Yelp Restaurant Reviews: Analyzing and Predicting the Funny, Cool and Useful Review Scores

Edith Gomez, Margaret Earnest, Bao Huynh

Members Contributions:

Member	Contributions
Margaret Earnest	Data Loading, cleanup, and sampling Task 3 solutions Final Report & Slides
Edith Gomez	Task 1 Solutions Final Report & Slides
Bao Huynh	Data Loading Task 2 Solutions Final Report & Slides

Proposed Problem

The focus of this project is to analyze, and predict the useful, cool, and funny review scores present on each restaurant review of the Yelp Dataset. To tackle this problem, we propose the following tasks:

1. Predict the most popular sentiment of a review (multiclass classification)
2. Predict whether a review hits thresholds for sentiments (binary classification)
3. Predict the numerical scores for Useful, Funny, and Cool (regression)

Each one of these problems will provide a better understanding of how the users of Yelp qualify other user's reviews. Our approaches will be based on just natural language processing, with the objective of seeing if we would be able to replicate the scores based on text reviews.

Proposed solution

For each task proposed, multiple approaches are provided:

Task 1: Predict the most popular sentiment of a review

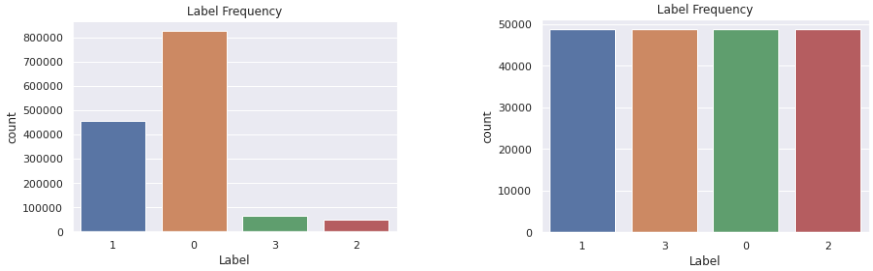
Proposed Solutions

1. Creation of Term Frequency – Inverse Document Frequency (TFIDF) features to train different machine learning classifiers (Naïve Bayes Classifier, Random Forest Classifier, Logistic Regression)
2. Fine-Tune BERT
3. Long-Short Term Memory (LSTM) Neural Network with word2vec input embeddings

For the implementation of these solutions, four target classes were created based on the useful, funny, and cool scores of the restaurant reviews. The classes were assigned to the text reviews depending on their highest score, if a draw was found between the different scores, then the review is assigned to the "equal" class indicating there is no clear predominant score for the review.

After assigning the defined classes, some data imbalance was encountered. Most of the reviews were assigned to the equal class. To tackle this situation, since the dataset has a considerably high amount of

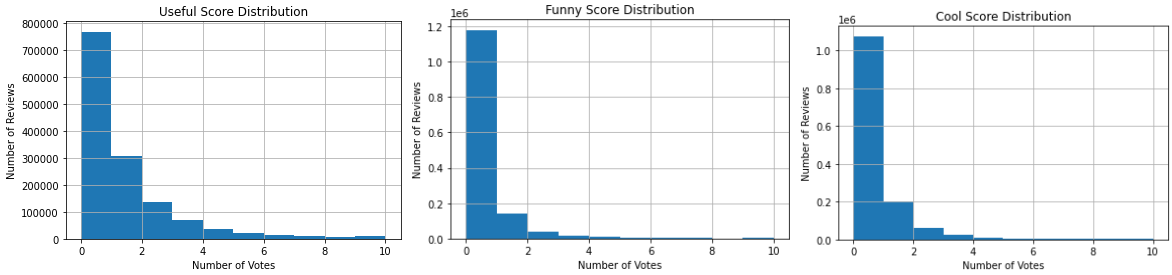
data, the under-sampling technique was implemented. The effect of the data balancing technique is shown in the following graphs, the resulting dataset contains 184,960 data instances. (1 = useful, 2 = funny, 3 = cool, 0 = none or equal votes)



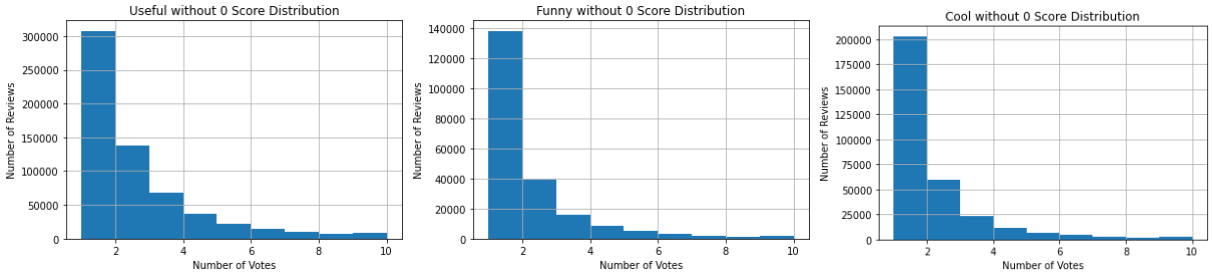
Additionally, the text reviews were preprocessed by deleting the stop words, capital letters, and special characters, as well as stemming each word. For the creation of the TF-IDF features a minimum document frequency of 100 documents was settled, with the purpose of reducing the sparsity of the data. The dataset was split into 80% training set and 20% test set for all proposed solutions.

Task 2: Predict whether a review hits thresholds for sentiments

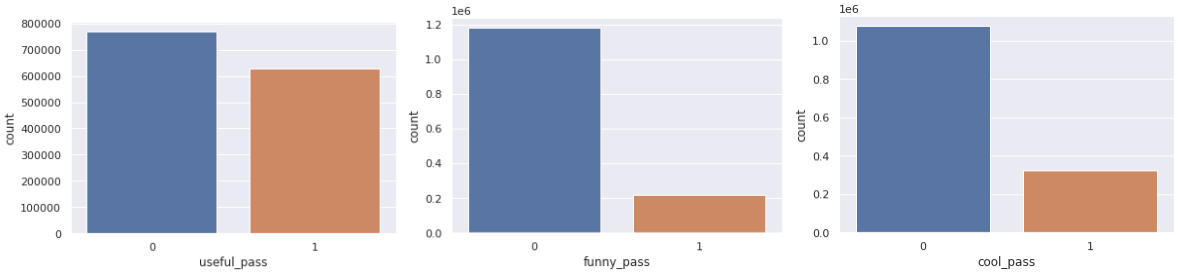
We approached this task with the solution of multi-label text classification. We first need to determine the thresholds, so we reviewed the distribution of each sentiment (Useful, Funny, and Cool):



We observed that all sentiments are heavily skewed toward 0 scores, which is reasonable since not many Yelp users provide these sentiments on reviews. However, even for the dataset without 0 scores, the sentiments distribution is still skewed toward low score values:



The sentiment distribution was imbalanced regardless of 0 scores, so we agreed to proceed with the original dataset to maintain the majority of the data instances and decided on threshold of 1 for all sentiments. Applying a threshold of 1 to each sentiment yields the following labels, with 0 representing a score of 0 and 1 representing scores of 1 or higher:



Like sentiment distribution above, we also did not balance these passing labels as balancing one would conflict with others. We then proceed with two solutions, each solution split 80-20 for training/testing:

1. Long-Short Term Memory (LSTM) Neural Network with GloVe word embeddings

The sentiment labels were extracted and combined into a single output dense layer for the model. For input, the review text was first processed, truncated, and mapped to word embeddings using a GloVe embedding dictionary. The embedding resulted in 22,152,700 parameters from text, which was then applied to a LSTM layer. The models were optimized through stochastic gradient descent with the Adam algorithm with binary cross entropy for loss. This is the general structure of the model:

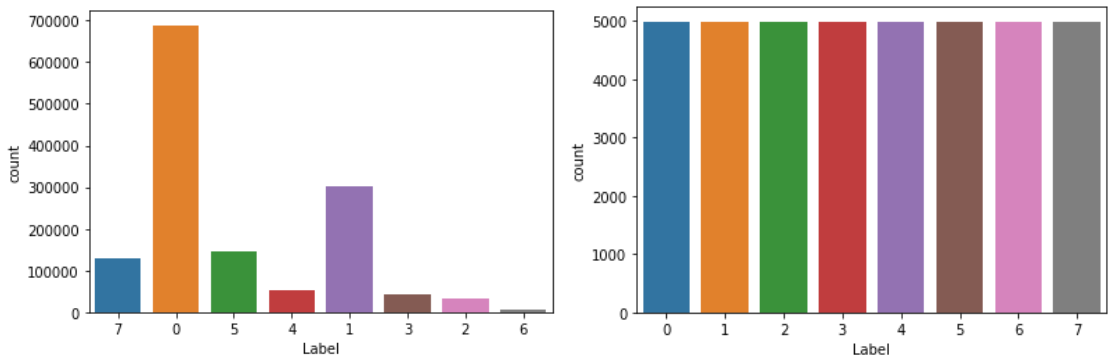
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 200)]             0

 embedding (Embedding)       (None, 200, 100)          22152700

 lstm (LSTM)                 (None, 128)               117248

 dense (Dense)               (None, 3)                 387

=================================================================
Total params: 22,270,335
Trainable params: 117,635
Non-trainable params: 22,152,700
_________________________________________________________________
```

2. Fine tune BERT (Multi-class classification)

We then approached this task with a different strategy – multi-class. The 3 sentiment labels were converted into 8 different classes as combinations of labels passing the threshold:

0-None | 1-Useful | 2-Funny | 3-Cool | 4-Useful & Funny | 5-Useful & Cool | 6-Funny & Cool | 7-All

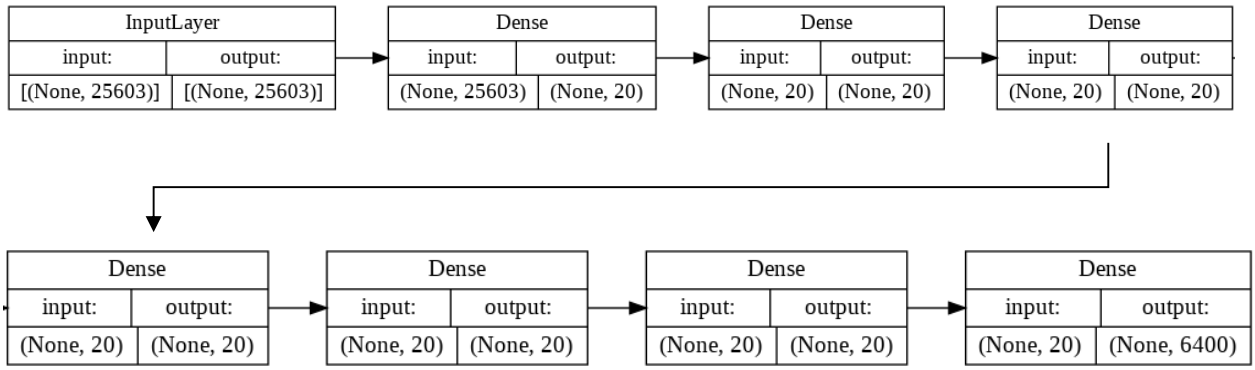
These classes also suffered from imbalanced distribution, so under-sampling was applied for balancing.



Task 3: Predict the numerical scores for Useful, Funny, and Cool

For this task, the data was under sampled to achieve a balanced class distribution in the input data. Since all three sentiment ratings heavily lean toward a value of zero, we wanted to balance the ratio of zero scores for each sentiment and non-zero scores. To do this, the data was divided into eight classes, which were every permutation of the three sentiment ratings with the possible values of zero or non-zero. 1,000 pieces of data were taken from each class for a total of 8,000 data points, which was the maximum amount that could be held in memory for this task. The text of each review was then converted to lowercase, clipped to its first 256 words, and matched against a dictionary of word embeddings. We used GloVe word embeddings with 100 dimensions to represent the text, resulting in 25,600 features from the text.

Next, the author of each review was identified in the Yelp user dataset and their useful, funny, and cool scores were recorded from their profile. These scores indicate a user's likelihood of creating a particular sentiment of review, but since it was unclear whether the magnitude of these user ratings corresponded directly to the number of upvotes a user received in a given sentiment category, these ratings were transformed into ratios, with all of a user's sentiment ratings adding up to one. For example, a user might score 0.1 for useful, 0.4 for funny, and 0.5 for cool. These scores were appended to the text features alongside the review's star rating, resulting in a total of 25,604 features.



This data was fed into three identical Keras Sequential models, with the only difference between them being the sentiment results that the model fit the data to. We initially tried training all three sentiment values together, but we had difficulty determining the mean absolute error of each individual sentiment, so we separated them out to see their scores. Each model had seven dense layers with an output size of 20 using rectified linear activation. The models were optimized through stochastic gradient descent with the Adam algorithm and used mean absolute error for loss.

Evaluation Results

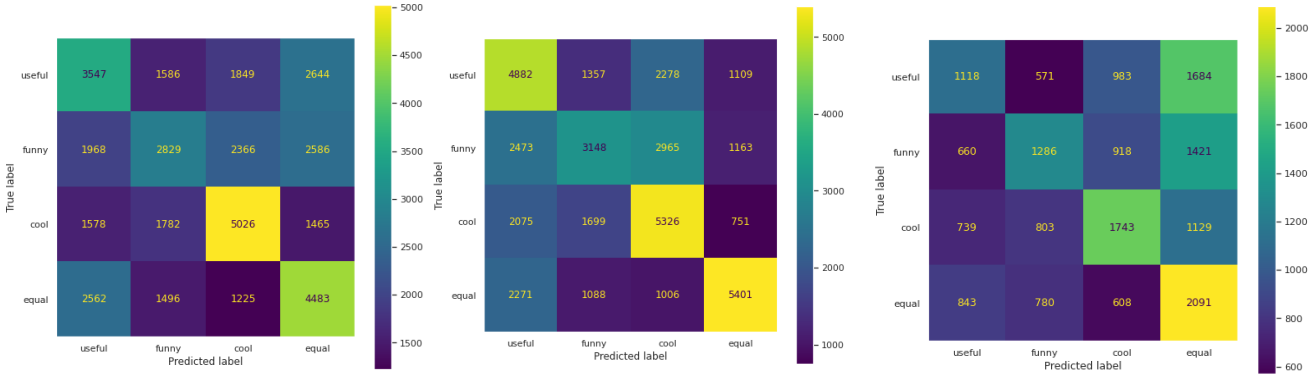
Task 1: Predict the most popular sentiment of a review

Solution 1: TFIDF features with Machine Learning Classifiers

The performance metrics for each of the classification models trained are shown in the following table:

	Logistic Regression	Random Forest	Naïve Bayes
F1 - score	0.40415	0.47973	0.39454
Accuracy	0.40739	0.48104	0.40444
Precision	0.40441	0.48777	0.40059
Recall	0.40739	0.48105	0.40444

We can observe that the Random Forest Model has higher performance metrics than Logistic Regression and Naïve Bayes, this could be attributed to the fact that Random Forest is an ensemble method that trains in our case 100 trees and takes as output the majority votes of all the trees. All of our models are able to distinguish between the classes better than a random classifier (accuracy = 0.25), however the performance is not optimal. To better illustrate the differences between the three models, confusion matrixes for each one is shown below:



Logistic Regression Random Forest Naïve Bayes

Note that all models can classify each label to some degree but have a harder time differentiating between the useful and equal classes, and the funny and equal classes.

Solution 2: Fine tune BERT

For this approach, a pre-trained BERT base uncased was fine tuned to perform classification between four classes. The tuning process was made with two epochs, it took around 7 hours to tune, which is the reason why the number of epochs was not increased. 109,485,316 parameters were trained. The evaluation metrics for this approach are shown below:

Accuracy	0.4059
Precision	0.4197
Recall	0.4059
F1 score	0.4079

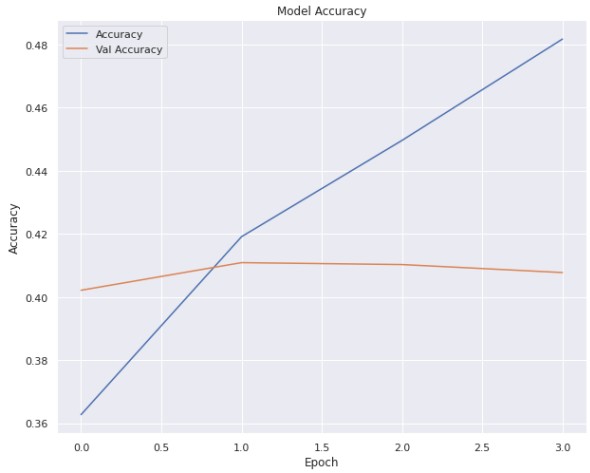
Compared with our previous solution, the fine tune BERT performed better than Logistic Regression and Naïve Bayes models, however, our previous Random Forest Model has higher accuracy, F1 score, precision, and recall. Seems like a simpler approach to the problem performs better in this case.

Solution 3: LSTM with Word2vec embeddings

For this solution, embeddings for the reviews were created based on word2vec with the skip-gram method (predicting neighbor words based on a center word). Next, a LSTM network was trained based on these embeddings. The architecture for the LSTM is shown below.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_1 (Embedding)      (None, 50, 300)           5092200

 dropout (Dropout)            (None, 50, 300)           0

 lstm (LSTM)                  (None, 50, 50)            70200

 lstm_1 (LSTM)                (None, 4)                 880

 dense (Dense)                (None, 4)                 20

=================================================================
Total params: 5,163,300
Trainable params: 5,163,300
Non-trainable params: 0
_________________________________________________________________
```

The embedding size is 300, a dropout layer was integrated to avoid overfitting. This simple LSTM should be able to capture relations between words along the text reviews. The training process was done with 10 epochs, with an early stopping condition for when the validation loss of the model did not experience any improvement. This condition made the training process terminate at 4 epochs. The resulting test metrics are presented in the table below.



Accuracy	0.33132
Precision	0.34385
Recall	0.33132
F1 score	0.24069

For this solution we observe a low F1 score, and an accuracy score that is not far from a Random Classifier. Compared to the fine-tuned BERT, this approach seems to not identify well between the classes. From all the implemented solutions, this is the one that has lowest performance in all metrics.

Task 2: Predict whether a review hits thresholds for sentiments

Running labels of passing sentiments through LSTM (trained for 5 epochs) and BERT (trained for 2 epochs) models yield the following results:

	LSTM Multi-labels	**Fine tune BERT Multi-class**
F1 - score	0.5009	0.2541
Accuracy	0.9424	0.2596
Precision	N/A	0.2566
Recall	N.A	0.2596

The accuracy of LSTM model is surprisingly high considering the imbalanced dataset and relatively low F1 score. This performance is clearly superior to the multi-class approach with BERT as the BERT model achieves low scores across all performance metrics.

Task 3: Predict the numerical scores for Useful, Funny, and Cool

We ran the Sequential model for task three on each sentiment for 100 epochs with 200 steps per epoch. This produced the following mean absolute errors:

	Mean Absolute Error
Useful	2.653
Funny	1.657
Cool	1.864

Analysis and Conclusion

	Task 1					**Task 2**	
	Logistic Regression	Random Forest	Naïve Bayes	Fine tune BERT Multi-class	LSTM Multi-class	LSTM Multi-labels	Fine tune BERT Multi-class
F1 score	0.40415	0.47973	0.39454	0.4059	0.33132	0.5009	0.2541
Accuracy	0.40739	0.48104	0.40444	0.4197	0.34385	0.9424	0.2596
Precision	0.40441	0.48777	0.40059	0.4059	0.33132	N/A	0.2566
Recall	0.40739	0.48105	0.40444	0.4079	0.24069	N.A	0.2596

We expected complexity to grow between tasks 1 and 2, so we anticipated that task 1 would have a highest accuracy score than task 2. However, the performance of task 2 proved otherwise with its oddly high accuracy through LSTM. Considering the nature of task 2, it makes sense that the multi-label approach would perform better than multi-class since each sentiment passing the threshold correlates closely with the other sentiment values to make into a group of related classes. Moreover, the dataset for multi-class is under-sampled for balancing purposes and that may also have affected performance. Since task 3 is evaluated using a continuous value, mean absolute error, it is difficult to translate the performance of the task into something that is comparable with the others.

Limitations

Our largest limitation in this project was that working in Google Colab limited our available memory and processing power. Given the lack of memory, we were only able to work with a small subset of the Yelp dataset. Likewise, lack of computational power limited the types and complexity of models we could try. Another limitation was the distribution of our input data. Although we did some simple balancing based on whether a sentiment had zero votes, we did not balance further based on values above zero, which could result in non-zero values being skewed and negatively affecting our predictions. For task three, the input data did not include much information that would indicate how many people could engage with a review, like view count or restaurant popularity. This makes it more difficult for the model to predict exact values for sentiments, even if it can determine that one review embodies a given sentiment more than another.

Future Work

If we were to continue this work on analyzing Yelp sentiments, we would make some changes to our process and models. First, we would use the WPI Cluster to train our models since it has much more memory and computational power than Google Colab and is not likely to shut off at random because the user is utilizing too many resources. We would also like to implement bidirectional LSTM architectures and multi-task learning to obtain better results. Finally, to better compute actual values for sentiment ratings, we would incorporate additional data such as review count or number of check ins for a business to get a sense of how many people might be exposed to a given review