**PA TASK 2: Sentimental Analysis Using Neural Networks**

**D213 – Advanced Data Analytics**

Michael Lawson

Western Governors University

D213 – Advanced Data Analytics

PA Task 2: Sentiment Analysis Using Neural Networks

Dr. Festus Elleh

09/26/2022

Michael C. Lawson | Student ID: #001479557 | Program: MS Data Analytics (5/1/21) |

Program Mentor: Emil Stoica | 859-265-1888 (EST) | [mlaws77@wgu.edu](mailto:mlaws77@wgu.edu)

**Part I:**

A.  1.  Is it possible to predict whether a customer’s review will be positive or negative using a neural network trained on the reviews already written by customers?

2.  The goal of the analysis is to predict whether customer reviews will be negative or positive using sentiment analysis on the words from reviews in the ‘sentiment’ column.

3.  Sentiment analysis and text-classification tasks such as handwriting recognition, speech recognition, tagging and sequence labeling are accomplished using recurrent neural networks (RNN). Outputs of RNN are allowed to cycle back in as inputs. RNN allows the processing of input of any length without increasing the size of the model, but computation can be slow (Amidi, n.d.). RNN is designed for data that comes in sequences, such as a sentence.

**Part II:**

B.  1.  a. Part of exploratory data analysis (EDA) is finding and removing special characters such as ‘!@#&\*’. In this analysis, special characters were identified, and then removed leaving only letters. Characters will be converted to lower case later during tokenization. You can see the before cleaning and after groups of characters below. Only alphabetical letters remain.

# Check for unusual characters

reviews = df\_amz['review']

list\_of\_characters = []

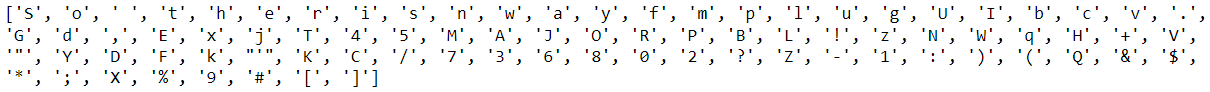
for comment in reviews:

for character in comment:

if character not in list\_of\_characters:

list\_of\_characters.append(character)

print(list\_of\_characters)



#Remove special characters

def remove\_punctuation(text):

final ="".join(u for u in text if u not in("?","'",".",":",";","!",'+','"','/','-','&','$','\*','%','#','[',']','0','1','2','3','4','5','6','7','8','9','\n', ',', '(',')'))

return final

df\_amz['review'] = df\_amz['review'].apply(remove\_punctuation)

# Check for special characters that were missed the first time

reviews = df\_amz['review']

list\_of\_characters = []

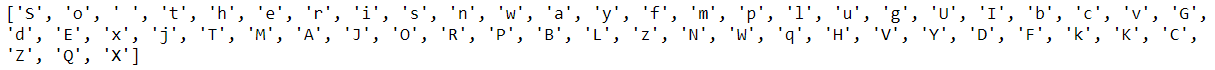
for comment in reviews:

for character in comment:

if character not in list\_of\_characters:

list\_of\_characters.append(character)

print(list\_of\_characters)



1. The number of unique words in a data set is referred to as “vocabulary size”. Tokenizer in the Keras library was used in this analysis to find the vocabulary size. The vocabulary size after removing special characters from the data was 1857. Below you can see a sample of the cleaned vocabulary all in lower case with token value assigned to each word in the data set.

Word clouds were then generated to preview the most common words in the data set. The larger a word is in the word cloud, the more that word appears in the data set. Word clouds are a great way to easily see which words in the vocabulary are most prominent. In this model’s positive reviews word cloud, the words that appear to be used the most are: ‘phone’, ‘great’ & ‘good’. The negative reviews word cloud displayed ‘Phone’, ‘work’, ‘battery’ and ‘product’ as the most used words in the negative reviews.

# Tokenize and convert characters to lower case

keras\_token = Tokenizer(oov\_token='[UNK]', lower=True)

# Call fit\_on\_texts to fit a dictionary of comments based on frequency

keras\_token.fit\_on\_texts(reviews)

# Create dictionary variable

word\_count = keras\_token.word\_index

print('Vocabulary size: ', len(word\_count))

# Print sample of word counts

list(word\_count.items())[950:975]

Text

Description automatically generated

# Generate word cloud to visualize most frequently used words

stopwords = set(STOPWORDS)

plt.figure(figsize=(12, 6))

stopwords.update(["br", "href"])

pos = " ".join(review for review in positive.review)

wordcloud = WordCloud(stopwords=stopwords).generate(pos)

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis("off")

plt.savefig('wordcloud11.png')

plt.show()

Text

Description automatically generated

# Generate word cloud to visualize most frequently used words for negative reviews

stopwords = set(STOPWORDS)

plt.figure(figsize=(12, 6))

stopwords.update(["br", "href"])

neg = " ".join(review for review in negative.review)

wordcloud = WordCloud(stopwords=stopwords).generate(neg)

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis("off")

plt.savefig('wordcloud11.png')

plt.show()

Text

Description automatically generated

1. Using the bag of words method, word embedding refers to the position of the word & words with similar meaning, from the beginning of the learned vector. The word embedding *length* is the squared root of the square root of the vocabulary size. Below, the code identifies the embedding length, (√(√1857)), as 7. Later 7 will be used as the embedding dimension when padding.

# Word embedding length

max\_sequence\_embedding=int(round(np.sqrt(vocab\_size)), 0

max\_sequence\_embedding



1. Sequence length refers to the number of words in the longest input sentence of text in the data used as text for the model. Choosing the max sequence length preserves the input data so that the model doesn’t generate conclusions that do not generalize well (Elleh, 2022). The max sequence length in this data set is 30.

The sequence length is another dimension needed for the code used for padding. Padding allows a list of sentences to be fit in a model by causing each sentence to have the same number of tokens. The reviews column was tokenized when searching for the vocabulary size. The code for padding is shown below with the embedding dimension set at 7 as discussed earlier. The max length

#Statisitcal justification for the chosen max sequence length

commentary\_length=[]

for char\_len in commentary:

commentary\_length.append(len(char\_len.split(' ')))

commentary\_max = np.max(commentary\_length)

commentary\_min = np.min(commentary\_length)

commentary\_median = np.median(commentary\_length)

print('The max sequence length would be: ', commentary\_max)

print('The min sequence length would be: ', commentary\_min)

print('The median sequence length would be: ', commentary\_median)

A screenshot of a computer

Description automatically generated with low confidence

# Padding

oov\_tok = "<OOV>"

embedding\_dim = 7

max\_length = 30

trunc\_type='post'

padding\_type='post'

B2.  Tokenizer separates the text data into smaller chunks called “tokens”. Tokens can include words, parts of words and even characters. Tokenizer helps the model during the training process by assigning a unique index to each word in the word\_index. Tokenization can also be used to replace abnormal characters, transform the text into sequences and pad the sequence lengths and transform them in preparation for the model. Below is the code and a couple different ways to visualize the vocabulary list after tokenization. The number corresponding to each word is the value assigned to that token; for example, the word ‘file’ is token number 951 (see below). Unknown words are assigned ‘[UNK]’ and token value 1. The first visualization shows how to take a sample of a range of specific token values. The second visualization displays a window with all tokens. Tokenizer can be imported from keras.

# Import Tokenizer

From tensorflow.keras.preprocessing.text import Tokenizer

#Tokenizer package

from tensorflow.keras.preprocessing.text import Tokenizer

# Tokenization

tokenizer = Tokenizer(num\_words=vocab\_size, oov\_token=oov\_tok)

tokenizer.fit\_on\_texts(X\_train)

word\_index=tokenizer.word\_index

# Print sample of word counts with assigned token values between 950-975

list(word\_count.items())[950:975]

Text

Description automatically generatedA picture containing background pattern

Description automatically generated

B3.  Each sequence in the data has a different sequence length; however, neural networks require input to have the same size and shape. This is where the Padding technique is needed for neural networks to improve model performance by preserving the shape of tensor dimensions. The longest sequence in the data set is 30, so the pad\_sequences library in Python can be used to “pad” each sequence to match. The pad\_sequences library can also be found in keras in tensorflow.keras.preprocessing.sequence. In the code shown below, padding follows the text sequence using the parameter ‘post’.

# Set the dimensions

oov\_tok = "<OOV>"

embedding\_dim = 7

max\_length = 30

trunc\_type='post'

padding\_type='post'

# Split the data into 80/20 train/test sets

X = df\_amz['review']

y = df\_amz['sentiment']

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size = 0.2, random\_state=7)

tokenizer.fit\_on\_texts(X\_train)

vocab\_size = len(tokenizer.word\_index)+1

word\_index = tokenizer.word\_index

# Padding to improve performance

train\_sequences = tokenizer.texts\_to\_sequences(df\_amz['review'])

train\_padded = pad\_sequences(sequences\_train, maxlen=max\_length, padding=padding\_type, truncating=trunc\_type)

test\_sequences = tokenizer.texts\_to\_sequences(df\_amz['review'])

test\_padded = pad\_sequences(sequences\_test, maxlen=max\_length, padding=padding\_type, truncating=trunc\_type)

A picture containing calendar

Description automatically generated

B4.

Two categories of sentiment will be used for sentiment analysis output in the neural network. ‘0’ represents negative feedback in the ‘sentiment’ column of the data, and ‘1’ represents positive feedback.

The ‘softmax’ function will calculate the relative possibilities for the final dense layer of the neural network.

*# Activate 'softmax'*

*activation = 'softmax'*

*loss = 'categorical\_crossentropy'*

*optimizer = 'adam'*

*num\_epochs = 7*

*#Define Early Stopping Monitor*

*early\_stopping\_monitor = EarlyStopping(patience=2)*

*# Create model*

*model=tf.keras.Sequential([*

*tf.keras.layers.Embedding(vocab\_size,embedding\_dim, input\_length=max\_length),*

*tf.keras.layers.GlobalAveragePooling1D(),*

*tf.keras.layers.Dense(100, activation='relu'),*

*tf.keras.layers.Dense(50, activation='relu'),*

*tf.keras.layers.Dense(2, activation=activation)*

*])*

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

B5.  Steps to prepare1:

1. Import packages/libraries
2. Import data
3. Clean the data by:
   1. removing special characters
   2. converting to lowercase
   3. removing stopwords
   4. tokenizing and padding
4. Explore the data by:
   1. checking dimensions
   2. checking statistics
   3. creating word clouds
   4. creating a histogram
   5. reviewing characters and words in the data
   6. finding the vocabulary size, max sequence size and embedding length
5. Split the data into 80/20 train/test sets
6. Apply tokenizer() to training set with the fit\_on\_texts method
7. View training set word index
8. Post-padding the data sequences to the max sequence size calculated earlier
9. Fit the model & convert both split data sets into NumPy arrays

B6.  Export data

pd.DataFrame(train\_padded).to\_csv(r"C:\Users\mlaws\OneDrive - Western Governors University\Documents\WGU\D213\training\_padded.csv")

pd.DataFrame(train\_sequences).to\_csv(r"C:\Users\mlaws\OneDrive - Western Governors University\Documents\WGU\D213\training\_label.csv")

pd.DataFrame(test\_padded).to\_csv(r"C:\Users\mlaws\OneDrive - Western Governors University\Documents\WGU\D213\testing\_padded.csv")

pd.DataFrame(test\_sequences).to\_csv(r"C:\Users\mlaws\OneDrive - Western Governors University\Documents\WGU\D213\testing\_label.csv")

**Part III:**

C.

1.

Graphical user interface, text

Description automatically generated

2.

Five layers with 18,951 trainable parameters and 0 non-trainable parameters.

Table

Description automatically generated

3.

ReLU (Rectified Linear Unit): ReLu was used on hidden layers because it is the most popular activation function which is used in hidden layer of the neural network (Chaudhary, 2020). ReLu was used for this model because it is simple to use and has less limitations than other activation functions. ReLu is standard practice and performs better than Sigmoid

Softmax: The softmax function was used for the last layer of the neural network. The advantage of Softmax is its ability to handle multiple classes. Where Sigmoid’s probabilities are produced independently, Softmax’s outputs are interrelated (Chaudhary, 2020).

The number of nodes was determined using stopping criteria in keras which runs the model with various numbers of layers and metrics until the optimal model is found with the highest accuracy and before overfitting occurs. Stopping criteria automatically prevents the model from overfitting. The number of epochs was initially set at 20 because that is what was used in the course material coding example. This number was adjusted several times until the most accurate model with the lowest loss was found. With epochs set at 8, the stopping criteria automatically determined that 5 total layers generated the optimal model.

Binary crossentropy was used because the rating sentiments were a binary 1 & 0 classification.

The optimizer ‘adam’ was chosen because it is easy and straightforward and highly adaptable. It is the most popular optimizer and generally performs better than other options. In bigger data sets, adam would be more efficient and faster than other options, but efficiency was not part of the decision-making process of this data set with only 1000 tuples.

The package early\_stopping\_monitor from the library tensorflow.keras.callbacks was used to create stopping criteria and prevents the model from overfitting. The number of epochs was set at 8 after several trials as described above, and this determined how many times the model would run with varying layers until the model was most accurate without overfitting. The stopping criteria automatically determined that 5 total layers generated the optimal model.

Graphical user interface, text, application, email

Description automatically generated

A picture containing application

Description automatically generated





The evaluation metric chosen for the NN was ‘accuracy’, which is a measurement of the percentage of times the predicted outcome matched the test outcome. The accuracy of the model is 95% with a loss metric of 69.3%. The high loss metric with the high accuracy indicates that a small number of predictions were very far from the test data, but most of the predictions were accurate.   
  
Graphical user interface, text, application, Word

Description automatically generated

**Part IV:  Model Evaluation**

D.

1.  After setting the number of epochs, the model will continue to be trained until the chosen max epochs is reached, even if the validation scores drop. An early stopping monitor was used so that the model will stop running when the validation score is no longer improving. The code is shown above in 3e. The level of patience tells the NN how many times to continue to run the model after it stops improving. A patience of 2 was chosen because 2 or 3 is generally used, and 2 produced a better model when retrying the code. The argument ‘callbacks’ passes the early stopping monitor to fit the function.

2.  Using 8 epochs as described earlier, and an early stopping monitor with a patience of 2, the model stopped at 6 epochs when the model wasn’t improving after 6 epochs of training. Graphical user interface

Description automatically generated with low confidence

Chart, line chart

Description automatically generated

3.  The early stopping monitor was used to prevent overfitting. Layers and nodes to the layers were adjusted until the optimal accuracy of 95% was achieved.

4.  The accuracy of the trained model on the test data set is 95%. The prediction loss for the model is 69.3%. The high accuracy along with the high loss is an indication that there is some data that was far from the prediction, but for the most part, the prediction was accurate to the test data.   
Graphical user interface, text, application, Word

Description automatically generated

**Part V:**

E.

Graphical user interface, text

Description automatically generated

F.  1000 customer reviews were split into 800 reviews for the training model and 200 for the test model . The sentiment analysis technique of natural language processing was used to predict if a customer’s sentiment was negative or positive. The model was trained on the training data set split and then the hyperparameters were tuned until the model performed with the highest accuracy on the test data set, which ended up being 95%.

G.  Using the NN, it is possible to predict the customer’s sentiment of the product using the words from their review with a 95% accuracy. The stakeholders could use the model to evaluate text only reviews of products to evaluate what the customer’s sentiment was, and make changes in the product or seek to market the product in a different way if reviews are overall negative.

**PART IV**

**Code has been provided as an HTML attachment with the submission.**

**References**

Amidi, Afshine. Amidi, Shervine. N.d. *Recurrent Neural Networks cheatsheet.* <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>

Elleh, Festus. May 15, 2022. *D213 Task 2 Webinar Video May 15, 2022.* <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=cedbd86a-2543-4d9d-9b0e-aec4011a606d>

Chaudhary, Mukesh. Aug 28, 2020. *Activation Functions: Sigmoid, Tanh, ReLU, Leaky ReLU, Softmax.* [https://medium.com/@cmukesh8688/activation-functions-sigmoid-tanh-relu-leaky-relu-softmax-50d3778dcea5#:~:text=Generally%20%2C%20we%20use%20ReLU%20in,use%20in%20last%20output%20layer%20.](https://medium.com/@cmukesh8688/activation-functions-sigmoid-tanh-relu-leaky-relu-softmax-50d3778dcea5%23:~:text=Generally%20%2C%20we%20use%20ReLU%20in,use%20in%20last%20output%20layer%20.)

Sewell, William. 2022. Video Presentation: [*Sentiment Analysis # 2 StopWords WordClouds*](https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=01eb7ceb-0a69-4cbb-a437-aee200d0e439)*.* <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=01eb7ceb-0a69-4cbb-a437-aee200d0e439>