# Advanced Analytics

Task 2

Nathan Hefner

**A. Describe the purpose of this data analysis by doing the following:**

**1. Summarize one research question that you will answer using a neural network model and NLP techniques.**

One research question we will answer is how accurately a neural network model using NLP sentiment analysis can predict IMDb movie ratings from user-written reviews and how this can optimize future movie ratings.

**2. Define the objectives or goals of the data analysis.**

The goals of this data analysis will be to train the model to be able to differentiate between positive and negative text-based reviews so that the model can accurately and consistently categorize future reviews, whether they're positive or negative.

**3. Identify an industry-relevant type of neural network capable of performing a text classification task that can be trained to produce useful predictions on text sequences on the selected dataset.**

The Neural Network I will be using is Bidirectional LSTM because it is designed to process text sequences while remembering long strings of words, which is useful for longer reviews. It does this by processing text in both directions to capture the full meaning of a sentence and not have any memory leaks. This neural network is an industry standard, with tech giants like Amazon and Netflix both using it; therefore, it will suffice for our use case.

**B. Summarize the data cleaning process by doing the following:**

**1. Perform exploratory data analysis on the chosen dataset, and include an explanation of each of the following elements:**

- **presence of unusual characters (e.g., emojis, non-English characters)**

The first step in cleaning the data is to ensure all of the reviews are readable and to ensure unusual characters don't break the model's training. The presence of these characters also adds noise to the entirety of the data set, which could lead to unwanted problems, biased results, and longer computing times. For this data set, I checked for characters outside the English alphabet, emojis, and HTML Tags. Any unusual characters I found I removed along with punctuation and stopwords, as they are just noise for the actual model and don't carry a significant amount of sentiment information.

```
def find_unusual_characters(text):
    return re.findall(r'[^\x00-\x7F]+', text)
all_unusual_chars = data['sentence'].apply(find_unusual_characters).sum()
unusual_char_counts = Counter(all_unusual_chars)
print(unusual_char_counts)
```

```
Counter({'\x96': 5, 'é': 4, '\x85': 2, 'å': 1, '\x97': 1})
```

```
def clean_text(text):
    text = re.sub(r'[^\x00-\x7F]+', ' ', text)
    text = text.lower()
    text = text.translate(str.maketrans('','', string.punctuation))
    stop_words = set(stopwords.words('english'))
    text = ' '.join([word for word in text.split() if word not in stop_words])
    text = re.sub(r'\s+', ' ', text).strip()
    return text
data['cleaned_sentence'] = data['sentence'].apply(clean_text)
```

```
all_unusual_chars_cleaned = data['cleaned_sentence'].apply(find_unusual_characters).sum()
unusual_char_counts_cleaned = Counter(all_unusual_chars_cleaned)

print(unusual_char_counts_cleaned)
```

```
Counter()
```

- **vocabulary size**

The vocabulary size is the number of unique words within the dataset. The words then become tokenized for processing. Knowing this amount is useful for processing as it contributes to the model complexity, by having more words means more memory and computational power needed, and to generalization and overfitting as a huge vocabulary can lead to overfitting while a small vocabulary can lead to generalization. Our dataset has 3052 vocabulary words, which is a strong midpoint.

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(data['cleaned_sentence'])
vocab_size = len(tokenizer.word_index)
print("Vocabulary size:", vocab_size)
```

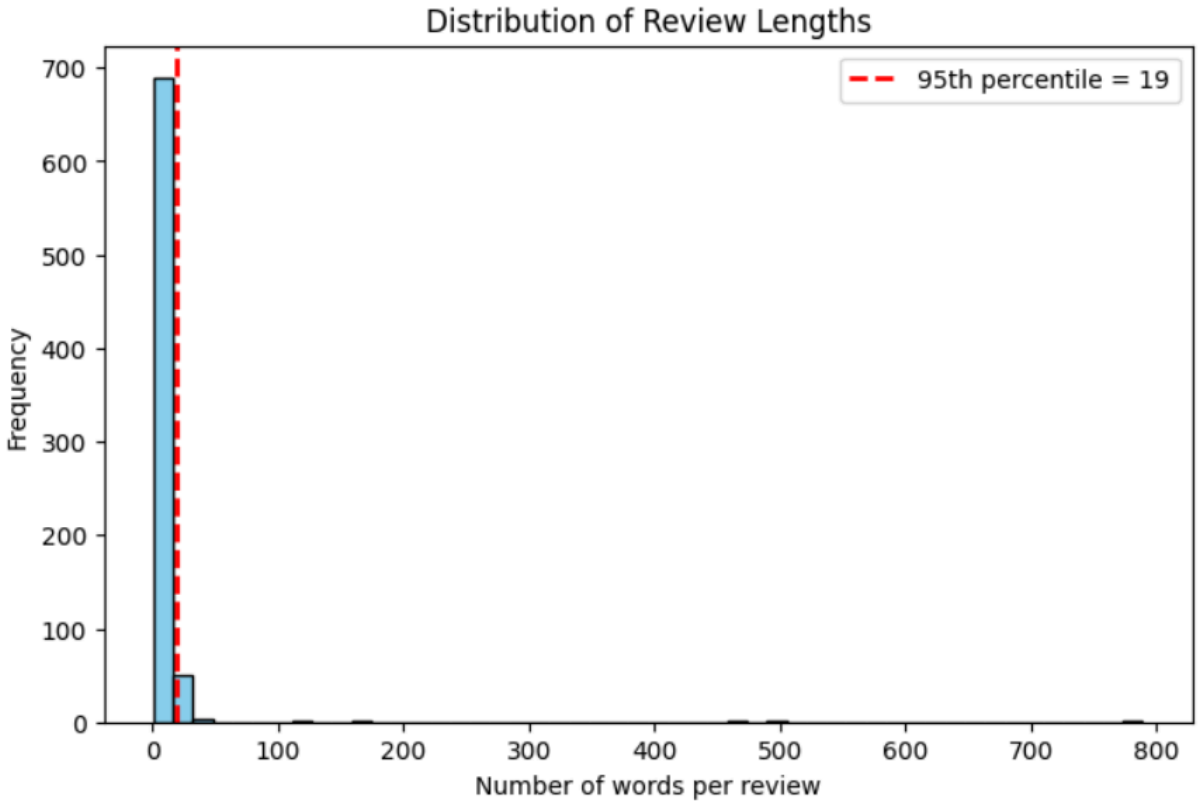
```
Vocabulary size: 3052
```

- **word embedding length**

Embedding length can be different for every model, but it is very important as it is a hyperparameter for our neural network. Since ours is in basic training and only uses the English language, the optimal range is 85-110, but the lower the length, the better the performance, so I will be using 85 for my embedding length (Finding the Optimal Number of Dimensions for Word Embeddings).

- **statistical justification for the chosen maximum sequence length**

Maximum sequence length is another hyperparameter to train the model. It counts the number of words per review and either cuts out words to meet the maximum or pads the review to meet the correct number count. We do this because the neural network needs all inputs to have the same shape. To pick an optimal amount instead of picking randomly, we will calculate the 95th percentile of the number of words per review. This will give us a good baseline because a too short sequence length may lose valuable data, while too long will add more computational power with little to no benefit.




The 95th percentile is 19 words, shown by the vast majority of the reviews to have under that amount, with some extreme outliers. I will pick 20 as the maximum sequence length for a clean, even amount.

**2. Describe the goals of the tokenization process**

Tokenization is the process of breaking text into smaller and more manageable units called tokens. The main goals of this process are to convert the raw text into a numeric format since neural networks work with numbers, not text. It also normalizes the input by changing all characters to lowercase, removing punctuation, and can even split contractions, to help reduce noise and vocabulary amount.

**3. Explain the padding process used to standardize the length of sequences.**

The padding process is an essential augmentation to the inputs for the model because all inputs need to be the same shape. So if the data doesn't reach that size, it will be padded until it does,o and on the flip side, if it is over the designated size, the data will be truncated, all to ensure the same consistent size. Using the standard Keras code, we will be post-padding, adding 0s at the end, to keep the original word order unchanged.

**4. Identify how many categories of sentiment will be used and provide an activation function for the final dense layer of the network.**

For my dataset, we will be separating IMDb movie reviews into positive or negative reviews, thus having two categories of sentiment and therefore being a binary classification. The activation function we will be using is the sigmoid activation because it is extremely compatible with binary outputs, as the sigmoid activation's outputs are a value between 0 and 1, with most of them being very near 1 or 0, thus allowing some differentiation (Sigmoid activation function).

**5. Explain the steps used to prepare the data for analysis, including the size of the training, validation, and test set split based on the industry average.**

For the training, testing, and validation split, I did a 70%, 15%, and 15% split because the training needs the most data for learning, and the validation and test data sets can be evenly split. I decided on these splits because if there was less training data, the model would show high variance in training, but with less testing/validation data, the model's performance would have greater variance. So this specific split is the most optimal to avoid both of these problems (Train test validation split: How to & best practices).

**C. Describe the type of neural network model used by doing the following:**

**1. Provide the output of the model summary of the function from TensorFlow or PyTorch.**

```
Model: "sequential_2"
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding_2 (Embedding) | (None, 20, 100) | 305,300 |
| spatial_dropout1d (SpatialDropout1D) | (None, 20, 100) | 0 |
| bidirectional_1 (Bidirectional) | (None, 256) | 234,496 |
| dense_2 (Dense) | (None, 64) | 16,448 |
| dropout_1 (Dropout) | (None, 64) | 0 |
| dense_3 (Dense) | (None, 1) | 65 |

```
Total params: 1,668,929 (6.37 MB)

Trainable params: 556,309 (2.12 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 1,112,620 (4.24 MB)
```

**2. Discuss the number of layers, the type of layers, and the total number of parameters.**

My model has a total of 6 layers:

**Embedding Layer**

This layer has the highest parameters, a total being 305,300, because of a calculation between the vocabulary size and the embedding dimensions. This layer learns a 100-dimensional vector representation for every word in the vocabulary.

**SpatialDropout1D Layer**

This layer is a regularization layer that helps reduce overfitting by randomly dropping entire embedding dimensions.

**Bidirectional LSTM Layer**

This layer is an important piece of the infrastructure of the model because it processes input sequences in both directions. By doing so, it learns to capture context from both past and

future tokens. The parameter count is on the higher end, 234,496 parameters, because the cells have multiple gates (input, output, forget, and cell state).

**Dense Layer**

This layer transforms the data and connects each data point to each of the 64 neurons. Using the 64 neurons and a simple calculation, we get to 16,448 parameters. This layer is fully connected with ReLU activation to combine the features extracted by the LSTM

**Dropout Layer**

This layer also helps with overfitting by randomly dropping 40% of the neurons' outputs to help improve generalization.

**Output Dense Layer**

This layer is the final prediction layer using sigmoid activation for binary classification. This layer has 1 unit and 65 parameters.

**3. Justify the choice of hyperparameters, including each of the following elements:**

**Activation Functions**

For the Dense Layer, we chose ReLU because it introduces non-linearity and helps the network learn complex relationships between features extracted by the LSTM; also, ReLU is efficient computationally and helps mitigate the vanishing gradient.

For the Output Layer, we chose Sigmoid because it is the ideal binary classifier, as it will output a value between 0 and 1 and is still differentiable.

**Number of Nodes per Layer**

For the Embedding Dimension, we have 100 nodes because a 100-dimensional embedding is an optimal and balanced amount between computational cost and training power. Larger ones may overfit the data, and smaller ones may lose semantic meaning.

LSTM Units have 128 nodes and are bidirectional, meaning they provide enough capacity to learn long-term dependencies in text, and by using a bidirectional LSTM, it doubles the context captured by reading forwards and backwards, which will improve sentiment analysis.

Our Dense Layer has 64 nodes; this layer acts like a feature combiner by allowing the model to learn non-linear interactions between features extracted by the LSTM.

**Loss Function**

The standard choice for binary classification problems is the Binary Cross-Entropy. It measures the difference between predicted probabilities and binary labels, a perfect choice for our model, as the output is binary.

**Optimizer**

The optimizer we chose was the Adam Optimizer because it combines AdaGrad, for handling sparse data, and RMSProp, for non-stationary objectives. It automatically adapts the learning rate for each parameter, which then leads to faster convergence and requires less manual tuning. It also helps with the gradient problem that can vary significantly across parameters.

**Stopping Criteria**

Our stopping criteria was EarlyStopping with a patience value of 3. This means the model will stop training when validation loss stops improving for three consecutive epochs, helping to prevent overfitting. To further improve generalization, we set restore_best_weights to true to ensure the model returns the best-performing epoch rather than the last trained epoch.

**D. Evaluate the model's training process and its relevant outcomes by doing the following:**

**1. Discuss the impact of using stopping criteria to include defining the number of epochs**

Using stopping criteria helps keep the integrity of the model by monitoring a validation metric and stopping the model before it starts memorizing training data noise. It does so because we set the patience value to three, which means the model stops after 3 consecutive losses in accuracy of the validation set. The optimal number of epochs for us is 4 based of our training and validation loss graph, because that is the point when the line stops decreasing and instead starts increasing.

```
Epoch 1/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 3s 84ms/step - accuracy: 0.5067 - loss: 0.6937 - val_accuracy: 0.5089 - val_loss: 0.6923 - learning_rate: 0.0010
Epoch 2/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.5793 - loss: 0.6881 - val_accuracy: 0.5000 - val_loss: 0.6911 - learning_rate: 0.0010
Epoch 3/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 0.6367 - loss: 0.6751 - val_accuracy: 0.5804 - val_loss: 0.6783 - learning_rate: 0.0010
Epoch 4/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 48ms/step - accuracy: 0.7648 - loss: 0.6140 - val_accuracy: 0.6161 - val_loss: 0.6590 - learning_rate: 0.0010
Epoch 5/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 48ms/step - accuracy: 0.8317 - loss: 0.4402 - val_accuracy: 0.6964 - val_loss: 0.5694 - learning_rate: 0.0010
Epoch 6/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - accuracy: 0.8987 - loss: 0.2561 - val_accuracy: 0.6964 - val_loss: 0.6799 - learning_rate: 0.0010
Epoch 7/20
7/9 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.9066 - loss: 0.2413
Epoch 7: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9197 - loss: 0.2160 - val_accuracy: 0.7411 - val_loss: 0.5718 - learning_rate: 0.0010
Epoch 8/20
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.9656 - loss: 0.1143 - val_accuracy: 0.7143 - val_loss: 0.6930 - learning_rate: 5.0000e-04
```

Here is a screenshot of the epochs training.

**2. Assess the fitness of the model and any actions taken to address overfitting or underfitting.**

```
y_pred = (model.predict(X_test) > 0.5).astype("int32")

f1 = f1_score(y_test, y_pred)
print(f"F1 Score: {f1:.4f}")
```

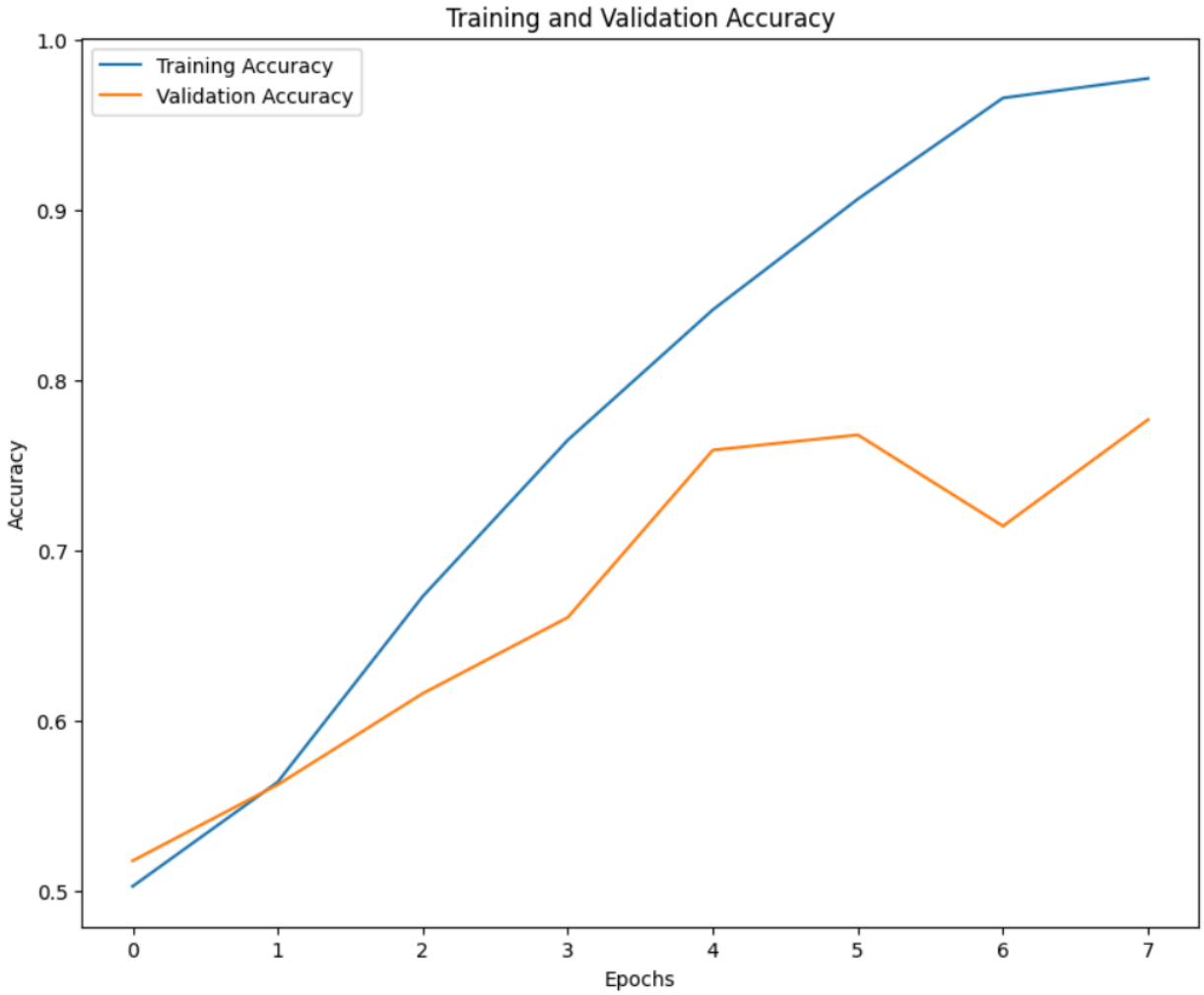
```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step
F1 Score: 0.8030
```

Having an F1 score above 0.8 means the model is accurately predicting the connotation of reviews the strong majority of the time. We achieved this result by the precise action of balancing both underfitting and overfitting.

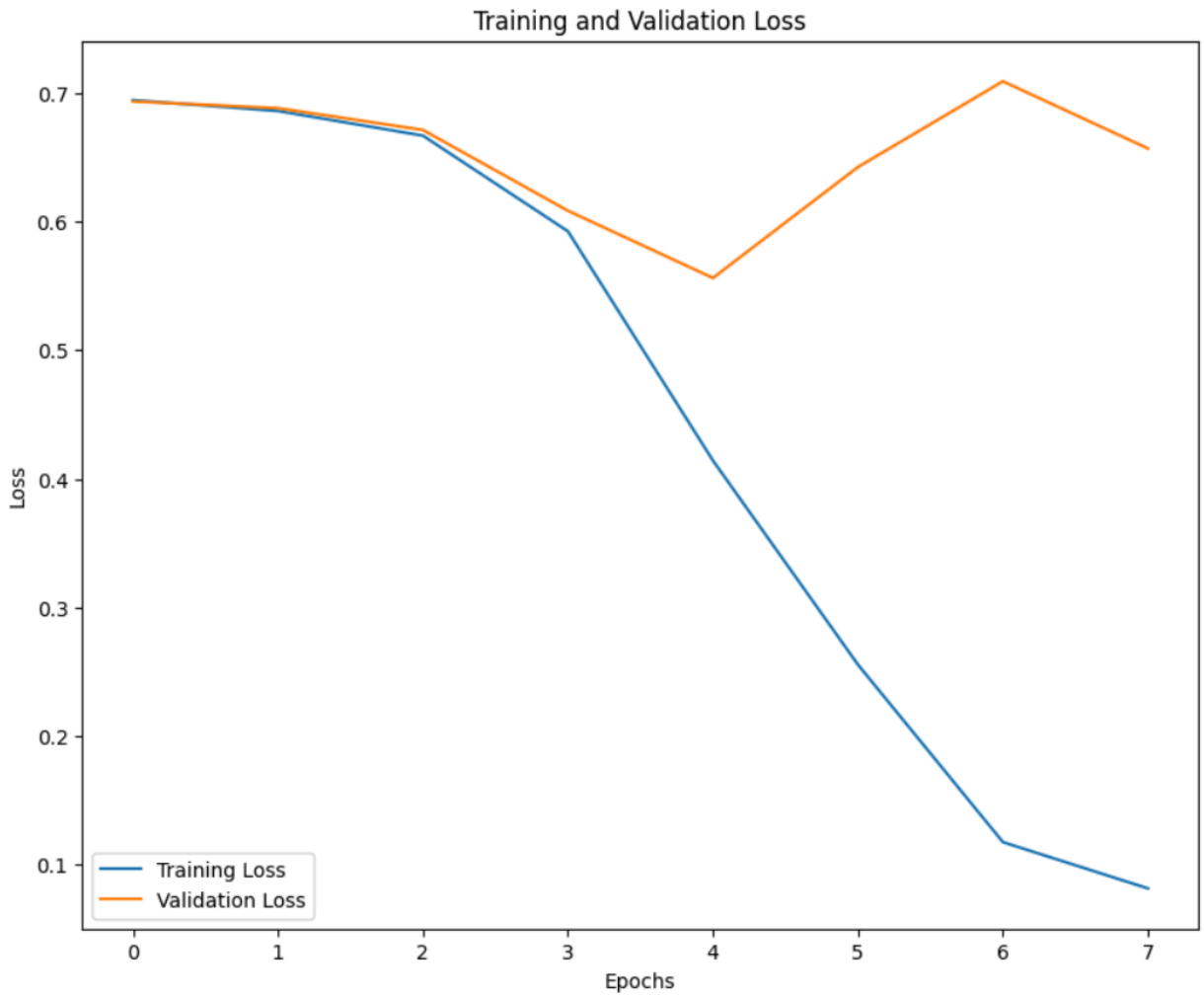
For overfitting, we prevented this by having early stopping, which stops training if validation loss stops improving for 3 epochs, and by having a dropout layer, which randomly drops 50% of neurons during training.

For underfitting, we prevented this by implementing an LSTM and embedding layer, which captures long-term dependencies in text and is bidirectional, allowing to see context from left to right and vice versa, and learns meaningful vector representations of words, improving the model's ability to understand semantic meanings, respectively.

**3. Provide clearly labeled visualizations of the model's training process and show the loss and accuracy metric.**




This graph shows the accuracy metric. The training data set continues to increase and the validation stops increasing after 5 epochs. Showing the importance of early stopping.




This graph shows the training and validation loss metric. Preferably, you would want both these values to decrease, but the validation set stops decreasing at 4 epochs and instead increases once again showing the importance of early stopping.

**4. Discuss the predictive accuracy of the trained model using the chosen evaluation metric from part D3.**

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy:.4f}")
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7699 - loss: 0.4726
Test Accuracy: 0.7699
```

With an accuracy metric just under 77% it is a highly accurate model that is more right than wrong, making its predictions reliable. It also means that it is unlikely the model is underfitting or overfitting.

**5. Explain how the analysis complies with artificial intelligence (AI) global ethical standards and mitigates bias.**

This analysis follows several widely recognized AL ethics principles, notably from the OECD and the EU AI Act. We followed these ethics by keeping transparency and non-discrimination, because the data preparation and training are clearly documented, and the dataset has a balanced spread of positive and negative reviews, keeping both sides proportional. A huge reason as well is that the data does not include any PII (personable identifiable information) since the nature of the dataset is anonymized movie reviews, there's no confidential data that needs to be handled carefully, so my data analysis follows all ethical guidelines.

**E. Provide the code you used to save the trained model within the neural network.**

```
model.save(r"C:\Users\Nathan\Documents\WGU\D604\Task 2\sentiment_analysis_model.keras")
```

**F. Discuss the functionality of your model, including the impact of your choice of network architecture.**

The model performs and functions very well, as the accuracy and F1 score are both high and well above an acceptable threshold. This is in part because of the network architecture we chose. The neural network we chose, Bidirectional LSTM, was an exceptional choice for its utility in capturing the context of a sentence from both left to right and vice versa, which is crucial for sentiment detection. Along with the sigmoid activation function, which was crucial for binary output computations. This model not only learned and can predict well it is also computationally efficient and optimal.

**G. Recommend a course of action based on your results as they relate to the research question.**

I would recommend implementing this model to help auto-sort the positive and negative movie reviews, as it is highly robust with a high accuracy score. Manually sorting may be necessary for the few reviews that are misplaced. If you want the model to be even more accurate, I would suggest a few modifications to achieve this goal. The standard first step is to gather more data; with more data, the model has more of a baseline to learn and grow off of. Next, I would increase the embedding dimension because more expressive word vectors can help, along with increasing the LSTM Units to capture dependencies. As of now, our model is great at generalizing and not over- or underfitting, but if this does become a problem because of the aforementioned tweaks, then you can change the batch size; smaller sizes may improve generalization, and lowering the dropout if the model starts to underfit.

**Code Sources**

https://www.tensorflow.org/guide/keras/working_with_rnns

**Sources**


Kwan, Matthew. “Finding the Optimal Number of Dimensions for Word Embeddings.” *Medium*, Medium, 6 Sept. 2023, medium.com/@matti.kwan/finding-the-optimal-number-of-dimensions-for-word-embeddings-f19f71666723.

“Sigmoid Activation Function: An Introduction.” *Built In*, builtin.com/machine-learning/sigmoid-activation-function. Accessed 20 Sept. 2025.

“Train Test Validation Split: How to & Best Practices [2024].” *V7*, www.v7labs.com/blog/train-validation-test-set. Accessed 9 Sept. 2025.