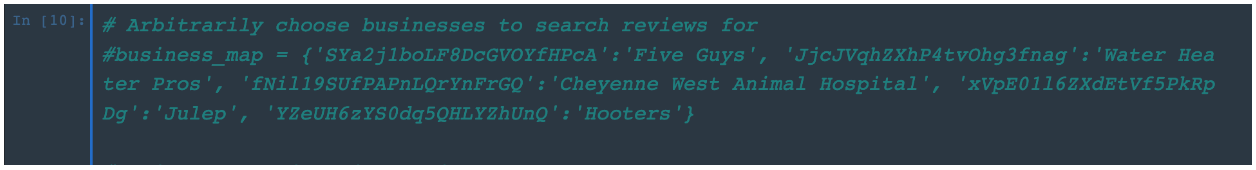
Hundreds of thousands, if not millions, of businesses are reviewed on Yelp every single day. Many of these businesses are *small* businesses, without the benefit of investment backing as compared to larger corporations, and require the services of platforms like Yelp in order to keep their business afloat and relevant. For a potential customer, Yelp is also an integral service used in order to discern businesses that can provide the strongest value *to them*, as the consumer. Given the review text of an arbitrarily chosen business, can a neural network discern the rating that business based on the words described in that text?

Methodology

In order to train this dataset, several Python libraries were utilized: Scikit Learn, TensorFlow, Keras, Numpy, and Pandas. To prepare the dataset for training, two preliminary steps were necessary: preparing the list of businesses, and the list of review text. First, the businesses were imported into the program, and those with *less* than 20 reviews were not considered for training. The following step was to filter out the review text for only JSON objects that matched the remaining businesses and storing them into tab separated values (TSV) files. Any leftover review text is discarded and not used for training.

For initial testing, in order to improve the speed of my training program, I had limited the training data to use only an arbitrarily selected set of businesses; for the final regression model, the neural network was trained over the entire corpus.

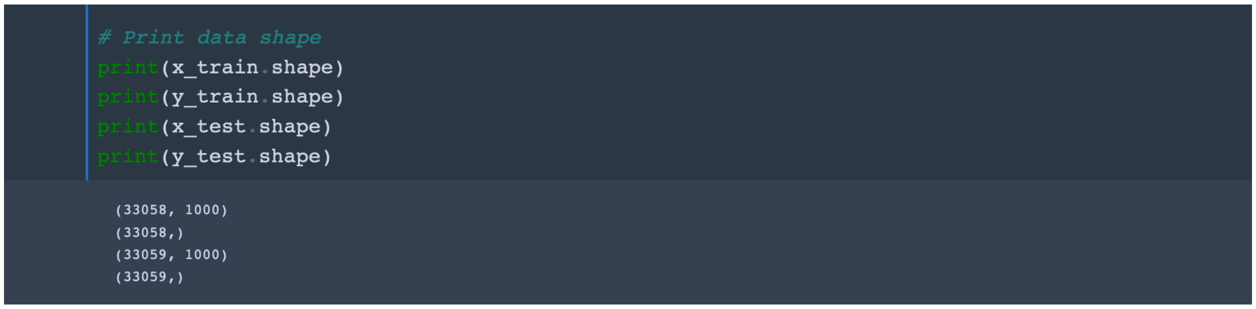
* https://www.kaggle.com/yelp-dataset/yelp-dataset



```
In [10]: # Arbitrarily choose businesses to search reviews for
         #business_map = {'SYa2j1boLF8DcGVOYfHPcA':'Five Guys', 'JjcJVqhZXhP4tvOhg3fnag':'Water Hea
         ter Pros', 'fNil19SUfPAPnLQrYnFrGQ':'Cheyenne West Animal Hospital', 'xVpE01l6ZXdEtVf5PkRp
         Dg':'Julep', 'YZeUH6zYS0dq5QHLYZhUnQ':'Hooters'}
```

After the preliminary steps are accomplished, the dataset is now ready to be trained. A TF-IDF Vectorizer was then prepared using Scikit Learn's feature extraction. Stop words are set to English, and only the first 1,000 features are extracted. The review vector is then fit, and merged with the original data frame by its review text, and unnecessary columns are dropped to prepare the data frame for the neural network.

Before making the neural network, however, an intermediate step is taken. The data frame containing the information that will be used by the neural network needs to be split into training and test data. This is accomplished through Scikit Learn's train_test_split function from its Model Selection library, and the test size is set to 0.5, in order to split the data into even halves, and random state is set to 40.



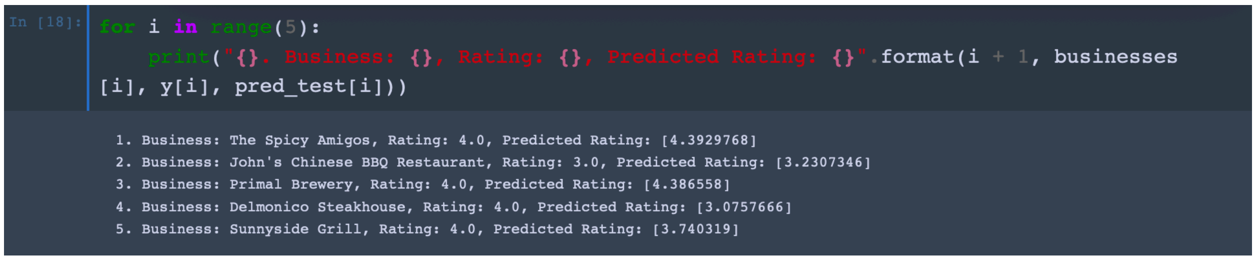
```
# Print data shape
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)

(33058, 1000)
(33058,)
(33059, 1000)
(33059,)
```

Lastly, a neural network is prepared with four layers. Each layer, except the output layer, is using a different activation function. In order, these are: tanh, sigmoid, and relu, and the model is compiled using the adam optimizer. With these settings, and an early stopping at the 25th epoch, the model has an RMSE of 0.2978.

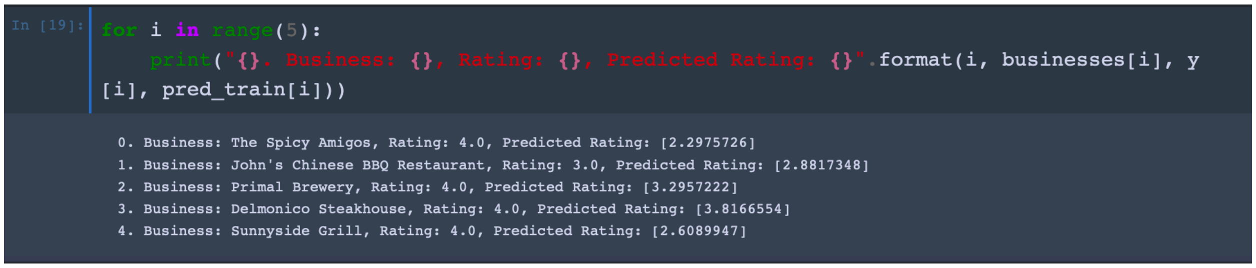
Experimental Results and Analysis

Now that the pred_test and pred_train models are complete, I verify if it can accomplish what was outlined in the problem statement: if this model can accurately predict a star rating based on review text. For the following arbitrarily chosen businesses, the pred_test model has been somewhat successful at predicting the review score, typically with an error of ± 0.4, save for the 4th business, in which it was off by nearly an integer amount. Interestingly, pred_train had inverse luck: it was inaccurate for each sampled business, save for the 4th, in which its margin of error was ± 0.2.



```
In [18]: for i in range(5):
             print("{}. Business: {}, Rating: {}, Predicted Rating: {}".format(i + 1, businesses
         [i], y[i], pred_test[i]))

         1. Business: The Spicy Amigos, Rating: 4.0, Predicted Rating: [4.3929768]
         2. Business: John's Chinese BBQ Restaurant, Rating: 3.0, Predicted Rating: [3.2307346]
         3. Business: Primal Brewery, Rating: 4.0, Predicted Rating: [4.386558]
         4. Business: Delmonico Steakhouse, Rating: 4.0, Predicted Rating: [3.0757666]
         5. Business: Sunnyside Grill, Rating: 4.0, Predicted Rating: [3.740319]
```



```
In [19]: for i in range(5):
             print("{}. Business: {}, Rating: {}, Predicted Rating: {}".format(i, businesses[i], y
         [i], pred_train[i]))

         0. Business: The Spicy Amigos, Rating: 4.0, Predicted Rating: [2.2975726]
         1. Business: John's Chinese BBQ Restaurant, Rating: 3.0, Predicted Rating: [2.8817348]
         2. Business: Primal Brewery, Rating: 4.0, Predicted Rating: [3.2957222]
         3. Business: Delmonico Steakhouse, Rating: 4.0, Predicted Rating: [3.8166554]
         4. Business: Sunnyside Grill, Rating: 4.0, Predicted Rating: [2.6089947]
```

In testing the model, I had also sampled the RMSE of different activation function combinations, for a sample data frame much smaller than the full set (~20,000 records), 1000 features, and a smaller input size for the neural network’s layers. The following table outlines the results of each test.

Activator Functions	Optimizer	RMSE
relu	adam	0.7571
relu	sgd	0.7536
relu, sigmoid	adam	0.7492
relu, sigmoid	sgd	0.9131
relu, sigmoid, tanh	**adam**	**0.7136**
relu, sigmoid, tanh	sgd	0.7576
sigmoid	adam	0.7504
sigmoid	sgd	0.7492
sigmoid, tanh	adam	0.7519
sigmoid, tanh	sgd	0.7575
tanh	adam	0.7412
tanh	sgd	0.7548
tanh, relu	adam	0.7575
tanh, relu	sgd	0.7512

Task Division and Project Reflection

I am not working on a team, so I am solely responsible for this project. I will concur that it is a strong challenge to work on this assignment on my own, but I feel that by doing so, I'm better able to hold myself accountable and better understand the course material. I believe this was a very interesting project to begin the semester with — I spent a lot of time reading more supplementary topics regarding different approaches and applications of models such as the one trained, such as Naive Bayesian Classification. I would be open to approaching fellow students for future projects, depending on my perceived difficulty of them at that time. However as a base, I believe that what I learned about natural language processing and neural networks will provide a strong foundation to tackle the rest of this course.

Challenges

One of the primary challenges that I experienced was attempting to perform TF-IDF Vectorization, as my system had kept running out of memory. On my limited dataset test using 20k records, I was able to capture each feature without error, but against the full dataset, I had to progressively lower the maximum features to record. This value went from 100,000 features, to 50,000 features, to 25,000 features, with each attempt being N/2 the previous amount. As of this writing, on my machine, I was able to perform this operation with 10,000 features, but in the actual submission, I used 1,000 features. Another challenge would be increasing the accuracy of my model's predictions. I had experimented with different neural network layer sizes, activation functions, patience, and optimizers in order to arrive at an optimal recorded weight.

Another challenge, ancillary to the first, is the time it took in order to train the model and load the dataset. For the sake of testing the program itself, I had limited the data to five arbitrarily chosen businesses to train with, expanding it to 10,000, and then 20,000, before moving on to the full dataset. I had first attempted a run using the full dataset, but it would take anywhere from 10 minutes to an hour to complete, depending on if I am using my Macbook Pro, or my Windows machine.

Other Insights

I believe that there is a lot more that can be done with this data than simply predicting business scores. With how heavily small businesses rely on Yelp for their public image, I believe that Yelp's academic dataset can be trained to predict, not just a star rating for a business, but whether or not that establishment will remain in business for long. Alternatively, this dataset could be used to instead *eliminate* faulty, or false, reviews by determining the likelihood of a review being written from someone who had never used that service, or from an actual experience with the business.