Yelp Review Rating Prediction

Cynthia Du, Meiyi Ye, Nam Truong

1. Introduction

Traveling, and you don't know where to eat? Getting tired of your go-to spots and want to try a new restaurant tonight? Yelp is here to help distinguish which restaurants you might or might not like. As an online platform with over 200 million reviews, Yelp has become a helpful and reliable rating platform, allowing customers to decide which establishments they want to go to. Generally, a Yelp review will contain two components:

1. An overall rating (on an integer scale of 1-5), with 5 being the best and 1 being the worst. This comes in the form as "star" ratings.
2. A couple sentences or short paragraph detailing the customer's thoughts about and experiences at the business.

In this project, we take a deep dive into the Yelp dataset with over 6.9 million reviews and ratings and over 150,000 businesses. Our goal of the project is to explore the relationship between the text review and the star rating, and ultimately formulate models to predict the star rating through the review's words and sentiment. Our motivation is to predict the true star rating of a business, therefore accounting for individual bias and variance as some reviewers tend to rate more highly than others for the same level of service, quality, and product received. Despite their usefulness, Yelp reviews often exhibit bias and subjectivity, which can skew perceptions and lead to misleading ratings. There is a need for a model that can predict the true rating of a Yelp review by mitigating individual biases and emphasizing the content's sentiment, thereby providing a more accurate reflection of the business's quality.

These reviews significantly influence consumer decisions and play a crucial role in shaping the reputations of businesses. Accurately predicting the ratings of these reviews helps in understanding consumer sentiment and preferences, which is vital for both users and business owners. The objective of this project is to develop a predictive model that can determine the true star rating of a Yelp review based on its textual content. By analyzing patterns in word usage and sentiment, the model aims to offer a normalized rating that more accurately reflects the actual quality of the service.

2. Data Preprocessing

Our dataset obtained was the Yelp dataset, which comprised of 3 different files: yelp_academic_dataset_review.json, yelp_academic_dataset_user.json, and yelp_academic_dataset_business.json. Here is a breakdown of each dataset:

1. Reviews dataset: Contains the full review data (both components 1 and 2 as stated above) with the corresponding user_id that wrote the review and the business_id the review was written for
2. User dataset: Contains user data including their name and their average yelp ranking of all reviews
3. Business dataset: Contains business data and its attributes such as location, hours open, business categories (ie 'Italian', 'Sushi', etc)

First, we had to convert all of the .json files to .csv files, and then we merged the three of these datasets. We merged user data with review data based on 'user_id' to connect each review with its user's ratings, forming the df_review_user dataframe. Then, we combined this with business data on 'business_id' to ensure each review accurately matches its respective business, resulting in the df_combined dataframe. df_combined data frame has 18 features:
user_id: the id that is unique for each reviewer. The user who wrote the review
business_id: the id unique for each business, who the review was written for
stars_review: the number of stars given in this particular review
text: the text / content of the review
average_stars: the average star ranking the particular user_id gives
name: name of the business
address: address of the business
city: the city the business is located in
state: the state the business is located in
postal_code: the postal code the business is located in
latitude: latitude of the business
longitude: longitude of the business
stars_business: the average star rating of the business
review_count: number of reviews for the business
is_open: (0 or 1) if the business is currently open
attributes: specific attributes of the business (ie. 'ByAppointmentOnly', 'OutdoorSeating', 'RestaurantsDelivery')
categories: the categories the business falls under (ie. 'Food', 'Shopping', 'Restaurants', 'Tea')
hours: the hours the business is open.

After conducting a value count analysis of the number of reviews per state from our combined dataset, we observed significant variations in the volume of reviews across states. Notably,

California stood out with a substantial total of 348,856 reviews, making it an ideal candidate for our analysis. This substantial data volume ensures a robust dataset, providing a comprehensive insight into consumer sentiments and behaviors specific to California. Therefore, we decided to focus our predictive modeling efforts on this state. Furthermore, we wanted to concentrate only on restaurants as our business type, so we filtered out only the businesses with 'Restaurants' as one of their categories and 'state' as California. Finally, our data frame ended up with 211,748 observations and 18 features.

Now that we had the data we wanted to use, we removed stopwords and normalized the data by removing HTML tags, URLs, hashtags, emojis, punctuation, and non-alphabetic tokens. Furthermore, we converted all the text to lowercase, tokenized the text, and lemmatized the tokens so that our models could evaluate the text of the reviews properly.

3. Exploratory Data Analysis

To get a better understanding of the data we were working with, we constructed some descriptive statistics.

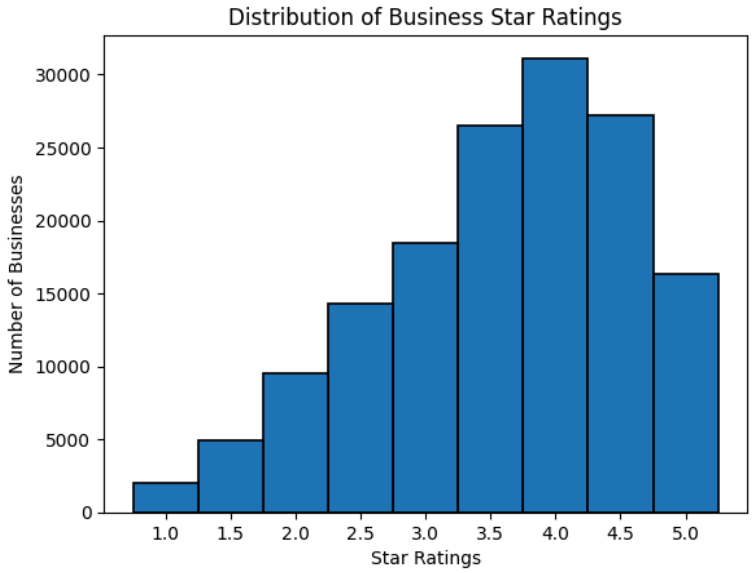

Figure 1: Distribution of Star Ratings

We can see that the distribution of business star ratings is skewed left, with the majority of the ratings between 4-5 stars. This may pose an issue as it means our dataset is imbalanced.

Table 1: Average Star Rating by City

City	Average Star Rating
Truckee	4.077692
Santa Barbara	3.918465
Carpinteria	3.894687
Summerland	3.888293
Montecito	3.873211
Isla Vista	3.842938
Goleta	3.841440

By looking at the average star rating by city, we are able to highlight the regions with particularly high customer satisfaction, which also may lead to some bias in our dataset. If the majority of the reviews come from a specific city that tends to rate more favorably, then our models might be overestimating the true rating for the restaurant. Thus, we also explored the count of reviews by city.

Table 2: Review Count by City

City	Review Count
Santa Barbara	162430
Goleta	26584
Carpinteria	11416
Isla Vista	6330
Montecito	3638
Summerland	1324
Truckee	26

After comparing the two tables, we see that some bias could be evident as mentioned above. Truckee has the highest average star rating, but it only comprises twenty-six of the thousands of reviews we have. Thus, it makes sense why Truckee has the highest star rating because it might be inflated and the use of Yelp might not be common there. We learn from statistics that the more samples an observation has, the more accurate it becomes. Thus, it is not reasonable to assume

that the true star rating of all restaurants located in Truckee is a 4.077. Furthermore we notice that most of the reviews are concentrated around Santa Barbara. Knowing this, we may not be able to generalize the results of our model to reviews in other parts of California, since our model may be slightly biased in that most of the training data comes from Santa Barbara.

Figure 2: Frequency of Words in Reviews

After standardizing the data set and removing stopwords, the words that appear the most in written reviews are ‘love’, ‘think’, ‘know’, ‘highly recommend’, ‘amazing’, ‘customer service’, etc. However, some of these words don’t seem as useful, so we underwent sentiment analysis to find the most positive and negative words in the dataset.

4. Sentiment Analysis

The process of sentiment analysis involved splitting the dataset into positive and negative reviews based on ratings, cleaning and filtering the text, and counting the frequency of words in each set. We defined a positive rating as those with 3, 4, or 5 stars, and a negative rating as those with 1 or 2 stars. Then, sentiment scores for each word (excluding pre-defined irrelevant terms) were calculated by subtracting its occurrences in negative reviews from those in positive reviews.

We removed all of the city names from this analysis, as a preliminary test yielded that ‘Santa’ and ‘Barbara’ were amongst the top positive words, which would be incorrect as the model might assume that ‘Santa’ and ‘Barbara’ are positive words, and falsely classify restaurants in Santa Barbara a higher rating.

The top words were ranked according to their sentiment scores, revealing that words like ‘good’, ‘great’, ‘food’, and ‘delicious’ were most positively associated, while ‘gross’, ‘refund’, ‘tasteless’, and ‘disgusting’ were most negatively associated. This analysis provides a clearer understanding of the words that strongly influence the sentiment in customer reviews, and

identifying these sentiment-driving words helped us gain deeper insights into customer perceptions and highlighted distinct elements that impact customer satisfaction and dissatisfaction.

Figure 3: Word Cloud of Most Positive Words

Figure 4: Word Cloud of Most Negative Words

5. Feature Selection

Due to the extensive number of words in the dataset, we explored whether feature selection would be beneficial to our model. As our data is text data and not numerical, we conducted TF-IDF vectorization, which converts text data into numerical features that machine learning models can use. Each word in the text is represented as a numerical value based on its importance within each review and across all reviews, which results in a feature matrix where each row corresponds to a review and each column corresponds to a unique word or n-gram.

Words that are important for predicting star ratings, such as ‘delicious’, ‘great’, or ‘rude’, will have higher TF-IDF scores in reviews where they are significant, whereas common words that appear in many reviews but do not contribute to distinguishing the star rating (ie. ‘menu’, ‘sit’, etc) get lower TF-IDF scores, effectively reducing their impact. Words with high TF-IDF scores in positive reviews versus negative reviews help the model learn associations between words and star ratings.

Principal Component Analysis, or PCA, was then conducted to reduce the number of TF-IDF features, focusing on the words that contribute most to the model’s performance. PCA projects the high-dimensional TF-IDF space into a lower-dimensional space, retaining the components that explain the most variance. We use the explained variance ratio to decide how many components to keep. This balances between reducing dimensionality and retaining enough information.

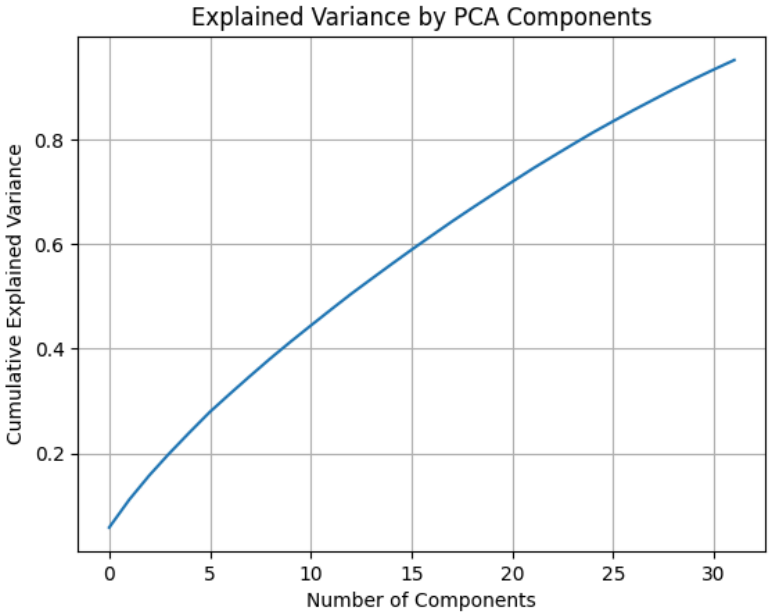

Figure 5: Explained Variance by Number of PCA Components

Based on our graph, we were able to conclude that PCA could reduce the dataset down to 30 components to retain 95% explained variance. However, when integrating PCA into our classification models, the models were unable to perform well and were unable to predict the star rating given by the user based on the text review given. In the classification models, the 5 categories were the 5 star ratings. The purpose of trying these classification models was to ensure that the models were learning and were able to correctly identify what star rating a review should be based on the sentiment and words in it. Both the Random Forest and K-Nearest Neighbors models performed poorly with an accuracy of only around 50%, even with hyperparameter tuning and finding the optimal number of neighbors for KNN through cross-validation.

Thus, we decided that PCA was unnecessary for our dataset as reducing the dataset down to 30 components doesn't make intuitive sense. It makes more sense for the model to be trained on the sentiment of all words in reviews and how they relate to each other, especially since we already removed stop words and predetermined non-influential words.

6. Modeling

In our analysis of the Yelp reviews dataset, which predominantly contained 3-star and 4-star ratings, we faced an imbalance issue. To address this, we implemented the RandomOverSampler technique, which aims to balance the dataset by randomly duplicating instances of the minority class until all classes have the same number of instances. This approach initially expanded our dataset to 500,000 observations. To manage this large dataset, we employed stratified sampling to create a more manageable subset of 50,000 observations, ensuring this smaller sample maintained the same proportion of classes as the larger dataset. Unfortunately, when we processed this balanced subset through our predictive models, we encountered repeated errors. This led us to conclude that for our specific analytical needs and model compatibility, maintaining the original, unbalanced distribution of the dataset was more effective.

6.1 LSTM Network

We first developed a Long Short-Term Memory (LSTM) network to predict the star ratings of Yelp reviews based on their textual content, utilizing LSTM's proficiency in handling sequence data, which is crucial for processing text where the order of words significantly impacts sentiment and meaning interpretation. The model is composed of several layers: an initial embedding layer that translates words into a 64-dimensional vector space to capture semantic similarities between words. This is followed by two LSTM layers; the first with 64 units to capture contextual dependencies across the text sequence, and the second with 32 units to condense this information, summarizing the critical features of the review. A dropout layer with a 50% rate is integrated to mitigate overfitting by randomly deactivating parts of the neural network during training, enhancing the model's ability to generalize better to new, unseen data. Additionally, the network includes a dense layer with 64 neurons and ReLU activation to introduce non-linearity, concluding with a single-unit linear activation layer that outputs the predicted star rating. The model, incorporating 687,617 parameters, uses the mean squared error (MSE) as the loss function to quantify the average squared discrepancy between the actual and predicted ratings, which is well-suited for regression tasks like ours.

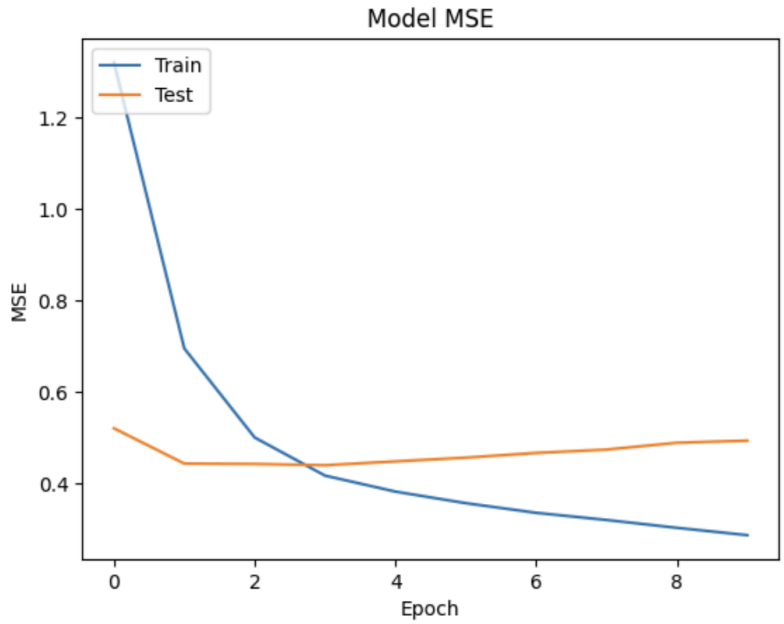

Figure 6: LSTM Training and Validation Loss Plot

The model was trained and evaluated over 10 epochs, and while it showed promising accuracy on the training dataset, the performance on the validation set depicted signs of overfitting. This was evident from the training and validation loss plot where the training loss consistently decreased, indicating the model was learning effectively, but the validation loss started to plateau and then diverged from the training loss. This divergence suggests that while the model was becoming increasingly precise on the training data, it was not performing equivalently on new, unseen data.

6.2 LSTM Network with Early Stopping and Dropout Layers

Next, we developed a Long Short-Term Memory (LSTM) network with early stopping implemented, as well as dropout layers to predict Yelp review ratings, capitalizing on the LSTM's capability to effectively handle sequential text data. Our model features an embedding layer that maps words into a 64-dimensional vector space from a vocabulary of 10,000 words, helping to reduce dimensionality and capture semantic relationships. It includes two LSTM layers, with the first layer containing 64 units to process sequences for contextual dependencies, and the second layer with 32 units to condense this information. To combat overfitting, we incorporated multiple dropout layers with a rate of 0.5 after each LSTM layer and before the final output layer, which randomly deactivates certain neurons during training. This strategy prevents the model from depending too heavily on specific features of the data. The architecture is completed with a dense layer of 64 neurons with ReLU activation for learning complex patterns and a single-neuron linear activation output layer for predicting star ratings. The model

has a total of 687,617 parameters. We implemented an Early Stopping callback that halts training if the validation loss does not improve after three epochs, effectively preventing overtraining. Training was planned for 50 epochs, but early stopping terminated it at the 6th epoch when no further improvement in validation loss was observed.

6.3 LSTM network with L2 Regularization

To address overfitting in our LSTM model for Yelp review ratings, we incorporated L2 regularization, which adds a penalty to the loss function based on the squared values of the model parameters. This method helps in smoothing the learned parameter values, thus discouraging the model from fitting too tightly to the training data and promoting better generalization on unseen data. Our LSTM model is structured with an Embedding layer, two LSTM layers with L2 regularization, a Dense layer also utilizing L2 regularization, followed by a Dropout layer to further mitigate overfitting, and concludes with a Dense output layer. We applied a regularization value of 0.01 to the LSTM and Dense layers. The model comprises a total of 687,617 trainable parameters, providing a substantial learning capacity while maintaining a balance to avoid overfitting thanks to the regularization techniques employed.

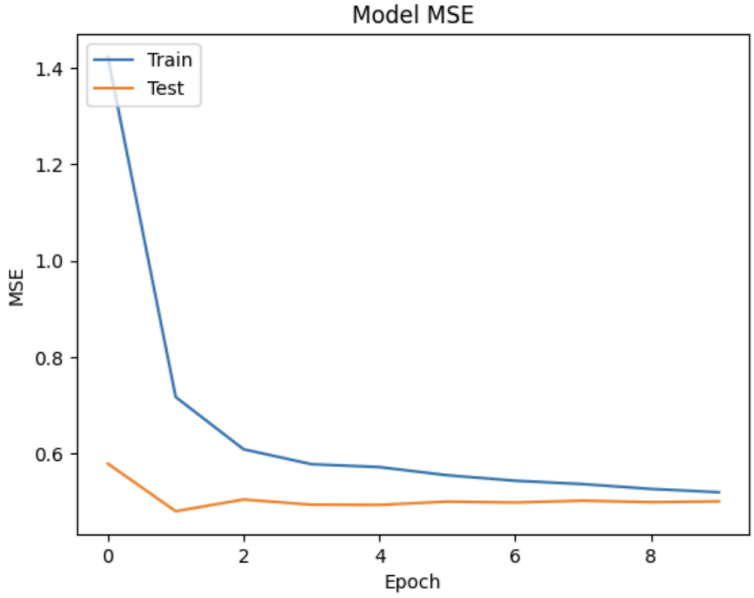

Figure 7: LSTM with L2 Regularization Training and Validation Loss Plot

The training and validation loss plots demonstrate the efficacy of L2 regularization in our setup. Both training and validation mean squared errors (MSE) closely converge as the training progresses, and the loss curves descend to a point where they almost parallel each other. This behavior suggests that our model, supported by L2 regularization, not only minimized overfitting but also retained its capability to generalize well, evident from the stable loss levels throughout the training epochs.

The final MSE value on the test set appears to stabilize around 0.4. This means that, on average, the squared difference between the predicted star ratings and the actual star ratings is 0.4. To put this into perspective, the root mean squared error (RMSE), which provides an error in the units of the target variable (ratings in this case), would be the square root of 0.4, which is approximately 0.63. This indicates that the typical prediction is about 0.63 stars away from the actual rating.

6.4 Convolutional Neural Network (CNN)

We also explored the utility of a Convolutional Neural Network (CNN), known for its effectiveness in handling data with spatial hierarchy, which can be advantageous for text data structured in sequences. Our CNN model was built with a structure comprising an initial Embedding layer configured for 10,000 vocabulary size, producing 64-dimensional embeddings. It was followed by a Convolutional layer with 128 filters of size 5, designed to capture local patterns within sequences of text. A Global Max Pooling layer was then used to reduce the dimensionality, ensuring the most significant features from each feature map were preserved for the final prediction. The network also included a Dense layer with 64 units and 'ReLU' activation to introduce non-linearity, a Dropout layer at 0.5 to mitigate overfitting by randomly omitting a subset of features during training, and a final Dense layer for regression output. The model contained a total of 689,409 trainable parameters.

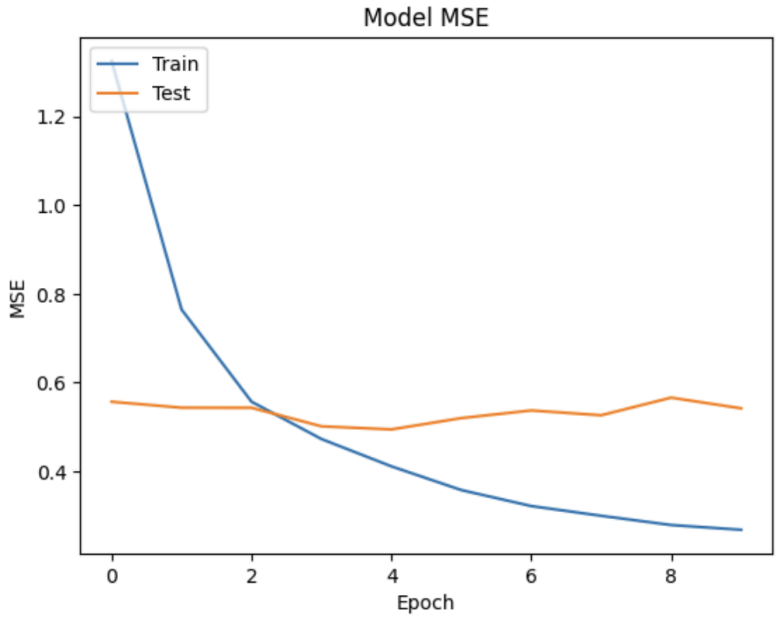

Figure 8: CNN Training and Validation Loss Plot

Upon training, the model's performance over epochs revealed initial rapid improvements in mean squared error (MSE), indicating strong learning. However, the plots showed that while the

training loss continued to decrease, the validation loss began to stabilize and then slightly increase, a classic indication of overfitting.

6.5 CNN with Early Stopping and Dropout Layers

To combat overfitting in our model for Yelp review sentiment analysis, we introduced early stopping and dropout layers into our architecture. The model is structured as follows: an input layer with an embedding of 64 dimensions for a vocabulary size of 10,000 words, followed by a 1D convolutional layer with 128 filters and a kernel size of 5. This is intended to capture spatial hierarchies in data by applying the convolution operation across the text sequences. After convolution, a global max pooling layer is used to reduce the dimensionality and summarize the features extracted by the convolutional layer. This is followed by a dense layer with 64 neurons activated by ReLU to introduce non-linearity to the model, a dropout layer with a rate of 0.5 to prevent overfitting by randomly omitting units during training, and finally, a dense output layer with a linear activation for regression tasks. The model has a total of 689,409 parameters. To further prevent overfitting, the early stopping mechanism monitors the validation loss and stops the training when it ceases to decrease, effectively reducing excessive training epochs and resource consumption. For our model, early stopping kicked in during epoch 8. This setup ensures that the model learns generalizable patterns rather than memorizing the training data.

6.6 CNN with L2 Regularization

In our Yelp review analysis, we also explored using a Convolutional Neural Network (CNN) enhanced with L2 regularization to tackle the challenge of overfitting. The model includes an embedding layer set to handle up to 10,000 unique words with an embedding dimension of 64. Following this, a convolutional layer with 128 filters and a kernel size of 5 is applied, which is ideal for capturing local patterns in the text. The CNN architecture also incorporates a global max pooling layer, which helps to reduce the dimensionality by capturing the most important feature from each feature map, and two dense layers with ReLU activation function. Dropout layers with a rate of 0.5 are strategically placed to reduce overfitting by randomly turning off a fraction of neurons during training. The final layer is a dense layer with linear activation used for the regression output. This configuration totals 689,409 trainable parameters.

To ensure the robustness of our model, we employed L2 regularization in the convolutional and dense layers, which penalizes large weights and encourages a simpler model that should generalize better on unseen data. The MSE plot from our training session shows that after initial training epochs, both training and validation losses decrease and stabilize without a significant gap between them, suggesting that L2 regularization effectively mitigated the risk of overfitting, as the model performs consistently across both training and validation datasets.

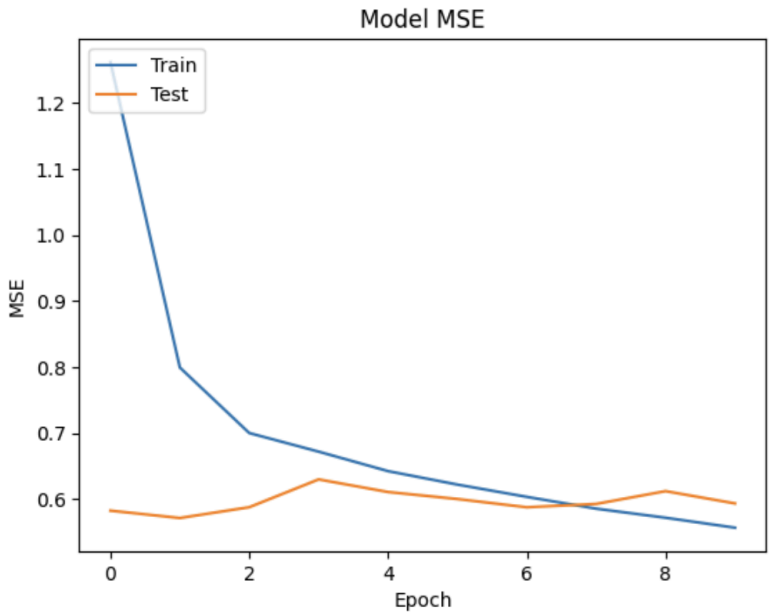

Figure 9: CNN with L2 Regularization Training and Validation Loss Plot

The final MSE value on the test set appears to stabilize around 0.6. This means that, on average, the squared difference between the predicted star ratings and the actual star ratings is 0.6. If we take the square root of the MSE 0.6, it is approximately 0.77. This tells us that the typical prediction is about 0.77 stars away from the actual rating.

6.7 BERT (Bidirectional Encoder Representations from Transformers)

Up until this point we have trained our models from scratch. This time we wanted to leverage BERT's pre-trained knowledge to achieve a greater performance. BERT was trained on a massive corpus of text data from Wikipedia and BookCorpus which is a dataset of books. When BERT was trained, some of the input tokens were randomly replaced with a [MASK] token which the model was tasked to predict. This allows the model to have a deep understanding of language.

When we decided to use BERT there were a couple of variants of the model that we had to choose from notably: Bert-base, RoBERTa, and DistilBERT.

Table 3: Parameters of BERT Models

Models	**Parameters**
Bert-base	340M
RoBERTa	355M
DistilBERT	66M

In anticipation of the model being very resource-hungry and computationally expensive, we have opted for the DistilBERT model with the least amount of parameters (66 million). Instead of feeding our model with our pre-processed data, we wanted to see how well the model would do on its pre-trained knowledge if we were to input our raw dataset. After inputting in our raw dataset, we wanted to explore the text length of the reviews in our dataset. Our findings indicated that the 95th percentile of the review length is 1398 words which led us to specify the max_length of the DistilBERT tokenizer to that value. When we started training our model with a max_length value of 1398 we quickly encountered an issue where we ran out of memory almost instantly. We tried to use the max_length value that would encompass the 90th, 80th, 70th percentile of the review length but those values were still too large, causing us to face the same issue.

A solution we came up with was to create a stratified subset of our dataset. Our entire dataset consists of 211,748 reviews. Our subset is 20% of the entire dataset consisting of 42,349 reviews. After reducing the amount of reviews in our dataset we repeated the same process of setting the max_length value in the DistilBERT tokenizer to that of the 90th, 80th, and 70th percentile of the review length. Unfortunately, we faced the same issue and quickly ran out of memory. Ultimately, we settled on a max_length value of 512 which was large enough to where we could capture the reviews and not run into an out-of-memory error.

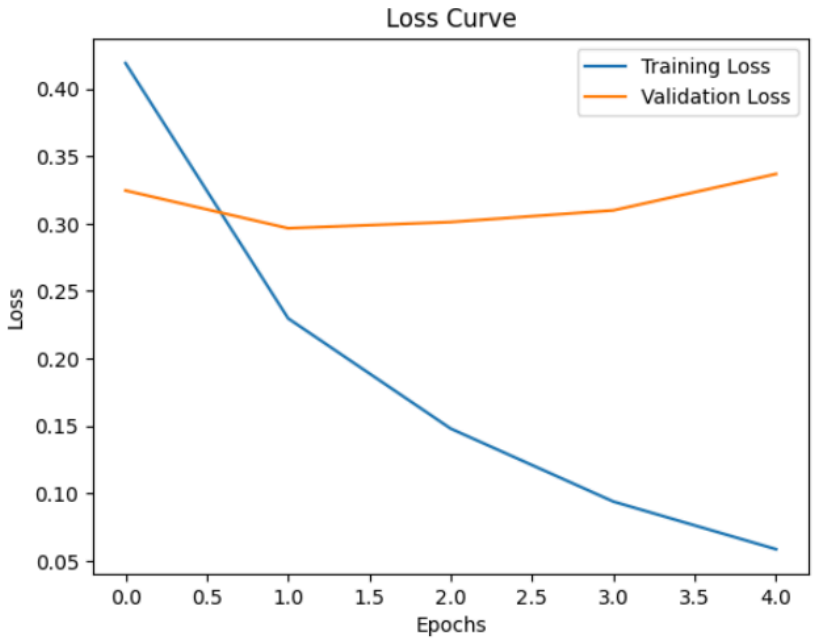

Figure 10: DistilBERT Loss Curve

The performance of the DistilBERT was shocking in ways that we were not expecting. We expected the model to perform much better since it was pre-trained on a large corpus of text data. However, seeing how we had to compromise on the max_length due to our lack of resources it is also no surprise that our model was overfitting and performed poorly on the validation result. We

were limited to using about 30% of the review length to try and predict the star ratings from the review.

7. Predicting True Star Rating of Review

Finally, to test our models and find the true star ratings of yelp reviews, we made predictions of the star rating based on the review text, which were converted to sequences. First, we tested if our model would work by testing our own review and seeing what the model would give us. The review we tested was, “The food was so so good but I was sick and I will not go back”. This review was predicted to have a 2-star rating by the model, reflecting its ability to grasp the nuanced sentiment in the sequence of words. The positive remark about the food is offset by the negative sentiment about the overall experience, which indicates the model’s capacity to understand and weigh conflicting sentiments in the text. The model was then applied to real review data from the dataset, and our model was able to predict the star rating pretty well. For example, some users gave a 3 star rating, while the model predicted 2 stars. Another user gave a 4 star rating, while our model predicted the review to reflect a 5 star rating. This aligns with the results of the models as stated above, where the typical prediction of our model is about 0.63 stars away from the true rating given.

The performance indicates that the model can effectively process and analyze the textual content of reviews, identifying key sentiment indicators and translating them into star ratings. This ability is especially valuable for quickly assessing the sentiment behind large volumes of customer reviews, which would be impractical to analyze manually. By predicting the star ratings with a reasonable degree of accuracy, the model provides an automated tool that can assist businesses in understanding customer satisfaction and pinpointing areas of improvement. This insight is vital for businesses aiming to enhance customer experience, as it helps in identifying common praise or complaints without sifting through each review manually.

To further enhance the model's accuracy, the training data can be refined and the model parameters could be adjusted. This iterative improvement process will help the model better handle varied expressions and nuances in customer feedback, aiming for even closer alignment with actual user ratings. Ultimately, the goal is to create a robust model that can provide consistent, accurate predictions of star ratings, contributing to more effective sentiment analysis and customer insights.

8. Conclusion

Our project successfully developed a model for predicting the star ratings of Yelp reviews based on the text review content. By transforming review text into sequences and applying a trained machine learning model, we were able to predict star ratings with a reasonable degree of accuracy. This method demonstrated its utility by aligning predicted ratings closely with actual

user ratings in most cases, providing a valuable tool for automating sentiment analysis in large datasets. Through the use of various models and techniques, including Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs), along with methods like early stopping, dropout layers, and L2 regularization, we refined our approach to analyzing Yelp reviews. The LSTM model, enhanced with L2 regularization, was particularly effective. Its performance on the validation dataset showed that the mean squared error (MSE) was significantly reduced, indicating that the typical prediction is approximately 0.63 stars away from the actual rating. This level of accuracy demonstrates the model's effectiveness in handling and analyzing the nuances of textual data in Yelp reviews.

Some advantages of our project are that our model is able to offer an efficient way to gauge customer sentiment from textual reviews, which is highly valuable for both businesses and customers. By predicting star ratings, the model provides actionable insights into customer satisfaction and areas of approvement. Furthermore, our model was able to capture the context and nuances within reviews, allowing it to balance mixed sentiments and produce a coherent rating. Some disadvantages of our project could include that the accuracy of predictions heavily depends on the quality and representativeness of the training data (i.e. what words are considered positive and negative in the training data). Thus, biases in the training data can affect model performance. Furthermore, reviews with complex or ambiguous sentiments may pose challenges, such as sarcastic reviews that could only be discerned through context.

This leads into the limitations of our project, where the model may not fully capture the entire range of emotions or the intensity of sentiments expressed in reviews, as it relies on text alone without additional context. Furthermore, variations in language, slang, and writing styles across different reviews can impact the model’s performance. As mentioned earlier, because most of our data is from Santa Barbara, the model’s performance might be affected by Santa Barbara slang and customs, resulting in the model not being as generalizable to other regions. Some personal limitations we encountered while doing the project were that the dataset was too large even after trimming it down significantly, so the model training time was very extensive and thus made it difficult to test different batch sizes, epochs, and hyperparameters. The extensive training time restricted the ability to perform comprehensive testing and validation of different model architectures or approaches. Furthermore, we were unable to upload our dataset to github (including the preprocessed and trimmed down version) due to its size, so we were unable to upload our data using a github link as stated in the project outline.

In conclusion, our model provides a valuable tool for understanding and analyzing customer feedback through automated star rating predictions. While there are inherent challenges and limitations in handling diverse and complex sentiments, the advantages it offers in scalability and efficiency make it a powerful asset for businesses looking to enhance their customer experience strategies.

9. Contributions Statement

Throughout the project, all three of us worked collaboratively to come up with ideas and constructive criticism of the models and overall direction of the project. We frequently had project work sessions, either on Zoom or in person, to go over the project and work on it together.

Cynthia focused mainly on the EDA, feature selection, and sentiment analysis. Meiyi and Nam focused mainly on the data processing and utilizing our model to predict ratings from our dataset. However, all three of us collaborated well on everything together (ie answering questions, providing feedback, and helping with implementing the code). All three of us worked together to build the models and gave each other feedback on how to potentially improve its performance.

For more information please visit our GitHub repository:
https://github.com/NamTTruong/PIC-16B-Project