Yelp Review to Rating Prediction using Recurrent Long-Short Term Memory Network

1. INTRODUCTION

Yelp allows its users to share reviews of local businesses. For each business, the reviews and star ratings are used to display some key quotes from reviews, and an average star rating. Yelp ratings brings us a new way to choose a business as a customer or run a business as an owner in our daily life. We prefer restaurants or hotels with higher ratings which determine our choice most time. However, we know that not all user ratings are objective all the time. It is possible that a very positive review may come with a five-star rating while another similar one just has a three-star rating. Our goal in this project was to improve Yelp's user experience, by predicting the rating of a business given its reviews. For our experiments, we used the Yelp Academic Dataset.

2.1 TASK DEFINITION

We use Yelp's recently released academic dataset[1], which provides over one hundred fifty thousand reviews and their corresponding ratings for restaurants centered near many different universities. For the sake of training, we did not consider correspondences between the 2 users or the restaurants with reviews because we wanted to provide a purely objective analysis of semantic information. We treated each review as its separate document and assumed each document corresponded to a single review and its single star rating. Each rating was rounded to a whole star, so the set of ratings only included 1, 2, 3, 4, and 5. We divided the dataset into 46 smaller chunks due to the limitations of system memory. We used the first 5 out of the 46 files as input the train the model. We used the 44400 reviews to test the model. We measure the accuracy by computing the Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

2.2 ALGORITHM DEFINITION

The need of recurrent neural networks is when there is a sequence of data and that temporal dynamics that connects the data is more important than the spatial content of each individual frame. Long Short-Term Memory (LSTM) networks are extended version of recurrent neural networks (RNN), which extends memory of network units. The basic unit in hidden layer of an LSTM network is the memory block; it replaces hidden units in a traditional RNN as shown in Figure 1[2]. LSTM's enable RNN's to remember their inputs over a long period of time. In a RNN there are three gates: input, forget and output gate. These gates determine whether to let new input in (input gate), delete the information because it isn't important (forget gate) or to let it impact the output at the current time step (output gate). Each memory cell has its own self-connected linear unit called "Constant Error Carousel" whose activation is cell state. When activation is zero, gates are closed, and irrelevant inputs and noise are not perturbed in the network.

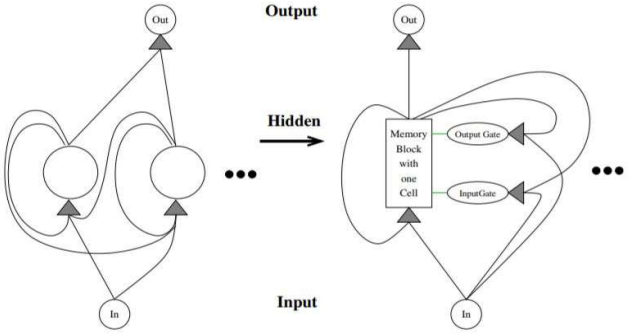


Figure 1: RNN with 1 recurrent hidden layer (left). LSTM network with memory block in hidden layer.

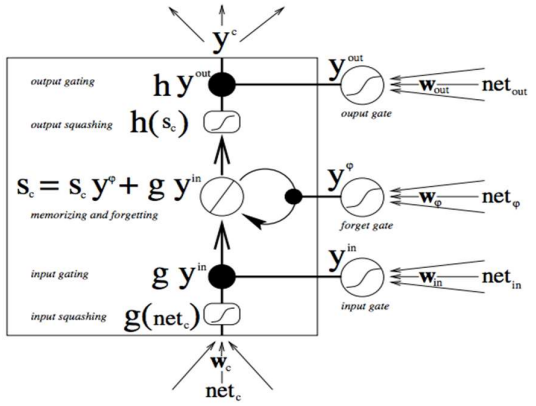


Figure 2: The traditional LSTM cell has a linear unit with a recurrent self-connection with weight 1.0.

We used LSTM because we need to take into account the dependency that exists between words in the reviews in order to classify them appropriately to its class (or rating). We also used the keras library in python to build the model of the LSTM network[3].

The pseudocode for LSTM algorithm is as follows:

Algorithm LSTM ()

```
1. words_in_dataset = tf.placeholder(tf.float32,
[time_steps, batch_size, num_features])
2. lstm = tf.contrib.rnn.BasicLSTMCell
(lstm_size)
    # Initial state of the LSTM memory.
3. hidden_state = tf.zeros([batch_size,
lstm.state_size])
4. current_state = tf.zeros([batch_size,
lstm.state_size])
5. state = hidden_state, current_state
   probabilities = []
6. loss = 0.0
7. for current_batch_of_words in
words_in_dataset do
    # The value of state is updated after
processing    each batch of words.
8. output, state = lstm(current_batch_of_words,
state)
   # The LSTM output can be used to make
next   word predictions
9. logits = tf.matmul(output, softmax_w) +
softmax_b
10. probabilities.append(tf.nn.softmax(logits))
11. loss += loss_function(probabilities,
target_words)
```

3. EXPERIMENTAL EVALUATION

3.1 Methodology

The Yelp data set came with following data:

1. Restaurant information
2. Customer information who rated these restaurant
3. The rating information with text reviews

After collecting the data various steps were performed in training and testing the model:

Step 1:
The Yelp dataset contained different JSON files, out of which the reading is performed on review.json file, since it contained the reviews and ratings given by the customers on which the prediction is to be performed. The script for reading is 'readingData.py'. The input file to this script was 2.5 GB so we had to chunk it into smaller pieces to be able to work with the data due to limitation of memory.

Step 2:
We generate the vector representation of the words in the review, using word2vec model, using logging library. We do not necessarily have to train our own model. Instead we can use a global library. However, we preferred to do so as it gives more insight into the problem as to what is going on. The second script after reading is performed is 'word2vect.py'. The input to this script was 'review1', 'review2', 'review3', 'review4', 'review5'. Due to limitation of memory we gave only first five review files for training as input, and these five files contain around half a million reviews. The output of this was a model called '100features_10minwords_10context'

Step 3:
Mapped the individual reviews to the word vectors and stored them in the pandas dataframe using pandas and re library. The third script is 'text to vector conversion as per trained word2vec.py'. The input to this script was model generated in the previous script

called '100features_10minwords_10context'. The output is a pickle file called 'business_data' with vectorised words for the chunked review files.

Step 4:
Prepared the input and output data for answering the modelling question using numpy library. The fourth script is- 'dataPreparation.py'. The input is 'business_data' obtained from previous script model and the output are two files 'X.npy' and 'y.npy'.

Step 5:
Trained a neural net-based model in Keras to predict ratings - a multinomial classification problem- using Keras library.
The file is 'neuralNetwork.py'. The input are two files 'X.npy' and 'y.npy', and the output is the prediction for the 44000 reviews (test set). We considered 1-5 rating as multinomial classification problem. We used LSTM version of deep neural net to accommodate NLP based input which could be of varied length and complex. We measured the accuracy using AUC.

There were some challenges that we encountered while implementing the project:

1. Our local system did not have enough RAM to support the operations on available data size(4GB).
2. We chunked data into small pieces and used only five chunks for training and only 44000 of these chunks for testing.
3. Even the review text beyond 30 words was not handled with the memory of our system so we truncated the text after 30 words.

3.2 Results

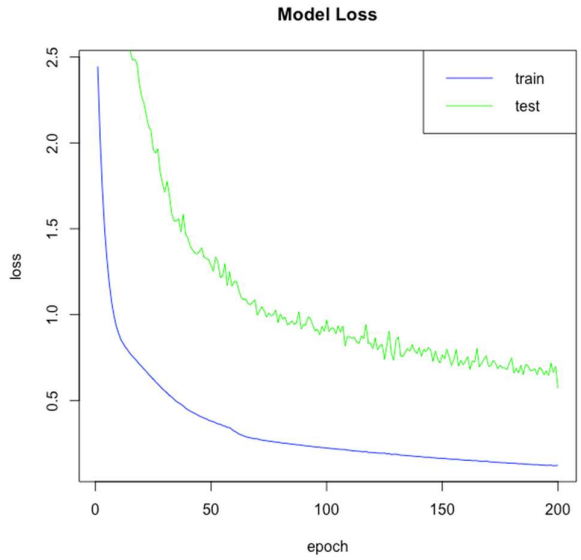
After running the project, we found the accuracy came out to be **89.796%**. We also calculated the loss function using the above formula:

$$\text{Loss}= \frac{-1}{N} \sum_{i=1}^{N} p * target_i$$

The output is as follows:

```
gument in `fit` has been renamed `epochs`.
  warnings.warn('The `nb_epoch` argument in `fit` '
Epoch 1/10
2018-04-27 11:25:57.025186: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:140] Your CPU
rts instructions that this TensorFlow binary was not compiled to use: AVX2
44400/44400 [==============================] - 47s 1ms/step - loss: 1.2388
Epoch 2/10
44400/44400 [==============================] - 42s 954us/step - loss: 1.1258
Epoch 3/10
44400/44400 [==============================] - 42s 947us/step - loss: 1.0794
Epoch 4/10
44400/44400 [==============================] - 43s 963us/step - loss: 1.0584
Epoch 5/10
44400/44400 [==============================] - 43s 962us/step - loss: 1.0413
Epoch 6/10
44400/44400 [==============================] - 42s 950us/step - loss: 1.0273
Epoch 7/10
44400/44400 [==============================] - 42s 950us/step - loss: 1.0150
Epoch 8/10
44400/44400 [==============================] - 42s 951us/step - loss: 1.0037
Epoch 9/10
44400/44400 [==============================] - 42s 950us/step - loss: 0.9945
Epoch 10/10
44400/44400 [==============================] - 42s 947us/step - loss: 0.9882
probability_predictions = model.predict(X[44400:], verbose=0)  Successfull
Calculating AUC Score / Accuracy Check....
Accuracy is
0.8979591836734694
```

Here we see, as the number of epochs increases, the loss function decreases. This relation can be depicted by the graph below.



The predictions made by our model can be shown by the above table. It shows the real ratings and the predicted ratings based on the given reviews.

	real	predicted
0	4	4
4	5	5
12	1	1
14	5	5
16	5	5
17	5	5
20	5	5
21	5	5
22	1	1
26	5	5
27	5	5
28	4	4
31	5	5
33	5	5
35	5	5
37	5	5
38	4	4
39	5	5
45	5	5
47	5	5
48	1	1
49	5	5
51	5	5

These are just 51 rows out of 44000. Based on this difference between predicted value and real value, the accuracy was calculated.

The results could be different and much more improved if we increase the validation set, and the batch size.

4. RELATED WORK

Kou and Wu tried building models to predict Yelp star rating and used the root-mean-square error (RMSE) as the error metric[4]. The best prediction performance with a 0.8223 RMSE is produced by ridge regression with TF-IDF feature selection method. Li and Zhang worked on both binary and 5-class classification problems to predict the star rating[5]. They used the mean square error (MSE) for evaluation and concluded that the best model is to use Support Vector Regression removing stop word with a test MSE of 0.0179 on the binary classification problem. Xu, Wu and Wang also looked into the 5-class classification problem and did binary sentiment analysis[6]. Their results showed that the perceptron algorithm has better performance than multi-class SVM and Nearest Neighbour on precision and recall, and the prediction results are good for a star rating of 1 and 5, but relatively poor for a rating of 2, 3, and 4. Lunkand's research suggests that Logistic Regression and SVM have similar performances but Logistic Regression performs slightly better when the model is tuned[7].

5. CONCLUSION AND FUTURE WORK

In the project we tackle the Review Rating Prediction problem for restaurant reviews on Yelp. We used the Long-Short Term Memory (LSTM) network to build the model. We trained it using around half a million reviews and tested it using 44400 reviews. We used n-gram of 30 words. We found that the accuracy of the correctly classified reviews was around 89%. We found that the ratings 4 or 5 was easily predicted, even without using the entire data set. We measured accuracy using the ROC AUC score. For accuracy calculation, the highest AUC score among all scores were chosen for all 5 classes of rating[8].

There are many avenues for improvements and future work. We list them below –

- We can try more sophisticated feature engineering methods, such as Parts-of-Speech (POS) tagging and spell-checkers, to obtain more useful n-grams.
- We can perform SVD on specific text-constructs only, such as the nouns or the adjective-noun pairs.
- We can implement an alternative for logistic regression called the ordered/ordinal logistic regression. It is an extension of logistic regression that specifically caters to classification problems where the class labels are ordered.
- To get a more detailed performance evaluation, we can try other metrics, such as precision, recall, F-score and confusion matrix. Also, for probabilistic models, we can analyse the confidence scores.
- We can compare our classifiers' performance with the performance of the traditional recommendation techniques such as collaborative filtering.
- Another avenue for future work is to test how our prediction models would perform on other business categories, such as shopping, travel, etc.

6. REFERENCES

[1] https://www.yelp.com/dataset

[2] Felix Gers, Long Short-term Memory in Recurrent Neural Networks, 2001.

[3] https://keras.io/layers/recurrent

[4] Jiyou Kou, and Luyan Wu. Building Prediction Models on Yelp Dataset. 2014

[5] Chen Li, and Jin Zhang. Prediction of Yelp Review Star Rating using Sentiment Analysis. 2014.

[6] Yun Xu, Xinhui Wu, and Qinxia Wang. Sentiment Analysis of Yelp's Ratings Based on Text Reviews. 2014.

[7] Karthik Lunkad, Prediction of yelp rating using reviews. 2015

[8] http://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html