UNIVERSITY OF CALIFORNIA
Los Angeles

Prediction of Yelp Score from Reviews with Machine Learning Model

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Applied Statistics

by

Yuandong (David) Sun

2022

ABSTRACT OF THE THESIS

Prediction of Yelp Score from Reviews
with Machine Learning Model

by

Yuandong (David) Sun
Master of Applied Statistics
University of California, Los Angeles, 2022
Professor Yingnian Wu, Chair

Yelp has been a dominating company in the business rating industry. In the current days, reviews and ratings critically affect people's decisions. A model that effectively predicts the ratings could be supportive to businesses. In this research, we develop several models based on various NLP and regression methods based on the text review and other traits of the reviews to predict the final star rating of each. Models including LSTM, GRU, BERT, and multinomial logistic regression are utilized. The fine-tuning BERT model has the best result of 68.8% on the testing dataset. Although the model predicts well in general, we could further investigate and evaluate text reviews on multiple aspects.

The thesis of Yuandong (David) Sun is approved.

Guani Wu

Michael Tsiang

Guang Cheng

Yingnian Wu, Committee Chair

University of California, Los Angeles

2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

In the early 21st Century, people used printed maps, advertising newsletters, billboards, and touring guides to acquire knowledge of new opening businesses and local attractions. It took much effort for an individual to develop and consume the information from these sources. With world wide web development, information was transformed on the Internet. Nowadays, people can easily search and browse maps, the local businesses, and travel guides online within seconds. The Internet also promotes more interactions among people, and people can share their thoughts and knowledge with a broader group. Under these circumstances, a new industry was invented: the local business search, rating, and advertising industry. As of 2020, 63.6% of the consuming public is likely to check online reviews before visiting, and 90% of respondents claimed that positive online reviews influenced their buying decisions [Ant22]. Many companies are involved in this industry, such as Google Reviews, Yelp, Trustpilot, Yellow Pages, TripAdvisor, and Foursquare.

Yelp was launched as a local business rating and review website in 2004, and it quickly gained popularity and dominated the local review business industry within the last two decades. Besides advertising, Yelp keeps offering new products such as smartphone App, Yelp Waiting List, Yelp Connect, Online food delivery, and many more. As of December 31, 2020, Yelp has over 224 million cumulative reviews, over 31 million App unique devices, and covers over 219 major cities. In addition, Yelp users are more active social media users according to statistics [And22].

As previously mentioned, the importance of reviews and scores could significantly affect

consumers' decisions, and Yelp is one of the dominating companies within the local business review industry. Thus, we are interested in creating a model based on the text reviews to predict the business rating score. Within the research, we used the 'Yelp Open Dataset' for the creation of the evaluation model [Inc]. We created the model based on NLP methods known as LSTM, GRU, and BERT, along with multinomial logistic regression.

This article is structured as follows: Chapter 2 provides the methodology; Chapter 3 offers a thorough description of data wrangling and exploratory analysis; Chapter 4 shows the process of the model creation; finally, the conclusion and discussion are given in Chapter 5.

CHAPTER 2

Methodology

2.1 LSTM Networks

LSTM networks stand for Long Short-Term Memory networks. It is a subset of Recurrent Neural Network (RNN). LSTM solves RNN's problem that RNN cannot learn long-term dependency in the data by having memory cells and gates that control the information flow and the memory cells [Leeb]. Figure 2.1 shows the framework of LSTM.

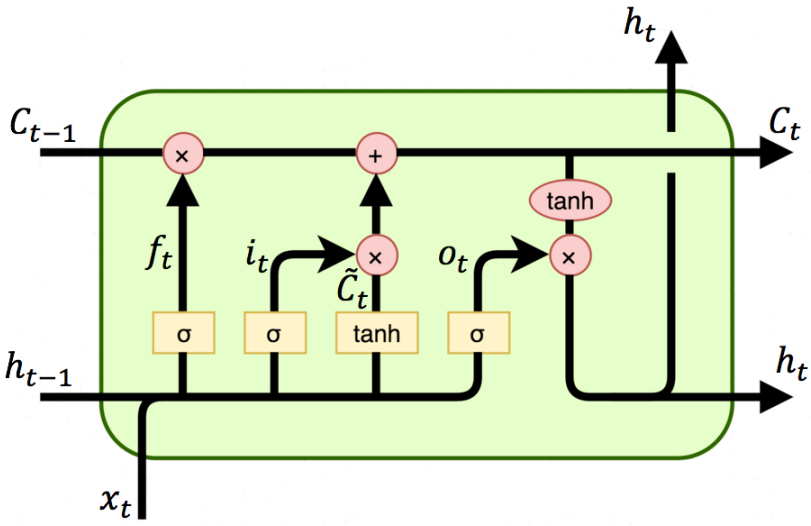


Figure 2.1: The framework for LSTM [Leeb]

The contents of the memory cells C_t are regulated by various gates:

- Forget gate f_t

- Input gate i_t
- Output gate O_t

The followings are the corresponding formulas.

$$
\begin{aligned}
f_t &= \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \\
\tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \\
i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\
C_t &= f_t * C_{t-1} + i_t v \tilde{C}_t \\
o_t &= \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \\
h_t &= o_t \otimes \tanh\left(C_t\right)
\end{aligned}
$$

W_n is the weight matrices of the input and recurrent connections for the input gate, the output gate, the forget gate, or the cell state. b_n is the bias terms for the same components. $\otimes$ is the element-wise product of two vectors. σ is a sigmoid function [FBW19].

2.2 GRU

GRU stands for Gated Recurrent Unit, which is a simplified LSTM by merging forget (f_t) and input gate (i_t) into update gate (z_t). GRU is another method that solves RNN's long-term dependency problem. The update Gate controls the forgetting factor and the decision to update the state unit. Figure 2.2 displays the framework of GRU.

The corresponding formulas are shown in the following.

$$
\begin{aligned}
z_t &= \sigma\left(W_z \cdot [h_{t-1}, x_t] + b_z\right) \\
r_t &= \sigma\left(W_r \cdot [h_{t-1}, x_t] + b_r\right) \\
\tilde{h_t} &= \tanh\left(W \cdot [r_t \otimes h_{t-1}, x_t] + b\right) \\
h_t &= (1 - z_t) \otimes h_{t-1} + z_t \otimes \tilde{h_t}
\end{aligned}
$$

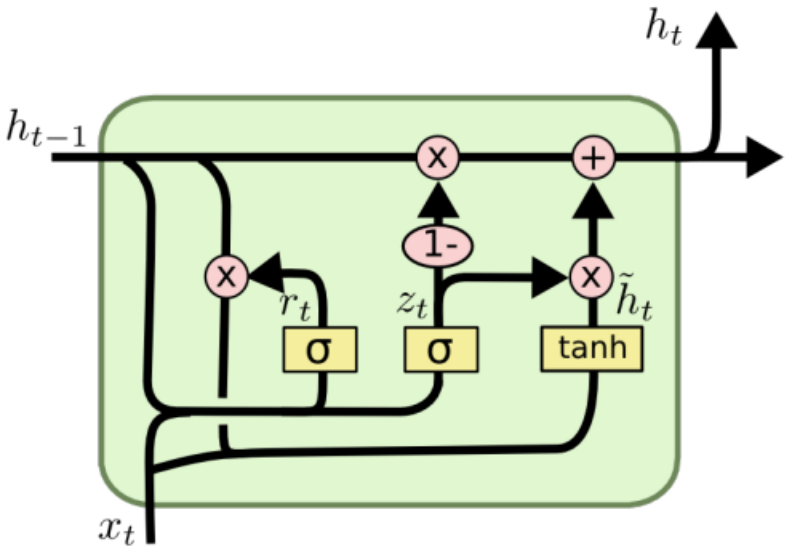


Figure 2.2: The framework for GRU [Leeb]

W_n is the weight matrices of the input and recurrent connections for the update gate, the output gate, or the cell state. b_n is the bias terms for the same components. $\otimes$ is the element-wise product of two vectors. σ is a sigmoid function [FBW19].

2.3 BERT

Language model training plays a vital role in natural language processing tasks. Currently, pre-trained representations are one-way language models. In other words, each word can only be trained using the information that precedes it. This constraint severely limits the expression ability of pre-trained representations. The pre-trained representations will result in a locally optimal solution for the sentence-level natural language task. For token-level natural language tasks, it would be a devastating disaster. To solve the above problems, BERT was presented by the researchers at Google [DCL19]. It is a MASK model, which puts a random mask into a sequence of words. The goal is to predict the masked word using contextual information. It can also be used in a Transformer with a bidirectional encoder.

This resolves the "language model is one-way" limitation. In addition, the BERT model introduces the task of "predicting whether it will be the next sentence" to learn pre-trained representations together. Hence, the goal of BERT Pre-Trained Remedies is to predict both what the masked word will be and whether the sentence pair will be the "next sentence."

BERT is an unsupervised pre-training model for NLP. The BERT model is a multi-tier Transformer. Each block is mainly composed of multi-header self-attention, normalization (Norm), residual connection, and Feed Forward. It is divided into two stages: model pre-training and model fine-tuning. In the pre-training stage of the model, the vast model parameters are usually on the order of tens of millions or even hundreds of millions. Hence a lot of data training is needed. Fortunately, the model is unsupervised at this time, and it only needs to crawl or use open-source data sets. In the fine-tuning stage of the model, it is necessary to fine-tune the model for specific tasks, which has achieved good results.

Figure 2.3 gives a pipeline of The BERT model. Clearly, the BERT adopted the double self-attention, which can federate the contextual information in all layers.

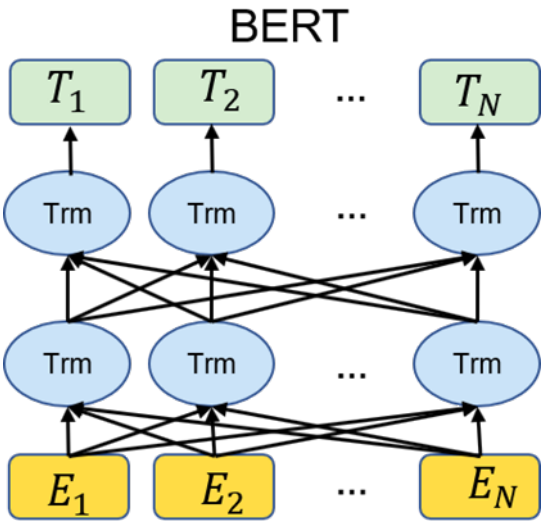


Figure 2.3: The framework for BERT [DCL19]

2.3.1 Model Structure

As shown in Figure 2.4, the encoding part includes input, add location encoding, and three blocks, i.e., Self-Attention, Add & Norm, Feed Forward.

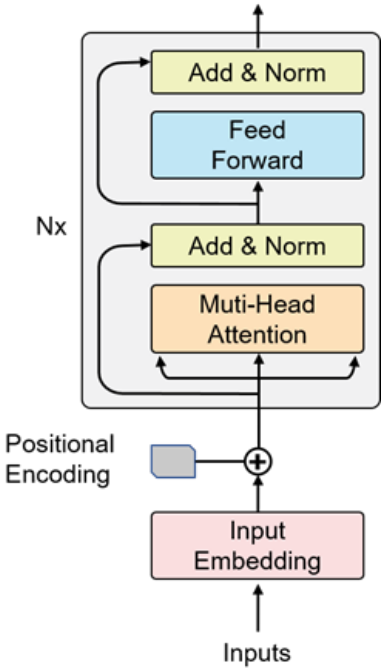


Figure 2.4: BERT Architecture [Fac]

2.3.2 Position Encoding

Position encoding is used to capture temporal associations between texts. The highest correlation is the closest location. At the same time, when NLP text is processed, text closer to the location is generally more relevant, so it is necessary to incorporate location encoding into the data. Specifically, the Embedding+Positional approach was adopted to integrate the relevance of the data. Embedding is textual data embedded into the corresponding dimension. Linear transformation of sine and cosine functions was used in the thesis to provide Positional information of the model. In the structure of self-attention, the data weight of each dimension is calculated in the form of a dot product, which is essential to

calculate the correlation between vectors. The location coding increases the correlation of adjacent data by adding the location coding of close frequency to adjacent data.

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

2.3.3 Self-Attention

Self-attention is an essential step of BERT, which combined with location encoding. It solves the problem of temporal correlation of text data, thus ending at one full network model, which has been used to solve temporal problems by relying on RNN, LSTM, GRU, etc [VSP17]. Self-attention is generally said to be a way to calculate weights dynamically when information is disseminated forward. The attention model is a structure that automatically adjusts weights when different information is transmitted after training. The specific structure of self-attention is shown in Figure 2.5.

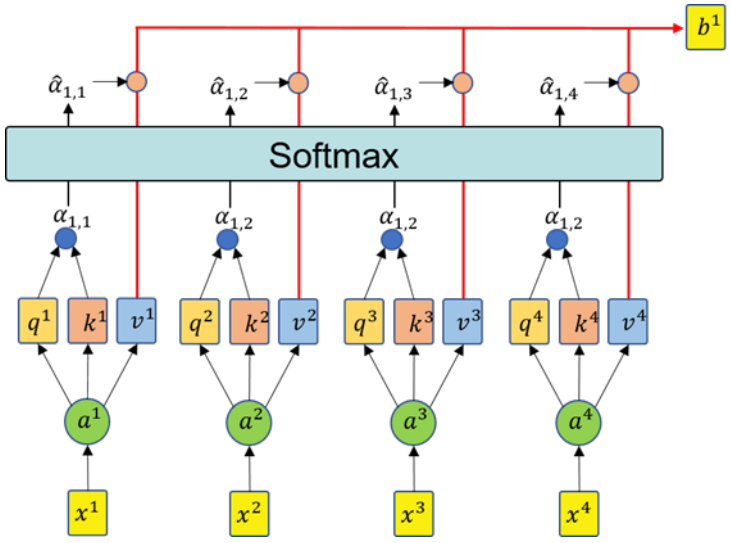


Figure 2.5: Self-Attention Mechanism [Leea]

where x^1, x^2, x^3, x^4 represents four sequential text messages via embedding. a^1, a^2, a^3, a^4 are obtained by adding four pieces of information and the position vector together so that the temporal correlation of text is modeled. And then assign three weights to each piece of

information, then we have:

$$q^i = a^i * w_q$$

$$k^i = a^i * w_k$$

$$v^i = a^i * w_v$$

Make q^1 dot product with $k^1, k^2, ..., k^i$ and we obtain which are $\alpha_{1,i}$. After softmax, $\hat{\alpha}_{1,i}$,

$$\alpha_{1,i} = q^1 * k^i$$
$$\hat{\alpha}_{1,i} = \frac{\exp(\alpha_{1,i})}{\sum_j \exp(\alpha_{1,j})}$$

Use the output of softmax to weigh v_i and obtain b^1. Similarly, we can also obtain $b^2; b^3; ...; b^i$ and finish the self-attention process.

$$b^k = \sum_i \alpha_{k,i} * v^i$$

2.4 Multinomial Logistic Regression

Multinomial logistic regression is a classification method used to predict more than two categorical unordered levels. It is similar to binary logistic regression. The method breaks the outcome variable down into a series of comparisons between two categories [HCS21].

$$\log(odds) = \text{logit}(P) = \ln\left(\frac{P}{1-P}\right) = a + b_1x_1 + b_2x_2 + b_3x_3 + ...$$

P = the probability that a case is in particular category,
exp = the exponential (approx 2.72),
a = the constant of the equation and,
b = the coefficient of the predictor or independent variables.

CHAPTER 3

Data

3.1 Data Source

The data originated from "Yelp Open Dataset." This dataset is a subset of Yelp businesses, reviews, and user data. It contains 8.6 million reviews, 160k businesses, 200k pictures, and eight metropolitan areas [Inc].

The JSON file "yelp_academic_dataset_review" is used for this research. The data contains the following columns: "review_id", "user_id", "business_id", "stars", "useful", "funny", "cool", "text", and "date". The following shows the name of the variables along with a brief description.

"**review_id**" is in string format, represents the unique ID that assigned by Yelp for each record of review.

"**user_id**" is in string format, represents the unique ID that assigned by Yelp for each user.

"**business_id**" is in string format, represents the unique ID that assigned by Yelp for each unique business location.

"**stars**" is in float format, represents the star rating score, ranging from 1 to 5.

"**useful**" is in float format, represents the number of useful votes received.

"**funny**" is in float format, represents the number of funny votes received.

"**cool**" is in float format, represents the number of cool votes received.

"**text**" is in string format, represents the text review .

"**date**" is in datetime format, represents the timestamp of corresponding review's publication.

3.2 Data Cleaning

The entire file is over 6 GB. Due to the system and environment limitations, we select the first three million records as our data. We notice that the result is in float format for the stars column. We convert the format from float to integer and add another column called "stars_str" by converting to string, such as from 3.0 to '3'.

3.2.1 Stars

As shown in Figure 3.1, the distribution of the stars rating is not equal. Ranking 5 Star is almost half of our data (43.9%). Rating 4 Star is 22.5%.

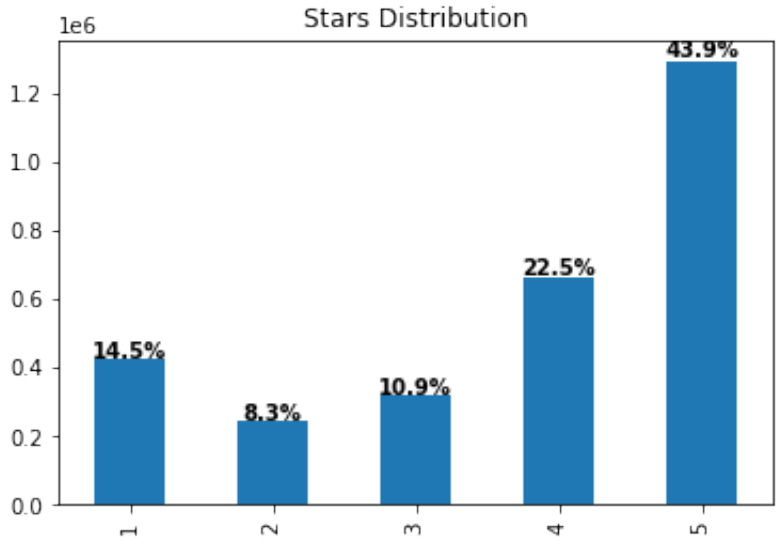


Figure 3.1: Stars Distribution

Several reasons could cause this situation:

- There are fake reviews that the business bought from third-party marketing agencies [DO13]. The reason is to create a high rating business reputation, and a large number of high rating reviews attract more customers because of the bandwagon effect.
- Business does in-store rating promotions and commercial solicitations such as offering free food, service, or discounts after customers review with a high star rating.
- Business pays reviewers to remove the low star reviews.

Regardless of the reasons, we need to resample our data to equal sample size for each star rating to have a sufficient model.

3.2.2 Null, English & Duplication

None of the records contain null values. We detect there are foreign languages in the text reviews, such as "第一家中式麻辣烫非常干净明亮食物选择很 very clean and bright..." and "Los mejores sándwich de brisket que eh probado...". For the accuracy purpose of the models, we decide to remove the non-English reviews. We also find that there are duplicated text reviews, which could be caused by general reviews such as "This is a good place." or by reviewers' incorrect usage of the app, such as submitting the same reviews multiple times.

3.2.3 Outliers

3.2.3.1 Useful

The range of useful votes is from 0 to 446. The unique values are almost continuous from 0 to 170 with the proportion 0.896 of all unique values, and we get scattered results from 172 to 446. As shown in Figure 3.2, most of the votes are clustered within 20 votes. We need to clear the outliers, and therefore we would only keep votes above 0.1% frequency proportion.

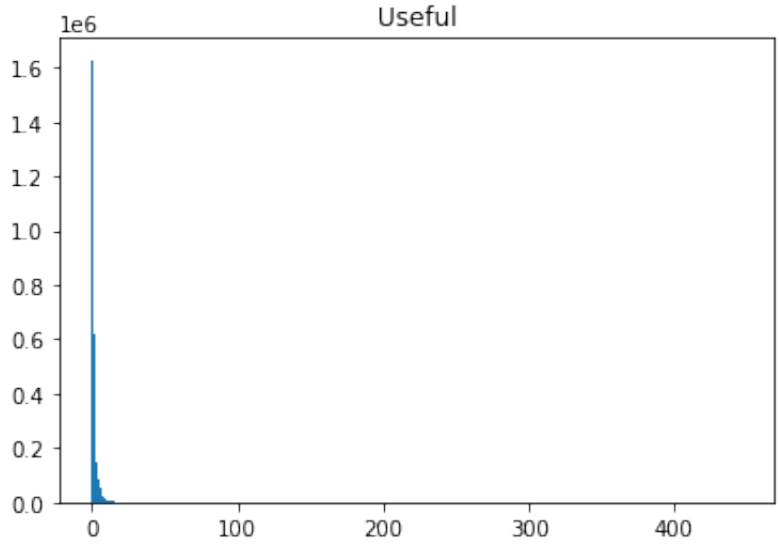


Figure 3.2: Useful Distribution

3.2.3.2 Funny

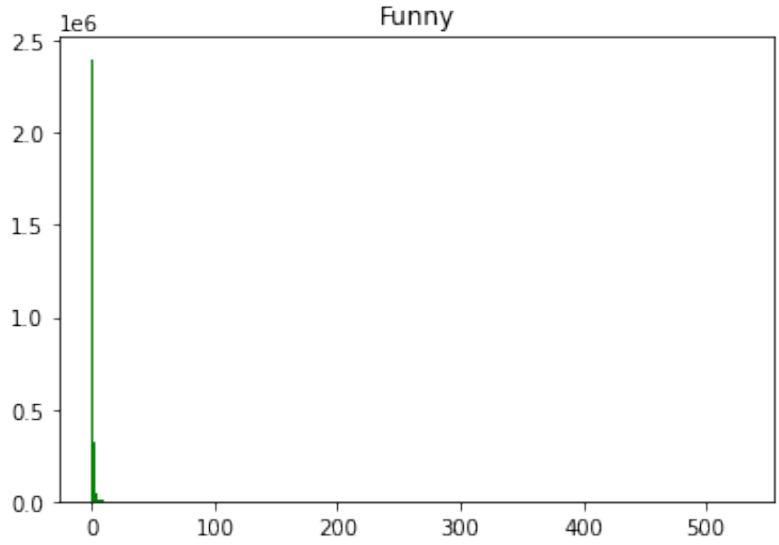


Figure 3.3: Funny Distribution

The range of funny votes is from 0 to 529. The unique values are almost continuous from 0 to 110 with the proportion 0.85 of all unique values, and we get scattered results from 113

to 529. As shown in Figure 3.3, most of funny votes are clustered within 10 votes. Again, we would remove records that are less than 0.1% frequency proportion.

3.2.3.3 Cool

The range of cool votes is from 0 to 530. The unique values are almost continuous from 0 to 157 with the proportion 0.925 of all unique values, and we get scattered results from 160 to 530. Similar to Funny, Figure 3.4 shows more than 99% of cool votes are clustered within 10 votes. We eliminate records that are less than 0.1% frequency proportion.

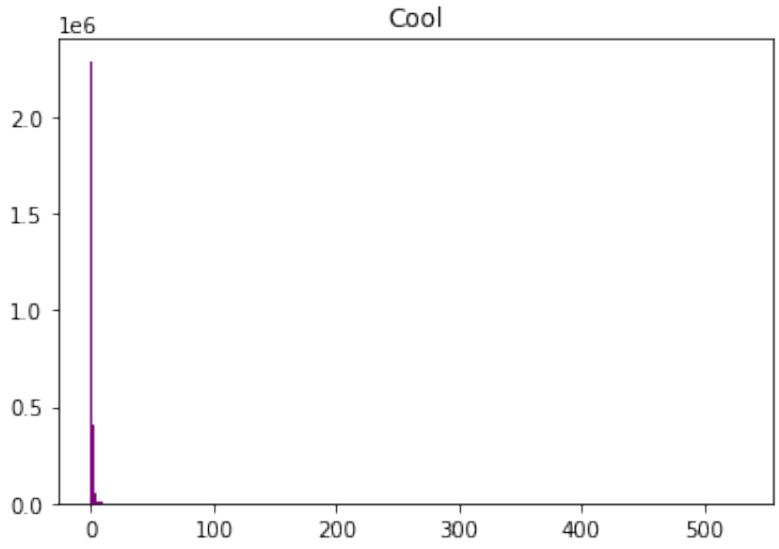


Figure 3.4: Cool Distribution

3.2.3.4 Text

For better results on models, we clean the text reviews by removing punctuation, changing to all lower case, and stemming sentences. After examining some low character count records, we decide to keep records with more than 50 characters. The reason is to eliminate the data to contain reviews simply flaming or praising the businesses without detailed and helpful content. For example, comments like “This place is perfect” and “This place sucks” would

not be counted in the sample selection. This is a simple way of selecting samples, and further discussion will be given in Chapter 5 Conclusion & Further Discussion.

3.3 Resample

As previously mentioned, it is necessary to clean and resample the data with an equal amount of records for each star, removing outliers for each variable as discussed above. Due to the system limitation and training time, 70,000 records are used for each star rating, which is 350,000 records in total. The resample data is randomly chosen by the "sample()" function with the random state equal to 200 for the reproductive purpose.

3.4 Exploratory Data Analysis

3.4.1 Useful

The frequency table (Table 3.1) contains useful votes that are more than 0.1% frequency proportion. From 0 to 18 votes cover 99.7% of overall useful votes. We want to highlight that 55.4% of the reviews have 0 useful vote, 21.1% have 1 useful vote, and 9.7% have 2 useful votes. The rest of 13.8% have 3 or more useful votes.

Table 3.1: Useful Frequency Table Proportion Percentage more than 0.1%

useful	count	freq %
0	1,628,090	55.4
1	620,731	21.1
2	284,491	9.7
3	146,120	5
4	82,188	2.8
5	49,704	1.7
6	31,775	1.1
7	21,395	0.7
8	15,371	0.5
9	11,250	0.4
10	8,337	0.3
11	6,381	0.2
12	5,042	0.2
13	4,214	0.1
14	3,357	0.1
15	2,732	0.1
16	2,275	0.1
17	1,924	0.1
18	1,511	0.1

3.4.2 Funny

The frequency table (Table 3.2) contains funny votes that are more than 0.1%. From 0 to 9 funny votes, in sum, cover 99.5% of overall funny votes. We want to highlight that 81.8% of the reviews have 0 funny vote, and 11.0% have 1 funny vote. The rest of 7.2% have 2 or more funny votes.

Table 3.2: Funny Frequency Table Proportion Percentage more than 0.1%

funny	**count**	**freq %**
0	2,400,963	81.8
1	324,532	11.0
2	100,291	3.4
3	42,639	1.4
4	22,351	0.8
5	13,046	0.4
6	8,386	0.3
7	5,723	0.2
8	4,207	0.1
9	3,052	0.1

3.4.3 Cool

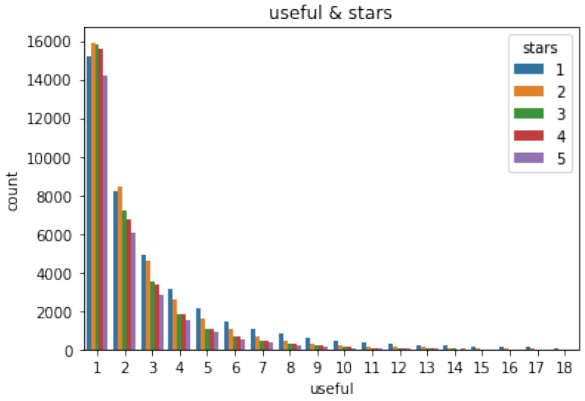
A frequency table (Table 3.3) is created for cool votes that are more than 0.1%. From 0 to 9 cool votes cover 99.5% of overall cool votes. We want to highlight that 77.9% of the reviews have 0 cool vote, and 13.8% have 1 cool vote. The rest of 8.3% have 2 or more cool votes.

Table 3.3: Cool Frequency Table Proportion Percentage more than 0.1%

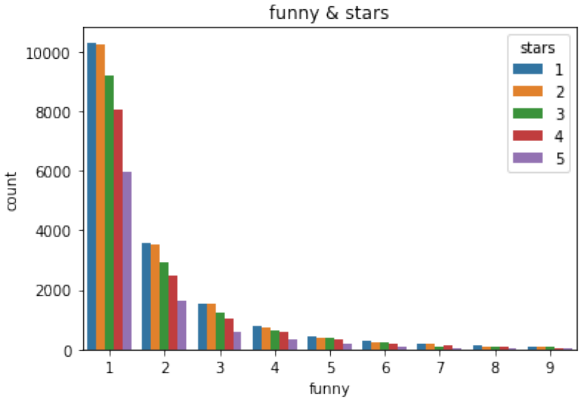
cool	**count**	**freq %**
0	2,287,227	77.9
1	407,113	13.8
2	122,024	4.1
3	48,279	1.6
4	23,699	0.8
5	13,346	0.5
6	8,521	0.3
7	5,895	0.2
8	4,127	0.1
9	3,082	0.1

3.4.4 Stars

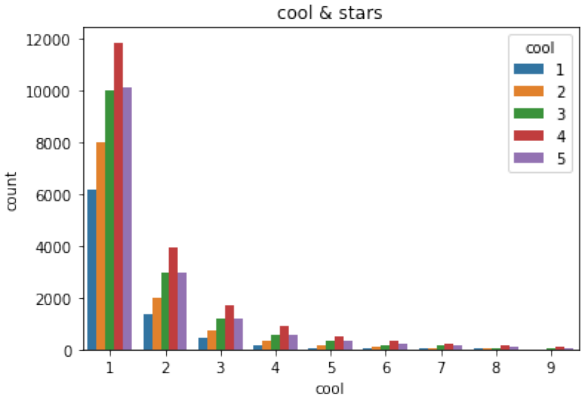
After resampled the data, we have an equal amount of records and a uniform distribution for each star. This would help us reduce biased results. Figure 3.5 shows the relationship between useful, funny, and cool with stars. Useful votes are equally distributed among stars. There are relatively more funny votes in lower stars. Lastly, there are more cool votes in star rating 4 comparing to other stars.



(a) Useful & Stars



(b) Funny & Stars



(c) Cool & Stars

Figure 3.5: Useful, Funny, Cool & Stars

3.4.5 Text

A new column "txt_len" is created to store the reviewed text character length. Table 3.4 shows the description of "txt_len". As previously mention ed, the minimum text length is set as 50. Half of the records (between 25% and 75% quantile) are in the 271 to 847 character length.

Table 3.4: Description of "txt_len"

count	250,000
mean	659.43
std	588.5
min	50
25%	271
50%	484
75%	847
max	5,000

3.4.6 BERT Sentimental Analysis

The reviews show different sentimental results. For example, some are positive for 5 stars, and some are negative for 1 star. The Hugging Face Transformer library has a "pipeline" function that analyzes the sentiment of the given sentences based on the pre-trained model "distilbert-base-uncased-finetuned-sst-2-english" [Fac]. The function's output contains a label: positive, negative, or neutral, along with a score out of 1, which demonstrates how confident the model is toward the output label.

Three columns are created for the output. "sent_label" stores the label: -1 for negative, 0 for neutral, and 1 for positive. Surprisingly, the output does not have any "neutral" labels for all the reviews. "sent_score" stores the score from the output. "sent_res" stores the result of "sent_label" times "sent_score".

CHAPTER 4

Models

4.1 Training, Validation & Testing Datasets

The training, validation, and testing datasets are divided into 7/1/2 proportions. Namely, we have 245,000 records in the training dataset, 35,000 records in the validation dataset, and 70,000 records in the testing dataset. The random state is set to 200 for reproductive purposes.

4.2 LSTM Model

The LSTM model uses "text", the text review, to predict star labels. The learning rate is initially set as the default, 0.001 (1e-3), with an early stop: delta rate at 0.0001 (1e-4) and 3 patient epochs. The validation split is set to 0.1. The batch size is set to 128. Fifty epochs are assigned, and eighteen epochs are performed. The diagram of the model is shown in Figure 4.1.

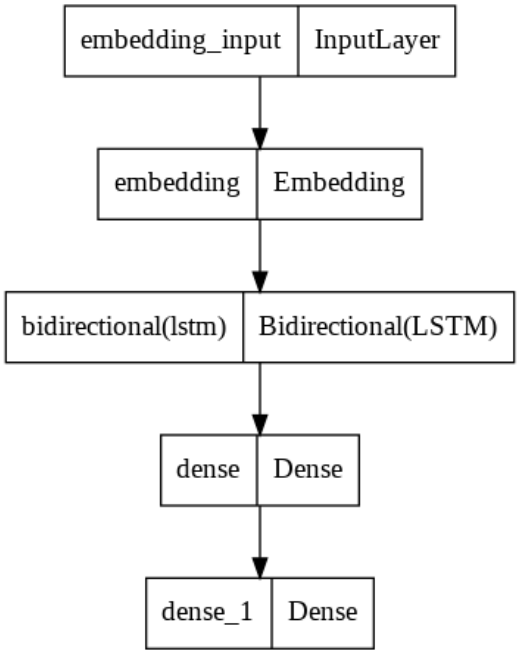
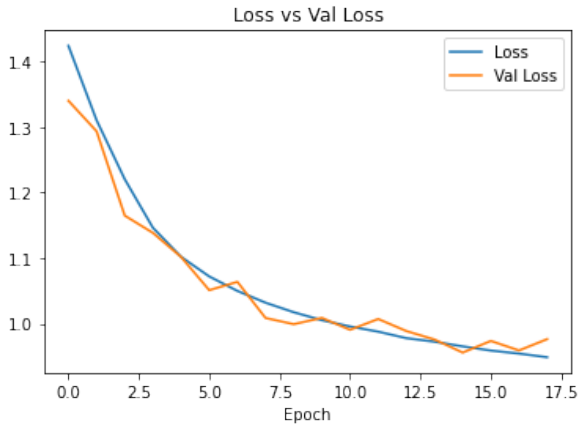
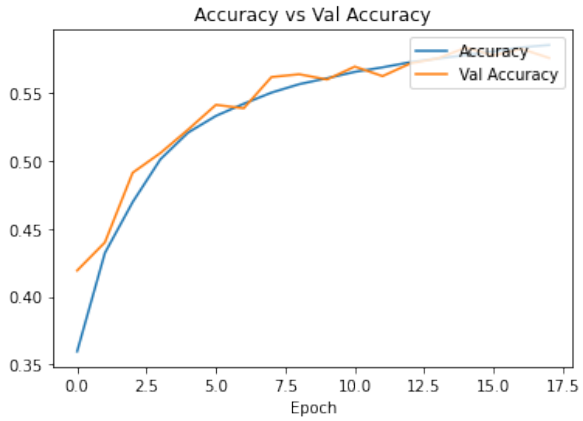


Figure 4.1: LSTM Model Structure

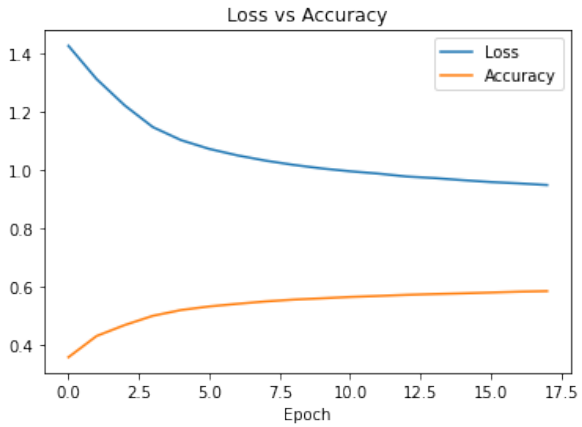
At the last epoch (18), the training and validation losses are 0.9485 and 0.9762, respectively. The training and validation dataset accuracies are 58.56% and 57.60%, respectively. The loss on the testing dataset is 0.979, and the accuracy is 57.2%. Plots are shown in Figure 4.2.



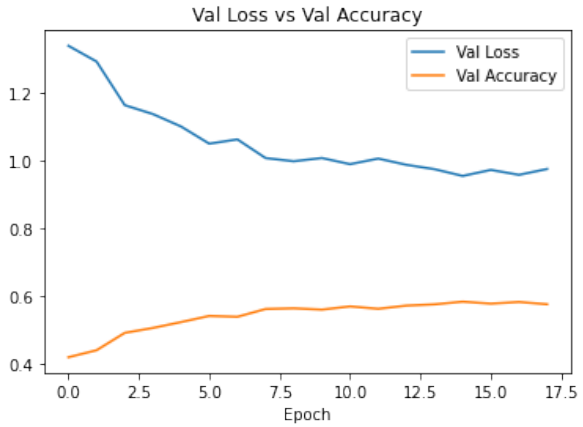
(a) Training and Validation Loss



(b) Training and Validation Accuracy



(c) Training Loss and Accuracy



(d) Validation Loss and Accuracy

Figure 4.2: LSTM model performance plots

4.3 GRU Model

The GRU model, similar to the LSTM model, uses business review to predict star labels. The parameters are the same as the LSTM model. Fifty epochs are assigned, and thirty-eight epochs are performed. The diagram of the model structure is shown in Figure 4.3.

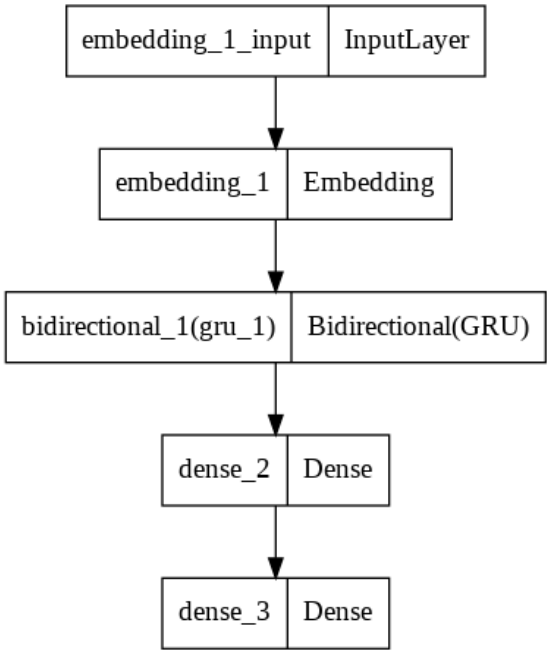
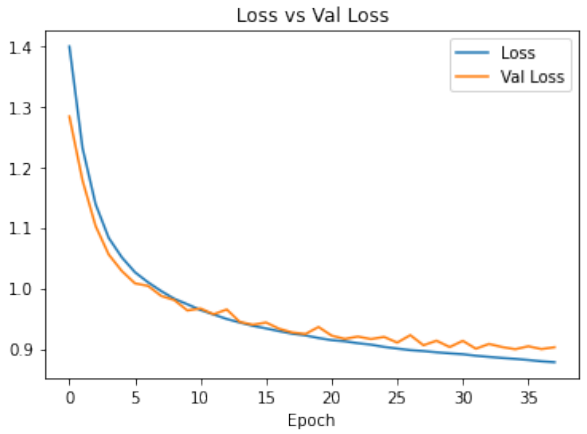
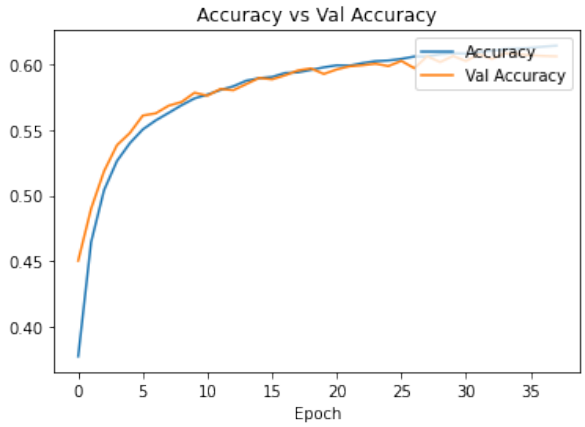


Figure 4.3: GRU Model Structure

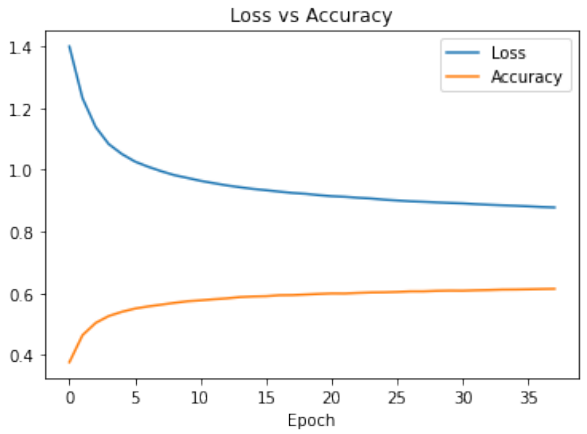
At the last epoch (38), The training and validation dataset losses are 0.8778 and 0.9025, respectively. The training and validation dataset accuracies are 61.45% and 60.64%, respectively. The loss on the testing dataset is 0.903, and the accuracy is 60.4%. Plots are shown in Figure 4.4.



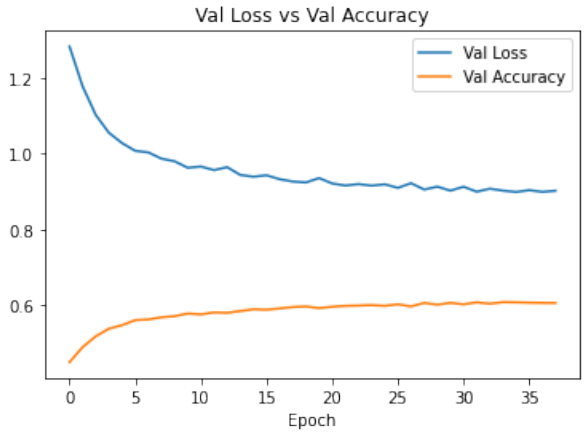
(a) Training and Validation Loss



(b) Training and Validation Accuracy



(c) Training Loss and Accuracy



(d) Validation Loss and Accuracy

Figure 4.4: GRU model performance plots

4.4 BERT Model

In this model, the fine-tuning feature of the BERT model is utilized. The model again uses the business review to predict star labels. The learning rate is initially set as 0.00001 (1e-5). Sixteen epochs are performed. The pre-trained model is based on "bert-base-cased". Three epochs are performed. We do not use many epochs like the earlier models because BERT

is already a well-trained model instead of a train from scratch; we are just fine-tuning the model in this case. The diagram of the model structure is shown in Figure 4.5.

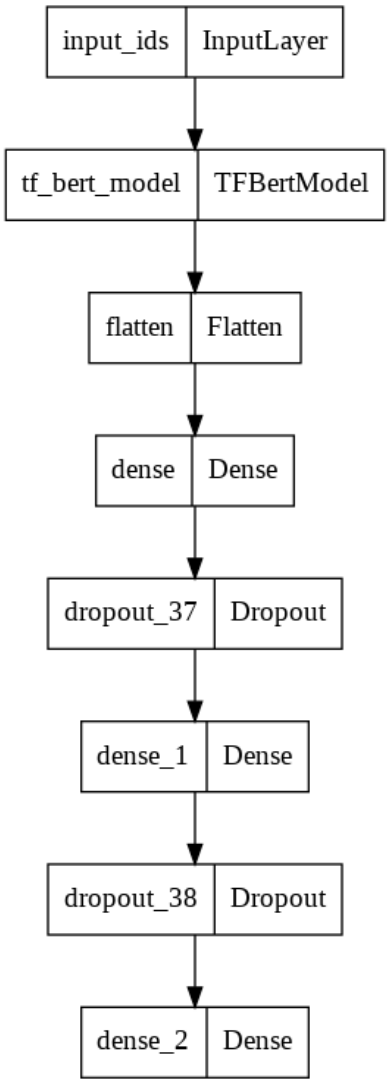
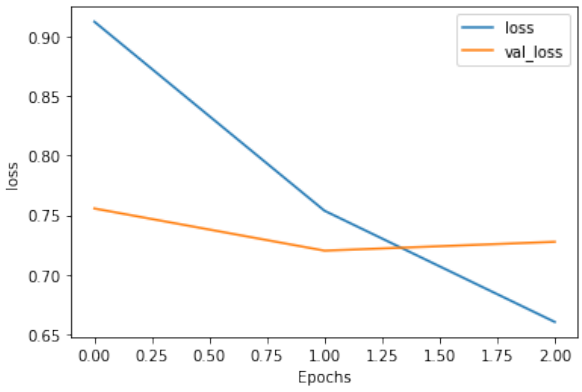
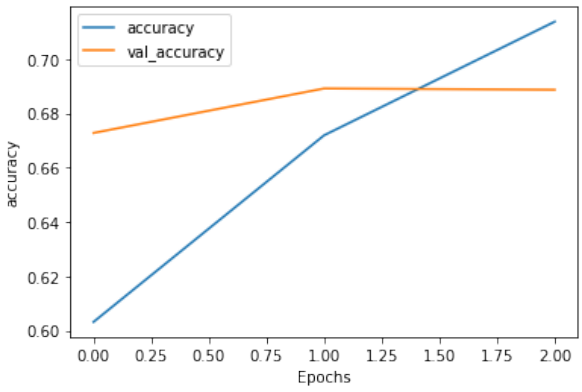


Figure 4.5: BERT Model Structure

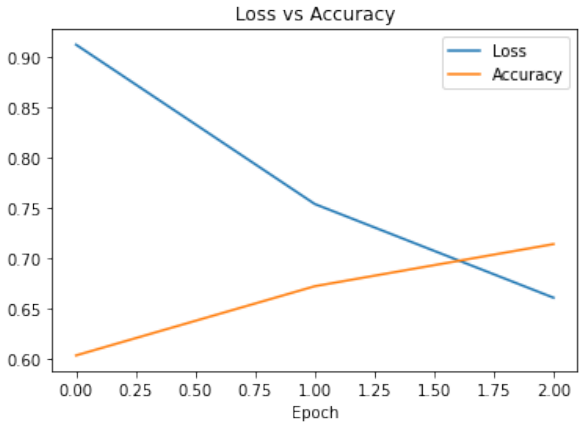
At the last epoch(3), the training and validation datasets losses are 0.6605 and 0.7276, respectively. The training and validation dataset accuracies are 71.38% and 68.87%, respectively. The loss on the testing dataset is 0.7277, and the accuracy is 68.57%. Plots are shown in Figure 4.6.



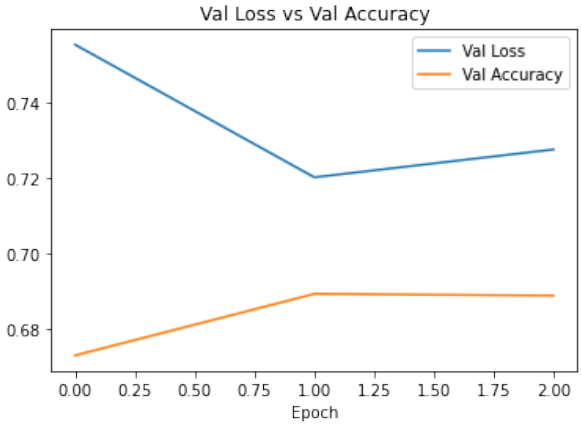
(a) Training and Validation Loss



(b) Training and Validation Accuracy



(c) Training Loss and Accuracy



(d) Validation Loss and Accuracy

Figure 4.6: BERT model performance plots

4.5 Multinomial Logistic Regression Model

For the multinomial logistic regression model, the first step is to check the assumption of non-multicollinearity; thus, Variance Inflation Factor (VIF) is calculated for each variable. The VIF is the kth diagonal element of the inverse of the correlation matrix of the variables [FLH07]. The formula is given as follows:

$$\mathrm{VIF}_k = \left(1 - \mathrm{R}_k^2\right)^{-1}$$

where R_k^2 is the multiple coefficient of determination of x_k regressed on all the other regressors.

Table 4.1 exhibits the VIF results. Each is about 1 and 2, so multicollinearity condition does not exist. Variables "useful", "funny", "cool", "txt_len", and "sent_res" are used in the model to predict the star label.

Table 4.1: VIF of variables

Variable	**VIF**
stars	2.142
useful	2.007
funny	1.8
cool	1.936
text_len	2.212
sent_res	1.31

According to the multinomial logistic regression model, based on the given variables, the accuracy score of the training, validation, and testing datasets is 38.84%, 38.61%, and 38.56%, respectively.

CHAPTER 5

Conclusion & Further Discussion

5.1 Conclusion

Four models are created based on LSTM, GRU, BERT, and Multinomial Logistic Regression. In terms of model selection, the BERT model based on BERT's fine-tuning would be the best because it provides the best accuracy and the minimum loss of all the datasets compared to those of the other models. Table 5.1 displays the detail of each models performance.

Table 5.1: Model comparison

	epoch	**train loss**	**train accu (%)**	**val loss**	**val accu (%)**	**test loss**	**test accu (%)**
LSTM	18	0.9485	58.56	0.9762	57.60	0.979	57.2
GRU	38	0.8778	61.45	0.9025	60.64	0.903	60.4
BERT*	3	0.6605	71.38	0.7276	68.87	0.7277	68.57
M.L.R			38.84		38.61		38.56

According to the accuracy score from the BERT model, the model reaches about 70% accuracy. After analyzing many reviews with different star ratings, the finding is that the minor difference among stars is irrelevant. For example, many reviews from Star 1 and Star 2, even some from Star 3, are alike. Similarly, reviews from Star 4 and Star 5 both show similar positive attitude statements. In addition, the stars are based on public reviews. Every reviewer has no agreed star rating, even with a similar experience. This situation makes the model more difficult to predict the precise star rating. However, the model can easily predict high or low stars.

Our model is essential since it would help businesses' performance and development. The

model is practical for companies to estimate their overall star ratings based on customer reviews from all the marketing rating sources, such as the survey and reviews collected from various sources: Swagbucks, Google Form, SurveyMonkey, etc. The companies could extract common words from low star and high star reviews to find their strengths and weaknesses, thus managing their business strategies and focuses. An incoming company can use the model to determine the ratings of potential competitors within the area, decide the location, and adjust business plans for success.

5.2 Further Discussion

Although comprehensive research and careful model training are done, many criteria still need to be improved. Three aspects for improvements are system environment, sample selection, and models.

From the system environment aspect, this research environment is based on the Google Colab Pro version. There is a 24-hour runtime restriction and graphic card limit. If conducted under a better environment, we could feed more data and faster training speed, and the models could probably yield better accuracy scores.

Another aspect is the sample selection. In terms of the sample size, in the resampling process, a total of 350,000 records is selected. The full dataset contains a lot more data than the selection. With more records, we might get a better model. Another one would be a better data cleaning strategy. As mentioned in Chapter 3, there are many fake reviews and unbalanced star ratings. Developing and utilizing a fake review classification model during the data cleaning process would get more trustworthy reviews for the models.

Last but not least, we only use "bert-base-cased" as the pre-trained model. There are many more transformer models for text classification on Hugging Face, such as XLNet, DistilBERT, and RoBERTa, which was introduced by facebook and contains 1000% more data than BERT [Sul21].

REFERENCES

[And22] Louie Andre. "78 yelp statistics you must know: 2022 market share & user profile analysis." https://financesonline.com/yelp-statistics/, Jan 2022.

[Ant22] James Anthony. "62 customer reviews statistics you must learn: 2021/2022 market share analysis & data." https://financesonline.com/customer-reviews-statistics/, Jan 2022.

[DCL19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional Transformers for language understanding." *arXiv.org*, May 2019.

[DO13] Jillian D'Onfro. "A Whopping 20% Of Yelp Reviews Are Fake." https://www.businessinsider.com/20-percent-of-yelp-reviews-fake-2013-9, 2013.

[Fac] Hugging Face. "Models." https://huggingface.co/models?pipeline_tag=text-classification\&sort=downloads.

[FBW19] Josef Fagerström, Magnus Bång, Daniel Wilhelms, and Michelle S. Chew. "LISEP LSTM: A machine learning algorithm for early detection of septic shock." *Scientific Reports*, **9**(1), 2019.

[FLH07] Ronald N. Forthofer, Eun Sul Lee, and Mike Hernandez. "13 - Linear Regression." In Ronald N. Forthofer, Eun Sul Lee, and Mike Hernandez, editors, *Biostatistics (Second Edition)*, pp. 349–386. Academic Press, San Diego, second edition edition, 2007.

[HCS21] Cheng Hua, Dr. YounJeng Choi, and Qingzhou Shi. *Companion to BER 642: Advanced regression methods*. Inprogress, University of Alabama, Apr 2021. https://bookdown.org/chua/ber642_advanced_regression/multinomial-logistic-regression.html.

[Inc] Yelp Inc. "Yelp open dataset." https://www.yelp.com/dataset.

[Leea] Hung-yi Lee. "Transformer Hung-yi Lee.".

[Leeb] Jeong Min Lee. "Recurrent neural networks and long-short term memory (LSTM)." https://people.cs.pitt.edu/~jlee/papers/cs3750_rnn_lstm_slides.pdf.

[Sul21] Ph.D. Suleiman Khan. "Bert, Roberta, Distilbert, XLNet-which one to use?" https://towardsdatascience.com/bert-roberta-distilbert-xlnet-which-one-to-use-3d5ab82ba5f8, May 2021.

[VSP17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in neural information processing systems*, pp. 5998–6008, 2017.