Sentimental Analysis and Prediction based on Natural Language Processing Algorithms

Zixi Wang, Zhijin Fang, Chuxuan He, Ruibin Lyu

1. Introduction:

In this paper, we focus on the following question: how to predict the rating, or the stars, users would give to a restaurant based on his or her review. In order to tackle this question, we first found a dataset from Yelp, an online forum where people submitted their reviews on restaurants in the United States. The dataset contains three components: (1) Review Data, which includes User ID, Business ID, the review and stars given by users;(2) User Data, which is more users-focused, including User ID, number of the user's fans, number of reviews on the user, etc; (3) Business Data, including the location, business ID, and the average stars given by users. We integrate these three components into one to better analyze the data.

2. Pre-processing Step:

To better answer our research question, visualizations are generated about the dataset. We have drawn a graph of 3000 random text arrays through the word cloud library, showing the most frequent words, where the more frequently the word appears the larger the size will be. This graph gives us some basic information about what made up a user's comment and some frequent diction used. We also manually filtered out some intuitively uninformative words for the purpose of presenting some common themes.

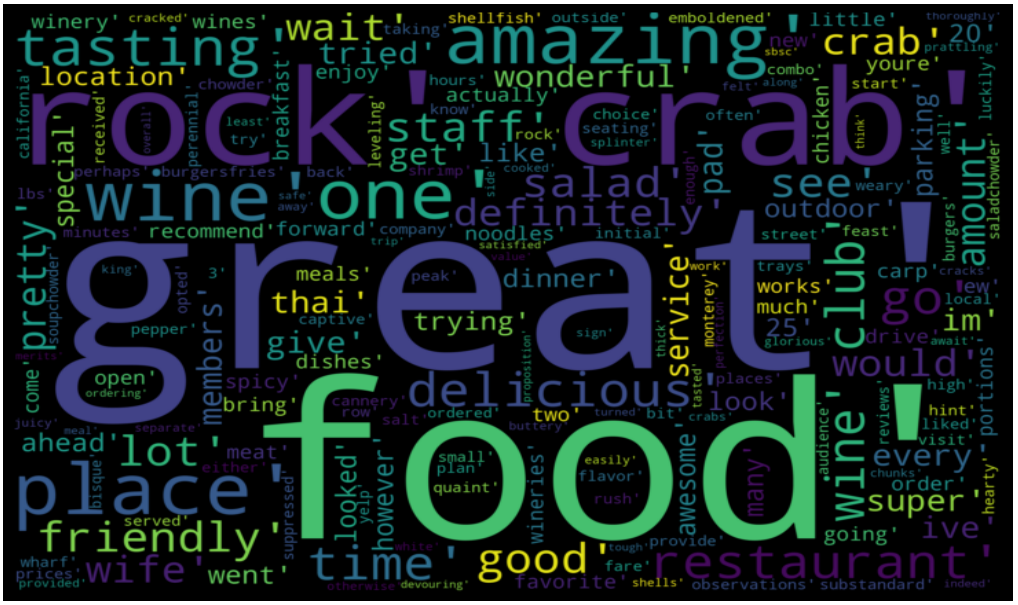


Figure 1: Wordcloud showing the most common words used in review texts

Preprocessing the review texts includes some fundamental mechanics of Natural Language Processing (NLP), so libraries around NLP like NLTK and Textblob were used. In this

dataset, the “Review" column records each user's subjective evaluation of the restaurant. These reviews contain some irrelevant words, including stopwords, punctuation marks, slang words, and so on. To make our models more accurate, we take in a string of text and perform the following: remove all punctuation, remove all stopwords, and finally return a list of the cleaned text.

Furthermore, customer reviews of restaurants were translated into concrete scores – polarity and subjectivity. To generate the scores, we use the TextBlob library. To be specific, polarity is ranged from -1 and 1, where “-1” refers to negative sentiment and “1” refers to positive sentiment. Subjectivity returns a score in the 0-1 range, and subjectivity quantifies the amount of personal opinion and objective information contained in the text. Therefore, it’s clear that these two new features are good parameters that can be used in sentiment analysis and provide essential information for our classification models. Generally, word sentiment is defined based on the strength and semantic orientation of the word. We need a predefined lexicon of positive and negative words, and a review will be represented by a bag of words. After assigning each word an individual score, the final sentiment is calculated by some pooling operation like taking an average of all the sentiments. Another feature of TextBlob is the powerful analysis of semantic symbols. For example, it can analyze emoticons and exclamation marks. [1]

We then combine the word array into a sentence by using the “join” function, where each letter is separated by a space. This format allows us to use “textblob” to analyze the polarity of the words. In order to better classify the review from customers, we also transformed our “Star” column into binary 0-1 labeling. In the dataset, the “Star” column had five types of variables: one, two, three, four, and five stars. We try to ignore all three stars because when a customer gives a three - a star evaluation of a restaurant, we assume that the customer has neither positive nor negative attitudes towards that specific restaurant, which means the emotion of the customer is generally considered to be natural. Therefore, we decided to ignore three stars in the process of changing the data, after which we combined one and two stars together as 0, and four and five stars together as 1. For classification models used in this experiment, we evaluate their performance on both their ability to predict stars and their ability to predict sentiment labels.

Besides using TextBlob as our decoder of text data, Keras Tokenizer was used along with LSTM neural network to generate vectorized text corpus of the original review texts. It only retains the 1000 most-common words and converts texts into space-separated sequences of words, that are machine-readable language. First, the frequency of each word in the dataset was calculated and stored in a word index dictionary, then each target word was converted into lists of integers based on the generated dictionary. The output data frame along with the labels of each review text made up the eventual training samples for the LSTM classification model.

In [94]: df_final

Out[94]:

	0	1	2	3	4	5	6	7	8	9	...	191	192	193	194	195	196	197	198	199	label
0	0	0	0	0	0	0	0	0	0	0	...	114	46	69	130	55	21	55	65	9	1
1	0	0	0	0	0	0	0	0	0	0	...	184	837	299	51	570	24	1557	4338	2452	1
2	0	0	0	0	0	0	0	0	0	0	...	20	40	10	23	103	399	4344	23	3087	0
3	36	288	25	143	4345	4346	3088	960	11	1393	...	313	519	140	1395	100	98	1276	267	422	1
4	0	0	0	0	0	0	0	0	0	0	...	203	962	55	65	69	46	304	573	1562	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	0	0	0	0	0	0	0	0	0	0	...	2	624	133	1139	296	697	1081	4078	296	0
996	0	0	0	0	0	0	0	0	0	0	...	8913	8914	2013	391	1011	33	324	8915	49	0
997	0	0	0	0	0	0	0	0	0	0	...	1112	69	58	19	46	4	88	334	3	1
998	0	0	0	0	0	0	0	0	0	0	...	5	720	72	167	280	627	54	10	641	0
999	0	0	0	0	0	0	0	0	0	0	...	11	43	6	734	57	19	109	329	3	1

1000 rows × 201 columns

Figure 2: LSTM input data processed by tokenizer

Note that the data wasn't balanced purposefully, because we have only a small amount of data, and some model of our choice tends to have better performance on a lengthier dataset. Instead, we implement weights on each of our models to cancel out the imbalance distribution's effect on our classification.

3. Experimental Setup:

1) NLTK and TextBlobt:
We regard the review prediction problem as a classification problem including binary, where we try to directly predict the star rating, and multi-classification, where we convert star ratings into a positive and a negative category. The tool and model we mainly use in this project are NLTK (Natural Language Toolkit, a well-known Python natural language processing tool), Long short-term memory (LSTM, a type of Recurrent Neural Network capable of learning order dependence in sequence prediction problems), and several machine learning models including Logistic Regression, Random Forest. The dataset was split into training data and testing (validation) data, then we used training data to train our models. At last, we use the test data to derive the accuracy of the model so that we can evaluate whether the model is reliable or not, and we can further run cross-validation on the entire dataset to tune the hyperparameter of our ideal model to increase the accuracy. For the rest of the paper, Part 2 is the pre-processing of the dataset which explains the process of cleaning up the target features so that it retains the maximum amount of information for further classification models. Part 3 is an experimental setup that detailedly describes classification methods used to predict labels. Part 4 is results and analysis, in this part we discuss the results, draw conclusions on these results and propose potential elaboration on the experiment.

To answer our research question, we utilize a powerful natural language processing tool "NLTK, which provides over a hundred corpus resources including WordNet, as well as a

range of functions such as classification, word separation, stemming, lexical annotation, dependency analysis, semantic inference, etc." [2] We use NLTK to predict whether the users like the restaurant or not by evaluating the emotional component of the user's reviews. Intuitively, when a user likes a restaurant, the user will use complimentary words to evaluate the restaurant. A user may praise a restaurant by saying that they love the food or the environment in the restaurant. On the other hand, if a user is not satisfied with the service or other aspects of the restaurant, they are likely to use negative words to criticize the restaurant. NLTK can capture the emotion, as well as subjectivity, in users' reviews to interpret customer attitudes and classify the users in terms of their attitudes so that we can get more information about the restaurant and its impact on users.

2) Random Forest:
Random forest is a bagging ensemble of decision trees. The ultimate classification or regression was achieved by majority voting, as the prediction of each tree will impose a partial weight on deciding the final prediction. As an enhanced ensembling regressor, for each tree, we select a subset of the attributes (recommended square root of |A|) and build trees using just these attributes. Based on the wisdom of the crowd, each decision tree in the forest outputs a prediction, and the value with the most predictions becomes the final classification result of this model. In order to analyze text-based information, this research would have to use the non-textual features and generated polarity and subjectivity scores and input them into the random forest regression model to forecast the stars or labels.

3) Logistic Regression:
Logistic Regression is a typical statistical model originally used to describe properties of population growth in biology. The input of logistic regression will undergo numerical transformations by parameters, such as weights or coefficients, to predict the output suggested by the dataset. The key difference from a linear regression model is that it outputs only binary values (1 or 0) rather than numerical values. As it will be applied as a machine learning model, the algorithm will be adjusting these parameters to optimize the accuracy of the model's prediction based on the testing data (in this case, multiple features extracted from reading the text-based comments). A major characteristic of this model is that it can only be used to predict binary results, which enables us to predict the positive or negative labels that each review falls in. However, logistic regression is unable to directly predict the stars.

4) Long short-term memory（LSTM）
The Recurrent Neural Network (RNN) is a type of neural network that is able to remember the past and make decisions based on the information learned from the past. What makes it unique is that it operates under a temporal sequence (each node has a time-varying activation), which makes it a stronger model for many particular tasks such as analyzing speech data. As a special type of RNN, Long-Short Term Memory, a deep learning algorithm for neuro-linguistic computation, is particularly convenient for sentimental analysis and exploring customer satisfaction in this study. In comparison with the traditional RNN language model, LSTM is better at analyzing the emotion of long sentences. LSTM networks

are well suited for classification, processing, and forecasting based on time series data, as there may be lags of unknown duration between important events in the time series. In our experiment, we are analyzing users' reviews, so it is important to look at the entire context of the sentence. Therefore, bidirectional LSTM is the perfect fit because it is able to persist and utilize both recent information and the past one, making it a flexible learning model that doesn't depend too much on long-term memories. The bidirectional trait makes bidirectional LSTM the perfect fit for natural language processing because it provides the model with multiple contexts of the sentence. For example, in forward processing, the computer sees, "I have a..."; in backward processing, the computer sees, "... and a cat". Having information from the starting and ending parts of the sentence, it is easier for the network to understand the sentence and make predictions.

We use these methods in our experiments. First, the value of Review_polarity and Review_subjectivity is represented in the two columns of the graph below.

	Star	Review	Review_polarity	Review_subjectivity
0	5.0	great burgersfries salad burgers hint salt pep...	0.391667	0.416667
1	5.0	bit weary trying shellfish company wharf often...	0.316921	0.543040
2	3.0	tough one merits wine fairly average understan...	0.289090	0.582319
3	5.0	love trying fresh seafood piers wharfs seaside...	0.300470	0.573266
4	4.0	stopped hungry snacks browsed store since spar...	0.261458	0.456250
...	...	...	...	...
995	3.0	much hype bland food nicely prepared dont get ...	0.308144	0.615152
996	3.0	cheapest hotel sb carpinteria goleta area staf...	0.252083	0.504167
997	4.0	beautiful tasting rooms amazing view great sel...	0.539583	0.733333
998	3.0	since dont think anyone cares think food innou...	0.189502	0.490978
999	4.0	came friends suggestion knowledgable beer help...	0.180500	0.436000

Figure 3: Training data integrated with review polarity score and review subjectivity score

We combine these two features with other features as our explanatory variable. The Star column is used as the response variable. First, the linear regression algorithm is used to predict Stars, but the score (0.2665) is very low so other methods are used to analyze the data. Thus, random forest is used, which increases the accuracy to 0.4797.

Apart from predicting the stars of a given review, we also predict the labels 0-1 of the review. We also try to use the Logistic regression to run the labels 0-1 of the review, we try to run 5000 data from the dataset, at first we just run the default Logistic Regression, and the result is 0.8053 which looks relatively high. However, since our data is unbalanced, the sum of the one and two stars obviously lower than the sum of four and five stars, so we try to set weight for our Logistic regression, the weight would balance the data to make the negative part of the conclusion more importance, then we weight one and two starts in 0.15, and weight four and five stars in 0.85, our modified Logistic regression results the accuracy in 0.7519, which shows the default accuracy is just inflated.

We use Random Forest Classifier with a default value, and we also add weight size, where the predicted accuracy score for our test data is 0.7868. Cross-validation was used with a Randomized Search CV for every hyperparameter in the random forest to get the best parameters. In the end, we fit 3 folds for each of the 100 candidates, and there are a total of 300 fits. Our final result parameters include 1000 n_estimator, 5 minimum sample split, 4 minimum sample leaf, auto max features, 100 max depth, and activation of bootstrapping. We fit these parameters into the Random Forest Classifier, and our final accuracy for predicting our random testing data is 0.8095. If we compare this result with our original default Random Forest Classifier accuracy, we get a round two percent improvement. Random Forest Classifier also produces the top five important columns: “Review_polarity”, “Review_subjectivity”, “Users_Ave_Star”, “User_Review_count, and “User_Useful_count”. This further substantiates our earlier assumption that text polarity and subjectivity are informative features for classification models.

We also try to use the LSTM (Long short-term memory) method. After we get the final_data(include the top common words from “Review” in integer-encoded vocabulary list form and the “Label” column. We split our final data into a 2:8 ratio of testing and training data. To make up our LSTM integrated neural network, we implemented one layer of embedding that parses through the vocabulary list and turns them into real-valued vectors of fixed length, one layer of LSTM and one dense layer performs a matrix-vector multiplication that finalizes the dimension of the previous layers. Adam was chosen as the optimizer and binary cross entropy loss function was used to predict labels and sparse cross entropy was used to predict stars.

Table 1: Summary of LSTM neural network model

Layer	Output Shape	Parameter Number
embedding_8 (Embedding)	(None, 200, 64)	640000
lstm_7 (LSTM)	(None, 64)	33024
dense_7 (Dense)	(None, 1)	65

```
Epoch 1/15
270/270 [==============================] - 113s 404ms/step - loss: 0.5045 - acc: 0.7553 - val_loss: 0.3961 - val_acc: 0.8153
Epoch 2/15
270/270 [==============================] - 106s 391ms/step - loss: 0.3388 - acc: 0.8570 - val_loss: 0.3461 - val_acc: 0.8498
Epoch 3/15
270/270 [==============================] - 110s 407ms/step - loss: 0.2745 - acc: 0.8891 - val_loss: 0.3412 - val_acc: 0.8496
Epoch 4/15
270/270 [==============================] - 107s 395ms/step - loss: 0.2432 - acc: 0.9042 - val_loss: 0.3393 - val_acc: 0.8548
Epoch 5/15
270/270 [==============================] - 107s 395ms/step - loss: 0.2113 - acc: 0.9190 - val_loss: 0.3779 - val_acc: 0.8574
Epoch 6/15
270/270 [==============================] - 110s 407ms/step - loss: 0.1833 - acc: 0.9309 - val_loss: 0.3759 - val_acc: 0.8558
Epoch 7/15
270/270 [==============================] - 106s 393ms/step - loss: 0.1687 - acc: 0.9370 - val_loss: 0.4664 - val_acc: 0.8471
Epoch 8/15
270/270 [==============================] - 108s 399ms/step - loss: 0.1471 - acc: 0.9450 - val_loss: 0.4607 - val_acc: 0.8519
Epoch 9/15
270/270 [==============================] - 109s 404ms/step - loss: 0.1268 - acc: 0.9535 - val_loss: 0.4768 - val_acc: 0.8513
Epoch 10/15
270/270 [==============================] - 106s 394ms/step - loss: 0.1116 - acc: 0.9604 - val_loss: 0.5612 - val_acc: 0.8477
Epoch 11/15
270/270 [==============================] - 106s 394ms/step - loss: 0.1071 - acc: 0.9623 - val_loss: 0.5437 - val_acc: 0.8434
Epoch 12/15
270/270 [==============================] - 120s 443ms/step - loss: 0.1008 - acc: 0.9654 - val_loss: 0.5449 - val_acc: 0.8424
Epoch 13/15
270/270 [==============================] - 107s 397ms/step - loss: 0.0985 - acc: 0.9651 - val_loss: 0.6482 - val_acc: 0.8432
Epoch 14/15
270/270 [==============================] - 111s 411ms/step - loss: 0.0809 - acc: 0.9720 - val_loss: 0.6087 - val_acc: 0.8413
Epoch 15/15
270/270 [==============================] - 106s 393ms/step - loss: 0.0711 - acc: 0.9764 - val_loss: 0.6397 - val_acc: 0.8387
```

Figure 5: LSTM neural network training process with accuracies and losses based on epochs

We train the model with 20 epochs and a batch size of 128, and start to see that the model is evidently overfitting after 6 epochs. When we perform the model on testing data, the training accuracy increases while the validation accuracy remains at 0.84. Most importantly, this accuracy shows that a model that only trains on text data with no other informative features is somewhat having limitations.

4. Results and Analysis:

After attempting to use optimized simplistic LSTM, Random Forest, and Logistic Regression, it is substantiated that Random Forest and LSTM performs best and it is able to forecast label of ratings precisely based on the reviews. Star means the prediction of star rating from 1-5 and label means the prediction of the label from 0-1 where 0 is a negative review (Star 1, 2) and 1 is a positive review (Star 4, 5).

Table 2: Methods with their respective testing accuracy

Methods	Test Accuracy
Linear Regression(Label)	0.2665
Random Forest (Star)	0.4797
Random Forest (Label)	0.8095
LSTM (Star)	0.8388
LSTM (Label)	0.5539
Logistic Regression(Label)	0.7431
Logistic Regression with weight(Label)	0.3845

From the graph, we found that Logistic Regression gives us a decent test accuracy; even though this test accuracy is high, Logistic Regression is still not the best method. Our dataset is a highly unbalanced dataset, we may see that there is a high percentage for our and five stars. The ratio between one and two stars with four and five stars is near 3:20.

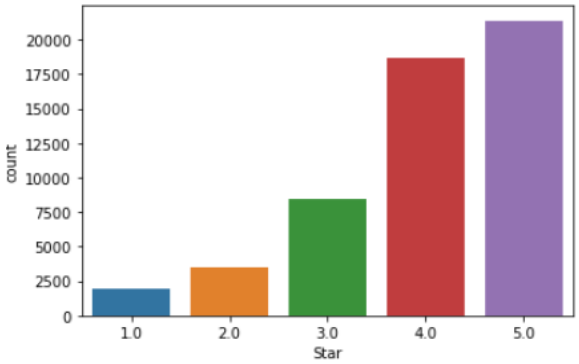
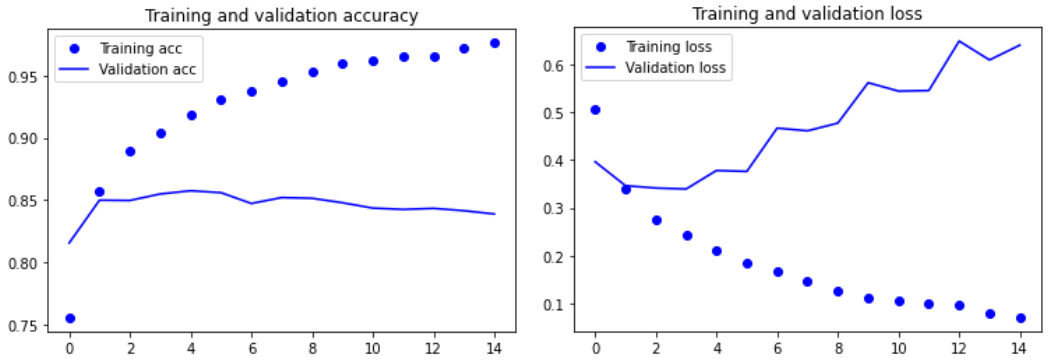


Figure 6: Bar graph showing the distribution of dataset

Logistic regression does not support imbalanced classification directly. After we add weight to both data, we can see our accuracy drop rapidly. Our sample size is still not big enough, this situation might affect the result of the Logistic regression; in addition, Logistic regression assumes there is a linearity between the predicted variable and the predictor variables. However, in the real world, it is unlikely that observations such as restaurants are linear and separable.

From our experiments, we found that LSTM would give a higher testing accuracy than Random Forest both on predicting stars and prediction labels. It is also observed that LSTM often has a tendency to overfit the training data. In our case, as soon as the fourth epoch, training accuracy reaches almost 90.4% training accuracy and 85.73% validation accuracy eventually for label prediction; as soon as the sixth epoch, 85.73% training accuracy and 55.43% validation accuracy eventually for star prediction.



Figures 7 & 8: Training and validation accuracy and loss through LSTM neural network training epochs

We have seen potential in implementing LSTM neural networks on predicting ratings based on only text data. However, its performance on the testing dataset wasn't optimized.

LSTM-integrated neural networks often require tuning and modification (e.g. adding dropout layers to avoid overfitting) but we didn't have enough time for the optimization. Also, during Tokenizater processing vocabulary through raw texts, because the words that appear most frequently are not necessarily full of emotional color, and may also be words without any emotional tendency. Some words may be meaningless, or may even be biased with our conclusion tendency. Secondly, when we convert these 10000 words into integers, we only get a lot of irregularities and related data, so our conclusions based on LSTM are prone to overfitting. On the other hand, random forest's performance on a wide variety of features beside text data suggests that generated features such as polarity and subjectivity are as informative as the review text itself. It is important that none of the original text dictions was a part of the input into the random forest model; instead, we merely chose polarity and subjectivity along with some features from the original dataset that describes the reviewing user and evaluation. This suggests that random forest has a strong ability in seeking hidden patterns among seemingly unrelated features and handling predictions based on a wide range of features, while LSTM is suitable when we only have linguistic data and a relatively lower-dimensional dataset. This is further supported by LSTM's remarkable performance on predicting stars. While normal classification models aren't able to directly process text, they can only use generated feature from NLP processors like NLTK, LSTM neural network is able to abstract more insights correlated with user rating from text emotion, which shows NLP preprocessing is powerful for building a strong classifier to predict user ratings or labels.

When human laborers are parsing sentences, their analysis lenses are often affected by subjective biases and may result in unrepresentative conclusions drawn from the data.
Nowadays, people may express their subjective opinions implicitly, without directly using words that demonstrate negative emotions, and this might result in a higher polarity score for negative comments and a lower polarity score for positive comments.

Potential improvement on this task can be elaborated on optimizing the neural network layers to avoid overfitting, e.g. adding more dropout layers, due to the fact that neural networks can attain high test accuracy with only one feature included. The neural network's loss function should also penalize positive comments more since we have a higher proportion of positive reviews in our datasets like weighting done by random forest and logistic regression.

Reference

(1) Shah, Parthvi. “My Absolute Go-to for Sentiment Analysis - Textblob.” *Medium*, Towards Data Science, 6 Nov. 2020, https://towardsdatascience.com/my-absolute-go-to--for-sentiment-analysis-textblob-3ac3a11d524.

(2) Lutkevich, Ben, and Ed Burns. “What Is Natural Language Processing? An Introduction to NLP.” *Enterprise AI*, TechTarget, 2 Mar. 2021, https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP.

(3) Saxena, Shipra. “LSTM: Introduction to LSTM: Long Short Term Memor.” *Analytics Vidhya*, 2 Dec. 2022, https://www.analyticsvidhya.com/blog/2021/03/introduction-to–long-short-term-memory-lstm/.