Sentiment Analysis

Douglas Ehlert

D213 Task Two

Part One: Research Question

Neural networks and natural language processing (NLP) techniques have a variety of use cases. The research question that will be explored in this analysis is whether a prediction can be made on a user's opinion of a product or service as either positive or negative, using previous reviews from a variety of industries and products. The goal of this analysis is to predict consumer sentiment of products from a variety of industries. This would be particularly important for a large organization that has multiple divisions in different segments of industry. In this case, datasets from Yelp, IMDB, and Amazon will be utilized and combined to form one sentiment analysis model. Trained neural networks are capable of processing natural language to identify sentiment; in the case of this analysis that sentiment will be whether a review was positive or negative. This analysis can lead to the organization addressing concerns that are revealed in customer reviews across disparate consumer goods and industries.

The type of neural network used in this analysis is a sequential neural network. This type of neural network is often used for sentiment analysis. Keras, which is a neural network API for Python, running on top of TensorFlow will be utilized.

Part Two: Data Preparation

To prepare the data, three datasets (Yelp reviews, IMBD reviews, and Amazon reviews) will be combined into one data frame.

```
In [30]: df = pd.concat([df_a, df_i, df_y], ignore_index=True)
         print(df.shape)
         pd.set_option('display.max_colwidth', 5000)
         df.head(100)
```

```
(3000, 3)
```

Out[30]:

	Comment	Label	Source
0	So there is no way for me to plug it in here in the US unless I go by a converter.	0	Amazon
1	Good case, Excellent value.	1	Amazon
2	Great for the jawbone.	1	Amazon
3	Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!	0	Amazon
4	The mic is great.	1	Amazon
...	...	...	...
95	Will order from them again!	1	Amazon
96	If you plan to use this in a car forget about it.	0	Amazon
97	I found this product to be waaay too big.	0	Amazon
98	Best I've found so far .... I've tried 2 other bluetooths and this one has the best quality (for both me and the listener) as well as ease of using.	1	Amazon
99	I'm very disappointed with my decision.	0	Amazon

100 rows × 3 columns

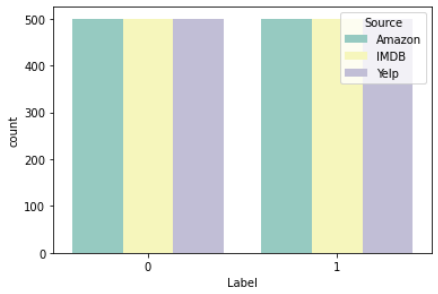
The data frame consists of 3,000 records. The records consist of 1,500 positive records (labeled as ‘1’) and 1,500 negative records (labeled as ‘0’).

```
In [60]: import seaborn as sns
         df.Label = df.Label.astype(int)
         df.info()
         sns.countplot(df['Label'], hue =df['Source'], palette ='Set3')

         #Data is split; 1,500 Positive and 1,500 Negative Reviews
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Comment  3000 non-null   object
 1   Label    3000 non-null   int32
 2   Source   3000 non-null   object
dtypes: int32(1), object(2)
memory usage: 58.7+ KB
```

Out[60]: <AxesSubplot:xlabel='Label', ylabel='count'>



It must be determined if unusual characters are present. It appears that no emojis are present in the data.

```
In [44]: import advertools as adv
         emoji_summary = adv.extract_emoji(df)
         print(emoji_summary.keys())

         dict_keys(['emoji', 'emoji_text', 'emoji_flat', 'emoji_flat_text', 'emoji_counts', 'emoji_freq', 'top_emoji', 'top_emoji_text',
         'top_emoji_groups', 'top_emoji_sub_groups', 'overview'])
```

```
In [47]: emoji_summary['emoji_counts']

Out[47]: [0, 0, 0]
```

It is immediately apparent that the data contains a mixture or upper and lower case letters. Additionally, various types of punctutation are present.

```
In [5]: #B-1.A
        pd.set_option('display.max_colwidth', 5000)
        df.head(10)
```

Out[5]:

	Comment	Label	Source
0	So there is no way for me to plug it in here in the US unless I go by a converter.	0	Amazon
1	Good case, Excellent value.	1	Amazon
2	Great for the jawbone.	1	Amazon
3	Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!	0	Amazon
4	The mic is great.	1	Amazon
5	I have to jiggle the plug to get it to line up right to get decent volume.	0	Amazon
6	If you have several dozen or several hundred contacts, then imagine the fun of sending each of them one by one.	0	Amazon
7	If you are Razr owner...you must have this!	1	Amazon
8	Needless to say, I wasted my money.	0	Amazon
9	What a waste of money and time!.	0	Amazon

The code shown below removes all punctutation and converts all text to lower case.

```
In [63]: #Convert all text to lowercase

         df['comment'] = df['Comment'].apply(lambda x: " ".join(x.lower() for x in x.split()))
         df['comment'].head()

Out[63]: 0    so there is no way for me to plug it in here in the us unless i go by a converter.
         1                                                good case, excellent value.
         2                                                     great for the jawbone.
         3          tied to charger for conversations lasting more than 45 minutes.major problems!!
         4                                                               the mic is great.
         Name: comment, dtype: object
```

```
In [64]: #remove all punctuation

         df['comment'] = df['comment'].str.replace('[^\w\s]','')
         df['comment'].head()

Out[64]: 0    so there is no way for me to plug it in here in the us unless i go by a converter
         1                                                good case excellent value
         2                                                     great for the jawbone
         3          tied to charger for conversations lasting more than 45 minutesmajor problems
         4                                                               the mic is great
         Name: comment, dtype: object
```

As a precautionary step, any emojis that escaped the initial analysis will be removed by the code below.

```
In [65]: #remove emoji's

def remove_emoji(text):
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # transport & map symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags
                               u"\U00002702-\U000027B0"
                               u"\U000024C2-\U0001F251"
                               "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)
df['comment'] = df['comment'].apply(lambda x: remove_emoji(x))
```

Next, stop words are removed. Stop words consist of words that do not add any meaning to a sentence or are “extremely common words which would appear to be of little value” (*Dropping Common Terms: Stop Words*, n.d.)

```
In [66]: #(Kosaka, 2020)

import nltk
nltk.download('stopwords')

from nltk.corpus import stopwords
stop = stopwords.words('english')
df['comment'] = df['comment'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
df.head(20)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\dbehl\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[66]:

	Comment	Label	Source	comment
0	So there is no way for me to plug it in here in the US unless I go by a converter.	0	Amazon	way plug us unless go converter
1	Good case, Excellent value.	1	Amazon	good case excellent value
2	Great for the jawbone.	1	Amazon	great jawbone
3	Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!	0	Amazon	tied charger conversations lasting 45 minutesmajor problems
4	The mic is great.	1	Amazon	mic great
5	I have to jiggle the plug to get it to line up right to get decent volume.	0	Amazon	jiggle plug get line right get decent volume
6	If you have several dozen or several hundred contacts, then imagine the fun of sending each of them one by one.	0	Amazon	several dozen several hundred contacts imagine fun sending one one
7	If you are Razr owner...you must have this!	1	Amazon	razr owneryou must
8	Needless to say, I wasted my money.	0	Amazon	needless say wasted money
9	What a waste of money and time!.	0	Amazon	waste money time
10	And the sound quality is great.	1	Amazon	sound quality great
11	He was very impressed when going from the original battery to the extended battery.	1	Amazon	impressed going original battery extended battery
12	If the two were seperated by a mere 5+ ft I started to notice excessive static and garbled sound from the headset.	0	Amazon	two seperated mere 5 ft started notice excessive static garbled sound headset
13	Very good quality though	1	Amazon	good quality though
14	The design is very odd, as the ear "clip" is not very comfortable at all.	0	Amazon	design odd ear clip comfortable
15	Highly recommend for any one who has a blue tooth phone.	1	Amazon	highly recommend one blue tooth phone
16	I advise EVERYONE DO NOT BE FOOLED!	0	Amazon	advise everyone fooled

The next step is to perform lemmatization. This will convert the words in the data to their base forms. For example, “was” is turned into “be” (*SpaCy 101: Everything You Need to Know · SpaCy Usage Documentation*, 2016)

In [67]:
```
#(SpaCy 101: Everything You Need to Know · SpaCy Usage Documentation, 2016)
#Lemmitization
import spacy
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
def space(comment):
    doc = nlp(comment)
    return " ".join([token.lemma_ for token in doc])
df['comment']= df['comment'].apply(space)
df.head(20)
```

Out[67]:

	Comment	Label	Source	comment
0	So there is no way for me to plug it in here in the US unless I go by a converter.	0	Amazon	way plug we unless go converter
1	Good case, Excellent value.	1	Amazon	good case excellent value
2	Great for the jawbone.	1	Amazon	great jawbone
3	Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!	0	Amazon	tie charger conversation last 45 minutesmajor problem
4	The mic is great.	1	Amazon	mic great
5	I have to jiggle the plug to get it to line up right to get decent volume.	0	Amazon	jiggle plug get line right get decent volume
6	If you have several dozen or several hundred contacts, then imagine the fun of sending each of them one by one.	0	Amazon	several dozen several hundred contact imagine fun send one one
7	If you are Razr owner...you must have this!	1	Amazon	razr owneryou must
8	Needless to say, I wasted my money.	0	Amazon	needless say waste money
9	What a waste of money and time!.	0	Amazon	waste money time
10	And the sound quality is great.	1	Amazon	sound quality great
11	He was very impressed when going from the original battery to the extended battery.	1	Amazon	impressed go original battery extend battery
12	If the two were seperated by a mere 5+ ft I started to notice excessive static and garbled sound from the headset.	0	Amazon	two seperate mere 5 ft start notice excessive static garbled sound headset
13	Very good quality though	1	Amazon	good quality though
14	The design is very odd, as the ear "clip" is not very comfortable at all.	0	Amazon	design odd ear clip comfortable
15	Highly recommend for any one who has a blue tooth phone.	1	Amazon	highly recommend one blue tooth phone
16	I advise EVERYONE DO NOT BE FOOLED!	0	Amazon	advise everyone fool
17	So Far So Good!.	1	Amazon	far good
18	Works great!.	1	Amazon	work great
19	It clicks into place in a way that makes you wonder how long that mechanism would last.	0	Amazon	click place way make wonder long mechanism would last

The vocabulary size of the data will be found using the Tokenizer classs in the Keras library. The vocabulary size of this data is 4,445 and is the number of unique words in the dataset.

In [68]:
```
#Calculate vocabulary size
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['comment'])
print("Vocabulary size: ", len(tokenizer.word_index)+1)
```

```
Vocabulary size:  4445
```

Determining the word embedding length is an important part of the data processing process because this helps the neural network to determine which words occur together.

Embedding length is the fourth root of the vocabular size and is calculated in the code below.

```
In [69]: #B-1.C
         #Word Embedding Length is the fourth root of the vocabulary size (4445
         max_sequence_embedding=int(round(np.sqrt(np.sqrt(4445))))
         print("Word embedding length:")
         max_sequence_embedding

         Word embedding length:

Out[69]: 8
```

Next, the maximum sequence length is determined to be 41. This is shown in the code below. The maximum length is chosen because it will preserve the available data input. Some inputs will be shorter than the maximum length; these will be addressed by padding.

```
In [13]: #B-1.D
         comment_length = []
         for char_len in df['comment']:
             comment_length.append(len(char_len.split(' ')))

         comment_max = np.max(comment_length)
         comment_min = np.min(comment_length)
         comment_median = np.median(comment_length)

         print("The maximum length of our sequences would be:", comment_max)
         print("The minimum length of our sequences would be:", comment_min)
         print("The median length of our sequences would be:", comment_median)

         max_length=41

         The maximum length of our sequences would be: 41
         The minimum length of our sequences would be: 1
         The median length of our sequences would be: 5.0
```

The data is split into train and test sets for model validation.

```
In [16]: #split into train/test test
         from sklearn.model_selection import train_test_split
         x = df['comment']
         y = df['Label']
         x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.20, random_state = 42)
```

With the data processing steps described above, the data is most likely ready for modeling. But before modelling, the data will be tokenized an additional time in order to separate the text into tokens. A unique “word index” is assigned to each word which will help the model during the training process. This is performed on the training data. At the same time, the training data will be padded. Padding improves model performance by standardizing the

input sentences. This analysis uses a post padding technique.

```
In [71]: #B2 & B3

         from tensorflow.keras.preprocessing.text import Tokenizer
         from tensorflow.keras.preprocessing.sequence import pad_sequences
         from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import LSTM,Dense, Dropout, SpatialDropout1D
         from tensorflow.keras.layers import Embedding




         #B-1.B
         from keras.preprocessing.text import Tokenizer
         tokenizer = Tokenizer()
         tokenizer.fit_on_texts(x_train)
         vocab_size = len(tokenizer.word_index) + 1
         word_index = tokenizer.word_index
         train_sequences = tokenizer.texts_to_sequences(x_train)
         test_sequences = tokenizer.texts_to_sequences(x_test)
         train_padded = pad_sequences(train_sequences, padding ='post', maxlen=max_length)
         test_padded [No Title]sequences(test_sequences, padding ='post', maxlen=max_length)

         train_padded.shape

Out[71]: (2400, 41)
```

The labels (categories) that will be used for sentiment analysis are 0 for a negative review and 1 for a positive review. These two categories will be displayed in the neural network output layer. The activation function that will be used in the final dense layer is ‘sigmoid’, which works well for binary classification.

There were numerous steps taken to prepare the data for analysis. First, the data was read into a pandas dataframe from the three data sources.

```
In [55]: import re
         import pandas as pd
         from nltk.corpus import stopwords
         import pandas as pd
         column_names=["Comment", "Label", "Source"]
         df_a = pd.read_csv(r'C:\Users\dbehl\OneDrive\WGU\D213-Task 2\amazon_cells_labelled.txt', names= column_names, sep='\t', header =
         df_a = df_a.assign(Source='Amazon')
         df_i = pd.read_csv(r"C:\Users\dbehl\OneDrive\WGU\D213-Task 2\imdb_labelled.csv", sep =',', names=column_names, header = None)
         df_i = df_i.assign(Source='IMDB')
         df_y = pd.read_csv(r"C:\Users\dbehl\OneDrive\WGU\D213-Task 2\yelp_labelled.txt", sep = '\t', names=column_names, header = None)
         df_y = df_y.assign(Source = 'Yelp')

         print(df_a.shape)
         print(df_i.shape)
         print(df_y.shape)

         (1000, 3)
         (1000, 3)
         (1000, 3)
```

Next, the data was concatated together.

```
In [56]: df = pd.concat([df_a, df_i, df_y], ignore_index=True)
         print(df.shape)
         pd.set_option('display.max_colwidth', 5000)
         df.head(100)

         (3000, 3)
```

The data was then checked for special characters.

```
In [61]: import advertools as adv
         emoji_summary = adv.extract_emoji(df)
         print(emoji_summary.keys())

         dict_keys(['emoji', 'emoji_text', 'emoji_flat', 'emoji_flat_text', 'emoji_counts', 'emoji_freq', 'top_emoji', 'top_emoji_text',
         'top_emoji_groups', 'top_emoji_sub_groups', 'overview'])

In [62]: emoji_summary['emoji_counts']

Out[62]: [0, 0, 0]

In [5]: #B-1.A
        pd.set_option('display.max_colwidth', 5000)
        df.head(10)
```

Out[5]:

	Comment	Label	Source
0	So there is no way for me to plug it in here in the US unless I go by a converter.	0	Amazon
1	Good case, Excellent value.	1	Amazon
2	Great for the jawbone.	1	Amazon
3	Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!	0	Amazon
4	The mic is great.	1	Amazon
5	I have to jiggle the plug to get it to line up right to get decent volume.	0	Amazon
6	If you have several dozen or several hundred contacts, then imagine the fun of sending each of them one by one.	0	Amazon
7	If you are Razr owner...you must have this!	1	Amazon
8	Needless to say, I wasted my money.	0	Amazon
9	What a waste of money and time!.	0	Amazon

The data was then cleaned by converting all text to lowercase, removing all punctutation, removing any emojis that were missed in in the seach for special characters, removing stop words, performing lemmitization, calculating vocabulary size, calculating word embedding length, calculating max sequence length, and finally dropping the ‘Source’ and uncleaned ‘Comment’ columns. The code for the above steps is shown in the screenshots below.

```
In [63]: #Convert all text to lowercase

         df['comment'] = df['Comment'].apply(lambda x: " ".join(x.lower() for x in x.split()))
         df['comment'].head()

Out[63]: 0    so there is no way for me to plug it in here in the us unless i go by a converter.
         1                                                 good case, excellent value.
         2                                                        great for the jawbone.
         3      tied to charger for conversations lasting more than 45 minutes.major problems!!
         4                                                              the mic is great.
         Name: comment, dtype: object

In [64]: #remove all punctuation

         df['comment'] = df['comment'].str.replace('[^\w\s]','')
         df['comment'].head()

Out[64]: 0    so there is no way for me to plug it in here in the us unless i go by a converter
         1                                                  good case excellent value
         2                                                     great for the jawbone
         3       tied to charger for conversations lasting more than 45 minutesmajor problems
         4                                                              the mic is great
         Name: comment, dtype: object
```

In [65]:
```
#remove emoji's

def remove_emoji(text):
    emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags
                           u"\U00002702-\U000027B0"
                           u"\U000024C2-\U0001F251"
                           "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)
df['comment'] = df['comment'].apply(lambda x: remove_emoji(x))
```

In [66]:
```
#(Kosaka, 2020)

import nltk
nltk.download('stopwords')

from nltk.corpus import stopwords
stop = stopwords.words('english')
df['comment'] = df['comment'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
df.head(20)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\dbehl\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[66]:

	Comment	Label	Source	comment
0	So there is no way for me to plug it in here in the US unless I go by a converter.	0	Amazon	way plug us unless go converter
1	Good case, Excellent value.	1	Amazon	good case excellent value
2	Great for the jawbone.	1	Amazon	great jawbone
3	Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!	0	Amazon	tied charger conversations lasting 45 minutesmajor problems
4	The mic is great.	1	Amazon	mic great
5	I have to jiggle the plug to get it to line up right to get decent volume.	0	Amazon	jiggle plug get line right get decent volume
6	If you have several dozen or several hundred contacts, then imagine the fun of sending each of them one by one.	0	Amazon	several dozen several hundred contacts imagine fun sending one one
7	If you are Razr owner...you must have this!	1	Amazon	razr owneryou must
8	Needless to say, I wasted my money.	0	Amazon	needless say wasted money
9	What a waste of money and time!.	0	Amazon	waste money time
10	And the sound quality is great.	1	Amazon	sound quality great
11	He was very impressed when going from the original battery to the extended battery.	1	Amazon	impressed going original battery extended battery
12	If the two were seperated by a mere 5+ ft I started to notice excessive static and garbled sound from the headset.	0	Amazon	two seperated mere 5 ft started notice excessive static garbled sound headset

In [67]:
```
#(SpaCy 101: Everything You Need to Know · SpaCy Usage Documentation, 2016)
#Lemmitization
import spacy
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
def space(comment):
    doc = nlp(comment)
    return " ".join([token.lemma_ for token in doc])
df['comment']= df['comment'].apply(space)
df.head(20)
```

Out[67]:

	Comment	Label	Source	comment
0	So there is no way for me to plug it in here in the US unless I go by a converter.	0	Amazon	way plug we unless go converter
1	Good case, Excellent value.	1	Amazon	good case excellent value
2	Great for the jawbone.	1	Amazon	great jawbone
3	Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!	0	Amazon	tie charger conversation last 45 minutesmajor problem
4	The mic is great.	1	Amazon	mic great
5	I have to jiggle the plug to get it to line up right to get decent volume.	0	Amazon	jiggle plug get line right get decent volume
6	If you have several dozen or several hundred contacts, then imagine the fun of sending each of them one by one.	0	Amazon	several dozen several hundred contact imagine fun send one one
7	If you are Razr owner...you must have this!	1	Amazon	razr owneryou must
8	Needless to say, I wasted my money.	0	Amazon	needless say waste money
9	What a waste of money and time!.	0	Amazon	waste money time
10	And the sound quality is great.	1	Amazon	sound quality great
11	He was very impressed when going from the original battery to the extended battery.	1	Amazon	impressed go original battery extend battery
12	If the two were seperated by a mere 5+ ft I started to notice excessive static and garbled sound from the headset.	0	Amazon	two seperate mere 5 ft start notice excessive static garbled sound headset
13	Very good quality though	1	Amazon	good quality though
14	The design is very odd, as the ear "clip" is not very comfortable at all.	0	Amazon	design odd ear clip comfortable
15	Highly recommend for any one who has a blue tooth phone.	1	Amazon	highly recommend one blue tooth phone
16	I advise EVERYONE DO NOT BE FOOLED!	0	Amazon	advise everyone fool
17	So Far So Good!.	1	Amazon	far good
18	Works great!.	1	Amazon	work great
19	It clicks into place in a way that makes you wonder how long that mechanism would last.	0	Amazon	click place way make wonder long mechanism would last

In [68]:
```
#Calculate vocabulary size
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['comment'])
print("Vocabulary size: ", len(tokenizer.word_index)+1)
```

```
Vocabulary size:  4445
```

In [69]:
```
#B-1.C
#Word Embedding length is the fourth root of the vocabulary size (4445
max_sequence_embedding=int(round(np.sqrt(np.sqrt(4445))))
print("Word embedding length:")
max_sequence_embedding
```

```
Word embedding length:
```

Out[69]:
```
8
```

In [13]:
```
#B-1.D
comment_length = []
for char_len in df['comment']:
    comment_length.append(len(char_len.split(' ')))

comment_max = np.max(comment_length)
comment_min = np.min(comment_length)
comment_median = np.median(comment_length)

print("The maximum length of our sequences would be:", comment_max)
print("The minimum length of our sequences would be:", comment_min)
print("The median length of our sequences would be:", comment_median)

max_length=41
```

```
The maximum length of our sequences would be: 41
The minimum length of our sequences would be: 1
The median length of our sequences would be: 5.0
```

```
In [14]: df= df.drop('Comment', axis =1)
```

```
In [15]: df=df.drop('Source', axis =1)
         df.head(10)
```

Out[15]:

	Label	comment
0	0	way plug we unless go converter
1	1	good case excellent value
2	1	great jawbone
3	0	tie charger conversation last 45 minutesmajor problem
4	1	mic great
5	0	jiggle plug get line right get decent volume
6	0	several dozen several hundred contact imagine fun send one one
7	1	razr owneryou must
8	0	needless say waste money
9	0	waste money time

The data was then split into training and test sets. 80% of the data was assigned to training the model and 20% of the data was set aside for testing.

```
In [70]: #split into train/test test
         from sklearn.model_selection import train_test_split
         x = df['comment']
         y = df['Label']
         x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.20, random_state = 42)
```

A tokenizer was then applied to the training set. At the same time, a word index of the training set was retrieved, and post-padding was applied to the sequence.

```
In [71]: #B2 & B3

         from tensorflow.keras.preprocessing.text import Tokenizer
         from tensorflow.keras.preprocessing.sequence import pad_sequences
         from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import LSTM,Dense, Dropout, SpatialDropout1D
         from tensorflow.keras.layers import Embedding




         #B-1.B
         from keras.preprocessing.text import Tokenizer
         tokenizer = Tokenizer()
         tokenizer.fit_on_texts(x_train)
         vocab_size = len(tokenizer.word_index) + 1
         word_index = tokenizer.word_index
         train_sequences = tokenizer.texts_to_sequences(x_train)
         test_sequences = tokenizer.texts_to_sequences(x_test)
         train_padded = pad_sequences(train_sequences, padding ='post', maxlen=max_length)
         test_padded = pad_sequences(test_sequences, padding ='post', maxlen=max_length)

         train_padded.shape
```

```
Out[71]: (2400, 41)
```

Finally, training and test data sets were converted to arrays using NumPy and exported as CSV files.

```
In [86]: df_x_training = pd.DataFrame(train_padded)
         df_x_testing = pd.DataFrame(test_padded)
         df_y_training = pd.DataFrame(y_train)
         df_y_testing = pd.DataFrame(y_test)

         df_x_training.to_csv(r"C:\Users\dbehl\OneDrive\WGU\D213-Task 2\Padded_x_training.csv")
         df_x_testing.to_csv(r"C:\Users\dbehl\OneDrive\WGU\D213-Task 2\Padded_x_testing.csv")
         df_y_training.to_csv(r"C:\Users\dbehl\OneDrive\WGU\D213-Task 2\Label_train.csv")
         df_y_testing.to_csv(r"C:\Users\dbehl\OneDrive\WGU\D213-Task 2\Label_test.csv")
```

Part Three: Network Architecture

The model was created, and the model summary is shown in the screenshot below.

```
In [88]:
from keras.callbacks import EarlyStopping
activation = 'sigmoid'
loss = 'binary_crossentropy'
optimizer = 'adam'

num_epochs = 20

#define early_stoopping monitor
early_stopping_monitor = EarlyStopping(patience=2)

model = keras.Sequential(
    [
        layers.Embedding(vocab_size, max_sequence_embedding, input_length=max_length),
        layers.GlobalAveragePooling1D(),
        layers.Dense(100, activation="relu"),
        layers.Dense(50, activation="relu"),
        layers.Dense(1,activation=activation),
    ]
)

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

model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 41, 8)             30896

 global_average_pooling1d_3  (None, 8)                 0
 (GlobalAveragePooling1D)

 dense_9 (Dense)             (None, 100)               900

 dense_10 (Dense)            (None, 50)                5050

 dense_11 (Dense)            (None, 1)                 51

=================================================================
Total params: 36,897
Trainable params: 36,897
Non-trainable params: 0
```

This model has five layers. The first is an Embedding layer. This layer "takes the integer-encoded vocabulary and looks up the embedding vector for each word-index. These vectors are learned as the model trains." (*Word Embeddings*, n.d.). The next layer is a 'GlobalAveragePooling1D' layer. "The GlobalAveragePooling1D layer returns a fixed-length output vector for each example by averaging over the sequence dimension. This allows the model to handle input of variable length, in the simplest way possible." (*Word Embeddings*, n.d.). Next, there are three dense layers. The dense layers are commonly found, deeply connected neural network layers. There are 36,897 total parameters and all of them are trainable.

The activation function chosen for this model is 'Sigmoid'. This was chosen for the final layer activation because the model is predicting a binary class (0 or 1) (Ronaghan, 2019). The Rectified Linear Activation ('relu') is used for the hidden layers.

Nodes are often selected by experiment. In this case, nodes and hidden layers were gradually increased until the model accuracy was not improved by additional nodes and layers.

Binary Correntropy was chosen for a loss function. This loss function is terrific for classification analysis; in this case a 0 or a 1. The 'adam' optimizer was chosen for this analysis to its high level of adaptability and its ability to reduce overfitting.

Early Stopping was utilized to stop the neural network from overfitting. At its most basic functionality, Early Stopping will monitor the validation score and stops training if accuracy does not increase after two epochs. The number of epochs is set by the patience factor (Team, n.d.).

The evaluation metric utilized for this model was 'accuracy'. This metric evaluates how well the neural network can classify comment based on the sentiment in the training data set. The test set was used to produce the test accuracy metric of 0.81. This model can predict the

output based on the input about 81% of the time.

```
In [168]: score=model.evaluate(test_padded, y_test, verbose=0)
          print(f'Test loss: {score[0]}/Test accuracy: {score[1]}')

          Test loss: 0.45785582065582275/Test accuracy: 0.8100000023841858
```

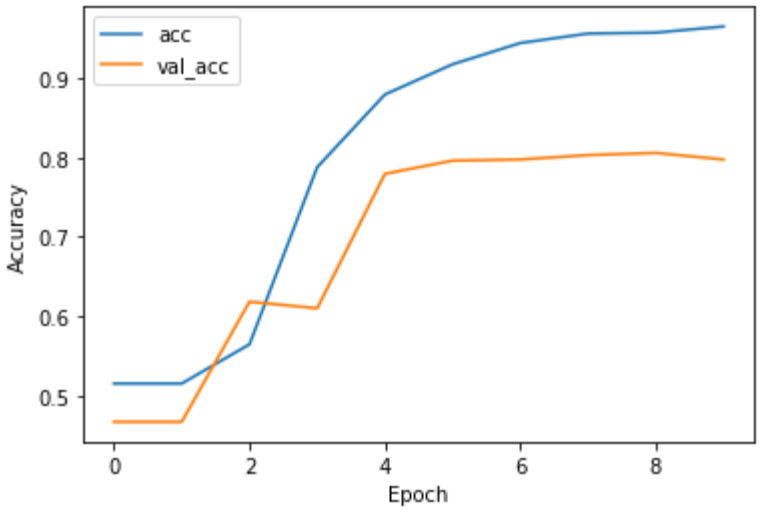
Part Four: Model Evaluation

The decision was made to use stopping criteria instead of defining the number of epochs. This method is preferable because it helps to eliminate overfitting. If the model simply ran for the standard 10 epoch, overfitting is a likely result, and the model must then be manually adjusted. The epoch number was defined as 20 and stop criterion was implemented with a of "patience=2". This results in the cessation of training when the validation score does not improve for two epochs.

```
In [167]: history = model.fit(train_padded, y_train, epochs=num_epochs,
                               validation_split = 0.3, callbacks=[early_stopping_monitor], verbose = True)

          Epoch 1/20
          53/53 [==============================] - 1s 5ms/step - loss: 0.6932 - accuracy: 0.5149 - val_loss: 0.6941 - val_accuracy: 0.466
          7
          Epoch 2/20
          53/53 [==============================] - 0s 3ms/step - loss: 0.6918 - accuracy: 0.5149 - val_loss: 0.6935 - val_accuracy: 0.466
          7
          Epoch 3/20
          53/53 [==============================] - 0s 3ms/step - loss: 0.6798 - accuracy: 0.5643 - val_loss: 0.6702 - val_accuracy: 0.618
          1
          Epoch 4/20
          53/53 [==============================] - 0s 3ms/step - loss: 0.5893 - accuracy: 0.7875 - val_loss: 0.6162 - val_accuracy: 0.609
          7
          Epoch 5/20
          53/53 [==============================] - 0s 3ms/step - loss: 0.4090 - accuracy: 0.8792 - val_loss: 0.4753 - val_accuracy: 0.779
          2
          Epoch 6/20
          53/53 [==============================] - 0s 2ms/step - loss: 0.2677 - accuracy: 0.9173 - val_loss: 0.4586 - val_accuracy: 0.795
          8
          Epoch 7/20
          53/53 [==============================] - 0s 2ms/step - loss: 0.1927 - accuracy: 0.9440 - val_loss: 0.4673 - val_accuracy: 0.797
          2
          Epoch 8/20
          53/53 [==============================] - 0s 2ms/step - loss: 0.1499 - accuracy: 0.9560 - val_loss: 0.4779 - val_accuracy: 0.802
          8
          Epoch 9/20
          53/53 [==============================] - 0s 3ms/step - loss: 0.1245 - accuracy: 0.9571 - val_loss: 0.4966 - val_accuracy: 0.805
          6
          Epoch 10/20
          53/53 [==============================] - 0s 3ms/step - loss: 0.1139 - accuracy: 0.9649 - val_loss: 0.5299 - val_accuracy: 0.797
          2
```

The following visualization shows the 'val_accuracy' score remaining relatively flat after the fifth epoch. This matches the summary from the model fitting shown in the above code snippet.



Overfitting occurs when the model learns the training data set too well and performs well on the training set but does not perform well on the test set. The measures taken to address overfitting were to start with a smaller neural network and then increase the capacity by adding nodes and layers. The analysis also provided an accuracy metric on test data that had not been evaluated by the model during training. This provides an accurate accuracy metric.

The final model performed well with an accuracy score of 0.81 and a prediction loss of .46. The accuracy score indicates that the model will predict the correct sentiment 81% of the time.

```
In [168]: score=model.evaluate(test_padded, y_test, verbose=0)
          print(f'Test loss: {score[0]}/Test accuracy: {score[1]}')

          Test loss: 0.45785582065582275/Test accuracy: 0.8100000023841858
```

Part Five: Summary and Recommendations

The code used to save the model is shown below.

```
In [171]: model.save('D213-Task 2- Sentiment Analysis.h5')
```

This model utilized 2,400 customer reviews as an input for training. The model was then tested on 600 reviews that had not evaluated before; this provided an accuracy metric that is

technically sound. The model can predict sentiment (positive or negative) with 81% accuracy from a written review. This model could be tuned to perform predictions on new customer reviews to help business stakeholders determine customer sentiment. Utilizing a sequential neural network allowed the model to detect sentiment with an acceptable degree of accuracy.

Based on the results of this analysis, it is recommended that the organization utilize the model to address customer concerns that are discovered through sentiment analysis of their submitted reviews.

References

Dropping common terms: stop words. (n.d.). Nlp.stanford.edu. https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html

Kosaka, M. (2020, November 23). *Cleaning & Preprocessing Text Data for Sentiment Analysis*. Medium. https://towardsdatascience.com/cleaning-preprocessing-text-data-for-sentiment-analysis-382a41f150d6

Ronaghan, S. (2019, August 1). *Deep Learning: Which Loss and Activation Functions should I use?* Medium. https://towardsdatascience.com/deep-learning-which-loss-and-activation-functions-should-i-use-ac02f1c56aa8

spaCy 101: Everything you need to know · spaCy Usage Documentation. (2016). SpaCy 101: Everything You Need to Know. https://spacy.io/usage/spacy-101

Team, K. (n.d.). *Keras documentation: EarlyStopping*. Keras.io. https://keras.io/api/callbacks/early_stopping/

Word embeddings, (n.d.). TensorFlow. Retrieved November 23, 2022, from https://www.tensorflow.org/text/guide/word_embeddings#:~:text=The%20Embedding%20layer%20takes%20the

In [53]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [54]:
```
import pandas as pd
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense, Dropout, SpatialDropout1D
from tensorflow.keras.layers import Embedding
from tensorflow.keras import layers
import tensorflow as tf
import tensorflow
import warnings


from tensorflow import keras
from keras.layers import Dense
```

In [55]:
```
import re
import pandas as pd
from nltk.corpus import stopwords
import pandas as pd
column_names=["Comment", "Label", "Source"]
df_a = pd.read_csv(r'C:\Users\dbehl\OneDrive\WGU\D213-Task 2\amazon_cells_labelle
df_a = df_a.assign(Source='Amazon')
df_i = pd.read_csv(r"C:\Users\dbehl\OneDrive\WGU\D213-Task 2\imdb_labelled.csv",
df_i = df_i.assign(Source='IMDB')
df_y = pd.read_csv(r"C:\Users\dbehl\OneDrive\WGU\D213-Task 2\yelp_labelled.txt",
df_y = df_y.assign(Source = 'Yelp')

print(df_a.shape)
print(df_i.shape)
print(df_y.shape)
```

```
(1000, 3)
(1000, 3)
(1000, 3)
```

In [56]:
```
df = pd.concat([df_a, df_i, df_y], ignore_index=True)
print(df.shape)
pd.set_option('display.max_colwidth', 5000)
df.head(100)
```

```
(3000, 3)
```

Out[56]:

	Comment	Label	Source
0	So there is no way for me to plug it in here in the US unless I go by a converter.	0	Amazon
1	Good case, Excellent value.	1	Amazon
2	Great for the jawbone.	1	Amazon
3	Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!	0	Amazon
4	The mic is great.	1	Amazon
...	...	...	...
95	Will order from them again!	1	Amazon
96	If you plan to use this in a car forget about it.	0	Amazon
97	I found this product to be waaay too big.	0	Amazon
98	Best I've found so far .... I've tried 2 other bluetooths and this one has the best quality (for both me and the listener) as well as ease of using.	1	Amazon
99	I'm very disappointed with my decision.	0	Amazon

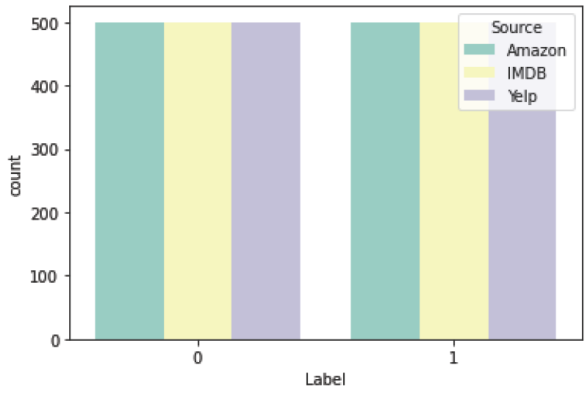
100 rows × 3 columns

In [60]:
```
import seaborn as sns
df.Label = df.Label.astype(int)
df.info()
sns.countplot(df['Label'], hue =df['Source'], palette ='Set3')

#Data is split; 1,500 Positive and 1,500 Negative Reviews
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   Comment  3000 non-null   object
 1   Label    3000 non-null   int32 
 2   Source   3000 non-null   object
dtypes: int32(1), object(2)
memory usage: 58.7+ KB
```

Out[60]: <AxesSubplot:xlabel='Label', ylabel='count'>



In [61]:
```
import advertools as adv
emoji_summary = adv.extract_emoji(df)
print(emoji_summary.keys())
```

```
dict_keys(['emoji', 'emoji_text', 'emoji_flat', 'emoji_flat_text', 'emoji_count
s', 'emoji_freq', 'top_emoji', 'top_emoji_text', 'top_emoji_groups', 'top_emoji
_sub_groups', 'overview'])
```

In [62]:
```
emoji_summary['emoji_counts']
```

Out[62]: [0, 0, 0]

In [5]:
```
#B-1.A
pd.set_option('display.max_colwidth', 5000)
df.head(10)
```

Out[5]:

	Comment	Label	Source
0	So there is no way for me to plug it in here in the US unless I go by a converter.	0	Amazon
1	Good case, Excellent value.	1	Amazon
2	Great for the jawbone.	1	Amazon
3	Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!	0	Amazon
4	The mic is great.	1	Amazon
5	I have to jiggle the plug to get it to line up right to get decent volume.	0	Amazon
6	If you have several dozen or several hundred contacts, then imagine the fun of sending each of them one by one.	0	Amazon
7	If you are Razr owner...you must have this!	1	Amazon
8	Needless to say, I wasted my money.	0	Amazon
9	What a waste of money and time!.	0	Amazon

In [63]:
```
#Convert all text to lowercase

df['comment'] = df['Comment'].apply(lambda x: " ".join(x.lower() for x in x.split
df['comment'].head()
```

Out[63]:
```
0    so there is no way for me to plug it in here in the us unless i go by a co
nverter.
1                                                          good case, excellen
t value.
2                                                               great for the
jawbone.
3       tied to charger for conversations lasting more than 45 minutes.major pr
oblems!!
4                                                                  the mic i
s great.
Name: comment, dtype: object
```

In [64]:
```
#remove all punctuation

df['comment'] = df['comment'].str.replace('[^\w\s]','')
df['comment'].head()
```

Out[64]:
```
0    so there is no way for me to plug it in here in the us unless i go by a co
nverter
1                                                          good case excellen
t value
2                                                              great for the
jawbone
3         tied to charger for conversations lasting more than 45 minutesmajor p
roblems
4                                                                  the mic i
s great
Name: comment, dtype: object
```

In [35]:
```
#remove emoji's

def remove_emoji(text):
    emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags
                           u"\U00002702-\U000027B0"
                           u"\U000024C2-\U0001F251"
                           "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)
df['comment'] = df['comment'].apply(lambda x: remove_emoji(x))
```

In [66]:
```
#(Kosaka, 2020)

import nltk
nltk.download('stopwords')

from nltk.corpus import stopwords
stop = stopwords.words('english')
df['comment'] = df['comment'].apply(lambda x: " ".join(x for x in x.split() if x
df.head(20)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\dbehl\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[66]:

	Comment	Label	Source	comment
0	So there is no way for me to plug it in here in the US unless I go by a converter.	0	Amazon	way plug us unless go converter
1	Good case, Excellent value.	1	Amazon	good case excellent value
2	Great for the jawbone.	1	Amazon	great jawbone
3	Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!	0	Amazon	tied charger conversations lasting 45 minutesmajor problems
4	The mic is great.	1	Amazon	mic great
5	I have to jiggle the plug to get it to line up right to get decent volume.	0	Amazon	jiggle plug get line right get decent volume
6	If you have several dozen or several hundred contacts, then imagine the fun of sending each of them one by one.	0	Amazon	several dozen several hundred contacts imagine fun sending one one
7	If you are Razr owner...you must have this!	1	Amazon	razr owneryou must
8	Needless to say, I wasted my money.	0	Amazon	needless say wasted money
9	What a waste of money and time!.	0	Amazon	waste money time
10	And the sound quality is great.	1	Amazon	sound quality great
11	He was very impressed when going from the original battery to the extended battery.	1	Amazon	impressed going original battery extended battery
12	If the two were seperated by a mere 5+ ft I started to notice excessive static and garbled sound from the headset.	0	Amazon	two seperated mere 5 ft started notice excessive static garbled sound headset
13	Very good quality though	1	Amazon	good quality though
14	The design is very odd, as the ear "clip" is not very comfortable at all.	0	Amazon	design odd ear clip comfortable
15	Highly recommend for any one who has a blue tooth phone.	1	Amazon	highly recommend one blue tooth phone
16	I advise EVERYONE DO NOT BE FOOLED!	0	Amazon	advise everyone fooled
17	So Far So Good!.	1	Amazon	far good
18	Works great!.	1	Amazon	works great

	Comment	Label	Source	comment
19	It clicks into place in a way that makes you wonder how long that mechanism would last.	0	Amazon	clicks place way makes wonder long mechanism would last

In [67]:
```
#(SpaCy 101: Everything You Need to Know · SpaCy Usage Documentation, 2016)
#Lemmitization
import spacy
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
def space(comment):
    doc = nlp(comment)
    return " ".join([token.lemma_ for token in doc])
df['comment']= df['comment'].apply(space)
df.head(20)
```

Out[67]:

	Comment	Label	Source	comment
0	So there is no way for me to plug it in here in the US unless I go by a converter.	0	Amazon	way plug we unless go converter
1	Good case, Excellent value.	1	Amazon	good case excellent value
2	Great for the jawbone.	1	Amazon	great jawbone
3	Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!	0	Amazon	tie charger conversation last 45 minutesmajor problem
4	The mic is great.	1	Amazon	mic great
5	I have to jiggle the plug to get it to line up right to get decent volume.	0	Amazon	jiggle plug get line right get decent volume
6	If you have several dozen or several hundred contacts, then imagine the fun of sending each of them one by one.	0	Amazon	several dozen several hundred contact imagine fun send one one
7	If you are Razr owner...you must have this!	1	Amazon	razr owneryou must
8	Needless to say, I wasted my money.	0	Amazon	needless say waste money
9	What a waste of money and time!.	0	Amazon	waste money time
10	And the sound quality is great.	1	Amazon	sound quality great
11	He was very impressed when going from the original battery to the extended battery.	1	Amazon	impressed go original battery extend battery
12	If the two were seperated by a mere 5+ ft I started to notice excessive static and garbled sound from the headset.	0	Amazon	two seperate mere 5 ft start notice excessive static garbled sound headset
13	Very good quality though	1	Amazon	good quality though
14	The design is very odd, as the ear "clip" is not very comfortable at all.	0	Amazon	design odd ear clip comfortable
15	Highly recommend for any one who has a blue tooth phone.	1	Amazon	highly recommend one blue tooth phone
16	I advise EVERYONE DO NOT BE FOOLED!	0	Amazon	advise everyone fool
17	So Far So Good!.	1	Amazon	far good
18	Works great!.	1	Amazon	work great
19	It clicks into place in a way that makes you wonder how long that mechanism would last.	0	Amazon	click place way make wonder long mechanism would last

In [68]:
```
#Calculate vocabulary size
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['comment'])
print("Vocabulary size: ", len(tokenizer.word_index)+1)
```

```
Vocabulary size:  4445
```

In [69]:
```
#B-1.C
#Word Embedding length is the fourth root of the vocabulary size (4445
max_sequence_embedding=int(round(np.sqrt(np.sqrt(4445))))
print("Word embedding length:")
max_sequence_embedding
```

```
Word embedding length:
```

Out[69]:
```
8
```

In [13]:
```
#B-1.D
comment_length = []
for char_len in df['comment']:
    comment_length.append(len(char_len.split(' ')))

comment_max = np.max(comment_length)
comment_min = np.min(comment_length)
comment_median = np.median(comment_length)

print("The maximum length of our sequences would be:", comment_max)
print("The minimum length of our sequences would be:", comment_min)
print("The median length of our sequences would be:", comment_median)

max_length=41
```

```
The maximum length of our sequences would be: 41
The minimum length of our sequences would be: 1
The median length of our sequences would be: 5.0
```

In [14]:
```
df= df.drop('Comment', axis =1)
```

In [15]:
```
df=df.drop('Source', axis =1)
df.head(10)
```

Out[15]:

	Label	comment
0	0	way plug we unless go converter
1	1	good case excellent value
2	1	great jawbone
3	0	tie charger conversation last 45 minutesmajor problem
4	1	mic great
5	0	jiggle plug get line right get decent volume
6	0	several dozen several hundred contact imagine fun send one one
7	1	razr owneryou must
8	0	needless say waste money
9	0	waste money time

In [70]:
```
#split into train/test test
from sklearn.model_selection import train_test_split
x = df['comment']
y = df['Label']
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.20, randon
```

In [71]:
```
#B2 & B3

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense, Dropout, SpatialDropout1D
from tensorflow.keras.layers import Embedding




#B-1.B
from keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer()
tokenizer.fit_on_texts(x_train)
vocab_size = len(tokenizer.word_index) + 1
word_index = tokenizer.word_index
train_sequences = tokenizer.texts_to_sequences(x_train)
test_sequences = tokenizer.texts_to_sequences(x_test)
train_padded = pad_sequences(train_sequences, padding ='post', maxlen=max_length)
test_padded = pad_sequences(test_sequences, padding ='post', maxlen=max_length)

train_padded.shape
```

Out[71]:
```
(2400, 41)
```

In [165]:
```
print(word_index)
```

```
{'not': 1, 'good': 2, 'movie': 3, 'great': 4, 'film': 5, 'do': 6, 'phone': 7,
'well': 8, 'bad': 9, 'one': 10, 'time': 11, 'work': 12, 'like': 13, 'i': 14,
'make': 15, 'go': 16, 'food': 17, 'place': 18, 'get': 19, 'service': 20, 'rea
lly': 21, 'would': 22, 'see': 23, 'use': 24, 'love': 25, 'even': 26, 'also':
27, 'quality': 28, 'ever': 29, 'could': 30, 'back': 31, 'come': 32, 'think':
33, 've': 34, 'look': 35, 'character': 36, 'sound': 37, 'recommend': 38, 'sa
y': 39, 'm': 40, 'much': 41, 'give': 42, 'headset': 43, 'price': 44, 'produc
t': 45, 'nice': 46, 'never': 47, 'thing': 48, 'waste': 49, 'excellent': 50,
'battery': 51, 'pretty': 52, 'try': 53, 'way': 54, 'buy': 55, 'still': 56, 'f
ind': 57, 'watch': 58, 'case': 59, 'enough': 60, 'feel': 61, 'people': 62, 'k
now': 63, 'be': 64, 'ear': 65, 'eat': 66, 'act': 67, 'minute': 68, 'first': 6
9, 'order': 70, 'take': 71, 'year': 72, 'scene': 73, 'want': 74, 'every': 75,
'two': 76, 'can': 77, '2': 78, 'little': 79, 'right': 80, 'call': 81, 'everyt
hing': 82, 'star': 83, 'lot': 84, 's': 85, 'play': 86, 'poor': 87, 'actor': 8
8, 'friendly': 89, 'long': 90, 'wait': 91, 'problem': 92, 'amazing': 93, 'rea
l': 94, 'happy': 95, 'enjoy': 96, 'piece': 97, 'nothing': 98, 'we': 99, 'expe
rience': 100, 'will': 101, 'far': 102, 'definitely': 103, 'story': 104, 'terr
ible': 105, 'day': 106, 'show': 107, 'end': 108, 'many': 109, 'restaurant': 1
10, 'money': 111, 'life': 112, '10': 113, 'plot': 114, 'easy': 115, 'new': 11
```

In [72]:
```
train_padded
```

Out[72]:
```
array([[ 369,  105,  201, ...,    0,    0,    0],
       [  27,    7,    6, ...,    0,    0,    0],
       [1037,    6,    1, ...,    0,    0,    0],
       ...,
       [ 686, 3857, 3858, ...,    0,    0,    0],
       [  10, 1556,  204, ...,    0,    0,    0],
       [ 122,   11,   55, ...,    0,    0,    0]])
```

In [74]:
```
np.asarray(y_train)
np.asarray(y_test)
np.asarray(x_test)
np.asarray(x_train)
```

Out[74]:
```
array(['obviously terrible customer service get pay',
       'also phone do not seem accept anything except cbr mp3s preferably rip w
indows medium player',
       'usually do not like headband one lightweight do not mess hair',
       ...,
       'chemistry ben affleck sandra bullock film could not understand would co
nsider even leave wifetobe chick supposedly knock',
       'one favourite director one talented director history cinema',
       'last time buy'], dtype=object)
```

In [86]:
```
df_x_training = pd.DataFrame(train_padded)
df_x_testing = pd.DataFrame(test_padded)
df_y_training = pd.DataFrame(y_train)
df_y_testing = pd.DataFrame(y_test)

df_x_training.to_csv(r"C:\Users\dbehl\OneDrive\WGU\D213-Task 2\Padded_x_training.
df_x_testing.to_csv(r"C:\Users\dbehl\OneDrive\WGU\D213-Task 2\Padded_x_testing.cs
df_y_training.to_csv(r"C:\Users\dbehl\OneDrive\WGU\D213-Task 2\Label_train.csv")
df_y_testing.to_csv(r"C:\Users\dbehl\OneDrive\WGU\D213-Task 2\Label_test.csv")
```

In [169]:

```
from keras.callbacks import EarlyStopping
activation = 'sigmoid'
loss = 'binary_crossentropy'
optimizer = 'adam'

num_epochs = 20

#define early_stoopping monitor
early_stopping_monitor = EarlyStopping(patience=2)

model = keras.Sequential(
    [
        layers.Embedding(vocab_size, max_sequence_embedding, input_length=max_ler
        layers.GlobalAveragePooling1D(),
        layers.Dense(100, activation="relu"),
        layers.Dense(50, activation="relu"),
        layers.Dense(1,activation=activation),
    ]
)

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

model.summary()
```

```
Model: "sequential_32"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_32 (Embedding)    (None, 41, 8)             30896

 global_average_pooling1d_32  (None, 8)                0
  (GlobalAveragePooling1D)

 dense_107 (Dense)           (None, 100)               900

 dense_108 (Dense)           (None, 50)                5050

 dense_109 (Dense)           (None, 1)                 51

=================================================================
Total params: 36,897
Trainable params: 36,897
Non-trainable params: 0
_________________________________________________________________
```

```
In [76]: print(y_train.shape)
```

```
(2400,)
```

```
In [77]: print(train_padded.shape)
```

```
(2400, 41)
```

```
In [78]: type(y_train)
         np.asarray(y_train)
```

```
Out[78]: array([0, 0, 1, ..., 0, 1, 0])
```

```
In [79]: type(train_padded)
```

```
Out[79]: numpy.ndarray
```

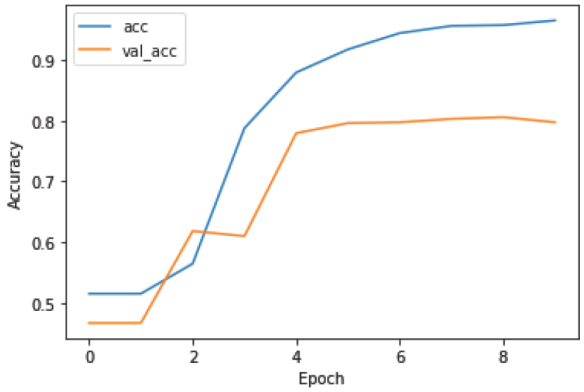
```
In [167]: history = model.fit(train_padded, y_train, epochs=num_epochs,
                              validation_split = 0.3, callbacks=[early_stopping_monitor], ver
```

```
Epoch 1/20
53/53 [==============================] - 1s 5ms/step - loss: 0.6932 - accuracy:
0.5149 - val_loss: 0.6941 - val_accuracy: 0.4667
Epoch 2/20
53/53 [==============================] - 0s 3ms/step - loss: 0.6918 - accuracy:
0.5149 - val_loss: 0.6935 - val_accuracy: 0.4667
Epoch 3/20
53/53 [==============================] - 0s 3ms/step - loss: 0.6798 - accuracy:
0.5643 - val_loss: 0.6702 - val_accuracy: 0.6181
Epoch 4/20
53/53 [==============================] - 0s 3ms/step - loss: 0.5893 - accuracy:
0.7875 - val_loss: 0.6162 - val_accuracy: 0.6097
Epoch 5/20
53/53 [==============================] - 0s 3ms/step - loss: 0.4090 - accuracy:
0.8792 - val_loss: 0.4753 - val_accuracy: 0.7792
Epoch 6/20
53/53 [==============================] - 0s 2ms/step - loss: 0.2677 - accuracy:
0.9173 - val_loss: 0.4586 - val_accuracy: 0.7958
Epoch 7/20
53/53 [==============================] - 0s 2ms/step - loss: 0.1927 - accuracy:
0.9440 - val_loss: 0.4673 - val_accuracy: 0.7972
Epoch 8/20
53/53 [==============================] - 0s 2ms/step - loss: 0.1499 - accuracy:
0.9560 - val_loss: 0.4779 - val_accuracy: 0.8028
Epoch 9/20
53/53 [==============================] - 0s 3ms/step - loss: 0.1245 - accuracy:
0.9571 - val_loss: 0.4966 - val_accuracy: 0.8056
Epoch 10/20
53/53 [==============================] - 0s 3ms/step - loss: 0.1139 - accuracy:
0.9649 - val_loss: 0.5299 - val_accuracy: 0.7972
```

```
In [168]: score=model.evaluate(test_padded, y_test, verbose=0)
          print(f'Test loss: {score[0]}/Test accuracy: {score[1]}')
```

```
Test loss: 0.45785582065582275/Test accuracy: 0.8100000023841858
```

In [170]:
```
plt.plot(history.history['accuracy'], label='acc')
plt.plot(history.history['val_accuracy'], label='val_acc')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
plt.savefig("Accuracy plot.jpg")
```




```
<Figure size 432x288 with 0 Axes>
```

In [171]:
```
model.save('D213-Task 2- Sentiment Analysis.h5')
```