**Performance Assessment- D213: Sentiment Analysis**

**A. Research Question**

**1.** For this assessment, the research question will be as follows: with sentiment analysis using neural networks, are we able to predict user reviews from Amazon?

**2.** The goal of the data analysis is to determine if we can use neural networks to perform sentiment analysis to determine if we are capable of predicting either positive or negative user reviews for Amazon products.

**3.** For this assessment, LSTM (long short-term memory) is the type of neural network that I will be using to perform sentiment analysis and make predictions on the Amazon review data set. LSTM was chosen because it is a “binary classification techniques which is a type of machine learning in which AI is trained to accurately predict the right outcome out of two possibilities, usually true or false, positive or negative, or 0 or 1” (“L Is for Logistic Regression to LSTM,” 2023). Since the goal of the analysis is to predict positive or negative reviews (sentiments) LSTM is an appropriate type of neural network to use for this analysis.

**B. Data Preparation**

**1.** The first step, after loading the data, is to prepare the data for sentiment analysis is to do some exploratory analysis to determine the presence of unusual characters, to determine the size of the vocabulary, to propose an embedding length for the model and to justify the maximum sequence length for the model.

* Python was used in order to check the amazon review file for unusual characters. For this assessment, unusual characters are defined to be anything that is not standard English and common punctuations, i.e. no emojis, etc. Using python, it was determined that our review set contained only 25 unusual characters. These were subsequently removed.
* By using Python to count the number of words (or vocabulary), it was determined that the size of our vocabulary is 1867.
* For this assessment, the embedding length was set to 100 because “the common embedding length that yields the most useful result is between 100 and 300” (Huseby, 2020) (Elleh, 2024).
* In order to determine what the maximum sequence length for the model should be, I used python to split the Amazon reviews into individual words and determine the sequence value that would fall into the 95th percentile of the length (words) of the reviews. By using the 95th percentile of the sequence length, the maximum sequence length can cover the majority of the data without having any outliers that might require more padding, which will overall improve the performance and accuracy of the model. When I did this in python, it was determined that the maximum sequence length should be set to 23.

The following screenshot showcases all of the code used to conduct exploratory data analysis and the resulting output:

A screenshot of a computer

Description automatically generated

**2-3.** The next part of preprocessing the data for sentiment analysis is what is referred to as tokenization. This uses the tokenizer function from the tensorflow.keras.preprocessing.text package in python in order to normalize the text reviews by converting them into a series of integers. This breaks down the text into what is referred to as “tokens” in order to make it easier for the model to understand and interpret the data. By turning the text data into integers, the model can then recognize, for example, repeating integers (same word in different reviews) in order to predict whether the user gave a positive or negative review. After tokenizing the review data, the next step in preprocessing is referred to as padding. “ Padding is the process of adding layers of zeros or other values outside the actual data in an input matrix. The primary purpose of padding is to preserve the spatial size of the input so that the output after applying filters (kernels) remains the same size” (DeepAI, 2019). This analysis padded the data based on the maximum sequence length of 23 that was determined in the previous steps. Padding was added to any sequences that were shorter than our value of 2023. This would be referred to as post padding, where the padding occurs after the text (int) sequence. Doing this keeps all of the data the same length which is critical for accurate analysis. The following screenshot showcases the code used to perform tokenization and padding as well as an example of a padded sequence (in its integer form). Notice the 0s at the end of the sequence where the padding took place:

A screenshot of a computer

Description automatically generated

**4.** The next step of preprocessing the data for analysis is to determine the number of categories of sentiment and to determine an activate function for the model. Python was used to determine the number of categories of sentiment as well as if we should use sigmoid or softmax as our activation function. By running this code, it was determined that there are two categories of sentiment, and the activation function should be sigmoid. This makes sense. Since the sentiment is binary (1 for positive, 0 for negative) it makes sense for there to only be two categories and for the use of sigmoid because you should “opt for softmax when dealing with multiple classes that require nuanced probability assignments for accurate predictions, and conversely, choose sigmoid for straightforward binary classifications demanding precise probability distinctions between two outcomes” (“Softmax vs. Sigmoid: Neural Networks Variation Explained” 2024). Since positive and negative (1s and 0s) is binary, it makes sense to use sigmoid as the activation function.

**5.** The last step of preprocessing the data is to split it into training, validation, and testing data sets. “The rough standard for train-validation-test splits is 60-80% training data, 10-20% validation data, and 10-20% test data” (Acharya, 2023). Based on this industry standard, the data was split into 60% training and 20% for both the validation and testing data sets. This was the final step of data preprocessing.

**6.** A copy of the cleaned data set as well as the training, testing, and validation data sets has been attached alongside this written assessment.

**C. Network Architecture**

**1.** The following screenshot is an output of the model summary as well as the full code used to build and train the model:

A screenshot of a computer program

Description automatically generated

**2.** Within the summary there are a total of 5 layers:

* Embedding layer: this layer embeds the text data into a format that is easier for the model to handle. It was defined by using the vocabulary size, our proposed embedding length, and the maximum sequence length that was based on the 95th percentile.
* LSTM layer: this layer is the type of neural network that we defined for the sequential model as discussed previously.
* Dense layer: this layer is important to help the model learn complicated patterns within the data that are not defined linearly.
* Dropout layer: this layer is important as it helps to prevent overfitting
* Output Dense layer: this layer uses sigmoid activation which we determined to be the appropriate final activation function during our exploratory data analysis section.

These 5 layers output a total of 219,501 trainable parameters, which is a sufficiently large number of parameters in which case the model can learn from the data in order to make our predictions off of.

**3.** The following discussion is related to the hyperparameters from our model:

* Activation functions: relu was used in the dense layer as it “ introduces the property of nonlinearity to a deep learning model and solves the vanishing gradients issue. It interprets the positive part of its argument” (Whitfield 2024). The other activation function used was the sigmoid activation function in the final output dense layer, which as discussed previously, was determined in our exploratory phase as it is suitable to binary classification.
* Number of nodes per layer: 50 nodes were used in the LSTM and dense layers. These numbers were determined through experimentation. I ran the model multiple times with an increasing number of nodes in the two layers and when the accuracy of the model was no longer improving, I determined that 50 and 50 was the best combination for the most efficient and accurate model.
* Loss function: binary crossentropy was used as the loss function since the research question involves predictions for binary classification (positive and negative, 1s and 0s).
* Optimizer: adam was used as the optimizer in the model as it is one of the most popular optimizers in neural networks and can efficiently adjust the learning rate when the model is trained
* Stopping criteria: the model was fit to stop if the model was not improving after 3 epochs.
* Evaluation metric: the model was evaluated by the accuracy, which measures the percentage in which the model predicted the correct value for sentiment

**D. Model Evaluation**

**1.** The following screenshot showcases the use of the stopping criteria to determine the best model epoch:

A screenshot of a computer

Description automatically generated

As discussed previously, the stopping criteria was defined to stop the model after 3 epochs if it was not improving. Based on that criteria, it was determined that the 6th epoch was the best model, with the following metrics:

* Accuracy: 0.8949
* Loss: 0.3086
* Val\_accuracy: 0.7650
* Val\_loss: 0.6014

Based on the aforementioned metrics, the model has a high accuracy of approximately 89% on the training data with a low loss of around 0.31 The accuracy against the validation was slightly less than the training data at around 76.5% with a loss of 0.6014. This suggests that the model is not as accurate when seeing new data, but 76% is still fairly high as it suggests it is still capable of predicting sentiment ¾ of the time. We can also look at the next epochs to determine why early stopping was initiated. Epoch 7 has a greater accuracy to the training set, but the validation loss is greater than 1 which shows that the model began to overfit and combined with the parameters of the subsequent epochs as well, the model was not improving, and epoch 6 was chosen as the best model. Epoch 6 has the highest validation accuracy with the least loss. This is a direct impact of our stopping criteria, as once the model showed signs of overfitting and was not improving, the algorithm was stopped. This also helps to preserve computational resources as well.

**2.** As discussed in the previous section, the 6th epoch had a training accuracy of 89.49% and a loss of 0.3086 with a validation accuracy of 76.5% and validation loss of 0.6014. The decrease in accuracy and increase in loss do show “some” signs of overfitting but not as significant as other epochs do. The stopping criteria was designed to reduce as much overfitting as possible and save the best model based on that criteria. A dropout layer of 0.5 was also included in order to reduce any overfitting. This means that when the model is being trained it turns off 50% of the neurons in the neural network at any time which allows the network to learn on various subsets within the data in order to prevent overfitting by ensuring the model is learning rather than memorizing.

**3.** The following again shows a screenshot of the entire model training process that was previously shown in part **C1**:

**A screenshot of a computer program

Description automatically generated**

Here is a plot that showcases the training and validation losses across epochs:

A graph of loss over epops

Description automatically generated

Here is a plot that showcases the training and validation accuracy across epochs:

A graph of a graph showing the performance of a performance

Description automatically generated with medium confidence

**4.** The following screenshot evaluates the model’s accuracy using the testing data:

A computer code with text

Description automatically generated with medium confidence

When predicting against the testing data, the accuracy of the model is approximately 0.8050 or 80.50% which is lower than the training data but higher than the validation data. The loss is also less. This suggests that the model was slightly overfitting to the training data, but it is still performing well against new data. Overall, 80% is a very high accuracy for making binary predictions.

**Summary and Recommendations**

**E.** As discussed previously, the code used to save the trained network within the neural network was initiated in the code for the early stopping, where the best model was then saved as a .keras file. The .py file for the entire code has also been attached alongside this written assessment. The following screenshot again shows the code used to save the best model within the early stopping criteria:

A close up of a text

Description automatically generated

**F.** The model was built using 1000 Amazon reviews that were split into training, testing, and validation splits on a 60-20-20 ratio. Using these data splits, a LSTM neural network was trained on the training data where the sentiments (positive/negative score) was used a labels. The LSTM model was trained to predict whether or not a particular amazon review was either positive or negative. Overall, the model was approximately 89.49% accurate against the training set, 76.5% against the validation set, and 80.5% against the testing set, which suggests some overfitting, but the model is still predicting correctly at least 3/4s of the time. Ultimately, the network architecture withing the LSTM model, as well as other key hyperparameters (such as early stopping criteria, binary crossentropy, etc.) definitely impacted the model in a positive way as these are designed to work with binary classification and sentiment analysis, resulting in a more efficient and better model than what other neural networks could have provided.

**G.** Based on everything discussed, I have several recommended future actions. I suggest fine tuning and experimenting with different parameters, specifically the dropout rate, as different rates might be less prone to overfitting. I also recommend using cross validation by including other review data sets to see how well that model performs across various reviews and sentiments. But overall, the model seemed the work appropriately and efficiently.

**Reporting**

**H.** An HTML copy of a Jupyter notebook showcasing the neural network has been attached alongside this written assessment.

**I-J. Sources**

Huseby, Kristin H. “Word Embeddings, What Are They Really?” Medium, Towards Data Science, 9 Dec. 2020, towardsdatascience.com/word-embeddings-what-are-they-really-f106e1ff0874.

Elleh, Festus Dr.. “Welcome to D213: Advanced Data Analytics- Task 2”. 2024. PowerPoint. Western Governors University. <https://westerngovernorsuniversity.sharepoint.com/:p:/r/sites/DataScienceTeam/_layouts/15/Doc.aspx?sourcedoc=%7B50F5B89A-72F1-4964-9942-1F3790EF1D59%7D&file=D213%20Task%202%20Cohort%20Webinar%20PPT.pptx&action=edit&mobileredirect=true>

“Padding (Machine Learning).” DeepAI, DeepAI, 17 May 2019, deepai.org/machine-learning-glossary-and-terms/padding#:~:text=Padding%20is%20the%20process%20of,to%20the%20desired%20output%20dimensions.

“Softmax vs. Sigmoid: Neural Networks Variation Explained.” RSS, 4 Apr. 2024, myscale.com/blog/neural-networks-softmax-sigmoid/.

Acharya, Akruti. “Training, Validation, Test Split for Machine Learning Datasets.” Encord, 13 June 2023, encord.com/blog/train-val-test-split/#:~:text=The%20training%20set%20data%20must,10%2D20%25%20test%20data.

Whitfield, Brennan. “An Introduction to the ReLU Activation Function.” Built In, 26 Feb. 2024, builtin.com/machine-learning/relu-activation-function.

“L Is for Logistic Regression to LSTM.” Jaid, 8 Nov. 2023, jaid.io/blog/l-is-for-logistic-regression-to-lstm/#:~:text=Logistic%20regression%20and%20LSTM%20%E2%80%94%20Long,negative%2C%20or%200%20or%201.