Shawn Wheeler

D213 Task II

000364855

Table of Contents

[Part I: Research Question 3](#_Toc170494146)

[A1: Research Question 3](#_Toc170494147)

[A2: Objectives or Goals 3](#_Toc170494148)

[A3: Prescribed Network 3](#_Toc170494149)

[Part II: Data Preparation 3](#_Toc170494150)

[B1: Data Exploration 3](#_Toc170494151)

[B2: Tokenization 5](#_Toc170494152)

[B3: Padding Process 6](#_Toc170494153)

[B4: Categories of Sentiment 6](#_Toc170494154)

[B5: Steps to Prepare the Data 7](#_Toc170494155)

[B6: Prepared Data Set 7](#_Toc170494156)

[Part III: Network Architecture 7](#_Toc170494157)

[C1: Model Summary 7](#_Toc170494158)

[C2: Network Architecture 8](#_Toc170494159)

[C3: Hyperparameters 8](#_Toc170494160)

[Part IV: Model Evaluation 10](#_Toc170494161)

[D1: Stopping Criteria 10](#_Toc170494162)

[D2: Fitness 10](#_Toc170494163)

[D3: Training Process 10](#_Toc170494164)

[D4: Predictive Accuracy 12](#_Toc170494165)

[Part V: Summary and Recommendations 13](#_Toc170494166)

[E: Code 13](#_Toc170494167)

[F: Functionality 13](#_Toc170494168)

[G: Recommendations 13](#_Toc170494169)

[Part VI: Reporting 13](#_Toc170494170)

[H: Reporting 13](#_Toc170494171)

[I: Sources for Third-Party Code 13](#_Toc170494172)

[J: Sources 14](#_Toc170494173)

## Part I: Research Question

### A1: Research Question

Is it possible to use past review data to be able to predict future reviews by utilizing neural networks and natural language processing?

### A2: Objectives or Goals

The goal is this project is to attempt to predict future reviews by using previous review data. In order to do that, a neural network model will be trained and then used for future prediction. Those predictions will then be tested for their accuracy to determine the value of the created model.

### A3: Prescribed Network

Recurrent Neural Networks (RNN) are capable of achieving the necessary tasks for this project. They has the ability to perform text classification, can be trained on past data, and then can also be used to make predictions. Other neural networks are not as well suited to this task as they lack the ability to retain previous inputs; RNNs, however, do have that ability making them ideal for sequential data analysis as their memory persists through time. (GeeksforGeeks, 2018).

## Part II: Data Preparation

### B1: Data Exploration

*Presence of Unusual Characters:*

After the data was concatenated, it was checked for the presence of unusual characters by using a for loop to iterate through the data and create a subset of characters used in reviews. Examining that information allowed for determination of the existence of many unusual characters that would need to be addressed.

*Screenshot detecting presence of unusual characters:*

A black text on a white background

Description automatically generated

*Vocabulary Size:*

Vocabulary size of the data refers to the quantity of unique words used throughout the data. In this case, this was determined to be 5272 words.

*Screenshot of vocabulary size:*

A screenshot of a computer code

Description automatically generated

*Proposed Word Embedding Length:*

Next, the data was investigated to determine the proposed word embedding length. This was achieved by using the vocabulary size and taking the 2nd square of its value. This value will be used later to determine one of the values needed in creating the model.

*Screenshot of word embedding length:*

*A screen shot of a computer

Description automatically generated*

*Statistical Justification for the Chosen Maximum Sequence Length:*

The maximum sequence length was selected by using a for loop once again to iterate through review data and locate the highest value. Values for the median and minimum sequences were also calculated.

*Screenshot of maximum sequence length:*

A screenshot of a computer code

Description automatically generated

### B2: Tokenization

Tokenization, in this case, means breaking down text into smaller pieces, which could be words or subwords, for instance, known as tokens which allow for a machine to better use and learn from language. These tokens can be used to allow for natural language processing tasks, such as, the sentiment analysis being attempted here. (Huzaifatahir, 2023). Tokenization allows for normalization and standardization. To achieve that, letters were made lowercase, punctuation and other unusual characters were removed, the data was lemmatized, and stop words were removed. Tokenizing was also used to determine vocabulary size via fit\_on\_texts and to pad data which will be described in B3 with texts\_to\_sequences. (Sharma, 2021).

*Packages used for normalizing text:*

|  |  |
| --- | --- |
| Packages / Libraries | Usage |
| pandas | Dataframe and Series manipulation |
| re | Regular Expressions module to remove any characters aside from a-z and A-Z |
| nltk.tokenize import word\_tokenize | Tokenize review data |
| nltk | Lemmatize review data using WordNetLemmatizer which “links words into semantic relations” (GeeksforGeeks, 2020) |
| nltk.corpus import stopwords | Remove common words to improve accuracy |
| tensorflow.keras.preprocessing.text import Tokenizer | Determining vocabulary size and padding data |

*Screenshot of initial tokenizing:*

*A screen shot of a computer program

Description automatically generated*

*Screenshot of vocabulary size using tokenization:*

*A screenshot of a computer code

Description automatically generated*

*Screenshot of padding using tokenization:*

*A screenshot of a computer code

Description automatically generated*

### B3: Padding Process

Padding is used to “make all sequences in a batch fit a given standard length” which is necessary for the data to be able to be used by the neural network. (TensorFlow, n.d.). Padding can be added to either the beginning or the end of the data and is added in the amount necessary to make all sequences a uniform length. In this case, the padding was added after the text sequence. Below is also a screenshot of a single padded sequence.

*Screenshot of single padded sequence:*

A screenshot of a number

Description automatically generated

### B4: Categories of Sentiment

Two categories of sentiment will be used for this analysis. There are only positive and negative sentiments only within the data; therefore, the sentiment is binary. The activation function for the final dense layer of the neural network was selected to be sigmoid. Sigmoid was selected because it allows for output to be binary (same as the categories of sentiment). To be more specific, sigmoid selects a probability between 0 and 1 and essentially rounds the number to select a negative or positive sentiment respectively. (TensorFlow Core, n.d.).

### B5: Steps to Prepare the Data

1. Load all three datasets and concatenate them into a singular dataset of 3000 rows and 2 columns.

1. Exploratory data analysis. Check for unwanted characters, identify vocabulary size and proposed word embedding length, determine max, median, and minimum lengths of sequences, and visualize sentiment ratings.

1. Clean data for processing. Remove unwanted characters leaving only lower and upper case letters. Convert upper case letters to lower case. Remove stop words.
2. Tokenize review data and then lemmatize the data to further prepare it for the neural network.
3. Split data set into training and test data using train\_test\_split with an 80% training and 20% testing data ratio.
4. Pad the X\_test and X\_train data so that all data is of equal length as required.
5. Convert all split data into individual numpy arrays.
6. Save each of the four individual data sets.

### B6: Prepared Data Set

*See Attached Files:* training\_padded.csv, training\_label.csv, test\_padded.csv, test\_label.csv

## Part III: Network Architecture

### C1: Model Summary

*Screenshot of output of model summary:*

A screenshot of a computer

Description automatically generated

### C2: Network Architecture

For the constructed neural network, five layers were used. The first layer is the embedding layer. The embedding layer consisted of 168,704 parameters. These correspond to a function of the size of the vocabulary and the embedding dimension.

Secondly, a SpacialDropout1D layer was implemented. This layer is a dropout layer which probabilistically removes inputs from a previous layer which has the effect of “simulating a large number of networks with very different network structure” making it more robust to overfitting. (Brownlee, 2018. There are zero parameters in this layer. It was set to drop 20% of inputs.

Thirdly, a Long-Short-Term-Memory (LSTM) layer was added. LSTM is a specialized Recurrent Neural Network used especially to classify sequential data and is quite useful in sentiment analysis. (Mathworks, n.d.). The number of parameters within this layer was 22,320.

Finally, two dense layers were added to the end. The dense layers collected output based on the unit sizes selected, 20 and 1. The first dense layer had a set unit of 20 and consisted of 1,220 parameters. The second dense layer contained only 21 parameters.

### C3: Hyperparameters

*Activation Functions:*

There were two activation functions used in this analysis. Tanh was used for LSTM and attempts to transform the data into a normalized encoding of the data which is then passed along to the dense functions. Both of these use the sigmoid activation function which allow for a binary output. Since we are performing sentiment analysis with only two categories, that is ideal.

*Number of Nodes per Layer:*

The LSTM layer uses 60 nodes. The first dense layer uses 20, and the final dense layer is set at 1. These values were chosen as they were shown to fit the model well via experimentation.

*Loss of Function:*

The loss function selected was binary\_crossentropy. Binary\_crossentropy was chosen because it is ideal (and only suitable) for predicting binary values similarly to how sigmoid was described earlier. Using information about the predicted probability, it is used to evaluate prediction error. (Sorokin, 2024)

*Optimizer:*

Adam was selected as the optimizer for this model. Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) are combined in Adam providing greater performance on natural language processes than other options. It is a very popular optimizer, as well, and is often used as a default before testing other potential options. It proved to have good performance. (Brownlee, 2017)

*Stopping Criteria:*

The stopping criteria is controlled by the EarlyStopping method. EarlyStopping has a parameter called patience which dictates how many worse results are acceptable before halting iteration through epochs. For instance, if patience=3, if after 3 iterations, there is no increase in accuracy, the process of fitting the model is stopped and the highest value used even if it has not reached the set amount of epochs. Patience was set to 6 for this analysis in order to still have a decent amount of training while helping to prevent overfitting.

*Evaluation Metric:*

For evaluation purposes, accuracy was selected to determine the value of the trained neural network.

## Part IV: Model Evaluation

### D1: Stopping Criteria

The stopping criteria was handled by EarlyStopping and set at patience=6. This allows the training process to end early if there is no improvement shown in the set amount of iterations. In this case, if there were 6 attempts in a row that failed to provide a better value, then the process would be terminated regardless of the amount of epochs set to cycle through and the previously best value will be used by the model. This is used to prevent overfitting and save resources. Below shows this is action as the process stops at epoch 19 even though the fitting was set to 30 epochs.

*Screenshot of final training epoch:*

A table of numbers with numbers

Description automatically generated with medium confidence

### D2: Fitness

Overfitting is when the model performs far better on the training dataset but not nearly as well on the testing dataset. In other words, it is not useful for predictions. This was mitigated in a few ways. Using a SpatialDropout1D layer and setting the recurrent\_dropout in the LSTM layer were the first methods used to create a better fitting model. On top of that, early stopping detection was utilized as another step to prevent overfitting. Based on the visuals of accuracy and loss, it appears to be a good fit. Also, while the training accuracy was higher at 81.58%, the test accuracy was still 73% which is indicative of good fitness.

### D3: Training Process

*Screenshot of model’s training process:*

*A screenshot of a computer

Description automatically generated*

*Line graph of loss:*

*A graph of a line

Description automatically generated with medium confidence*

*Line graph of accuracy:*

A graph of a graph

Description automatically generated with medium confidence

### D4: Predictive Accuracy

The predictive accuracy of the model was shown to be 73% based on comparison to testing data which is respectably accurate. Using a confusion matrix, the accuracy was also tested and, as seen below, the model was shown to be correct in 438/600 instances and incorrect on 162/600 which mirrors the 73% accuracy calculated for the test data.

*Screenshot of confusion matrix:*

A diagram of a confusion matrix

Description automatically generated

*Screenshot of predictive accuracy:*

**

## Part V: Summary and Recommendations

### E: Code

model.save('d213task2model.keras') was used to save the trained network.

### F: Functionality

The created neural network performs well for the task it was created for. Using 3000 past reviews, the model was trained and proven to have a predictive accuracy of 73% based on the test data. With these predictions in mind, plans can be created to either potentially enhance the positive sentiments or decrease the negative ones. RNNs are perfect for the task of sentiment analysis as the architecture excels in text classification. LSTM and the dense layers were selected as well based on their unique suitedness to sentiment analysis with the dense layers providing the needed binary output and the LSTM as a specialized RNN. The combinations of these factors create great functionality for the model for its purpose of predicting either a positive or negative sentiment.

### G: Recommendations

With a predictive accuracy of 73%, the final model is good enough for its intended purpose. The model should be used for its predictions which, depending on the overall goal, should then be acted on. For instance, if the predicted positive sentiments are too low, an initiative could be implemented to stave that off. More data should also be collected and the model could be updated in, say, six months for additional analysis and updated, more accurate predictions.

## Part VI: Reporting

### H: Reporting

*See Attached File:* d213task2complete.html

### I: Sources for Third-Party Code

Couse Materials (n.d.)

### J: Sources

Brownlee. (2017, July 2). *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. Machine Learning Mastery. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Brownlee. (2018). https://www.facebook.com/jason.brownlee.39. (2018, December 4). How to Reduce Overfitting With Dropout Regularization in Keras. Machine Learning Mastery. https://machinelearningmastery.com/how-to-reduce-overfitting-with-dropout-regularization-in-keras/

GeeksforGeeks. (2018, October 3). Introduction to Recurrent Neural Network - GeeksforGeeks. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

GeeksforGeeks. (2020). Python - Lemmatization Approaches with Examples. (2020, September 4). GeeksforGeeks. https://www.geeksforgeeks.org/python-lemmatization-approaches-with-examples/

Huzaifatahir. (2023, September 14). Decoding the Magic: NLP Tokenization and Text Normalization. Medium; Medium. https://medium.com/@huzaifatahir7524/decoding-the-magic-nlp-tokenization-and-text-normalization-91c9b5a2089e

Mathworks. (n.d.). *Long Short-Term Memory (LSTM) Networks*. (n.d.). Www.mathworks.com. https://www.mathworks.com/discovery/lstm.html

Sharma, P. (2021, January 1). Keras Tokenizer Tutorial with Examples for Beginners. MLK - Machine Learning Knowledge. https://machinelearningknowledge.ai/keras-tokenizer-tutorial-with-examples-for-fit\_on\_texts-texts\_to\_sequences-texts\_to\_matrix-sequences\_to\_matrix/

Sorokin, M. (2024, January 17). *Binary Cross-Entropy: Mathematical Insights and Python Implementation*. Medium. https://medium.com/@vergotten/binary-cross-entropy-mathematical-insights-and-python-implementation-31e5a4df78f3

TensorFlow. (n.d.). Understanding masking & padding | TensorFlow Core. (n.d.). TensorFlow. https://www.tensorflow.org/guide/keras/understanding\_masking\_and\_padding

TensorFlow Core. (n.d.). Basic text classification | TensorFlow Core. (n.d.). TensorFlow. https://www.tensorflow.org/tutorials/keras/text\_classification