D213 Task 2 Sentiment Analysis Using Neural Networks

Western Governor's University

Part I: Research Question

A1. Question

Is it possible to analyze customer sentiment using neural networks and the NLP technique so that the company can take appropriate actions to improve customer satisfaction?

A2. Objectives and Goals

The goal of this analysis is to find patterns associated with reviews from data sources including: whether positive reviews are shorter or longer than negatives reviews; whether the reviews are constructed such that they observe common sentences patterns; or the type of punctuation marks used and how the upper case letters are used.

A3. Prescribed Network

Recurrent Neural Networks (RNN) is a type of neural network designed to process sequential data. It helps captures patterns in sequential data. RNN's have a feedback loop that allows them to maintain an internal memory of past inputs. RNN's have recurrent connections that allow information to persist and flow across different time steps.

Part II: Data Preparation

B1. Data Exploration

A. Presence of unusual characters:

Alpha Characters:
{'v', 'f', 'o', 'r', 'j', 'i', 'z', 'a', 'p', 'k', 'c', 'l', 'h', 't', 'm', 's', 'n', 'u', 'q', 'x', 'b', 'd', 'w', 'e', 'y', 'g'}
Total of 26 unique English letters in this dataset

Numeric Characters:
{'4', '1', '5', '9', '8', '3', '7', '2', '0', '6'}
Total of 10 unique numerical characters in this dataset

Non-alphanumeric characters:
{'*', '/', 'é', ':', '[', '"', 'ê', 'å', '(', '+', '`', '\x96', ')', '&', ']', ',', '?', '$', ';', '%', '#', '\x97', '-', '.', '!'}
Total of 25 unique special characters in this dataset

Emojis: 2748

B. Vocabulary Size:

Vocabulary size is defined as the number of unique words in the dataset. Vocabulary size is 6078.

C. Word embedding length:

Word embedding length is the distance of the position of that word from the beginning of the vector (Brownlee, 2021). Embedding length is the fourth root of the vocabulary size. This is the squared root of the square root (Brownlee, 2021). The proposed word embedding length is 9.

D. Statistical Justification for the chosen maximum sequence length

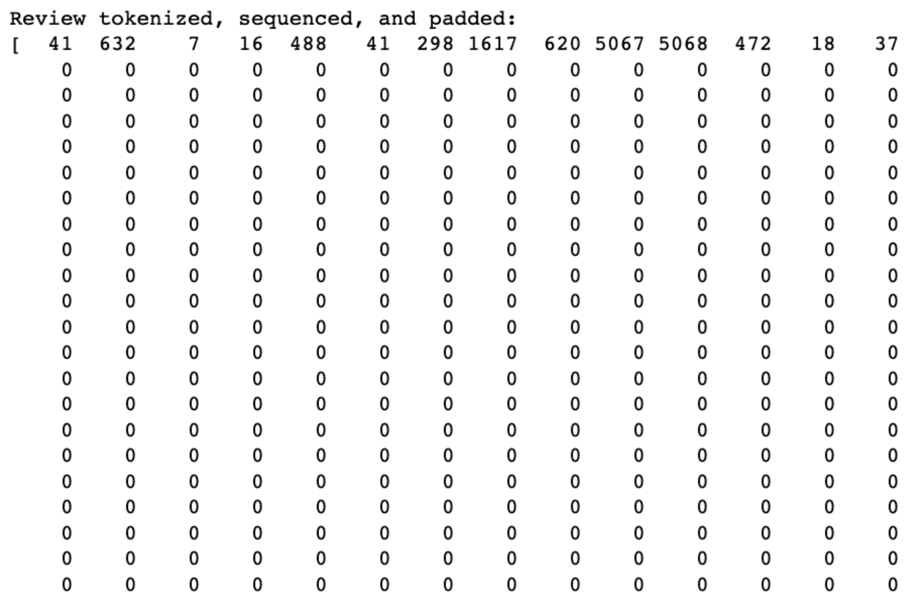
The maximum sequence length can be determined by finding the review with the highest number of words in the dataset. Inputs shorter than the maximum sequence length will be addressed by padding. The maximum word length is 140.

B2. Tokenization

The tokenization process is to separate the test into smaller chunks called tokens. These tokens can be individual words, sentences, or subword units. A unique index is called "word_index" which is assigned to each word in the text which helps the model during the training process. The goals of the tokenization process include: replacing any abnormal characters, formatting, and standardizing texts; lemmatizing words and transforming the text into the sequence; and preparing the transformed sequence to a maximum sequence length by padding (Saxena, 2021).

B3. Padding Process

Padding is the technique used to add extra elements to sequences to make them equal in length. The padding can come either before or after the test sequence. Padding improves performance by preserving the shape of the tensor dimensions. In this analysis, the padding occurs after the sequence indicated by zeros after the number sequence.



```
Review tokenized, sequenced, and padded:
[   41  632    7   16  488   41  298 1617  620 5067 5068  472   18   37
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

B4. Categories of Sentiment

The number of sentiment categories was determined to be two. This indicates there is a binary classification with positive/negative sentiment. The recommended activation function of the final dense layer of the network is the sigmoid activation function which maps the output to a value between 0 and 1. This can then be interpreted as the probability of belonging to the positive class.

B5. Steps to Prepare the Data

1. Read and load the data of each file into juypter notebook using Python and combine the three dataset into one data frame
2. perform exploratory data analysis including viewing: vocabulary size, view presence of unusual characters, and determine word embedding length,
3. Performed text cleaning operations including removing special characters, punctuation, and unwanted symbols
4. Converted the reviews into individual tokens that can be fed into the neural network to create a numerical representation of the text data
5. The tokens were then converted into numerical values to binary vectors.
6. To ensure all sequences had the same length padding was then completed
7. scaling was then completed by standardizing the data
8. The preprocessed data was split into training, validation, and test sets using 80/20 ratio. The training set is used to train the model. The validation set is used for hyperparameter tuning and model selection (Brownlee, 2020a). The test set is used to evaluate the final model performance. (Brownlee, 2020a)

B6. Prepared Dataset

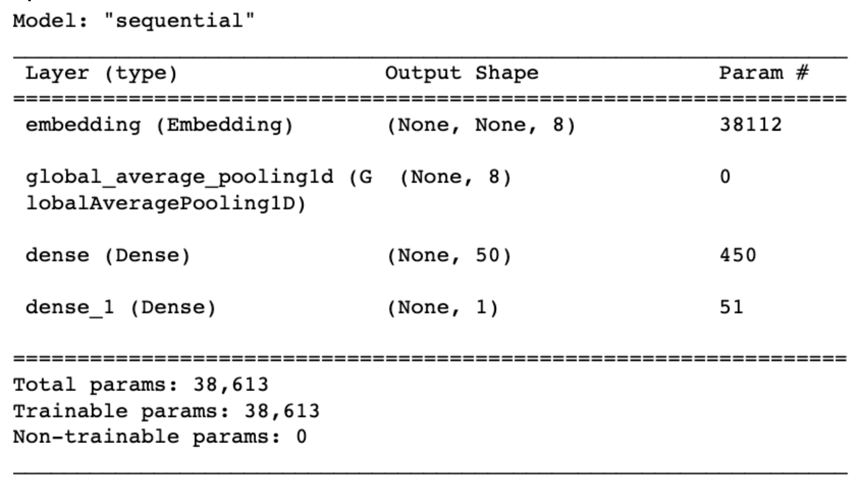
The prepared data sets are attached using the following code:

```
pd.DataFrame(X_train).to_csv('X_train.csv')
pd.DataFrame(X_test).to_csv('X_test.csv')
pd.DataFrame(y_train).to_csv('y_train.csv')
pd.DataFrame(y_test).to_csv('y_test.csv')
```

Part III: Network Architecture

C1. Model Summary

A Keras model is built using the Keras API that runs on top of TensorFlow. Keras provides an interface for creating and training the neural network. The model.summary() provides a quick way to visualize the components of the Keras model. It provides the name of the layers, the type, shapes, and the number of trainable parameters (Brownlee, 2019b). The Keras model represents the actual neural network model.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 8)           38112

 global_average_pooling1d (G  (None, 8)                0
 lobalAveragePooling1D)

 dense (Dense)               (None, 50)                450

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 38,613
Trainable params: 38,613
Non-trainable params: 0
_________________________________________________________________
```

C2. Network Architecture

The neural network was built as a sequential model from Keras. The model consists of four layers consisting of an embedding layer, a global average pooling layer, and two dense layers. The embedding layer consisted of 38,112 trainable parameters. These are trainable parameters that correspond to the vocabulary size multiplied by the embedding dimension (ActiveState, 2022). The embedding layers map each word in the input sequence to the vector size of 8. The global average pooling layers use the output of the embedding layer and computes the average value across the sequence dimensions (ActiveState, 2022). This layer has zero parameters. The first dense layer has 50 units which takes the output of the global average pooling layers and applies a linear transformation followed by a sigmoid activation function. This layer has 450 parameters. The second dense layer has one unit and 51 parameters.

C3. Hyperparameters

A. Activation Functions: There are two dense layers that require an activation function. The sigmoid function is good for binary classification problems since it exits between 0 and 1. This allows it to predict the probability of an output corresponding to a binary class (*Sigmoid Function*, n.d.)

B. Number of nodes per layer: The hidden dense layer has 50 nodes. This was determined through trial and error for best results. The output has one node for the binary class.

C. Loss of function: A binary cross-entropy is a model metric that tracks incorrect labeling of the data class. This loss function was selected due to its value use when predicting binary values that match values of 0 and 1 in the dataset.

D. Optimizer: The model utilizes the Adam optimizer. The Adam optimizer is efficient for parameter updates. It improves the accuracy and speech of deep learning models (Mahendra, 2023). Adom optimizer combines the scaled learning rate of RMSprop and the utilization of momentum like SGD and AdaGrade (Bushaev, 2018). Adom optimizers also has a high level of adaptability and is able to handle noise in the input model well to reduce overfitting.

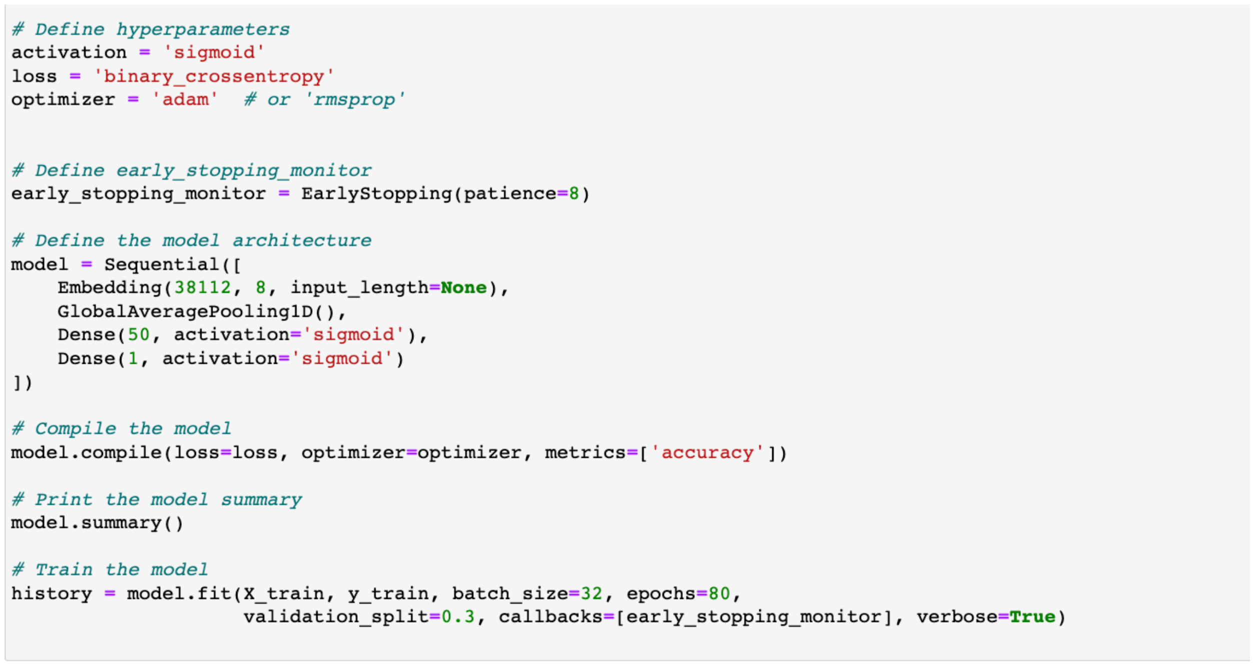
E. Stopping criteria: The stopping criteria define when to stop the training process to prevent overfitting and increase optimal performance. Early stopping monitors the validation accuracy score and stops training early in the event validation accuracy does not increase after 7 successive epochs (Brownlee, 2018). The patience setting used in this model is 8. Numbers from 1 through 10 were tested with a value of 8 chosen to give a suitable time to run the model to train while stopping it before overfitting. A large number of epochs was selected to ensure the model trains appropriately. 200 was chosen through trial and error. Anything larger could have been a sign of overfitting.

F. Evaluation Metric: Accuracy was used as a metric to evaluate how well the neural network could classify comments based on sentiment within the dataset.

Part IV: Model Evaluation

D1. Stopping Criteria

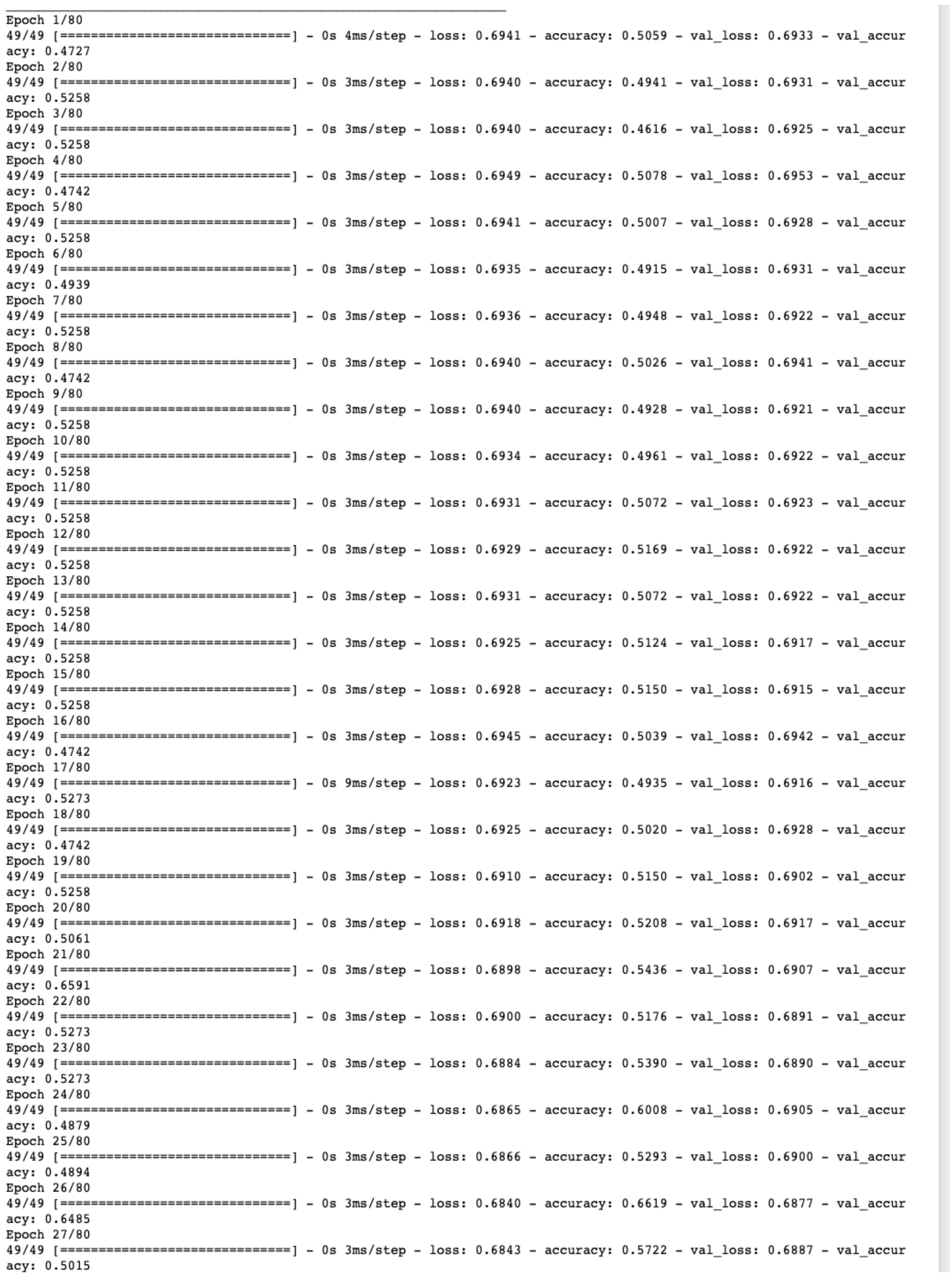
The stopping criteria allows the training process to terminate automatically. It will stop when specific conditions are met. Early stopping monitors the validation accuracy score and stops training early in the event validation accuracy does not increase after 7 successive epochs (Brownlee, 2021). This prevents overfitting. The model was set to 80 as the maximum number of epoch. Based on the stop criteria the model completed training by epoch 49.



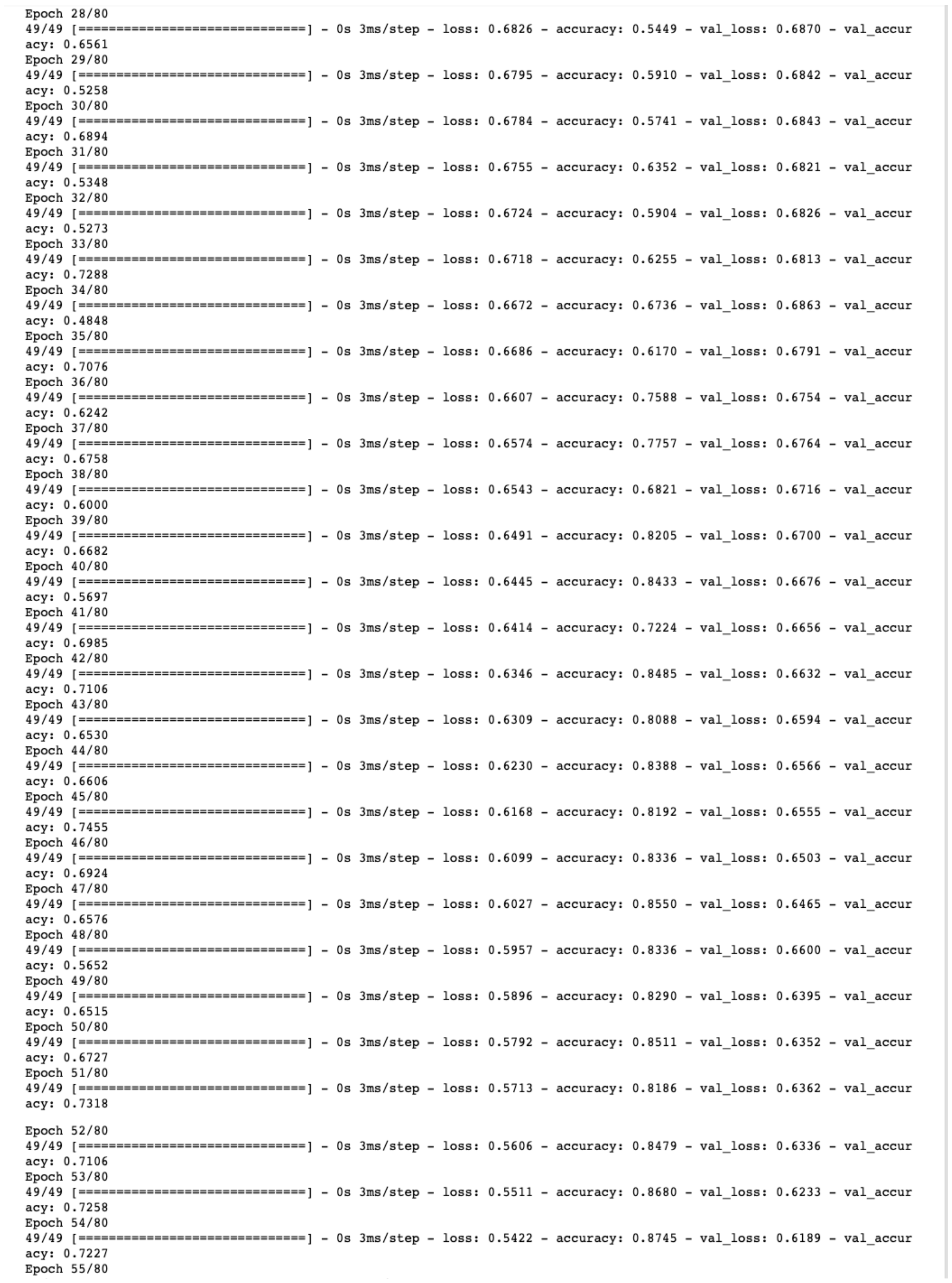
```
# Define hyperparameters
activation = 'sigmoid'
loss = 'binary_crossentropy'
optimizer = 'adam'  # or 'rmsprop'


# Define early_stopping_monitor
early_stopping_monitor = EarlyStopping(patience=8)

# Define the model architecture
model = Sequential([
    Embedding(38112, 8, input_length=None),
    GlobalAveragePooling1D(),
    Dense(50, activation='sigmoid'),
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(loss=loss, optimizer=optimizer, metrics=['accuracy'])

# Print the model summary
model.summary()

# Train the model
history = model.fit(X_train, y_train, batch_size=32, epochs=80,
                    validation_split=0.3, callbacks=[early_stopping_monitor], verbose=True)
```

D2. Training Process

Model's training process:

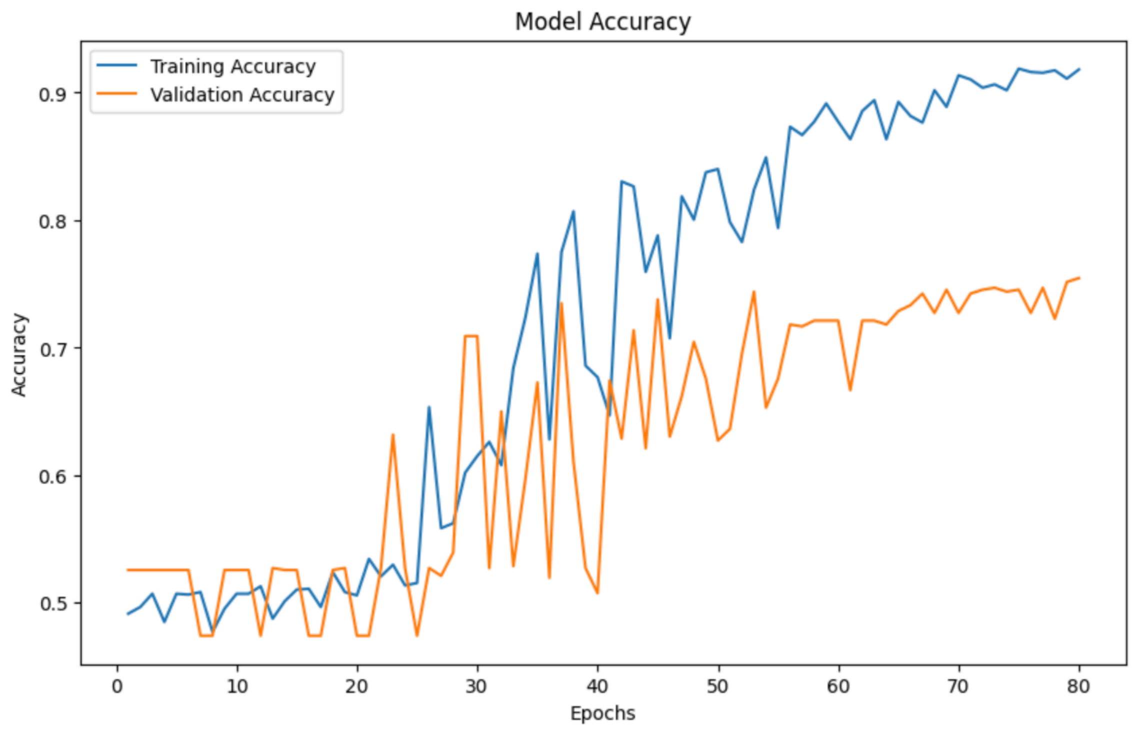


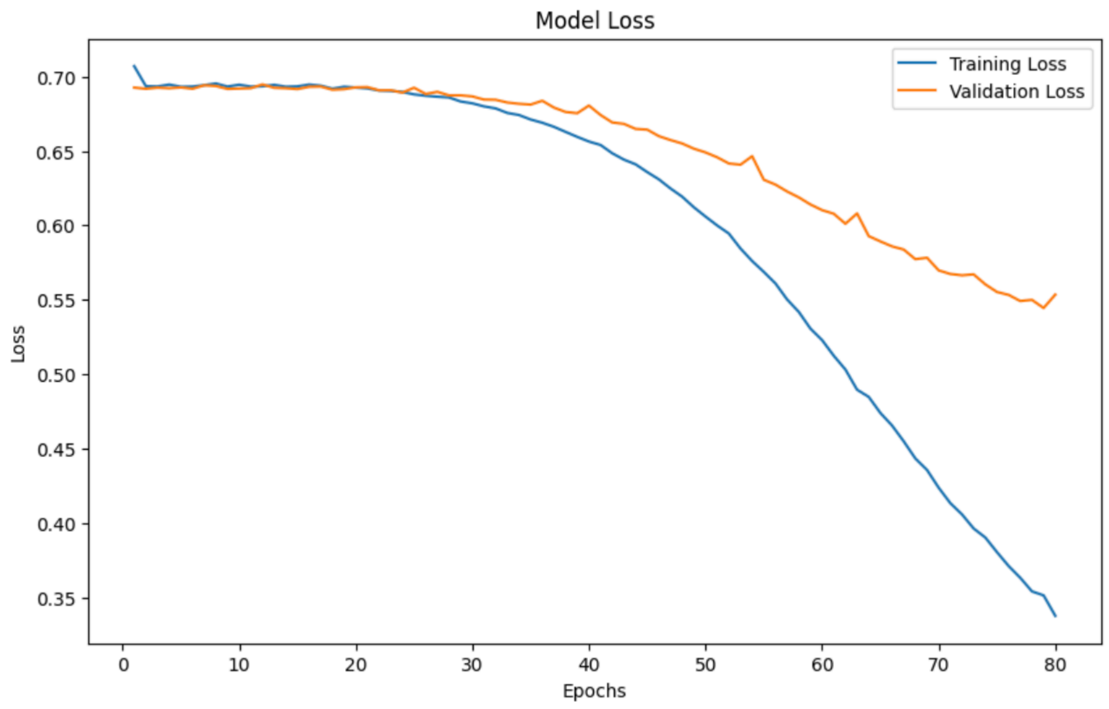
```
Epoch 1/80
49/49 [==============================] - 0s 4ms/step - loss: 0.6941 - accuracy: 0.5059 - val_loss: 0.6933 - val_accur
acy: 0.4727
Epoch 2/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6940 - accuracy: 0.4941 - val_loss: 0.6931 - val_accur
acy: 0.5258
Epoch 3/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6940 - accuracy: 0.4616 - val_loss: 0.6925 - val_accur
acy: 0.5258
Epoch 4/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6949 - accuracy: 0.5078 - val_loss: 0.6953 - val_accur
acy: 0.4742
Epoch 5/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6941 - accuracy: 0.5007 - val_loss: 0.6928 - val_accur
acy: 0.5258
Epoch 6/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6935 - accuracy: 0.4915 - val_loss: 0.6931 - val_accur
acy: 0.4939
Epoch 7/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6936 - accuracy: 0.4948 - val_loss: 0.6922 - val_accur
acy: 0.5258
Epoch 8/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6940 - accuracy: 0.5026 - val_loss: 0.6941 - val_accur
acy: 0.4742
Epoch 9/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6940 - accuracy: 0.4928 - val_loss: 0.6921 - val_accur
acy: 0.5258
Epoch 10/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.4961 - val_loss: 0.6922 - val_accur
acy: 0.5258
Epoch 11/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5072 - val_loss: 0.6923 - val_accur
acy: 0.5258
Epoch 12/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.5169 - val_loss: 0.6922 - val_accur
acy: 0.5258
Epoch 13/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5072 - val_loss: 0.6922 - val_accur
acy: 0.5258
Epoch 14/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6925 - accuracy: 0.5124 - val_loss: 0.6917 - val_accur
acy: 0.5258
Epoch 15/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.5150 - val_loss: 0.6915 - val_accur
acy: 0.5258
Epoch 16/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6945 - accuracy: 0.5039 - val_loss: 0.6942 - val_accur
acy: 0.4742
Epoch 17/80
49/49 [==============================] - 0s 9ms/step - loss: 0.6923 - accuracy: 0.4935 - val_loss: 0.6916 - val_accur
acy: 0.5273
Epoch 18/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6925 - accuracy: 0.5020 - val_loss: 0.6928 - val_accur
acy: 0.4742
Epoch 19/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6910 - accuracy: 0.5150 - val_loss: 0.6902 - val_accur
acy: 0.5258
Epoch 20/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6918 - accuracy: 0.5208 - val_loss: 0.6917 - val_accur
acy: 0.5061
Epoch 21/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6898 - accuracy: 0.5436 - val_loss: 0.6907 - val_accur
acy: 0.6591
Epoch 22/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6900 - accuracy: 0.5176 - val_loss: 0.6891 - val_accur
acy: 0.5273
Epoch 23/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6884 - accuracy: 0.5390 - val_loss: 0.6890 - val_accur
acy: 0.5273
Epoch 24/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6865 - accuracy: 0.6008 - val_loss: 0.6905 - val_accur
acy: 0.4879
Epoch 25/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6866 - accuracy: 0.5293 - val_loss: 0.6900 - val_accur
acy: 0.4894
Epoch 26/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6840 - accuracy: 0.6619 - val_loss: 0.6877 - val_accur
acy: 0.6485
Epoch 27/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6843 - accuracy: 0.5722 - val_loss: 0.6887 - val_accur
acy: 0.5015
```

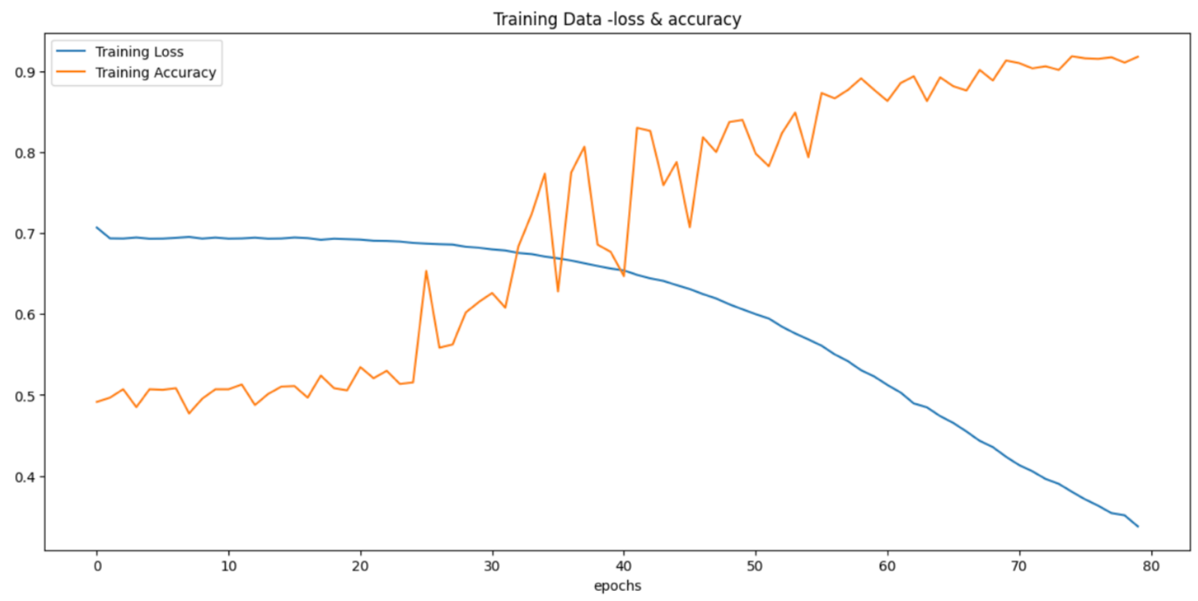


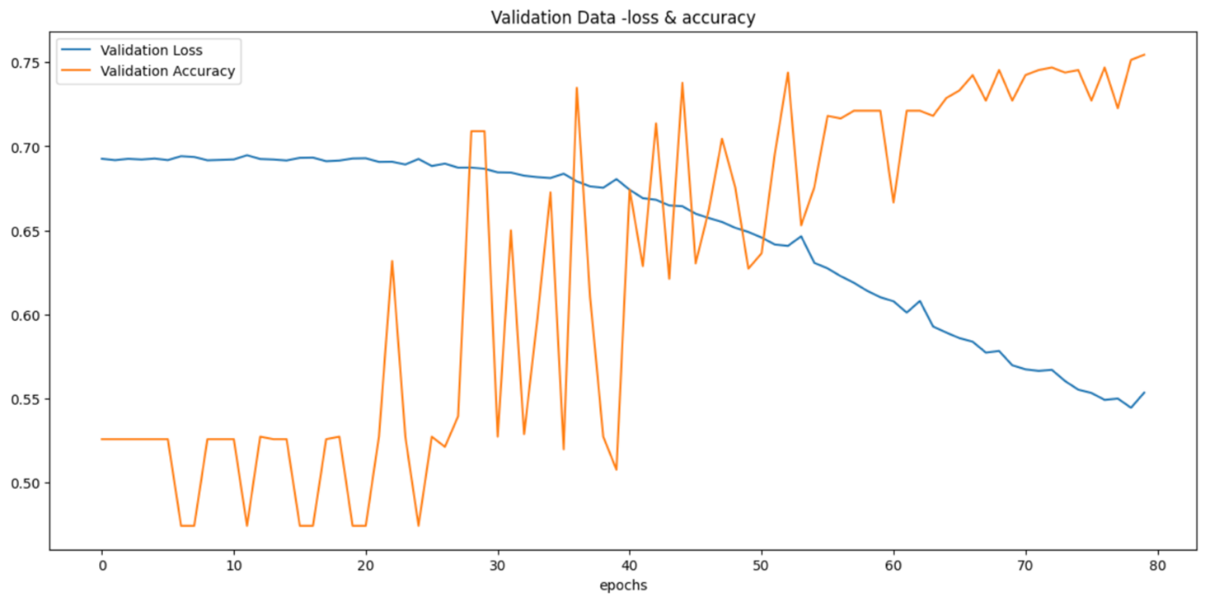
```
Epoch 28/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6826 - accuracy: 0.5449 - val_loss: 0.6870 - val_accur
acy: 0.6561
Epoch 29/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6795 - accuracy: 0.5910 - val_loss: 0.6842 - val_accur
acy: 0.5258
Epoch 30/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6784 - accuracy: 0.5741 - val_loss: 0.6843 - val_accur
acy: 0.6894
Epoch 31/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6755 - accuracy: 0.6352 - val_loss: 0.6821 - val_accur
acy: 0.5348
Epoch 32/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6724 - accuracy: 0.5904 - val_loss: 0.6826 - val_accur
acy: 0.5273
Epoch 33/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6718 - accuracy: 0.6255 - val_loss: 0.6813 - val_accur
acy: 0.7288
Epoch 34/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6672 - accuracy: 0.6736 - val_loss: 0.6863 - val_accur
acy: 0.4848
Epoch 35/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6686 - accuracy: 0.6170 - val_loss: 0.6791 - val_accur
acy: 0.7076
Epoch 36/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6607 - accuracy: 0.7588 - val_loss: 0.6754 - val_accur
acy: 0.6242
Epoch 37/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6574 - accuracy: 0.7757 - val_loss: 0.6764 - val_accur
acy: 0.6758
Epoch 38/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6543 - accuracy: 0.6821 - val_loss: 0.6716 - val_accur
acy: 0.6000
Epoch 39/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6491 - accuracy: 0.8205 - val_loss: 0.6700 - val_accur
acy: 0.6682
Epoch 40/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6445 - accuracy: 0.8433 - val_loss: 0.6676 - val_accur
acy: 0.5697
Epoch 41/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6414 - accuracy: 0.7224 - val_loss: 0.6656 - val_accur
acy: 0.6985
Epoch 42/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6346 - accuracy: 0.8485 - val_loss: 0.6632 - val_accur
acy: 0.7106
Epoch 43/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6309 - accuracy: 0.8088 - val_loss: 0.6594 - val_accur
acy: 0.6530
Epoch 44/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6230 - accuracy: 0.8388 - val_loss: 0.6566 - val_accur
acy: 0.6606
Epoch 45/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6168 - accuracy: 0.8192 - val_loss: 0.6555 - val_accur
acy: 0.7455
Epoch 46/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6099 - accuracy: 0.8336 - val_loss: 0.6503 - val_accur
acy: 0.6924
Epoch 47/80
49/49 [==============================] - 0s 3ms/step - loss: 0.6027 - accuracy: 0.8550 - val_loss: 0.6465 - val_accur
acy: 0.6576
Epoch 48/80
49/49 [==============================] - 0s 3ms/step - loss: 0.5957 - accuracy: 0.8336 - val_loss: 0.6600 - val_accur
acy: 0.5652
Epoch 49/80
49/49 [==============================] - 0s 3ms/step - loss: 0.5896 - accuracy: 0.8290 - val_loss: 0.6395 - val_accur
acy: 0.6515
Epoch 50/80
49/49 [==============================] - 0s 3ms/step - loss: 0.5792 - accuracy: 0.8511 - val_loss: 0.6352 - val_accur
acy: 0.6727
Epoch 51/80
49/49 [==============================] - 0s 3ms/step - loss: 0.5713 - accuracy: 0.8186 - val_loss: 0.6362 - val_accur
acy: 0.7318

Epoch 52/80
49/49 [==============================] - 0s 3ms/step - loss: 0.5606 - accuracy: 0.8479 - val_loss: 0.6336 - val_accur
acy: 0.7106
Epoch 53/80
49/49 [==============================] - 0s 3ms/step - loss: 0.5511 - accuracy: 0.8680 - val_loss: 0.6233 - val_accur
acy: 0.7258
Epoch 54/80
49/49 [==============================] - 0s 3ms/step - loss: 0.5422 - accuracy: 0.8745 - val_loss: 0.6189 - val_accur
acy: 0.7227
Epoch 55/80
```

Line Graphs:


Model Accuracy
Training Accuracy
Validation Accuracy
Accuracy
0.5
0.6
0.7
0.8
0.9
0
10
20
30
40
50
60
70
80
Epochs


Model Loss
Training Loss
Validation Loss
Loss
0.35
0.40
0.45
0.50
0.55
0.60
0.65
0.70
0
10
20
30
40
50
60
70
80
Epochs





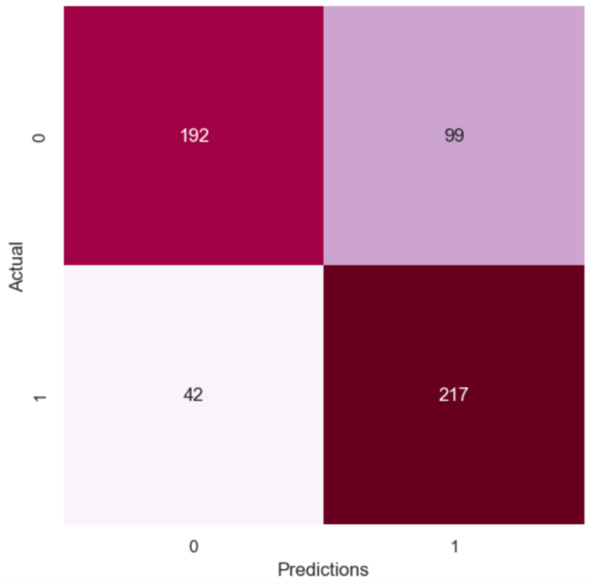
D3. Fit

Overfitting occurs when a model learns the training dataset too well. This is seen when the model performs well on the training dataset but not on the testing dataset. Thus, utilizing both training and test datasets can prevent overfitting. The model appears to be a good fit for the data. The stop criterion was utilized to prevent overfitting. Early stopping stopped the training process when the model's performance on the validation dataset started to degrade and no longer show significant improvements. When examining the model for overfitting, trends in the loss values against the validation data loss values were examined. The loss values continued to drop. If overfitting would have occurred, the validation data loss values would have diverged from the training loss values and begun to increase. Another way to avoid overfitting was to first start with a smaller network and then increase the capacity of the network by adding more layers and nodes to the layers. The model can be further improved by providing more data for training and optimizing the network further (Brownlee, 2018).

D4. Predictive Accuracy

The predictive accuracy of the model was 74.36%. This indicates that model is able to correctly classify approximately 74.36% of the training examples. The trained network has a final model loss of 0.53121 which suggest that the model has achieved a moderate level of accuracy in predicting the labels in the training dataset. For each response option, the model accurately predicted a positive review (1) in 217 out of 259 (84%) reviews. It predicted negative reviews (0) accurately in 192 out of 291 (66%) reviews.

Heatmap:



Part V. Summary and Recommendations

E. Code

Code used to save the trained network: model.save('D213_trained_model.h5')

F. Functionality

The neural network was trained using the customer reviews and their actual sentiments recorded as labels. The neural network successfully trained the 2198 text reviews and their score labels to create a model to predict future scores. This model can be used to predict customer reviews as either positive or negative to aid in business strategies. This will allow the business to address problem areas through customer satisfaction through the prediction of positive and negative reviews. The network architecture of the model is used for sentiment analysis. The RNN model architecture is suited for text classification making it well suited for classification of reviews. The layers were structured with considerations for binary score output that served as positive and negative reviews.

G. Recommendations

The predictive accuracy indicates this to be an acceptable neural network model for predicting customer sentiment. Thus, the model is recommended for use in data collection, reporting, and metrics of customer sentiment for business use in strategizing business needs based on customer feedback.

Part VI: Reporting

H. Reporting

An HTML file is attached to document the executed Jupyter Notebook presentation.

I. Sources for Third Party Code

Becker, D., Hull, I., & Halder, B. (n.d.). Advanced Data Analytics. DataCamp. Retrieved July 2022, from https://app.datacamp.com/learn/custom-tracks/custom-advanced-data-analytics

Chen, B. (2023, February 9). Emojis Aid Social Media Sentiment Analysis: Stop Cleaning Them Out! *Medium*. https://towardsdatascience.com/emojis-aid-social-media-sentiment-analysis-stop-cleaning-them-out-bb32a1e5fc8e

Team, K. (n.d.). *Keras documentation: The Model class*. https://keras.io/api/models/model/

J. Sources

ActiveState. (2022, July 11). *What is a Keras model and how to use it to make predictions- ActiveState*. https://www.activestate.com/resources/quick-reads/what-is-a-keras-model/

Brownlee, Jason. 2018. Use Early Stopping to Halt the Training of Neural Networks At the Right Time. https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

Brownlee, J. (2019b). How to Visualize a Deep Learning Neural Network Model in Keras. *MachineLearningMastery.com*. https://machinelearningmastery.com/visualize-deep-learning-neural-network-model-keras/

Brownlee, J. (2020a). What is the Difference Between Test and Validation Datasets? *MachineLearningMastery.com*. https://machinelearningmastery.com/difference-test-validation-datasets/

Brownlee, J (2021). How to Use Word Embedding Layers for Deep Learning with Keras. *MachineLearningMastery.com*. https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

Bushaev, V. (2018, October 24). Adam — latest trends in deep learning optimization. *Medium*. https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c

Mahendra, S. (2023). What is the Adam Optimizer and How is It Used in Machine Learning. *Artificial Intelligence +*. https://www.aiplusinfo.com/blog/what-is-the-adam-optimizer-and-how-is-it-used-in-machine-learning/#:~:text=problem%20being%20solved.-,Overall%2C%20the%20Adam%20optimizer%20is%20a%20powerful%20tool%20for%20improving,the%20cost%20or%20loss%20function

Saxena, S. (2021). Tokenization and Text Normalization. *Analytics Vidhya*. https://www.analyticsvidhya.com/blog/2021/03/tokenization-and-text-normalization/

Sigmoid Function. (n.d.). https://www.learndatasci.com/glossary/sigmoid-function/