Yelp Review Quality Prediction Project Report

Kushal D'Souza
kdsouza@usc.edu

Linda Wong
wonglind@usc.edu

1 Introduction

In this project, we aim to use the Yelp Dataset in order to predict how useful, funny and cool a new Yelp review is going to be. For the purpose of this project, we will restrict the dataset to only those businesses that are restaurants. The aim of this project is to enable Yelp to promote reviews that the prediction model determines to have a high score in the"useful", "funny" and "cool" categories. From our research, we understand that a lot of prior work has approached this problem by extracting numeric features (such as the number of words in the review, the number of exclamation marks, number of adjectives etc.) from the text of each review. In our project, we aim to use features that are strictly related to the word vector embedding or derived from word vector embeddings. Thus, we aim to quantify the "usefulness", "funniness" and "coolness" of a review solely based on how the words in the review are related to each other.

2 Data Prepossessing

During the first two weeks of the project, a considerable amount of time was spent in order to cleanse and format the data. This is an important step in our machine learning pipeline. This section of the report provides information on the steps taken to ensure that the data is ready to be provided as input to our model.

2.1 Data Extraction

The first step in our pipeline involves the extraction of relevant data for use in our model. We only use the data contained within the business.json and review.json files. From the review.json file, we extracted the data from every row for the following features: "review id", "business id", "text", "useful", "funny", "cool". The text data would eventually be processed and would serve as input to our prediction model. The values for the "useful" , "funny", and "cool" features would also be processed and would serve as labels to our model.

2.2 Feature Clipping

We noticed that the distribution of the ratings present in the dataset for "useful", "funny", and "cool" is heavily skewed towards 0, i.e. a large number of the ratings for each review have a value of either 0 and 1. The rest of the ratings have values for "useful", "funny" and "cool" that are mostly distributed between the numbers 0 to 30. We also noticed that ratings having a value larger than 30 are much fewer in number. Table 1 shows the distribution of the number of ratings across each of the features in the dataset. In order to ensure that anomalies (for example: value of 1300 for useful) in the dataset do not adversely affect our labeled data when training, we performed feature clipping on the values present in the "useful", "funny" and "cool" columns in the data. There are two methods of dealing with outliers in the data. One of the methods is to take the logarithm of every value. This is known as log scaling. However, we noticed that the distribution of data was heavily skewed even after applying logarithmic scaling. The other method is to clip the values at a certain number so that every review that has a count for "useful", "funny" and "cool" above a certain value is just set to a threshold value. This is known as feature clipping. We applied feature clipping on our data in order to minimize the effect of outliers on our prediction model. We clipped our data such that the values for "useful" range from 3 to 27, the values for "funny" range from 3 to 20 and the values of "cool" range from 3 to 24.

2.3 Data Normalization

The next step in our pipeline is to normalize the counts for "useful", "funny" and "cool" such that they fall between the ranges of 0 to 1. A value

Range	Useful	Funny	Cool
0-9	4148278	4180159	4173089
10-19	40402	15647	20001
20-29	7225	3281	4550
30-39	2674	1236	1778
40-49	1197	564	913
50-59	623	282	477
60-69	415	159	316
70-79	245	89	179
80-89	153	52	134
90-99	119	41	95
99+	353	174	152

Table 1: Initial distribution of data

of 1 for "useful" would mean that the review is extremely "useful" and a value of 0 would mean that the review is not "useful". Here, we used min-max scaling to perform normalization on the data. The formula to perform min-max scaling on each value x within a feature is provided below:

$$x_{ij} = \frac{x_{ij} - min(x_j)}{max(x_j) - min(x_j)}$$

2.4 Data Distribution

The final step in data pre-processing is to ensure that the data is distributed evenly across the spectrum from 0 to 1 for each of the labels in the normalized data set. The reason for this step is because we want to ensure that our model is exposed to a variety of data and want to avoid situations wherein there are a lot of reviews in the training data that are rated between 0 to 0.2 for being "useful", "funny" and "cool" and relatively fewer reviews having ratings from 0.7 to 1. We created three separate datasets for each of the "useful", "funny" and "cool" labels wherein we extracted 1519 rows for values from 0 to 1 with increments of 0.1 for the "useful" column, 1368 rows for "funny" and 1320 rows for "cool". Figure's 1, 2 and 3 show the final distributions for our dataset for each of the three labels.

3 Word Vector Embedding

The first task was to choose a library for obtaining the word vector embeddings which we will feed as input to our model. We have chosen Gensim's Word2Vec library. The Word2Vec library takes as input a list of lists where the outer list contains the list of reviews and the inner list contains the words for each individual review. This library will then

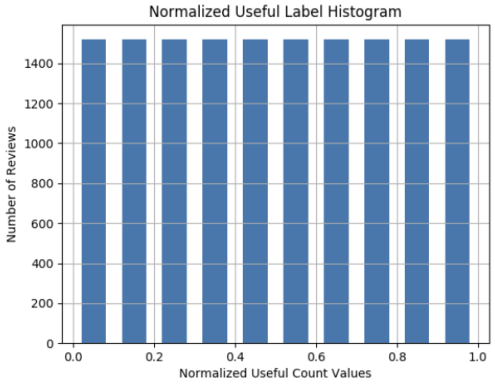

Figure 1: Histogram of the normalized dataset for useful. The dataset contains 15190 rows.

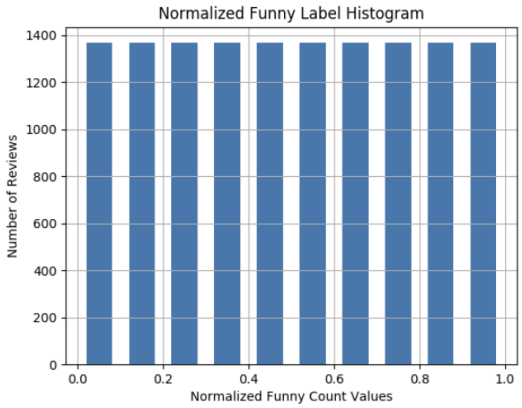

Figure 2: Histogram of the normalized dataset for funny. The dataset contains 13680 rows.

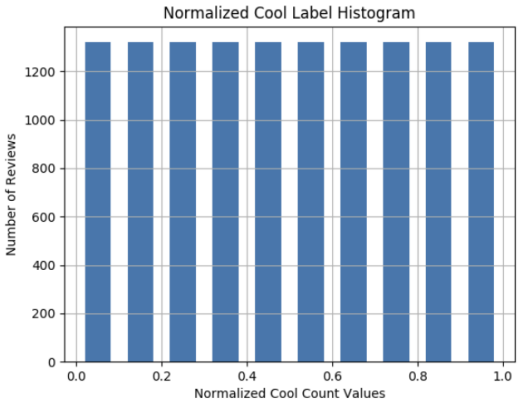

Figure 3: Histogram of the normalized dataset for cool. The dataset contains 13200 rows.

output a word vector embedding for words in the dataset. In order to train our Word2Vec model, we experimented with a variety of embeddings sizes to see which ones produced the best results. For our final model, we use an embedding size of 256, a context window of size 30 and a minimum word count of 50.

4 Model 1: Regression with LSTM

We first attempted to predict the pre-processed values of labels ("useful", "funny" and "cool") for each review. In order to do that, we created a recurrent neural network with LSTM units. We train this model using word vector embeddings of each word in the review as input and the normalized values of "useful", "funny" and "cool" as labels.

To convert review text data into a format that can be accepted by our model, we applied tokenization to each of the review text data by removing newlines, spaces and punctuation as well as lowercasing and separating all the words with commas. Next, we converted this cleansed text data to a word vector embedding representation using our Word2Vec model. The dimensions for the input to the model is restricted to the length of the longest review. For any input review that is shorter than this length, we padded it with zeros. We simply omitted words that are not in the vocabulary of our Word2Vec model.

The architecture of our model consists of an input layer which accepts the word vector embeddings of text data, two fully connected layers with ReLU as the activation function, a layer of LSTM units and an output layer of one neuron indicating the predicted value. We used mean absolute error (MAE) as our loss function and Adam for our optimizer. After experimenting with different hyperparameters, we chose 100 neurons for the LSTM layer and 32 for the batch size.

We split our dataset so that 70% of the data is used for training and 30% for testing. After training the model with different numbers of epochs, the best result from our test set gave us an accuracy of 0.11. We realized that our dataset would be more suitable for classification instead of regression and therefore, we converted our neural network model to perform binary classification of labels ("useful" vs "not useful", "funny" vs "not funny" and "cool" vs "not cool").

5 Model 2: Binary Classification with LSTM

After obtaining low accuracies for the regression model, we decided to approach the objective of identifying qualities of a review through binary classification. First, we had to convert the review label values which range from 0.0 to 1.0 to values of either 0 or 1 indicating, for example, "not useful" or "useful". We relabeled any values below 0.5 to 0 and any values equal to or above 0.5 to 1. We kept most of the architecture of our regression model the same, with the exception of the last layer and loss function. For the output layer, sigmoid is used as the activation function to indicate which binary class a sample belongs to. The loss function is changed to binary cross entropy and the updated evaluation metrics include accuracy, precision, recall and F1 score.

After training with various hyperparameters, the accuracy from our best classification model improved drastically to 0.50. We used these results as the baseline and we attempted to further improve the performance of our model using clustering and random forest techniques.

6 Model 3: Neural Network using Weighted Cluster Centers Based on Vector Embeddings

Based on the above results, we assumed that the recurrent neural networks are not able to learn much from the data due to the fact that not all words in a review contribute to whether a review is useful or not. Therefore, we tried a different approach wherein we determine if the presence or absence of certain words contribute towards the review being up voted as either "useful", "funny" and "cool". Therefore, instead of converting the words in the review into word vector embeddings and feeding these as input to the network, we would handpick a subset of words that we think are crucial to having a good review. For example, if we decide that the words: "cook", "bake" and "fry" would determine whether a review is useful or not, then for each review, we look at each word within the review and determine which of the above three crucial words it is closest to in terms of its vector embedding. Let us say that a

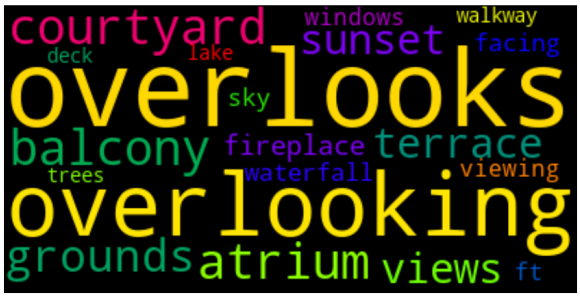

Figure 4: WordCloud for overlooks

Figure 5: WordCloud for varieties

review contains three words that are closest to "cook", four closest to "bake" and one closest to "fry". The input to the network is then the word vector embeddings for "cook", "bake" and "fry" weighted by how many words in the review are closest to them.

In order to determine what words we should use as crucial words, we took the word vector embeddings of the whole vocabulary that we calculated previously and ran K-Means clustering on these embeddings with the number of clusters set to 120. The intuition behind doing this is that, we can separate the vocabulary into 120 different groups, wherein each group represents some kind of terminology related to restaurants. Our Jupyter Notebook prints out WordCloud's for four of the clusters in order to illustrate this point. Figure 4 and 5 show the wordclouds for the cluster centers that represent the words "overlooks" and "varieties". The wordclouds show the 20 closest words by word vector embedding to these cluster centers.

The labels for each of the review's are converted into one of three classes, wherein values for "useful", "funny" and "cool" that lie between 0 to 0.33 are assigned a value of 0, values between 0.33 to 0.66 are assigned a value of 1 and values between 0.66 and 1 are assigned a value of 2. The model now predicts the most likely of these three classes for a given review as input.

The model itself consists of two dense layers having 500 and 300 hidden units respectively. There is a final dense layer that has 3 hidden units. The first two layers have ReLU activations and the last layer has a softmax activation. We use categorical crossentropy loss. On training and testing the model, we observe that the model returns an accuracy of 0.42 on the three class classification task.

7 Model 4: Random Forest Classifier with Cluster Weights as Inputs

We next tried to use a random forest classifier on our dataset. Random forest is an ensemble method that consists of multiple decision trees. For this classifier, the input data is only the weights that are assigned to each of the cluster centers. For example, if we decide that the words "cook", "bake" and "fry" would be words that determine whether a review is useful or not, then for each review, we would look at each word within the review and determine which of the above three words it is closest to in terms of its vector embedding. Let us say that the review contained three words that are closest to "cook", four closest to "bake" and one closest to "fry". The input to the network is then the array [3, 4, 1]. Notice the crucial difference here to the previous model. We still used the word vector clusters to determine the distribution of words from a review to each cluster. Previously, our input to the model was the weights multiplied by the word vector embeddings. Now, we simply passed the weights assigned to each cluster as an input to the classifier. Again, we tried to classify each review into one of three categories. We used 100 estimators and ended up with an accuracy of 0.38%.

As we can see, this model does not perform much better than the previous model. Therefore, we rephrased our classification task as a binary classification task wherein we either determine if a review is "useful", "funny" and "cool" or it is "not useful", "not funny" and "not cool". Therefore, any values for the "useful", "funny" and "cool" columns that fall below 0.5 are treated as not useful and any values in the range of 0.5 to

1 are treated as useful.

We observe that the accuracy increased significantly which is expected because we reduced the dimensionality of the output classifications. We tried the same classification task with the dense neural network to see whether the network improves the accuracy of the binary classification task. However, we did not expect to see a significant improvement using this network. We in fact observed that the dense neural network model performed marginally worse using our current input data on a binary classification task.

8 Building and Running The Model

In order to run the code, you need to have python3 installed along with Jupyter. In order to train and test our models, please install jupyter by running the following command:

```
pip install jupyter
```

You will then have to cd into the directory containing the train_and_predict.ipynb Python notebook file and run jupyter using the following command:

```
jupyter notebook
```

Once you have Jupyter running, it should open the following link in a web browser

```
http://localhost:8888/tree?
```

On opening the notebook, you can run each cells in order to use our models. On running the second cell, a text input box and a combo box should appear. Please paste the location of the yelp dataset folder using the absolute path of the folder into the text input box. Please choose a value from "useful", "funny" and "cool" within the combo box. A screenshot of this cell is shown in Figure 6. Once you have pasted the location of the yelp dataset folder and chosen a value in the combo box, you should not run this cell again. You must simply move on to running the next cell.

Some of the cells which format the data and convert each row to its corresponding input format take approximately 20 to 25 minutes to run. When a cell is running, it will have a star shown in the square brackets at the top left corner of the cell as shown in Figure 7.

Figure 6: Cell 2: Please input the location of the dataset and choose the type of data once you run this cell. Do not run this cell again after inputting your dataset location. Simply move to the next cell and continue running from there.

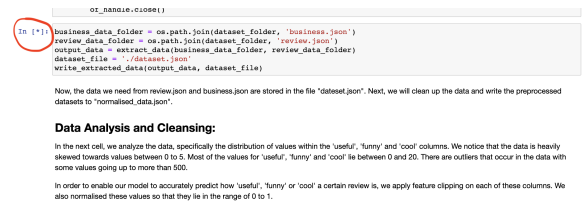

Figure 7: The circled star indicates that the cell has not completed running. Please wait for each cell to finish running before moving on to the next cell.

9 Evaluation

Tables 2 to 4 show the accuracy, precision, recall and F1 score for each of the models that we built:

Table 2: Results for Useful

	Accuracy	Precision	Recall	F1	MAE
Regression	0.11	N/A	N/A	N/A	0.25
Binary classification	0.50	0.25	0.50	0.33	N/A
3-class classification with clusters	0.40	0.40	0.40	0.39	N/A
3-class classification with random forests	0.38	0.43	0.38	0.27	N/A
Binary classification with random forests	0.59	0.59	0.59	0.59	N/A
Binary classification with clusters	0.51	0.60	0.51	0.37	N/A

10 Conclusion

In this project, we set out to create a model which can help Yelp in determining how useful, funny or cool a new review is. To start off, we tried to frame this as a regression task wherein we determine the usefulness, funniness or coolness of a review on a scale of 0 to 1, 0 being not useful and 1 being very useful. However, we soon learned that this was not feasible and we had to reframe our problem as a classification task. On running various models to classify each review, we have come to the conclusion that due to some inherent bias's present in the review data, it is difficult to model a good estimator for how useful, funny or cool a given review is. This happens due

Table 3: Results for Funny

	Accuracy	Precision	Recall	F1	MAE
Regression	0.1011	N/A	N/A	N/A	0.2612
Binary classification	0.50	0.25	0.50	0.33	N/A
3-class classification with clusters	0.44	0.44	0.44	0.41	N/A
3-class classification with random forests	0.35	0.45	0.35	0.26	N/A
Binary classification with random forests	0.59	0.59	0.59	0.59	N/A
Binary classification with clusters	0.55	0.56	0.55	0.53	N/A

Table 4: Results for Cool

	Accuracy	Precision	Recall	F1	MAE
Regression	0.10	N/A	N/A	N/A	0.25
Binary classification	0.50	0.25	0.50	0.33	N/A
3-class classification with clusters	0.35	0.12	0.35	0.18	N/A
3-class classification with random forests	0.39	0.46	0.39	0.28	N/A
Binary classification with random forests	0.57	0.57	0.57	0.57	N/A
Binary classification with clusters	0.49	0.24	0.49	0.33	N/A

to various factors, some of which are explained below:

- Two reviews that are equally similar in terms of content and length have vastly different counts for “useful“, “funny“ and “cool“ within the training data. This might be due to the fact that some restaurants are located in more populated areas and as a result have a larger number of upvotes on their reviews.

- Some reviews which are quite useful, funny, or cool simply do not have many upvotes. This could be due to the fact that they were not online for a large amount of time before the data was harvested or simply because the restaurants do not get many visitors and thus do not get many upvotes on Yelp.

- Usefulness, funniness and coolness are subjective criteria by which one measures the quality of a review. What one person or demographic finds as useful might not be useful at all in another demographic. From personal experiences of using Yelp in London, it was quite important to see the proximity of a certain restaurant to a local train station or underground station in London. We would consider a review that mentions such details as more useful than one that does not simply because public transport is the most widely used form of transport in London. However, when reading through reviews in LA for example, the proximity of the restaurant to public transport is not important to us at all. Due to the presence of such bias’s in the data, we surmise that there is no clear underlying pattern in the data that can help us in modeling a good estimator of “usefulness“, “funniness“ or “coolness“ of a review.

References

[2019] Anandarajan, Murugan and Hill, Chelsey and Nolan, Thomas. 2019. *Text Preprocessing: Maximizing the Value of Text Data.* 10.1007/978-3-319-95663-3_4.

Ben Isaacs and Xavier Mignot and Maxwell Siegelman *Predicting Usefulness of Yelp Reviews* Link

C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Weinberger *Advances in Neural Information Processing Systems 26 (pp. 3111–3119).*

Ma, Long and Zhang, Yanqing 2015. *Using Word2Vec to process big text data.* . 10.1109/BigData.2015.7364114

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. and Dean, J.. 2013. *Distributed Representations of Words and Phrases and their Compositionality.*

Mohammed K. Barakat *Predicting the Usefulness of a Yelp Review Using Machine Learning* Link

Patro, S Gopal and Sahu, Kishore Kumar. 2015 *Normalization: A Preprocessing Stage.* . IARJSET. 10.17148/IARJSET.2015.2305.

Xinyue Liu and Michel Schoemaker and Nan Zhang *Predicting Usefulness of Yelp Reviews* Link

Yao, Lirong and Guan, Yazhuo. 2018 *An Improved LSTM Structure for Natural Language Processing.* 565-569. 10.1109/IICSPI.2018.8690387

Yin, Wenpeng and Kann, Katharina and Yu, Mo and Schütze, Hinrich 2017 *Comparative Study of CNN and RNN for Natural Language Processing*

Young, Tom and Hazarika, Devamanyu and Poria, Soujanya and Cambria, Erik. 2018 *Recent Trends in Deep Learning Based Natural Language Processing [Review Article]. IEEE Computational Intelligence Magazine.* 13. 55-75. 10.1109/MCI.2018.2840738.