PERFORMANCE ASSESSMENT

Task 2 | Sentiment Analysis

KAILI HAMILTON

Masters of Science in Data Analytics, Western Governors University

Course: D213

Instructor: Dr. Kesselly Kamara

Program Mentor: Krissy Bryant

March, 2024

TABLE OF CONTENTS

A1

One research question that I will answer using neural network models and natural language processing techniques is, “Can I use reviews left by customers on Yelp to predict their sentiment as a positive or negative review?”

A2

The objective of this analysis is to use neural network models and natural language processing techniques is to analyze and predict customer sentiment based on the reviews they submit on Yelp.

A3

A type of neural network (NN) capable of performing a text classification task that can be trained to produce useful predictions on text sequences on a selected data is recurrent neural network (RNN). RNN works with sequential data but it is well-suited for natural language processing (NLP), including text classification (Kamara, n.d.).

B1

The data set used for this performance assessment is the Yelp data. Exploratory data analysis (EDA) was performed. This data contains 1000 rows containing the customers’ reviews and the sentiment conveyed, positive or negative. Through EDA I found the following (Kamara, n.d.):

- This data did not contain the presence of unusual characters such as emojis or non-English characters.
- The sentiment type is split evenly, 50% positive and 50% negative.
- The vocabulary size is 2072.
- The proposed word embedding length is 50. This is because the data set is relatively small, the data contains a smaller vocabulary, and we are performing binary classification in our sentiment analysis. Embedding length of 50-100 is standard for a dataset like the Yelp reviews data.
- The chosen maximum sequence length is 32. The statistical justification for this value is that once the data have been tokenized, lemmatized, and padded, the maximum sequence length is 32. This will ensure most sequences fit within this limit without significant information loss.

```
#Inital list of words/characters in reviews
reviews = df['Review']
list_of_chars = []
for comment in reviews:
    for character in reviews:
        if character not in list_of_chars:
            list_of_chars.append(character)
print(list_of_chars)
```

```
['Wow... Loved this place.', 'Crust is not good.', 'Not tasty and the texture was just nasty.', 'Stopped by during the late May bank holiday off Rick Steve recommen
dation and loved it.', 'The selection on the menu was great and so were the prices.', 'Now I am getting angry and I want my damn pho.', "Honeslty it didn't taste TH
AT fresh.)", 'The potatoes were like rubber and you could tell they had been made up ahead of time being kept under a warmer.', 'The fries were great too.', 'A grea
t touch.', 'Service was very prompt.', 'Would not go back.', 'The cashier had no care what so ever on what I had to say it still ended up being wayyy overpriced.',
'I tried the Cape Cod ravoli, chicken,with cranberry...mmmm!', 'I was disgusted because I was pretty sure that was human hair.', 'I was shocked because no signs ind
icate cash only.', 'Highly recommended.', 'Waitress was a little slow in service.', 'This place is not worth your time, let alone Vegas.', 'did not like at all.',
'The Burrittos Blah!', 'The food, amazing.', 'Service is also cute.', 'I could care less... The interior is just beautiful.', 'So they performed.', "That's righ
t....the red velvet cake.....ohhh this stuff is so good.", '- They never brought a salad we asked for.', 'This hole in the wall has great Mexican street tacos, and
friendly staff.', 'Took an hour to get our food only 4 tables in restaurant my food was Luke warm, Our sever was running around like he was totally overwhelmed.',
'The worst was the salmon sashimi.', 'Also there are combos like a burger, fries, and beer for 23 which is a decent deal.', 'This was like the final blow!', 'I foun
d this place by accident and I could not be happier.', 'seems like a good quick place to grab a bite of some familiar pub food, but do yourself a favor and look els
ewhere.', 'Overall, I like this place a lot.', 'The only redeeming quality of the restaurant was that it was very inexpensive.', 'Ample portions and good prices.',
'Poor service, the waiter made me feel like I was stupid every time he came to the table.', 'My first visit to Hiro was a delight!', 'Service sucks.', 'The shrimp t
ender and moist.', 'There is not a deal good enough that would drag me into that establishment again.', "Hard to judge whether these sides were good because we were
grossed out by the melted styrofoam and didn't want to eat it for fear of getting sick.", 'On a positive note, our server was very attentive and provided great serv
ice.', 'Frozen pucks of disgust, with some of the worst people behind the register.', 'The only thing I did like was the prime rib and dessert section.', "It's too
bad the food is so damn generic.", 'The burger is good beef, cooked just right.', 'If you want a sandwich just go to any Firehouse!!!!!', 'My side Greek salad with
the Greek dressing was so tasty, and the pita and hummus was very refreshing.', 'We ordered the duck rare and it was pink and tender on the inside with a nice char
```

```
def contains_non_english(text):
    # Regular expression pattern to match English characters and common punctuation marks
    pattern = re.compile(r'^[a-zA-Z0-9!@#$%^&*(),.?":{}|<>]+$')

    # Check if the text contains non-English characters
    if pattern.match(text):
        return True
    else:
        return False

print(contains_non_english('list_of_chars'))
```

```
False
```

```
# Identify vocabulary size

tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['Review'])

vocab_size = len(tokenizer.word_index) + 1
print("Vocabulary Size: ", vocab_size)
```

```
Vocabulary Size:  2072
```

```
review_len = []
for char in reviews:
    review_len.append(len(char.split(" ")))

review_max = np.max(review_len)
review_min = np.min(review_len)
review_median = np.median(review_len)

print("Max sequence length: ", review_max)
print("Median sequence length: ", review_median)
print("Min sequence length: ", review_min)
```

```
Max sequence length:  32
Median sequence length:  10.0
Min sequence length:  1
```

B2

The goals of the tokenization process and text normalization are as follows (Kamara, n.d).

1. Convert text to all lowercase.
2. Remove non-English characters or unusual characters, such as emojis.
3. Lemmatize words, which reduces them to their root form, such as “skiing” to “ski”.
4. Split up sentences into individual words, called tokens.
5. Remove stop words.

The code and packages used to complete the above process are found below.

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

import sklearn
from sklearn import preprocessing
from sklearn.preprocessing import OneHotEncoder
from sklearn import model_selection
from sklearn.model_selection import train_test_split
```

```
import re
import tensorflow as tf
from tensorflow import keras
from keras import preprocessing

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.models import load_model
from tensorflow.keras.layers import Dense, Embedding
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package stopwords to ...
```

```
nltk.download('punkt')
nltk.data.path.append("/path/to/nltk_data")
```

```
[nltk_data] Downloading package punkt to ...
```

```
nltk.download('wordnet')
nltk.data.path.append("/path/to/nltk_data")
```

```
[nltk_data] Downloading package wordnet to ...
```

```
description_list = []
stop_words = stopwords.words('english')
for description in df.Review:

    #Regular Expression removes puncuation and special characters
    description = re.sub("[^a-zA-Z]", " ", description)

    #convert to lowercase
    description = description.lower()

    #perform tokenization
    description = nltk.word_tokenize(description)

    #perform lemmatization
    lemma = nltk.WordNetLemmatizer()
    decription = [lemma.lemmatize(word) for word in description if word not in stop_words]

    description_list.append(description)

print(description_list)
```

```
[['wow', 'loved', 'this', 'place'], ['crust', 'is', 'not', 'good'], ['not', 'tasty', 'and',
'the', 'late', 'may', 'bank', 'holiday', 'off', 'rick', 'steve', 'recommendation', 'and', 'l
'and', 'so', 'were', 'the', 'prices'], ['now', 'i', 'am', 'getting', 'angry', 'and', 'i', 'w
at', 'fresh'], ['the', 'potatoes', 'were', 'like', 'rubber', 'and', 'you', 'could', 'tell',
```

```
# Removing stopwords

decription = [word for word in description if not word in stop_words]
description = " ".join(description)
description_list.append(description)
print(stop_words)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "
s', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's",
```

```
# Identify vocabulary size

tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['Review'])

vocab_size = len(tokenizer.word_index) + 1
print("Vocabulary Size: ", vocab_size)
```

```
Vocabulary Size:  2072
```

B3

The padding process used to standardize the length of sequences is as follows. To input data into our model, all inputs must have the same length. We can limit the size of the sequence by "truncating longer reviews and padding shorter reviews with a null value (0)" (Li, 2018). To continue the padding process, we then add the token, 0, to the training and testing data to create new padded training and testing sets from which you run the model. In this analysis the maximum sequence length is 32.

- The padding for the model in this analysis occurs after the sequence.
- The code below displays a screenshot of a single padded sequence.

Padding

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
padding_type = 'post'  # or 'pre'
trunc_type = 'post'    # or 'pre'

# Applying padding to training data
sequences_train = tokenizer.texts_to_sequences(X_train)
padded_train = pad_sequences(sequences_train, maxlen = review_max, padding=padding_type, truncating=trunc_type)

# Applying padding to testing data
sequences_test = tokenizer.texts_to_sequences(X_test)
padded_test = pad_sequences(sequences_test, maxlen = review_max, padding=padding_type, truncating=trunc_type)
```

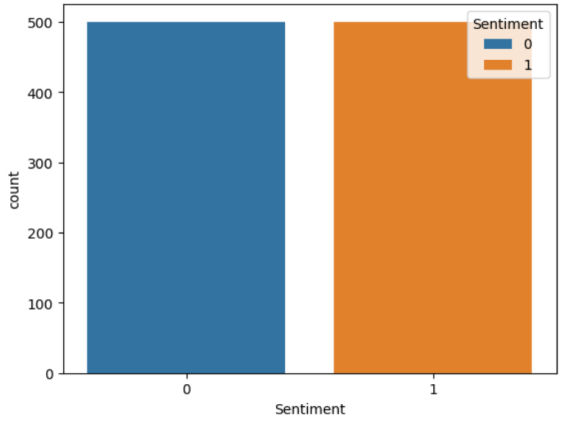
```
import sys
```

```
# Display the padded sequence

np.set_printoptions(threshold=sys.maxsize)
padded_train[1]
```

```
array([   9,   45,    7, 2056,   24,    1,   15, 1722, 2058,  155,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0])
```

B4

Two categories of sentiment will be used in this analysis, positive (1) and negative (0).



Softmax is the activation function used for the final dense layer of the neural network because it outputs probabilities, which can be interpreted easily (Kamara, n.d.).

```
# define hyperparameters
activation = 'softmax'
loss = 'sparse_categorical_crossentropy'
optimizer = 'adam'
num_epochs = 20
embedding_dim = 50

# define early_stopping_monitor
early_stopping_monitor = EarlyStopping(patience=2)

# define the model
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_shape=(review_max,)),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(50, activation='relu'),
    tf.keras.layers.Dense(2, activation=activation)
])

# train the model
model.compile(loss=loss, optimizer=optimizer, metrics=['accuracy'])
model.summary()
history = model.fit(training_padded, training_label, batch_size=32, epochs=num_epochs,
                    validation_split=0.3, callbacks=[early_stopping_monitor], verbose=True)
```

B5

The steps to prepare the data for analysis are as follows (Kamara, n.d.).

1. Import necessary libraries and packages and read in the data set.
2. Perform exploratory data analysis to determine the shape of the data, how many categories of sentiment are present, detect the presence of unusual characters, determine the vocabulary size and word embedding length.
3. Tokenize and normalize the text data by converting text to all lowercase, remove non-English characters or unusual characters, such as emojis, lemmatize words, which reduces them to their root form, such as “skiing” to “ski”, divide sentences into individual words, called tokens, and remove stop words.
4. Determine the maximum sequence length after tokenization and lemmatization. 32 in this analysis.
5. Split the data into training, validation, and testing sets based on industry standards. This analysis splits the data into 80% training, 0% validation, and 20% testing (Karama, n.d.).
6. Pad the data to standardize the length of sequences. This padding and truncating type of this analysis includes the tokens at the end of the sequence.
7. Apply the padding process to the training, validation, and testing data sets to be used to build the model.

B6

A copy of the prepared data sets are submitted with the submission of this performance assessment.

C1

The output of the model summary of the function form TensorFlow is provided below.

```
Model: "sequential_2"
```

Layer (type)	Output Shape	Param #
embedding_11 (Embedding)	(None, 32, 50)	103,600
global_average_pooling1d_2 (GlobalAveragePooling1D)	(None, 50)	0
dense_6 (Dense)	(None, 100)	5,100
dense_7 (Dense)	(None, 50)	5,050
dense_8 (Dense)	(None, 2)	102

```
Total params: 113,852 (444.73 KB)
Trainable params: 113,852 (444.73 KB)
Non-trainable params: 0 (0.00 B)
Epoch 1/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.4958 - loss: 0.6944 - val_accuracy: 0.4417 - val_loss: 0.6996
Epoch 2/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5280 - loss: 0.6882 - val_accuracy: 0.4417 - val_loss: 0.7027
Epoch 3/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5370 - loss: 0.6855 - val_accuracy: 0.4833 - val_loss: 0.6955
Epoch 4/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6112 - loss: 0.6625 - val_accuracy: 0.5708 - val_loss: 0.6662
Epoch 5/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6491 - loss: 0.6416 - val_accuracy: 0.6708 - val_loss: 0.6313
Epoch 6/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7724 - loss: 0.5449 - val_accuracy: 0.8250 - val_loss: 0.5241
Epoch 7/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9169 - loss: 0.3829 - val_accuracy: 0.7042 - val_loss: 0.5143
Epoch 8/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9333 - loss: 0.2412 - val_accuracy: 0.8333 - val_loss: 0.4077
Epoch 9/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9317 - loss: 0.1778 - val_accuracy: 0.8208 - val_loss: 0.4093
Epoch 10/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9785 - loss: 0.1065 - val_accuracy: 0.7708 - val_loss: 0.4489
```

This is the code that generated the output.

```
# define hyperparameters
activation = 'softmax'
loss = 'sparse_categorical_crossentropy'
optimizer = 'adam'
num_epochs = 20
embedding_dim = 50

# define early_stopping_monitor
early_stopping_monitor = EarlyStopping(patience=2)

# define the model
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_shape=(review_max,)),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(50, activation='relu'),
    tf.keras.layers.Dense(2, activation=activation)
])

# train the model
model.compile(loss=loss, optimizer=optimizer, metrics=['accuracy'])
model.summary()
history = model.fit(training_padded, training_label, batch_size=32, epochs=num_epochs,
                    validation_split=0.3, callbacks=[early_stopping_monitor], verbose=True)
```

C2

The model for this sentiment analysis has five layers. First, the input, or embedding layer, which converts input text tokens into dense word embeddings. Next, are three hidden layers, GlobalAveragePooling1D, and two dense layers. The GlobalAveragePooling1D layer takes the average of all words from the embedding layer so it's one dimensional. The two hidden layers are fully connected with 100 and 50 units respectively and use the ReLU activation function. Lastly, the output layer, which is a dense layer with 2 units (because of the binary classification of this analysis) using the Softmax activation function (Kamara, n.d.).

The total number of parameters generated are 113,852. From the input layer: 103,600. From the GlobalAveragePooling1D layer: 0. From the other two hidden layers: 5,100 and 5,050 respectively. From the output layer: 102. The optimizer function, Adam, adjusts these parameters during training to minimize loss and improve the model's performance (Kamara, n.d.).

C3

The justification for the choice of the hyperparameters used in this model are found below.

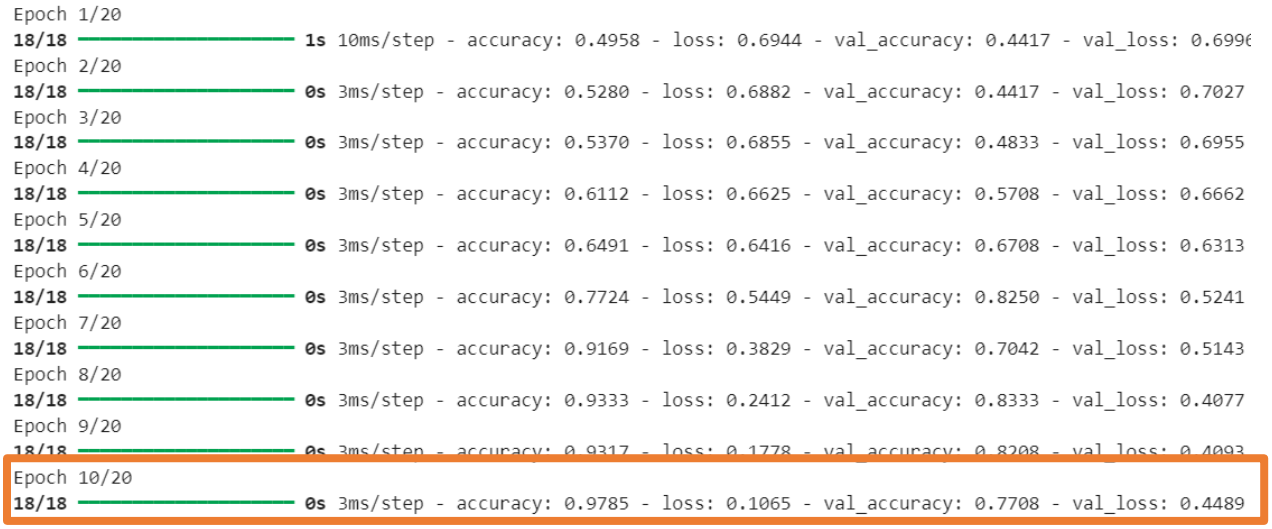
- Activation functions. The hidden layers use ReLU because it allows the model to capture nonlinearities, inputs 0 for negative values, and is the industry standard (Becker, n.d.). The final layer uses Softmax because it outputs probabilities, which can be interpreted easily and fits well with the binary classification of sentiment, positive and negative (Kamara, n.d.).
- Number of nodes per layer. The number of nodes in the embedding (input) layer is determined by the embedding dimension, 50 in this analysis. This number is used because the data set is relatively small, the data contains a smaller vocabulary, and we are performing binary classification in our sentiment analysis. Embedding length of 50-100 is standard for a dataset like the Yelp reviews data. Next, the dense layers use 100, 50, and 2 nodes respectively. 100 is a moderate number appropriate for this analysis to learn the patterns and features from the GlobalAveragePooling1D layer. Reducing the nodes to 50 in the next layer for higher-level learning and reducing model complexity. The output layer has 2 nodes because this is a binary classification task, positive and negative sentiments (Kamara, n.d.)..
- Loss function. Spare categorical entropy is the loss function used in this analysis because the sentiment is represented as integers, 0 or 1, and doesn't require manually doing one-hot encoding (Becker, n.d.).
- Optimizer. Adam, or Adaptive Moment Estimation, is the optimizer function because it can easily handle a model with many parameters, like the model in this analysis, and find an optimum output quickly (Kamara, n.d.).

- Stopping criteria. Early Stopping is used in this analysis because it helps prevent overfitting by stopping the training process when the validation accuracy starts to decrease. By stopping training at an early point, Early Stopping helps improve generalization in the final model and stops the model from memorizing noise in the data. The patience level is set to 2 for this analysis, so the model will run for two more epochs after the validation accuracy stops improving. The number of epochs is set to 20. So, the model will run for 20 epochs, or until the Early Stopping criteria are met (Kamara, n.d.).
- Evaluation metric. The chosen evaluation metric for this analysis is accuracy. The accuracy score is easily interpreted—it is the proportion of correctly assigned sentiment values. Furthermore, accuracy lends itself well to this binary classification task because it directly reflects the model's ability to distinguish between the two classes (Kamara, n.d.).

D1

The impact of using stopping criteria to include defining the number of epochs is discussed here, along with a screenshot showing the final training epoch.

The stopping criteria used in this analysis is Early Stopping. It is impactful to use primarily because it prevents overfitting by stopping the training process when the validation starts to decrease. By stopping training at an early point, Early Stopping helps improve generalization in the final model and stops the model from memorizing noise in the data. The patience level is set to 2 for this analysis, so the model will run for two more epochs after the validation accuracy stops improving. The number of epochs is set to 20. So, the model will run for 20 epochs, or until the Early Stopping criteria are met. The model ran for 10 epochs (Kamara, n.d.).



D2

The fitness of the model is good because steps were taken to address overfitting. These steps were hyperparameter tuning of the embedding dimension, the activation functions, the number of nodes per layer, and the loss and optimizer functions. Additionally, the use a stopping criteria, as discussed in parts C3 and D1, helps to prevent overfitting by stopping the training process when the validation starts to decrease. By stopping training at an early point, Early Stopping helps improve generalization in the final model and stops the model from memorizing noise in the data. Also monitoring the validation loss during training helps assess the model's generalization ability as well. Lastly, cross validation was performed across the training and testing data.

D3

Here are visualizations of the model's training process, including a line graph of the loss and chose evaluation metric, accuracy.

```
Model: "sequential_2"
```

Layer (type)	Output Shape	Param #
embedding_11 (Embedding)	(None, 32, 50)	103,600
global_average_pooling1d_2 (GlobalAveragePooling1D)	(None, 50)	0
dense_6 (Dense)	(None, 100)	5,100
dense_7 (Dense)	(None, 50)	5,050
dense_8 (Dense)	(None, 2)	102

```
Total params: 113,852 (444.73 KB)
Trainable params: 113,852 (444.73 KB)
Non-trainable params: 0 (0.00 B)
Epoch 1/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.4958 - loss: 0.6944 - val_accuracy: 0.4417 - val_loss: 0.6996
Epoch 2/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5280 - loss: 0.6882 - val_accuracy: 0.4417 - val_loss: 0.7027
Epoch 3/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5370 - loss: 0.6855 - val_accuracy: 0.4833 - val_loss: 0.6955
Epoch 4/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6112 - loss: 0.6625 - val_accuracy: 0.5708 - val_loss: 0.6662
Epoch 5/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6491 - loss: 0.6416 - val_accuracy: 0.6708 - val_loss: 0.6313
Epoch 6/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7724 - loss: 0.5449 - val_accuracy: 0.8250 - val_loss: 0.5241
Epoch 7/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9169 - loss: 0.3829 - val_accuracy: 0.7042 - val_loss: 0.5143
Epoch 8/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9333 - loss: 0.2412 - val_accuracy: 0.8333 - val_loss: 0.4077
Epoch 9/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9317 - loss: 0.1778 - val_accuracy: 0.8208 - val_loss: 0.4093
Epoch 10/20
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9785 - loss: 0.1065 - val_accuracy: 0.7708 - val_loss: 0.4489
```

```
# verify model accuracy on training data
score = model.evaluate(training_padded, training_label, verbose=0)

print(f'Training Loss: {score[0]} | Training Accuracy: {score[1]}')
```

```
Training Loss: 0.18936742842197418 | Training Accuracy: 0.9237499833106995
```

```
# verify model accuracy on test data
score = model.evaluate(test_padded, test_label, verbose=0)

print(f'Test Loss: {score[0]} | Test Accuracy: {score[1]}')
```

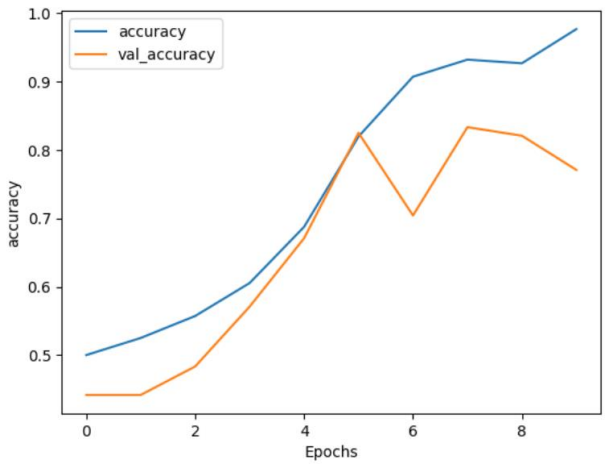
```
Test Loss: 0.481830894947052 | Test Accuracy: 0.7799999713897705
```

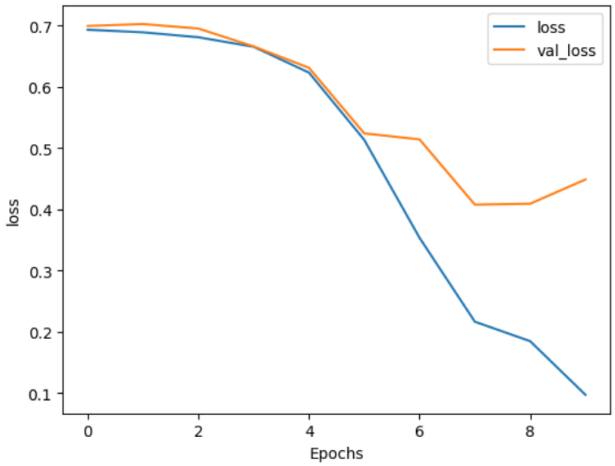
```
# verify the predicted sentiment by comparing to actual label from test data
# choose any text from data to verify

i = 89

print("Predicted Review Text: ", X_test[i], "\n")
print("Predicted: NEGATIVE" if predictions_test[i][0] >= 0.5 else "POSITIVE", "Review")
print("Actual: ", "NEGATIVE" if y_test[i] == 0 else "POSITIVE", "Review")
```

```
Predicted Review Text:  ['check', 'it', 'out']

POSITIVE Review
Actual:  POSITIVE Review
```

```
# verify the predicted sentiment by comparing to actual label from test data
# choose any text from data to verify

i = 199

print("Predicted Review Text: ", X_test[i], "\n")
print("Predicted: NEGATIVE" if predictions_test[i][0] >= 0.5 else "POSITIVE", "Review")
print("Actual: ", "NEGATIVE" if y_test[i] == 0 else "POSITIVE", "Review")
```

```
Predicted Review Text:  ['i', 'think', 'food', 'should', 'have', 'flavor', 'and', 'texture', 'and', 'both', 'were', 'lacking']

POSITIVE Review
Actual:  NEGATIVE Review
```





D4

The predictive accuracy of this model is good. The training accuracy is high, 92%. The testing accuracy is 14 points lower at 78%. This difference is within the 5-15% percent acceptable in the industry (Kamara, n.d.). The model correctly predicts the sentiment 78% of the time on new data.

```
# verify model accuracy on training data
score = model.evaluate(training_padded, training_label, verbose=0)

print(f'Training Loss: {score[0]} | Training Accuracy: {score[1]}')
```

```
Training Loss: 0.18936742842197418 | Training Accuracy: 0.9237499833106995
```

```
# verify model accuracy on test data
score = model.evaluate(test_padded, test_label, verbose=0)

print(f'Test Loss: {score[0]} | Test Accuracy: {score[1]}')
```

```
Test Loss: 0.481830894947052 | Test Accuracy: 0.7799999713897705
```

E

The code I used to save the trained network with the neural network is included in the submission of this performance assessment.

```
# saving the model
model.save('yelp_SA_model.keras')

yelp_SA_model = load_model('yelp_SA_model.keras')
```

Predictions

```
# predictions on training data
predictions_train = yelp_SA_model.predict(training_padded)
```

```
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 851us/step
```

```
# predictions on testing data
predictions_test = yelp_SA_model.predict(test_padded)
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 602us/step
```

F

The functionality of my neural network, including the impact of the network architecture is discussed here.

A neural network is a series of algorithms that endeavor to recognize underlying relationships in a data set through a process that mimics the way the human brain operates consisting of interconnected layers, called neurons, that process and transform input data to produce out predictions (Kamara, n.d.). The functionality of a neural network is impacted by its architecture. It should be designed to optimize loss and accuracy. Elements of the architecture are the number of layers and the number of nodes, the types of activation, loss, and optimization functions, and the hyperparameters. We want a model that does not over- or underfit the data and has a high accuracy on new data.

G

Based on the results of this sentiment analysis, a recommended course of action is as follows. The predictive accuracy of this model is good. The training accuracy is high, 92%. The testing accuracy is 14 points lower at 78%. This difference is within the 5-15% percent acceptable in the industry (Kamara, n.d.). The model correctly predicts the sentiment 78% of the time on new data and can be used to evaluate the sentiment of customers of the restaurant. You can use this sentiment to inform business practices to find ways to improve sentiment through things like better customer service, higher quality food, and restaurant cleanliness and atmosphere.

Yes, the model can be used with 78% accuracy on new data, but I also recommend improving this accuracy by retraining the model through hyperparameter tuning and adjusting the architecture of the neural network.

H

The neural network model was created in Jupyter Lab. An HTML document of this presentation is included with the submission of this performance assessment.

I

Becker, Dan et al., “Introduction to Deep Learning in Python” Datacamp, n.d., app.datacamp.com/learn/courses/introduction-to-deep-learning-in-python

Cecchini, David et al., “Recurrent Neural Networks (RNNs) for Language Modeling with Keras” Datacamp, n.d., app.datacamp.com/learn/courses/recurrent-neural-networks-rnn-for-language-modeling-with-keras

Kamara, Kesselly. “D213 Task 2 Building NN Model in Python”. D213 Western Governors University, n.d., wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=1b9aff54-735f-456a-a6b4-b11b00eb8d2f

Kamara, Kesselly. “D213 Task 2 Data Processing in Python”. D213 Western Governors University, n.d., wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=8639374a-964b-4ae9-b33b-b1210052c07d

J

Becker, Dan et al., “Introduction to Deep Learning in Python” Datacamp, n.d., app.datacamp.com/learn/courses/introduction-to-deep-learning-in-python

Cecchini, David et al., “Recurrent Neural Networks (RNNs) for Language Modeling with Keras” Datacamp, n.d., app.datacamp.com/learn/courses/recurrent-neural-networks-rnn-for-language-modeling-with-keras

Kamara, Kesselly. “D213 Task 2 Building NN Model in Python”. D213 Western Governors University, n.d., wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=1b9aff54-735f-456a-a6b4-b11b00eb8d2f

Kamara, Kesselly. “D213 Task 2 Data Processing in Python”. D213 Western Governors University, n.d., wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=8639374a-964b-4ae9-b33b-b1210052c07d

Li, Susan, “A Beginner’s Guide on Sentiment Analysis with RNN” Towards Data Science, 2018, towardsdatascience.com/a-beginners-guide-on-sentiment-analysis-with-rnn-9e100627c02e

K

Professional communication is demonstrated throughout the content and presentation of this PA.