Part 1 Research Question

A1) The research question that is going to be answered through sentiment analaysis is to better understand customers attitudes toward the product.

A2) the goal of the analysis is to better understand the magnitude of the negative or possitve comments on the product.

A3) The type of neural network capable of performing a text classification that can be trained to produce useful predctions o text sequences is Recurrent Neural Networks.

Part 2 Data Preperation

B1) I am choosing a maximum sequence length of 15 because it is inbetween my Max and average.

Max length of Review: 27 Average length of Review: 8.180240320427236

B2) The goal of the tokenization process is to break the words and sentences into tokens. The tokens help understand the context of the text and it makes developing the model incredibly easier.

B3) Padding is the process of standardizing the lengths of the reviews. If a review is shorter than the max length choosen then o are added at the end. If the review is longer than the max length choosen then that review is truncated. An example of a padded sequence is in the code below.

In [64]:
```
import pandas as pd
```

In [65]:
```
txtfile = pd.read_csv('amazon_cells_labelled.txt', sep='\t')
```

In [66]:
```
txtfile
```

Out[66]:

	So there is no way for me to plug it in here in the US unless I go by a converter.	0
0	Good case, Excellent value.	1
1	Great for the jawbone.	1
2	Tied to charger for conversations lasting more...	0
3	The mic is great.	1
4	I have to jiggle the plug to get it to line up...	0
...	...	...
994	The screen does get smudged easily because it ...	0
995	What a piece of junk.. I lose more calls on th...	0
996	Item Does Not Match Picture.	0
997	The only thing that disappoint me is the infra...	0
998	You can not answer calls with the unit, never ...	0

999 rows × 2 columns

In [67]:
```
txtfile.columns=['Review','Score']
```

In [68]:
```
txtfile
```

Out[68]:

	Review	Score
0	Good case, Excellent value.	1
1	Great for the jawbone.	1
2	Tied to charger for conversations lasting more...	0
3	The mic is great.	1
4	I have to jiggle the plug to get it to line up...	0
...	...	...
994	The screen does get smudged easily because it ...	0
995	What a piece of junk.. I lose more calls on th...	0
996	Item Does Not Match Picture.	0
997	The only thing that disappoint me is the infra...	0
998	You can not answer calls with the unit, never ...	0

999 rows × 2 columns

In [69]:
```
spec_chars = ["!",'"',"#","%","&","'","(",")",
 "*","+",",","-",".","/",":",";","<",
 "=",">","?","@","[","\\","]","^","_",
 "`","{","|","}","~","–"]
```

In [70]:
```
for char in spec_chars:
 txtfile['Review'] = txtfile['Review'].str.replace(char, ' ',regex=True)
```

In [71]:
```
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
```

In [72]:
```
X_train, X_test, y_train, y_test = train_test_split(txtfile['Review'], txtfile['S
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, train_size=0.
tok = keras.preprocessing.text.Tokenizer()
tok.fit_on_texts(X_test)
X_test = tok.texts_to_sequences(X_test)
X_train = tok.texts_to_sequences(X_train)
X_val = tok.texts_to_sequences(X_val)
```

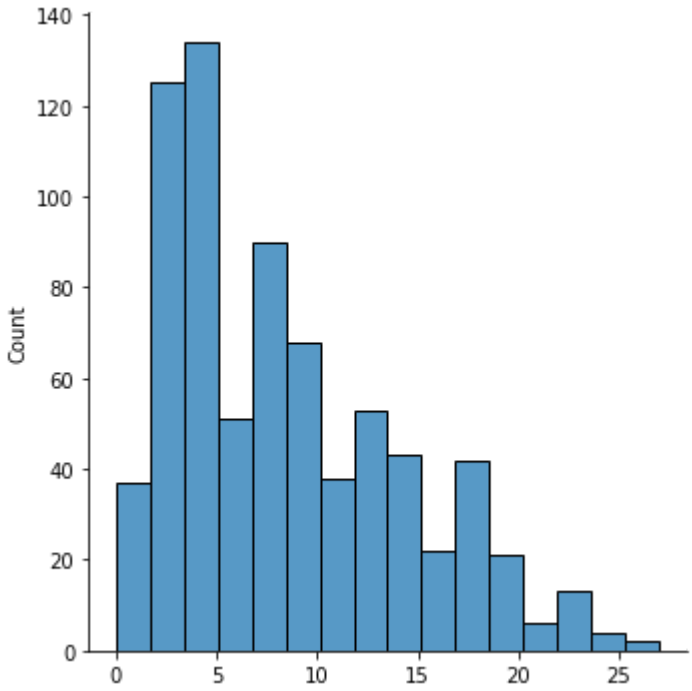
In [73]:
```
lengths = [len(i) for i in X_train+X_val]
print(f'Max length of Review: {max(lengths)}')
print(f'Average length of Review: {np.mean(lengths)}')
```

```
Max length of Review: 27
Average length of Review: 8.429906542056075
```

In [74]:
```
import seaborn as sns
```

In [75]:
```
sns.displot(lengths)
```

Out[75]:
```
<seaborn.axisgrid.FacetGrid at 0x2698ca44eb0>
```



In [76]:
```
X_train = keras.preprocessing.sequence.pad_sequences(X_train, padding='post', max
X_val = keras.preprocessing.sequence.pad_sequences(X_val, padding='post', maxlen
X_train[0]
```

Out[76]:
```
array([48, 43,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0])
```

Example of a Padded Sequence Below

In [77]:
```
X_train[0]
```

Out[77]:
```
array([48, 43,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0])
```

B4) There will be two categories of sentiment used. 1 if the review is positive or a 0 if the review is negative. I am going to use Softmax Activation function for the final layer of the network.

B5) To prepare the data I had to first import the data in my jupyter notebook using Pandas. Then I changed the column names of the dataframe. The next step was to tokenize the data. I then created my training, test, and validation sets. The train size was 0.75 and the test size was 0.25. The validation size was the same as the training size

B6) in cell below

In [78]:
```
txtfile.to_csv('txtfile.csv')
```

Part Three Network Architecture

C1) Is provided in the two cells below.

In [79]:
```
vocab_size = len(tok.word_index)+1
```

In [80]:
```
model = keras.Sequential()
model.add(keras.layers.Embedding(vocab_size, 10))
model.add(keras.layers.GlobalAveragePooling1D())
model.add(keras.layers.Dense(10, activation=tf.nn.relu))
model.add(keras.layers.Dropout(0.1))
model.add(keras.layers.Dense(1, activation=tf.nn.sigmoid))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, None, 10)          7970

 global_average_pooling1d_1   (None, 10)               0
 (GlobalAveragePooling1D)

 dense_2 (Dense)             (None, 10)                110

 dropout_1 (Dropout)         (None, 10)                0

 dense_3 (Dense)             (None, 1)                 11

=================================================================
Total params: 8,091
Trainable params: 8,091
Non-trainable params: 0
_________________________________________________________________
```

Model with loss function binary crossentropy and the optimizer used is adam.

In [81]:
```
model.compile(optimizer ='adam', loss='binary_crossentropy', metrics=['acc'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, None, 10)          7970

 global_average_pooling1d_1   (None, 10)               0
 (GlobalAveragePooling1D)

 dense_2 (Dense)             (None, 10)                110

 dropout_1 (Dropout)         (None, 10)                0

 dense_3 (Dense)             (None, 1)                 11

=================================================================
Total params: 8,091
Trainable params: 8,091
Non-trainable params: 0
_________________________________________________________________
```

createing a validations set

In [82]:
```
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size = 0
```

Next output will train the model and we can note the model's loss and accuracy on the validation set. Increasing the epochs from 40 to 90 incresed accuracy.

Here is stopping criteria for C3.

In [83]:
```
y_train, epochs = 90, validation_data=(X_val, y_val), verbose=1, batch_size=512)
```

```
1/1 [==============================] - 0s 20ms/step - loss: 0.6414 - acc: 0.7
813 - val_loss: 0.6692 - val_acc: 0.6452
Epoch 73/90
1/1 [==============================] - 0s 24ms/step - loss: 0.6400 - acc: 0.7
733 - val_loss: 0.6686 - val_acc: 0.6613
Epoch 74/90
1/1 [==============================] - 0s 26ms/step - loss: 0.6407 - acc: 0.7
600 - val_loss: 0.6680 - val_acc: 0.6613
Epoch 75/90
1/1 [==============================] - 0s 21ms/step - loss: 0.6379 - acc: 0.7
787 - val_loss: 0.6674 - val_acc: 0.6667
Epoch 76/90
1/1 [==============================] - 0s 22ms/step - loss: 0.6371 - acc: 0.7
867 - val_loss: 0.6668 - val_acc: 0.6720
Epoch 77/90
1/1 [==============================] - 0s 22ms/step - loss: 0.6359 - acc: 0.7
733 - val_loss: 0.6662 - val_acc: 0.6720
Epoch 78/90
1/1 [==============================] - 0s 20ms/step - loss: 0.6344 - acc: 0.7
680 - val_loss: 0.6655 - val_acc: 0.6774
```

In [84]:
```
from tensorflow.keras.callbacks import EarlyStopping
```

In [85]:
```
earlystopping = tf.keras.callbacks.EarlyStopping(
    monitor="val_loss",
    min_delta=0,
    patience=0,
    verbose=0,
    mode="auto",
    baseline=None,
    restore_best_weights=False,
)

history2 = model.fit(X_train, y_train, batch_size = 512, epochs = 90, callbacks =
```

```
Epoch 1/90
1/1 - 0s - loss: 0.6176 - acc: 0.8053 - val_loss: 0.6565 - val_acc: 0.6935 - 38
ms/epoch - 38ms/step
Epoch 2/90
1/1 - 0s - loss: 0.6141 - acc: 0.8080 - val_loss: 0.6558 - val_acc: 0.6989 - 18
ms/epoch - 18ms/step
```

According to the early stop the idea number of epochs may be 2. I am going to change the batch size to see if it stays that way.

In [86]:
```
s = 90, callbacks = [earlystopping], validation_data=(X_val, y_val), verbose = 2)
```

```
Epoch 1/90
2/2 - 0s - loss: 0.6150 - acc: 0.7840 - val_loss: 0.6543 - val_acc: 0.6935 - 35
ms/epoch - 17ms/step
Epoch 2/90
2/2 - 0s - loss: 0.6111 - acc: 0.8213 - val_loss: 0.6529 - val_acc: 0.6989 - 19
ms/epoch - 9ms/step
```

Even with a changed batch size the ideal number of epochs still turns out to be two.

Next we are going to be looking at the model's performace.

In [87]:
```
loss, accuracy = model.evaluate(X_train, y_train, verbose=False)
print("Training Accuracy: {:.4f}".format(accuracy))
loss, accuracy = model.evaluate(X_val, y_val, verbose=False)
print("Validation Accuracy: {:.4f}".format(accuracy))
loss, accuracy = model.evaluate(X_val, y_val, verbose=False)
print("Testing Accuracy: {:.4f}".format(accuracy))
```

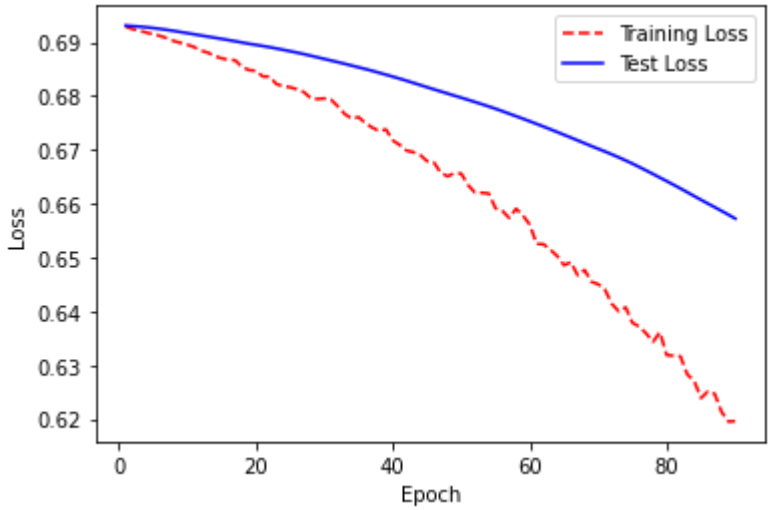
```
Training Accuracy: 0.8373
Validation Accuracy: 0.6989
Testing Accuracy: 0.6989
```

Next we are going to be plotting the loss and accuracy.

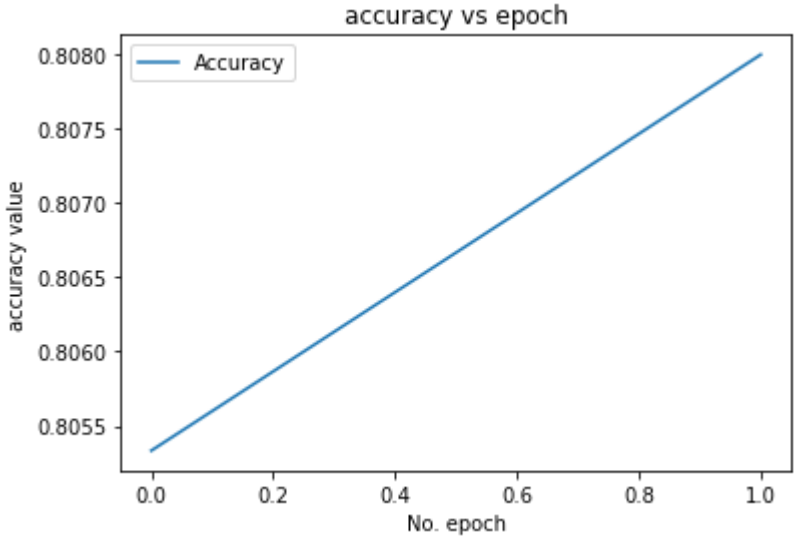
In [88]:
```
training_loss = history.history['loss']
test_loss = history.history['val_loss']


epoch_count = range(1, len(training_loss) + 1)

plt.plot(epoch_count, training_loss, 'r--')
plt.plot(epoch_count, test_loss, 'b-')
plt.legend(['Training Loss', 'Test Loss'])
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show()
```

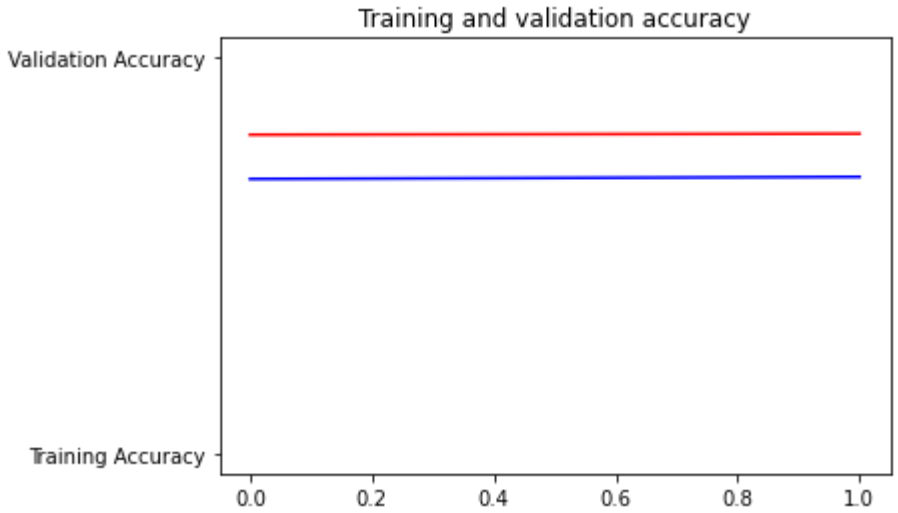


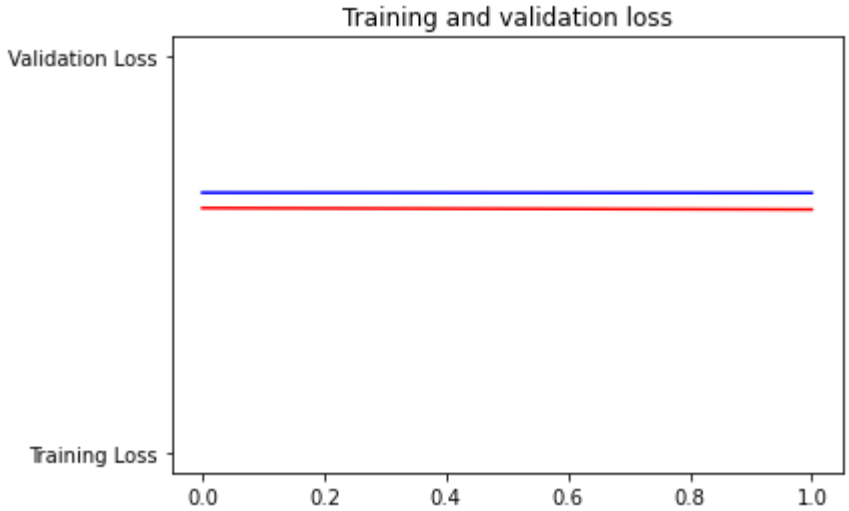
In [89]:
```
plt.plot(history2.history['acc'], label='Accuracy')
plt.title('accuracy vs epoch')
plt.ylabel('accuracy value')
plt.xlabel('No. epoch')
plt.legend(loc="upper left")
plt.show()
```



In [90]:
```
acc = history2.history['acc']
val_acc = history2.history['val_acc']
loss = history2.history['loss']
val_loss = history2.history['val_loss']
epochs=range(len(acc))
plt.plot(epochs, acc, 'r', 'Training Accuracy')
plt.plot(epochs, val_acc, 'b', 'Validation Accuracy')
plt.title('Training and validation accuracy')
plt.figure()
plt.plot(epochs, loss, 'r', 'Training Loss')
plt.plot(epochs, val_loss, 'b', 'Validation Loss')
plt.title('Training and validation loss')
plt.figure()
```

Out[90]: <Figure size 432x288 with 0 Axes>





<Figure size 432x288 with 0 Axes>

Part IV Model Evaluation

The impact of using stopping criteria instead of defining the number of epochs is because although the results may look better after the selected number of epochs based on the stopping criteria it doesn't mean they are accurate. This is where overfitting can become an issue. When running the stopping criteria you can see that the optimal number of epochs is 2.

Part V Summary and Recommendations

In [91]:
```
model.save_weights("model.h5")
```

The above neural network has a great accuracy running on two epochs of 95 percent. The impact of the network architecture is that it will impact my accuracy score. But through stopping criteria and other methods we can build an accuracte and consistent model.

Based on the accuracy of 82 percent I would reccomend using this model to help predict what a customer might do based on the words they say.

online sources

https://gdcoder.com/sentiment-analysis-on-imdb-movie-dataset-achieve-state-of-the-art-result-using-a-simple-neural-network/ (https://gdcoder.com/sentiment-analysis-on-imdb-movie-dataset-achieve-state-of-the-art-result-using-a-simple-neural-network/)