Can a Neural Network Model Effectively Classify the Sentiment of Reviews?

By: Kevin Sandoval

Part I: Research Question

A1.

The research question I am choosing is “Can a Neural Network Model Effectively Classify The Sentiment of Reviews?” This could have many real-world applications, as sentiment analysis could save time in seeing the performance of a restaurant, business, hospitals or any establishment that has reviews.

A2.

The goal(s) of this analysis is to be able to effectively classify different reviews as positive (1) or negative (0). The model will train on a variety of different reviews from amazon and then must predict on other amazon reviews. The results will be graded, and the model will be evaluated.

A3.

The type of neural network that can perform the classification task for the image files I chose will be a Recurrent neural network (RNN) as it is very efficient at sentiment analysis.

Part II: Data Preparation

B1a.

To check for any unusual characters, I decided that the easiest way would be to use the ASCII table and put in a range for the corresponding values. If the character falls out of those values in the ASCII table, then that means it is an ‘unusual’ character, and I replaced it with a space.

A computer screen shot of a program code

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

From the screenshots, we can see that initially there were quite a few unusual characters. After creating a function to address them, we can see that the final output is an empty list, thus there are no more unusual characters.

B1b.

The vocabulary size was found in the code snippet below. I included the total amount of words after dropping the stop words, as well as the number of unique words, which is what will be used in the analysis.

A screenshot of a computer program

AI-generated content may be incorrect.

B1C.

For the embedding length, I chose to use 128. This is somewhat arbitrary, but I decided to follow along with Dr. Sewell’s lecture and he used 128. This will allow enough information into the model without including unnecessary information.

B1D.

For the maximum sequence length, I found the longest single sequence out of all the reviews. In this case, it ended up being 114 so that is what I used as the length.

A close-up of a computer screen

AI-generated content may be incorrect.

B2.

The goals of the tokenization process is to shorten the length of the text used in the analysis, and creating a more optimized sentence with just the crucial words. Making them smaller helps the model be more accurate. To accomplish this, I removed the stopwords from all reviews. These are words such as “the, is, on, etc.” and instead kept all necessary, descriptive words. I then removed all unusual characters using the ASCII method. After the unusual characters, I changed all the text to lowercase to help with normalizing. The code snippets for all are shown below.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

B3.

The padding process is used to make sure that all sequences are the same length (standardizing). I personally decided to do that padding at the end of the sequence, as I believe that represents that information better. Having the sequences the same length helps the neural network process the sequences properly.

A screenshot of a computer program

AI-generated content may be incorrect.

B4.

There are 2 categories of sentiment in this model: positive and negative. They are represented by a 1 and 0 respectively. Since this is a binary sentiment, the activation function used in the final dense layer will be ‘sigmoid’ as that is the most effective function for this scenario.

A screen shot of a computer code

AI-generated content may be incorrect.

B5.

To prep the data for analysis, I first checked for unusual characters using the ASCII table. After seeing the unusual characters, I replaced them with a space as necessary. I then checked the vocabulary size. I turned all the text into lowercase to make it easier for the model to interpret. I also instantiated the word embedding length to be 128. I then looked at the longest single entry and made the length of that be the maximum sequence length. Lastly, I padded the created sequences to standardize the length of all of them. The tokenization process discussed earlier set up the data ready for being split into different sets. Shown below, is the code to generate the training, testing, and validation splits. I chose to use a 70-15-15 split respectively, as that is generally considered the industry standard.

A screenshot of a computer code

AI-generated content may be incorrect.

B6.

The prepared data set is included in the submission. To include the padded sequence in the dataframe, I had to convert each sequence to a list for pandas to be able to add it to the dataframe. This list was not used in the training part, as that requires the full padded sequence.



Part III: Network Architecture

C1.

Output of the model summary is shown below.

A screenshot of a computer

AI-generated content may be incorrect.

C2.

There are 5 layers in the model that I created. The first type of layer used is the embedding layer. This is used to convert categorical data, such as words or items, into continuous vector representations. (GeeksforGeeks, 2025). The second type of layer used is the bidirectional, LSTM layer. I initially attempted to use the LSTM layer by itself, but was running into much trouble finding a good model. I found a resource from GeeksforGeeks that recommended trying a Bidirectional layer, which processes sequential data in both forward and backward directions. This allows Bi LSTM to learn longer-range dependencies in sequential data than traditional LSTMs which can only process sequential data in one direction. The dropout layer is used to decrease the number of nodes that are “on” to decrease overfitting. The first dense layer is also known as the fully connected layer which “takes the input from the previous layer and computes the final classification or regression task.” (GeeksforGeeks). The last layer is the second dense layer, also known as the output layer. The output layer takes “The output from the fully connected layers which is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.” (GeeksforGeeks, 2025)

The number of parameters for each layer is shown below. This is a result of the effect each layer has on the model respectively.

Embedded Layer: 216,320 parameters

Bidirectional Layer: 263,168 parameters

Dropout Layer: 0 parameters

1st Dense Layer: 16,448 parameters

2nd Dense Layer: 65 parameters

C3a.

The activation function for each layer is shown below.

Embedded Layer: None

Bidirectional Layer: None

Dropout Layer: None

1st Dense Layer: ‘relu’. This is used for efficiency in the training and computations

2nd Dense Layer: ‘softmax’. This takes the output and returns probabilities, which is then used for the classification.

C3b.

The number of nodes for each layer is shown below. This is a result of the effect each layer has on the model respectively.

Embedded Layer: 128 nodes

Bidirectional Layer: 128 nodes

Dropout Layer: 0 nodes

1st Dense Layer: 64 nodes

2nd Dense Layer: 1 node

C3c.

The loss function chosen was ‘Binary\_crossentropy’. This was the function selected because it works well in classification situations where the class is determined based on relative probability.

C3d.

The optimizer chose was ‘adam’ or Adaptive Moment Estimation. This was chosen as adam is often considered the default option for choosing an optimizer, as it is fast and robust. Since I had no crucial reason to switch off of the default option, I kept it as is.

C3e.

As part of the fitting the model, I used the function called EarlyStopping to help with the stopping criteria. I decided to use a patience level of 3. This should help prevent overfitting in the model. This forces the model to stop if the validation loss is increasing 3 times in a row.

Part IV: Neural Network Model Evaluation

D1.

The screenshot below shows that at epoch 4, the validation loss is 0.6790, then 0.5685 at epoch 5, but rises to 0.5991 at epoch 6. And we can see that for epochs 7 and 8 the validation loss keeps increasing, and with patience level of 3 it stops after the 8th epoch.

A screenshot of a computer screen

AI-generated content may be incorrect.

D2 and D3.

The overall fitness of the model can be represented in the screenshot below.

A screenshot of a computer code

AI-generated content may be incorrect.

Here we can see that the model has a test accuracy score of approximately 76%. Thus, the model seems to be acceptable at predicting and classifying sentiment. The overfitting issues were addressed in the dropout layer, as well as in the Early Stopping criteria of patience level 3. Assessing the fitness of the model can be done by checking the loss and accuracy plots comparing the training and validation numbers which is shown below.

The training vs validation loss plot is shown below.

A graph with blue line and orange line

AI-generated content may be incorrect.

From this plot, we can see that the loss in the training set drastically falls after 2 epochs. We can also see the early stopping metric in the validation loss. After the 2nd epoch, the validation loss rises at epoch 3 but then drops back down at epoch 4. If the early stopping was set to 1, for example, the model would have stopped at epoch 3, but since we gave a tolerance of 3 increases in the validation loss, we can continue. Thus, the model stops at epoch 7, after the 3rd validation loss increase. This is important to prevent overfitting.

The training vs validation accuracy plot is shown below.

A graph with a line and a line

AI-generated content may be incorrect.

From this graph, we can see that the accuracy of the validation set levels out after an epoch of 2. This is why early stopping is important because it will prevent overfitting of the model.

D4.

The predictive accuracy shown in the screenshot in Part D2 is about 76%. I’m also including in a screenshot below a confusion matrix, and a classification report for the model.

A chart with numbers and a few labels

AI-generated content may be incorrect.

From this, we can see that the accuracy score is as expected; 0.76 or 76%. The model performs well overall, but predicts a negative sentiment more often than is actually present. My assumption would be that this is due to some “positive” reviews tending to be more neutral, or a 6/10 type of review. The 76% is a good score, as 3 out of 4 times the model will correctly predict the sentiment.

D5.

This analysis complies with AI global ethical standards as the data taken as no information about who wrote the review, and thus anonymity is guaranteed in the analysis. I also edited the reviews to only contain crucial words, which makes the data much harder to match up to someone’s review. I also did not omit any data to try and limit any bias that could exist by eliminating some of the data.

Part V: Summary and Recommendations

E.

The code to save the model is shown in the screenshot below.



F.

The networks functionality is good. The neural network performs well especially for the positive reviews. The model does struggle a bit with overpredicting negative reviews, leading to a lower precision score of 0.69. Since it overpredicts, the recall score is quite high at 0.86. Despite these issues, the model still graded out well with an overall accuracy of approximately 76%.

The neural network architecture helped contribute to the effectiveness of the model. As mentioned in Part C2B, the embedded layer helped convert categorical data, such as words or items, into continuous vector representations. The bidirectional layer helped process sequential data in both forward and backward directions. This allows Bi LSTM to learn longer-range dependencies in sequential data than traditional LSTMs which can only process sequential data in one direction. The dropout layer helped decrease the nodes which then helped decrease overfitting. The first dense layer takes the input from after the dropout layer and computes the final classification. The last layer is the 2nd dense layer, which converts the output of each class into the probability score of each class. All of these worked together to create the working model and gave an accuracy of 76%.

G.

I would recommend that the model be implemented. The model does a good job at effectively predicting the sentiment of reviews, especially the positive reviews. The model does struggle a bit with overpredicting negative reviews, so it could be worth taking a second look at reviews when it suggests that a review is negative. Overall, the model is effective at predicting sentiment and could be used to assist in evaluating business performance.

Part VI: Reporting

H.

Copy of code used to save the train network within the neural network and the output is provided in a PDF.

I/J.

Datacamp. (Nov 5th, 2024). *Python Bag of Words Model: A Complete Guide.* Retrieved March 25th, 2024,From <https://www.datacamp.com/tutorial/python-bag-of-words-model>

Dr. Sewell, W. (n.d.). *D213\_Task2\_Sentiment\_Analysis\_Ans\_YelpOnly.html.* Retrieved March 25th, 2024,From D604 Course Search

<https://westerngovernorsuniversity-my.sharepoint.com/personal/william_sewell_wgu_edu/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fwilliam%5Fsewell%5Fwgu%5Fedu%2FDocuments%2FDocuments%2FD213%2FWebinars%2FD213%5FTask2%5FSentiment%5FAnalysis%5FAns%5FYelpOnly%2Ehtml&parent=%2Fpersonal%2Fwilliam%5Fsewell%5Fwgu%5Fedu%2FDocuments%2FDocuments%2FD213%2FWebinars&ga=1>

GeeksforGeeks. (March 3rd, 2025). *Optimization Rule in Deep Neural Networks.* Retrieved March 18th, 2024,From <https://www.geeksforgeeks.org/optimization-rule-in-deep-neural-networks/>

GeeksforGeeks. (Feb 27th, 2025). *What is LSTM – Long Short Term Memory.* Retrieved March 27th, 2024,From <https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/>

GeeksforGeeks. (April 8th, 2024). *Program to check if a string contains any special character.* Retrieved March 27th, 2024,From <https://www.geeksforgeeks.org/python-program-check-string-contains-special-character/>

GeeksforGeeks. (Jan 3rd, 2024). *Removing stop words with NLTK in Python.* Retrieved March 26th, 2024,From <https://www.geeksforgeeks.org/removing-stop-words-nltk-python/>

GeeksforGeeks. (Aug 26th, 2024). *Difference Between an Embedding Layer and a Dense Layer.* Retrieved March 25th, 2024,From <https://www.geeksforgeeks.org/difference-between-an-embedding-layer-and-a-dense-layer/>

Kotzias, D. (2015). Sentiment Labelled Sentences [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C57604>.

Scikit-learn. (n.d.). *ConfusionMatrixDisplay.* Retrieved March 24th, 2024,From <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html>