Yelp Star Prediction and Application

12/ 20/ 2021

Deep Learning for Prediction of Business Outcomes

Final Project

Harry Qiu(489297) **Yuening Pan** (489424) **Yingzhi Chen** (489703)

Abstract

Users' review is one of the most crucial aspects for online services like Yelp. Yelp app provides a general reliable platform for user-produced content, mostly reviews about businesses and stores. Generally, people could make reservations and look for ratings of stores. Yelp reviews provide people with recommendations, real experience, directions, etc. In this project, we are focusing on the Yelp review data including reviews, ratings and evaluations for all kinds of businesses. The ratings ranging from 1 to 5 stars and the reviews are text provided by users. Yelp also has a feature of "useful" to let other users vote if the review is useful or not. Besides, "funny" indicates if the users think the review is funny or not and "Cool" indicates if the review is popular. The goal of the project is to use words frequency from the review text to predict the ratings for each business. 20,000 were randomly selected from the data to be implemented by our deep learning methods. Specifically, the model is trained with simple RNN, LSTM and CONV1D + LSTM. Experimental results show that Cov1D + LSTM is a better technique which gives a best val_MAE of 0.4944.

Key Words: MAE, LSTM, RNN, CONV1D

Introduction

Nowadays, content search-ability is critical for any business and could considerably affect organizational revenue levels. Yelp is a business organization that connects individuals with great local businesses and stores. Yelp's reviews consist of rating, detailed review text, helpful or not, funny or not and cool or not. These reviews help users to find their preference when choosing between different business stores based on other people's experience. Users of Yelp platform play a crucial role in generating the reviews of local businesses, including the boutiques, restaurant, saloons, dentist services, mechanics, plumbing and others. The organization's primary business includes creation of connections between consumers who read and write reviews and the local organizations reviewed or described.

The Yelp dataset is an important subset of Yelp Inc. which reviews and provides user data for use in personal, education and academic use. To achieve its strategic objectives, the platform uses innovative systems to create enhanced connections between business owners and the consumers. Research has shown that "86% of people will hesitate to purchase from a business that has negative online reviews."(Saleh) and "on average, a one-star increase on Yelp leads to a 5 to 9% increase in a business's revenue. At the same time, a single negative review can cost a business about 30 customers"(Saleh). Such innovative systems include AI and machine learning tools (Mehdiyev et al., 2020). These technologies are used to generate ads and advertisements for restaurants, hotels, salons and other businesses. However, as people tend to look at ratings for businesses as reading reviews could be time-consuming, the accuracy between review text and rating is extremely significant. Our model implemented deep learning methods to predict the

rating of business based on the word frequencies extracted from the review text. The Yelp review will help to demonstrate how Deep Learning models can be useful on both the consumers' side and on the advertising sides through useful reviews and display of relevant ads to users.

Problem Description

The project goal is to use existing review text to predict the rating given by users by analyzing the word frequencies in text. We plan to use the deep learning approach to analysis reviews from Yelp database in order to predict the star rating. We randomly selected 20,000 reviews from Yelp, covering nearly all industries of Yelp, and then got the overall text data through data cleaning. Later, the data is implemented through the RNN, LSTM and cov1D + LSTM models to predict the industry-wide star ratings based on the frequency of keywords in the review.

Literature Review

Deep learning methods could be very efficient in text mining. In text mining, the application of the deep learning tools could be very effective in text clustering and text classification (Wang et al., 2019). It would be easy to find the deep desired word frequency through deep learning.

Deep learning tools have superior performances particularly in handling images, speech and audio signal inputs. Kusiak, (2020) summarises benefits of deep learning tools and provides insights on their application in business, particularly manufacturing companies. In another study, Hosaka, (2019) demonstrated the business application of deep learning models in predicting bankruptcy. According to Hosaka (2019), the effectiveness of the deep learning tools in image analysis provides an opportunity to convert the financial statement numerical data and financial

ratios calculated from financial statements into an image. In such a study, each financial ratio is designed to correspond to a given fixed pixel position. The generated images are then used to train data on GoogleNet. Overall, such study's findings demonstrate that the deep learning model was superior to the representative conventional models in predicting business performances, bankruptcy and other critical contexts other than bankruptcy. Park and Song (2020) also used the deep learning model in construction of prediction models due to its competency in extracting essential features from the input images. The study, which uses three real-life events, demonstrates that the proposed RNN based models are effective in predicting business processes performances. The RNN based models can be used to forecast potential issues in business processes. Overall, the existing literature demonstrates that RNN, a deep learning technique, can be used to predict business process performances and help organizations react in a proactive manner.

Kusiak (2020) demonstrates that other predictive models can be used to address the static or failure to include time variable problems associated with past models. In addition, Kusiak (2020), demonstrates that deep learning tools can introduce the time dimension addressing the optimization of manufacturing systems problems. Another measure that is closely related to business performance is probability of bankruptcy. Hosaka (2019) noted that the existing models using financial indicators are incorrectly passed over since they could have complex non-linear relationships with probability of bankruptcy. Hosaka (2019) demonstrates that deep learning methods can be used to enhance business performance and bankruptcy prediction if compared to other models such as linear discriminant analysis or decision tree. In a different study, Park and Song (2020) addresses business problems experienced by managers as they try to manage singular process instances in a complex business environment. The three studies demonstrate that

RNN and other involved deep learning models can improve predictive capabilities influencing business performances.

However, the previous researchers did not focus on inclusion of process-related performance measures as well as contextual information incorporation in the predictive models. Apart from time dimensions, the predictive models used can include optimizations and simulation based data to enhance accuracies in performance predictive models. There are notable gaps in the studies since most of them do not include a predictive model that quantifies the prediction accuracies that could aid in decision making. The model does not consider the quality issues in business processes. It is also complex to determine the effective measures in prediction of business performances and bankruptcy. Therefore, future studies could focus on deployment of prediction models to ensure prediction accuracies of the deep learning methods. The existing literature supports application of deep learning models to improve predict rating based on the given review text.

Database Background and Data Processing

We used the “Yelp_reviews.csv” dataset from Kaggle, and it has 8 attributes to evaluate as we implement the different deep learning neural networks to it. The dataset is a subset of Yelp’s businesses, reviews and user data. It was originally put together for the Yelp Dataset Challenge which is a chance for us to implement analysis and research based on the given data (Yelp). A total of 20,000 reviews are randomly selected as our sample for model training. Yelp review consists of numerous kinds of text including slang, oral expression and causal dialect. Besides, misuse of punctuation and unhelpful words such as “the”, “he” etc. should also be taken into

consideration. The data is first cleaned to ensure removal of unhelpful words and focusing on the meaningful words.

Review_id: The unique ID for each review.

User_id: The ID of the Yelp reviewer. Identical for every user.

Business_id: The ID of the Yelp store. Identical for every store.

Stars: The overall ratings based on comprehension evaluation given to Yelp stores by reviewers on each review, ranging from 1 to 5.

Date: The published date of the review.

Text: The specific content of the review from Yelp reviewers.

Useful: Users will evaluate the usefulness of the store, ranging from 1 to 5.

Funny: Users will evaluate the funniness of the store, ranging from 1 to 5.

Cool: Users will evaluate if the merchant store is worth visiting, ranging from 1 to 5.

Model description

1. **Simple Recurrent Neural Network**

(1) Model Building

In this part, we built a simple Recurrent Neural Network (RNN) to solve the business problem. We choose RNN because the business problem our project wants to solve is similar to what we learned in class, Airline Tweets Sentiment Analysis for Improving Customer Satisfaction. In class, RNN shows a good fit for this kind of application, so we decided to use RNN to solve our problem.

We randomly selected 20,000 reviews from the Yelp database, dropped all the punctuations and calculated the word frequency to do rating evaluation. Our network would simply learn 8-dimensional embeddings, turn the input integer sequences (2D integer tensor) into embedded sequences (3D float tensor), flatten the tensor to 2D, and train a single Dense layer on top for classification.

Like all recurrent layers in Keras, SimpleRNN can be run in two different modes:

- It can return either the full sequences of successive outputs for each timestep (a 3D tensor of shape (batch_size, timesteps, output_features)),
- Or it can return only the last output for each input sequence (a 2D tensor of shape (batch_size, output_features)). These two modes are controlled by the return_sequences constructor argument.

In our model building process, because it is sometimes useful to stack several recurrent layers one after the other in order to increase the representational power of a network, we trained a simple recurrent network using an Embedding layer and three SimpleRNN layers. Then we compiled the model with steps_per_epoch = 128, epochs = 25 and x_train shape = (18000, 200).

```
[ ]  model = Sequential()
     # We specify the maximum input length to our Embedding layer
     # so we can later flatten the embedded inputs

     model.add(Embedding(max_features, 64, input_length=maxlen))
     # After the Embedding layer,
     # our activations have shape `(samples, maxlen, 64)`.

     model.add(SimpleRNN(32, return_sequences=True))
     model.add(SimpleRNN(16, return_sequences=True))
     model.add(SimpleRNN(32)) # default activation is "tanh"
     model.add(Dense(1))
     model.summary()
```

Figure 1

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 200, 64)           640000

 simple_rnn (SimpleRNN)      (None, 200, 32)           3104

 simple_rnn_1 (SimpleRNN)    (None, 200, 16)           784

 simple_rnn_2 (SimpleRNN)    (None, 32)                1568

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 645,489
Trainable params: 645,489
Non-trainable params: 0
_________________________________________________________________
```

Figure 2

(2) Results

For the Simple RNN, the model loss is around 0.32 and validation MAE is around 0.77, which is not a good performance.

```
Epoch 19/25
128/128 [==============================] - 19s 146ms/step - loss: 0.3831 - mape: 15.4621 - val_loss: 0.7853 - val_mape: 36.3943
Epoch 20/25
128/128 [==============================] - 19s 148ms/step - loss: 0.3832 - mape: 15.1116 - val_loss: 0.7183 - val_mape: 30.3061
Epoch 21/25
128/128 [==============================] - 19s 148ms/step - loss: 0.3715 - mape: 15.1941 - val_loss: 0.7136 - val_mape: 32.2595
Epoch 22/25
128/128 [==============================] - 19s 147ms/step - loss: 0.3836 - mape: 15.1142 - val_loss: 0.7468 - val_mape: 33.8021
Epoch 23/25
128/128 [==============================] - 19s 148ms/step - loss: 0.2977 - mape: 11.5055 - val_loss: 0.7290 - val_mape: 30.9199
Epoch 24/25
128/128 [==============================] - 19s 150ms/step - loss: 0.3246 - mape: 12.9832 - val_loss: 0.7092 - val_mape: 28.3415
Epoch 25/25
128/128 [==============================] - 19s 147ms/step - loss: 0.3227 - mape: 12.8767 - val_loss: 0.7666 - val_mape: 34.9993
```

Figure 3

We also display the training and validation loss and accuracy to evaluate the results.

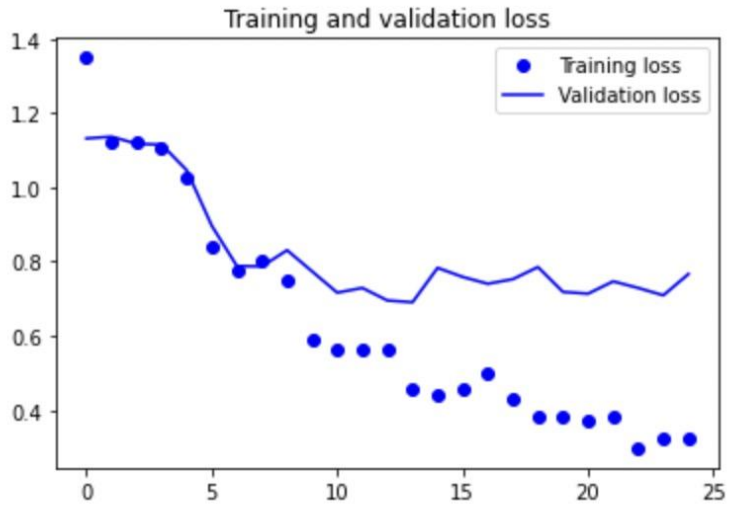

Figure 4

From the results above, we find that our small recurrent network model doesn't perform very well compared to this baseline around 0.77 validation MAE.

One remainder of the problem is simply that SimpleRNN isn't very good at processing long sequences, like text. Other types of recurrent layers perform much better.

2. **Long Short Term Memor**y

(1) Model Building

Because of the poor performance of simple RNN, we will set up a model using an LSTM layer and train it on the Tweet data. Here's the network, similar to the one with Simple RNN that we just presented. We only specify the output dimensionality of the LSTM layer, and leave every other argument (there are lots) to the Keras defaults.

Therefore, we use one embedding layer, one LSTM layer with 0.1 dropout rate, another LSTM layer and one Dense layer to build the model and do analysis with the results. Then we compiled the LSTM model with steps_per_epoch = 128, epochs = 25 and x_train shape = (18000, 200).

```
model = Sequential()
model.add(Embedding(max_features, 64, input_length=maxlen))
model.add(LSTM(32,return_sequences=True,dropout=0.1,recurrent_dropout=0.1))
model.add(LSTM(16))
model.add(Dense(1))

model.summary()
```

Figure 5

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 200, 64)           640000

 lstm (LSTM)                 (None, 200, 32)           12416

 lstm_1 (LSTM)               (None, 16)                3136

 dense (Dense)               (None, 1)                 17

=================================================================
Total params: 655,569
Trainable params: 655,569
Non-trainable params: 0
_________________________________________________________________
```

Figure 6

(2) Results

For the LSTM model, the model loss is around 0.40 and validation MAE is around 0.51, which is a significant improvement compared with simple RNN.

```
Epoch 19/25
128/128 [==============================] - 43s 337ms/step - loss: 0.4127 - mape: 16.3760 - val_loss: 0.5052 - val_mape: 20.7805
Epoch 20/25
128/128 [==============================] - 43s 337ms/step - loss: 0.4269 - mape: 16.5196 - val_loss: 0.4960 - val_mape: 19.3874
Epoch 21/25
128/128 [==============================] - 43s 337ms/step - loss: 0.4375 - mape: 16.8651 - val_loss: 0.5064 - val_mape: 19.3901
Epoch 22/25
128/128 [==============================] - 43s 335ms/step - loss: 0.4248 - mape: 16.8023 - val_loss: 0.5080 - val_mape: 19.8828
Epoch 23/25
128/128 [==============================] - 43s 336ms/step - loss: 0.3852 - mape: 14.3180 - val_loss: 0.4972 - val_mape: 20.1058
Epoch 24/25
128/128 [==============================] - 43s 337ms/step - loss: 0.3863 - mape: 14.9754 - val_loss: 0.5441 - val_mape: 22.2721
Epoch 25/25
128/128 [==============================] - 43s 337ms/step - loss: 0.4082 - mape: 15.7896 - val_loss: 0.5088 - val_mape: 21.9155
```

Figure 7

We also display the training and validation loss and accuracy to evaluate the results.

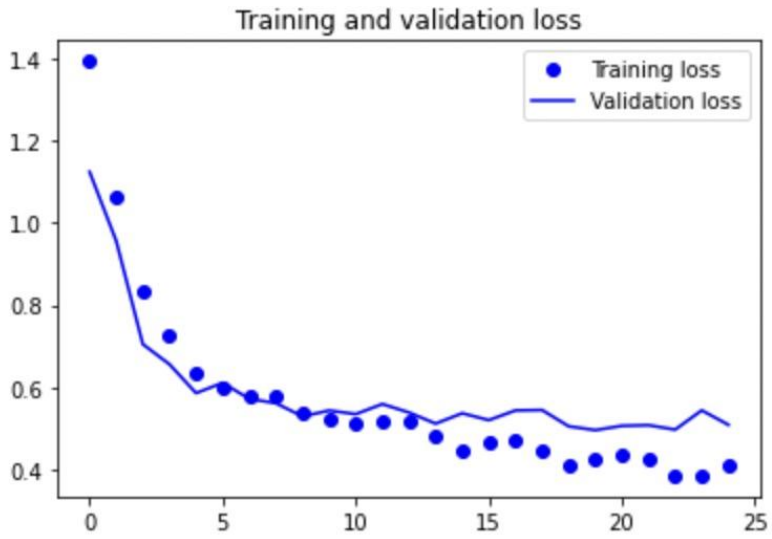

Figure 8

From the results above, we find that our LSTM model performs better with around 0.77 validation MAE than the simple RNN.

LSTM improves our result a lot, but we are considering if there are other kinds of layers that could be added into our model to improve the result accuracy.

3. 1D Convolutional Neural Networks and LSTM

(1) Model Building

From the results of the LSTM model, we still want to improve our model accuracy. We can see that the added layers do improve our results by a bit, but do not meet our expectations. So we choose to add one 1D Convolutional layer into our model. From what we learned in class, adding Cov1D layers can improve the model and solve the problem, so we want to have a try.

```
model = Sequential()
model.add(Embedding(max_features, 64, input_length=maxlen))
model.add(Conv1D(64,10,activation="relu"))
model.add(MaxPooling1D(3))
model.add(LSTM(32,return_sequences=True,dropout=0.1,recurrent_dropout=0.1))
model.add(LSTM(16))
model.add(Dense(1))
model.summary()
```

Figure 9

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 200, 64)           640000

 conv1d (Conv1D)             (None, 191, 64)           41024

 max_pooling1d (MaxPooling1D  (None, 63, 64)           0
 )

 lstm (LSTM)                 (None, 63, 32)            12416

 lstm_1 (LSTM)               (None, 16)                3136

 dense (Dense)               (None, 1)                 17

=================================================================
Total params: 696,593
Trainable params: 696,593
Non-trainable params: 0
_________________________________________________________________
```

Figure 10

Therefore, we use one embedding layer, one Cov1D layer, one LSTM layer with 0.1 dropout rate, another LSTM layer and one Dense layer to build the model and do analysis with the results. Then we compiled the Cov1D and LSTM model with steps_per_epoch = 128, epochs = 25 and x_train shape = (18000, 200).

(2) Results

For the Cov1D and LSTM model, the model loss is around 0.34 and validation MAE is around 0.49, which performs best among the three models with the lowest MAE. And we find that both the validation MAE and validation MAPE are lower than the results of the other two models.

```
Epoch 18/25
128/128 [==============================] - 21s 165ms/step - loss: 0.3991 - mape: 15.5342 - val_loss: 0.4932 - val_mape: 19.3488
Epoch 19/25
128/128 [==============================] - 21s 166ms/step - loss: 0.3737 - mape: 14.6905 - val_loss: 0.5118 - val_mape: 19.4609
Epoch 20/25
128/128 [==============================] - 21s 164ms/step - loss: 0.3763 - mape: 14.7867 - val_loss: 0.4908 - val_mape: 19.7545
Epoch 21/25
128/128 [==============================] - 21s 165ms/step - loss: 0.3819 - mape: 15.1527 - val_loss: 0.4985 - val_mape: 20.7357
Epoch 22/25
128/128 [==============================] - 21s 165ms/step - loss: 0.3976 - mape: 15.5718 - val_loss: 0.5022 - val_mape: 17.8115
Epoch 23/25
128/128 [==============================] - 21s 165ms/step - loss: 0.3603 - mape: 13.7586 - val_loss: 0.4922 - val_mape: 20.5943
Epoch 24/25
128/128 [==============================] - 21s 164ms/step - loss: 0.3493 - mape: 13.6379 - val_loss: 0.4876 - val_mape: 17.7611
Epoch 25/25
128/128 [==============================] - 21s 164ms/step - loss: 0.3420 - mape: 13.4512 - val_loss: 0.4944 - val_mape: 19.0081
```

Figure 11

We also display the training and validation loss and accuracy to evaluate the results.

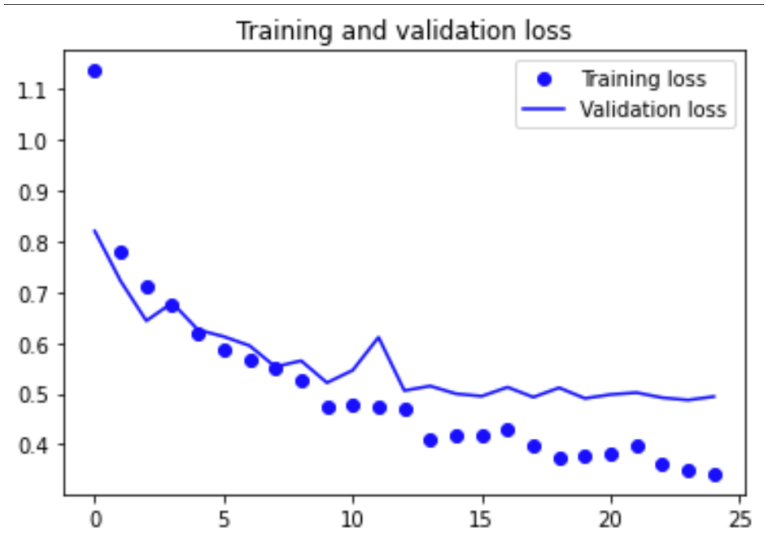

Figure 12

From the results above, we find that our Cov1D and LSTM model performs well with around 0.49 validation MAE and 19 validation MAPE. That means this kind of model would be a better fit for evaluating reviews to do star rating prediction.

Model Performance Comparison

To find the best model to solve our star rating problem, MAE and MAPE of the test data set are utilized as the performance measurement. Best performance of the Cov1D and LSTM model is

around 0.49 validation MAE and 19 validation MAPE. And the LSTM model has better performance than Simple RNN. In conclusion, for current dataset and algorithms, the Cov1D and LSTM model can have the best performance.

Model	Val MAE	Val MAPE
Simple RNN	0.77	35
LSTM	0.51	21.92
Cov1D+LSTM	0.49	19

Figure 13

Conclusions

1. For the simple RNN model, we got the validation MAE as around 0.77, which is a poor performance. So the simple RNN model may not be suitable to evaluate the reviews to do star rating prediction.
2. For the LSTM model, it has better results than simple RNN. We use two layers of LSTM and add dropout rate and recurrent dropout rate to solve the overfitting problems.
3. Combining the Cov1D and LSTM, we found that it is the best model with the validation MAE 0.4944 and the validation MAPE 19.0081. Both the two key validation key results are lowest among the three models when the validation MAE is lower than 0.5 and the validation MAPE is lower than 20, which is closer to testing results.

So we find that the model with both the Cov1D and LSTM layers is most suitable to solve the star rating prediction. The MAE is significantly reduced by adding one Cov1D layer and Maxpooling1D layer into the previous LSTM model.

Applications and Future work

(1) Applications

When we select data from the Yelp database, instead of choosing a specific industry, we use the random data covering every industry. So the model we conduct could apply to several industries which have review data and need to provide ratings for customers, such as hotels, restaurants, and so on. We can apply our model to solve similar business problems as doing sentiment analysis for airline companies.

(2) Future Work

Overall, for current deep learning algorithms and dataset, Cov1D + LSTM model works best for this star rating problem with the low MAE of 0.49. We can achieve our project objectives that predict star ratings well with Yelp's data in the future. And at the next step, we will use bidirectional RNN and explore better approaches to improve the performance of our model.

Reference

Saleh, Khalid. “The Importance of Online Customer Reviews [Infographic].” *Invesp*, 11 Apr. 2018, https://www.invespcro.com/blog/the-importance-of-online-customer-reviews-infographic/.

Wang, Haoriqin, et al. “Application of Deep Learning in Text Mining.” *Application of Deep Learning in Text Mining | Atlantis Press*, Atlantis Press, 1 Mar. 2014, https://www.atlantis-press.com/proceedings/mce-14/14128.

Hosaka, T. (2019). Bankruptcy prediction using imaged financial ratios and convolutional *Neural Networks. Expert Systems with Applications, 117*, 287-299. doi:10.1016/j.eswa.2018.09.039

Kusiak, A. (2019). Convolutional and generative adversarial neural networks in manufacturing. *International Journal of Production Research, 58*(5), 1594-1604. doi:10.1080/00207543.2019.1662133

Mehdiyev, N., Evermann, J., & Fettke, P. (2020). A novel business process prediction model using a deep learning method. *Business & information systems engineering, 62*(2), 143-157. https://link.springer.com/article/10.1007/s12599-018-0551-3

Park, G., & Song, M. (2020). Predicting performances in business processes using Deep Neural Networks. *Decision Support Systems, 129*, 113191. doi:10.1016/j.dss.2019.113191

Yelp, Inc. “Yelp Dataset.” *Kaggle*, 2 Mar. 2021,

https://www.kaggle.com/yelp-dataset/yelp-dataset.

Appendix

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import keras

df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/yelp_review.csv")
data=df.sample(n=20000)
data.head()
```

#get sample from the entire dataset

```
# We need to get rid of special marks from reviews
def  clean_text(text):

    text = text.str.lower()
    text = text.apply(lambda T: re.sub(r"(@[A-Za-z0-9]+)|([^0-9A-Za-z
\t])|(\w+:\/\/\S+)|^rt|http.+?", "", T))

    return text
data['text']= clean_text(data['text'])
data['text'] = data['text'].str.replace('#','')
data.text.head()
```

```
from sklearn.model_selection import train_test_split
rev=data["text"]
target=data["stars"]
x_train, x_test, y_train, y_test =
train_test_split(rev,target,test_size=0.1,random_state=0,stratify=target)
```

```
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer

# Number of words to consider as features
max_features = 10000
# Cut texts after this number of words
# (among top max_features most common words)
maxlen = 200
batch_size = 32

#Tokenizing
tokenizer=Tokenizer(max_features,oov_token="<?>")
tokenizer.fit_on_texts(x_train)
X_train=tokenizer.texts_to_sequences(x_train)
X_test=tokenizer.texts_to_sequences(x_test)

print(len(X_train), 'train sequences')
print(len(X_test), 'test sequences')

# X_train with upper case "X" is tokenized, x_train with lower case "x" is
not

# This turns our lists of integers
# into a 2D integer tensor of shape `(samples, maxlen)`
print('Pad sequences (samples x text length)')
X_train = pad_sequences(X_train, maxlen=maxlen)
X_test = pad_sequences(X_test, maxlen=maxlen)
print('X_train shape:', X_train.shape)
print('X_test shape:', X_test.shape)
X_train_df = pd.DataFrame(X_train)
X_train_df
```

#Simple RNN

```
model = Sequential()
# We specify the maximum input length to our Embedding layer
# so we can later flatten the embedded inputs

model.add(Embedding(max_features, 64, input_length=maxlen))
# After the Embedding layer,
# our activations have shape `(samples, maxlen, 64)`.

model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(16, return_sequences=True))
model.add(SimpleRNN(32)) # default activation is "tanh"
model.add(Dense(1))
model.summary()
```

```
model.compile(optimizer='adam', loss='mae', metrics='mape')
history = model.fit(X_train, y_train,
                    steps_per_epoch = 128,
                    epochs=30,
                    batch_size= batch_size,
                    validation_data=(X_test,y_test))
```

#LSTM

```
model = Sequential()
model.add(Embedding(max_features, 64, input_length=maxlen))
model.add(LSTM(32,return_sequences=True,dropout=0.1,recurrent_dropout=0.1)
)
model.add(LSTM(16))
model.add(Dense(1))

model.summary()
model.compile(optimizer='rmsprop', loss='mae', metrics='mape')
history = model.fit(X_train, y_train,
                    steps_per_epoch = 128,
                    epochs=30,
```

```
                    batch_size=batch_size,
                    validation_data=(X_test,y_test))
```

#CONV1D&LSTM

```
model = Sequential()
model.add(Embedding(max_features, 64, input_length=maxlen))
model.add(Conv1D(64,10,activation="relu"))
model.add(MaxPooling1D(3))
model.add(LSTM(32,return_sequences=True,dropout=0.1,recurrent_dropout=0.1)
)
model.add(LSTM(16))
model.add(Dense(1))
model.summary()

model.compile(optimizer='rmsprop', loss='mae', metrics='mape')
history = model.fit(X_train, y_train,
                    steps_per_epoch = 128,
                    epochs=25,
                    batch_size=batch_size,
                    validation_data=(X_test,y_test))
```

#Plot result

```
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(loss))

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```