WGU MSDA Program

Sentiment Analysis Using Natural Language Processing Neural Network

Write Up

Christopher Kamper

March 6th, 2024

**Part I: Research Question**

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

1. Can the sentiment of a customer review be classified as positive or negative by a neural network accurately?

1. The goal of this analysis is to create a neural network model that can predict the sentiment of a customer's text review as positive or negative. This will be done by using text reviews from three different companies (Amazon, IMDB, and Yelp) that are categorized with a score denoting the sentiment. These reviews will be used to train and test the model after the cleaning and preprocessing stages. A well-performing model will clearly answer the research question and would allow for it to be used in practice.

1. For this analysis, a Recurrent Neural Network (RNN) will be used. This one is chosen over other forms of neural networks as a big advantage is its ability to handle Natural Language Processing problems. (Thomas) Since we are working with text data and looking to make a binary classification, this is the best method to use.

**Part II: Data Preparation**

1. Summarize the data cleaning process by doing the following:

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

While there are no obvious unusual characters while reviewing the Excel files and Data Frame, it was noticed that the IMDB data set did end up with less than the 1000 rows expected. It appears there was a formatting issue when changing the .txt file sourced to a .csv that caused some rows to drop. Working under the assumption that there is a formatting reason that most likely caused this, those rows were just excluded from the data set.

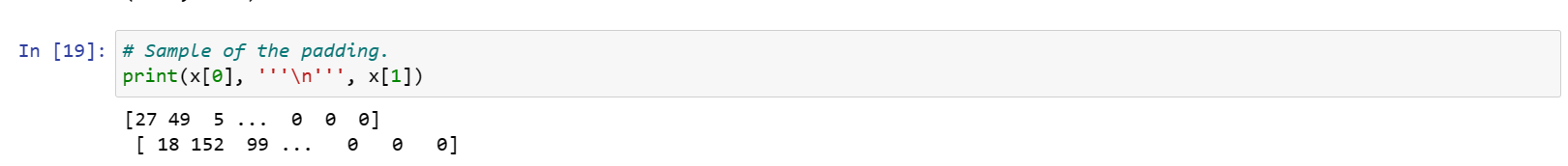
The vocabulary size chosen for this model is 50,000. This is done off the recommendation of Jason Brownlee. (Sewel) The goal with this is to ensure there is plenty in the vocabulary for the best performance out of the model. This eliminates the need for further calculations and gives a good baseline or starting point.

Once the text data set was prepared through the previous preprocessing steps, the vectorized values were padded so all had the same length. This is required by the TensorFlow model. It was chosen to use the maximum word count of the set to ensure that as many words were taken into consideration for the model. There was a test run using an average value for this parameter, but the result was a small drop in performance of the evaluation metric of the model and there was no noticeable difference in the efficiency based on the time it took the model to run. With this in mind, it was decided to keep as much of the data as possible that could have been cut if the length was adjusted to a smaller value that removed data from the longer strings.

The sequence length was chosen since the data set is not particularly large and utilizing all available text did not appear to have a large effect on the efficiency of the model. Another model was tested using an average value, but this had a slightly negative effect on the values of the validation accuracy and loss when compared to the model used that used the true maximum count from the lemmatized words.

1. The goal of the tokenization process is to transform the text string that makes up each review and break them down into "tokens" that comprise the string. Each word becomes its own token within the grouping that is the original string. This is useful in future steps such as creating a vocabulary for a model and vectorizing since each word becomes its own entity.

1. Padding occurs after the text sequence in this analysis. This was done since the data set is not extremely large and the goal was to ensure as much of the other preprocessing steps were utilized. This allowed for all words to make it through that process without the chance of losing any significance if padding restricted the values in each sequence prior to these steps being completed.



1. There are two categories of sentiment to be used as a result in this analysis. The final output (dense) layer will be 1 dimensional but use the sigmoid activation function to provide the desired binary output. A prediction of "1" denotes a positive sentiment and a prediction of "0" means a negative sentiment.

1. Steps used to prepare data:
   1. Load each of the three .txt files into their own data frame through reading each in as a tab-delimited .csv file and reviewing the data loaded. Ensure the columns are named the same for when they are combined.
   2. Combine the data frames using the pd.concat function so all data is now in one data frame.
   3. Add a column to the combined data frame to show the count of words in each review.
   4. Add a column to the combined data frame to show the count of characters in each review.
   5. Check for null values in each column.
   6. Set all text in each review to be all lower case.
   7. Remove any punctuation in the text of each review.
   8. Lemmatize the text in each review to bring each word to its root.
   9. Remove stop words from the text in each review.
   10. Tokenize the text in each review so every word is split into a separate data set.
   11. Vectorize each word from the tokenized data set.
   12. Split the data set so 80% goes into a x and y training set and the remaining 20% is put into temporary sets.
   13. Split the temporary sets so that 50% goes to a x and y validation set and the remaining 50% goes to separate test sets.

1. The "combined\_df" data frame and "x" data sets are provided as .csv files attached.

**Part III: Network Architecture**



1. Summary from the model:

A screenshot of a document

Description automatically generated

1. The model has a total of 5 layers. The first is an embedding layer that embeds each integer into a 128-dimensional vector. (Kalita) Then a flatten layer is added to transform the embedded values to a one-dimensional array for interactions with the later layers. From there the model utilizes 3 hidden layers that reduce the dimensionality from 32 to 8. Finally, an output layer is used that gives the binary result desired in 1 dimension. The parameters used were the embedding dimension, which was set to 128. The loss function applied was the "binary\_crossentropy". The optimizer was set to "adam”, and the number of epochs set was 20. The total number of parameters in this model is given in the summary as 12,069,441.

1. This analysis included the use of two activation functions. Both apply the non-linear calculation needed for each layer of the model but the output desired for each layer determines which is used. The hidden layers utilize the "relu" function since that is the most commonly used for those calculations. The output layer utilizes the "sigmoid" function since that will produce a binary result. Since the goal of the analysis is a binary prediction, this function is the best option. (SHARMA)

The nodes per layer were chosen simply based on cutting the total number in half with each layer. 32 were chosen for the first hidden dense layer so the next layer was set to 16 and then 8. From there it was decided to go to the output layer to see where the evaluation metric landed. Since the model was immediately providing a validation accuracy above 70%, it was decided to keep this architecture. If the performance of the model was more concerning, this would be an option to change to see the effect on the performance.

The loss function used for this analysis is the binary\_crossentropy function. This was chosen since the desired result is a binary classification and this function is best fit to handle this type of result.

The optimizer chosen for this analysis is the "adam" optimizer. It was picked as it is known as a great general option and is the industry go-to for NLP problems. This is due to its ease of implementation and ability to converge much quicker than other options. (“National Tech & Startups | Built In”)

Early stopping criteria was put in place for the model with a patience value of 3 to aid in minimizing overfitting. (GfG) This means that the model will stop once results for 3 consecutive epochs are not showing improvement of the accuracy. If the criteria are not met, all 20 requested epochs will run but the model did meet the criteria at 9 epochs. This was set to ensure that a reasonable number of epochs were available for the model to reach the best potential while also limiting without the worry of missing out on performance.

The evaluation metric used on the model is the validation accuracy calculation from each epoch. This is used since it is a direct measurement of the model's performance to predict on data it has not seen before. It can be argued that this is the most important metric that could matter to most stakeholders, so it was chosen to be the main factor for evaluating.

**Part IV: Model Evaluation**

1. Evaluate the model training process and its relevant outcomes by doing the following:

1. The stopping criteria was used to help limit the effect of overfitting the model during training. It stops running when the criteria are met which was set to a patience of 3 in this model based on the validation accuracy score. This means the model will stop if there are 3 consecutive epochs that an improvement to the score is not seen. In this analysis the criteria stopped the model after epoch 9, signifying the epoch 6 was the best performing while 7, 8, and 9 returned a lower score.

A white sheet with black text

Description automatically generated with medium confidence

1. The validation loss is the metric here that would provide the best insight to the fitness of the model. This score is on the higher end which would imply there could be some overfitting on the training data. The graph representing the loss across the epochs below backs this up as the training and validation lines drastically diverge with the training loss approaching zero while the validation loss is rising.

1. The validation accuracy score was chosen as the evaluation metric. Below is the plot showing the training and validation scores across the epochs.

A graph of a graph

Description automatically generated

The loss was also reviewed and the plot showing the training and validation loss scores across the epochs is below.

A graph of a graph with a line

Description automatically generated

1. The model is returning a test accuracy of 0.7855 or 78.55%. This means that the model is predicting the correct outcome almost 79% of the time. The loss is a little higher than what would be desired, but the accuracy metric was chosen for evaluation so based on that, this model is returning good accuracy.

**Part V: Summary and Recommendations**

1. Code used for network training:

# Set parameters.

embedding\_dim = 128

loss = 'binary\_crossentropy'

optimizer = 'adam'

num\_epochs = 20

# Define early stopping monitor.

early\_stopping = EarlyStopping(monitor='val\_accuracy', patience=3, restore\_best\_weights=True)

# Create the model.

model = tf.keras.Sequential()

model.add(Embedding(vocab\_size, embedding\_dim, input\_length=max\_length))

model.add(Flatten())

model.add(Dense(32, activation='relu'))

model.add(Dense(16, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

# Compile the model.

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

model.summary()

# Train the model.

model\_result = model.fit(x\_train, y\_train, epochs=num\_epochs, callbacks=early\_stopping, validation\_data=(x\_val, y\_val))

1. The functionality of a RNN is set up to use connections to create cycles within the network that retains memory of the previous values. This model falls into this category and uses the same functionality to make its predictions. This means the connections of the vectorized text inputs are analyzed through the hidden layers to find the best values that relate to the sentiment of the training data. The architecture of this network is what allows it to be functional and make the connections needed for predictions. The input or embedding layer sets up the connections for the vectorized text and the flatten layer is used to make the data a one-dimensional array. This feeds the hidden layers that analyze the connections to produce the binary output prediction if the sentiment of a review is positive or negative. This model is considered a many-to-one recurrent neural network since it takes the many inputs over the multiple layers and brings them on one single output. This architecture is the foundation of the model's functionality and allows for it to accomplish the goal of getting a binary prediction.

1. Based on the evaluation metric of the validation accuracy, it can be recommended that this model could be used in practice to predict the sentiment of customer reviews. It should be noted though that the validation loss score is a bit of a concern since it is on the higher end. This implies there could be some overfitting to the training data. If it is decided to utilize the model, ideally it would be further trained on more data not currently available or to have some tweaks made. The training could also be run using this metric rather than the validation accuracy, but there is still a chance that the results will be very similar, or the accuracy score will decrease. Despite this, it would be recommended to use this model as it does pass the validation accuracy metric with a good score and that is the value truly being used to evaluate this model.

**Part VI: Reporting**

1. The .ipynb file and a PDF printout of the Jupyter notebook used for this analysis is attached.

1. Code Sources:

Sewell, William. “D213 SA Webinar 5.” D213 SA Webinar. D213 SA Webinar 5, 28 Feb. 2024.

Elleh, Festus. “D213 Task 2 Building NN Model in Python\_default.” D213 Task 2. D213 Task 2 Building NN Model in Python\_default, 28 Feb. 2024.

“Neural Networks - Dense Layers.” Datacamp.com, DataCamp, 2022, campus.datacamp.com/courses/introduction-to-tensorflow-in-python/63344?ex=1. Accessed 28 Feb. 2024.

“Neural Networks - Activation Functions.” Datacamp.com, DataCamp, 2022, campus.datacamp.com/courses/introduction-to-tensorflow-in-python/63344?ex=5. Accessed 28 Feb. 2024.

“Neural Networks - Optimizers.” Datacamp.com, DataCamp, 2022, campus.datacamp.com/courses/introduction-to-tensorflow-in-python/63344?ex=8. Accessed 28 Feb. 2024.

“Neural Networks - Training a Network in TensorFlow.” Datacamp.com, DataCamp, 2022, campus.datacamp.com/courses/introduction-to-tensorflow-in-python/63344?ex=11. Accessed 28 Feb. 2024.

1. Sources:

GfG. “How Does Epoch Affect Accuracy in Deep Learning Model?” GeeksforGeeks, GeeksforGeeks, 31 July 2023, [www.geeksforgeeks.org/how-does-epoch-affect-accuracy-in-deep-learning-model/](http://www.geeksforgeeks.org/how-does-epoch-affect-accuracy-in-deep-learning-model/). Accessed 28 Feb. 2024.

Kalita, Debasish. “A Brief Overview of Recurrent Neural Networks (RNN).” Analytics Vidhya, 11 Mar. 2022, [www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/](http://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/). Accessed 28 Feb. 2024.

SHARMA, SAGAR. “Activation Functions in Neural Networks - towards Data Science.” Medium, Towards Data Science, 6 Sept. 2017, towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. Accessed 28 Feb. 2024.

Thomas, Christopher. “Recurrent Neural Networks and Natural Language Processing.” Medium, Towards Data Science, 9 June 2019, towardsdatascience.com/recurrent-neural-networks-and-natural-language-processing-73af640c2aa1. Accessed 28 Feb. 2024.

“Complete Guide to the Adam Optimization Algorithm.” Built In, 2023, builtin.com/machine-learning/adam-optimization. Accessed 28 Feb. 2024.

Sewell, William. “D213 SA Webinar 5.” D213 SA Webinar. D213 SA Webinar 5, 28 Feb. 2024.