In [104]:
```
import numpy as np
from sklearn.model_selection import train_test_split


from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
import tensorflow as tf
import keras


import pandas as pd
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences #for paddin
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense, Dropout, SpatialDropout1D
from tensorflow.keras.layers import Embedding

import nltk
import re
from nltk.corpus import stopwords
from nltk import word_tokenize
from nltk.stem import PorterStemmer
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
```

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

Out[104]:
```
True
```

In [7]:
```
nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/inde
x.xml (https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml)
```

Out[7]:
```
True
```

In [8]:
```
import pandas as pd
import gzip
import json

def parse(path):
  g = gzip.open('C:/Users/blien/Documents/WGU/D213/Task 2/Prime_Pantry_5.json
  for l in g:
    yield json.loads(l)

def getDF(path):
  i = 0
  df = {}
  for d in parse(path):
    df[i] = d
    i += 1
  return pd.DataFrame.from_dict(df, orient='index')

df = getDF('C:/Users/blien/Documents/WGU/D213/Task 2/Prime_Pantry_5.json.gz')
df.head(5)
```

Out[8]:

	overall	verified	reviewTime	reviewerID	asin	reviewerName	reviewText	s
0	4.0	True	09 24, 2015	A31Y9ELLA1JUB0	B0000DIWNI	Her Royal Peepness Princess HoneyBunny Blayze	I purchased this Saran premium plastic wrap af...	(
1	5.0	True	06 23, 2015	A2FYW9VZ0AMXKY	B0000DIWNI	Mary	I am an avid cook and baker. Saran Premium Pl...	
2	5.0	True	06 13, 2015	A1NE43T0OM6NNX	B0000DIWNI	Tulay C	Good wrap, keeping it in the fridge makes it e...	(
3	4.0	True	06 3, 2015	AHTCPGK2CNPKU	B0000DIWNI	OmaShops	I prefer Saran wrap over other brands. It does...	(
4	5.0	True	04 20, 2015	A25SIBTMVXLB59	B0000DIWNI	Nitemanslim	Thanks	F

In [9]:
```
df.shape
```

Out[9]:
```
(137788, 12)
```

In [10]:
```
df.describe()
```

Out[10]:

	overall	unixReviewTime
count	137788.000000	1.377880e+05
mean	4.546223	1.473494e+09
std	0.907137	3.252124e+07
min	1.000000	1.144541e+09
25%	4.000000	1.453594e+09
50%	5.000000	1.475021e+09
75%	5.000000	1.495498e+09
max	5.000000	1.538611e+09

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 137788 entries, 0 to 137787
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   overall         137788 non-null  float64
 1   verified        137788 non-null  bool   
 2   reviewTime      137788 non-null  object 
 3   reviewerID      137788 non-null  object 
 4   asin            137788 non-null  object 
 5   reviewerName    137772 non-null  object 
 6   reviewText      137611 non-null  object 
 7   summary         137727 non-null  object 
 8   unixReviewTime  137788 non-null  int64  
 9   vote            9437 non-null    object 
 10  image           665 non-null     object 
 11  style           1152 non-null    object 
dtypes: bool(1), float64(1), int64(1), object(9)
memory usage: 12.7+ MB
```

In [12]:
```
df_final = df.drop(['verified', 'reviewTime', 'reviewerID', 'asin', 'reviewer
```

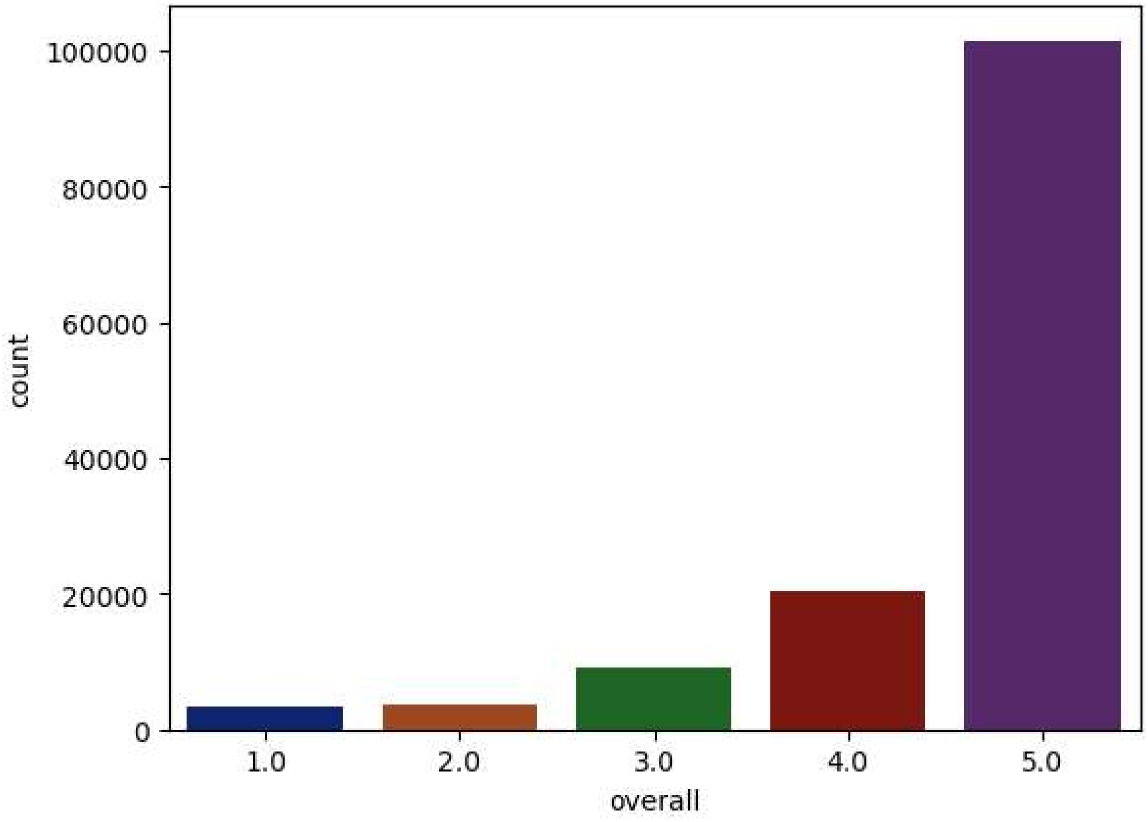
In [13]:
```
pd.set_option('display.max_colwidth', 7000)
df_final.head(5)
```

Out[13]:

	overall	reviewText
0	4.0	I purchased this Saran premium plastic wrap after trying Reynolds press and seal wrap which I would never use again.. There is less static cling to this wrap than I remember. To me this is a good thing because it doesn't stick to its self .\n\nThis is my typical complaint with all plastic wraps. When trying to cut them they ball all up and are useless. However they have improved this. Now Saran clings to the bowl or plate you wish to cover.\n\nNow if only they could improve the cutters on the boxes so that the cutters actually cut and scissors weren't required would be better..
1	5.0	I am an avid cook and baker. Saran Premium Plastic Wrap is a staple in my pantry and the only plastic wrap I purchase. I have tried other brands including Glad and have consistently found Saran Wrap to be far superior.\n\nSaran Wrap is easy to use. It's cutting bar cuts the wrap smoothly and the end of the wrap is easy to remove from the roll, doesn't get all sticky and impossible to remove like on some other brands. Some of the comments mention that Saran Wrap does not cling, but I have never had this problem when using this wrap at room temperature, in the refrigerator, or in the microwave.\n\nKeeps food stuffs fresh and wonderful to use to separate layers of freshly baked cookies and brownies stored in containers in the freezer. I also use this to tightly wrap partially used fruits and vegetable like apples and avocadoes. Saran Wrap excels at keeping these partially used fruits and vegetables fresh with no browning. Another great Amazon Prime Pantry value.
2	5.0	Good wrap, keeping it in the fridge makes it easier to tear. Learned this trick from my sister.
3	4.0	I prefer Saran wrap over other brands. It doesn't cling as well to dishes, but it tangles less when pulling it out of the box.
4	5.0	Thanks

In [14]:
```python
import seaborn as sns
sns.countplot(data = df_final, x = 'overall', palette = 'dark')
```

Out[14]:
```
<AxesSubplot:xlabel='overall', ylabel='count'>
```



In [15]:
```python
df_final['overall'] = df['overall'].astype(int)

df_text = df_final[['overall', 'reviewText']]
```

In [16]:
```
df_text.dropna()
```

Out[16]:

	overall	reviewText
0	4	I purchased this Saran premium plastic wrap after trying Reynolds press and seal wrap which I would never use again.. There is less static cling to this wrap than I remember. To me this is a good thing because it doesn't stick to its self .\n\nThis is my typical complaint with all plastic wraps. When trying to cut them they ball all up and are useless. However they have improved this. Now Saran clings to the bowl or plate you wish to cover.\n\nNow if only they could improve the cutters on the boxes so that the cutters actually cut and scissors weren't required would be better..
1	5	I am an avid cook and baker. Saran Premium Plastic Wrap is a staple in my pantry and the only plastic wrap I purchase. I have tried other brands including Glad and have consistently found Saran Wrap to be far superior.\n\nSaran Wrap is easy to use. It's cutting bar cuts the wrap smoothly and the end of the wrap is easy to remove from the roll, doesn't get all sticky and impossible to remove like on some other brands. Some of the comments mention that Saran Wrap does not cling, but I have never had this problem when using this wrap at room temperature, in the refrigerator, or in the microwave.\n\nKeeps food stuffs fresh and wonderful to use to separate layers of freshly baked cookies and brownies stored in containers in the freezer. I also use this to tightly wrap partially used fruits and vegetable like apples and avocadoes. Saran Wrap excels at keeping these partially used fruits and vegetables fresh with no browning. Another great Amazon Prime Pantry value.
2	5	Good wrap, keeping it in the fridge makes it easier to tear. Learned this trick from my sister.
3	4	I prefer Saran wrap over other brands. It doesn't cling as well to dishes, but it tangles less when pulling it out of the box.
4	5	Thanks
...	...	...
137783	5	great
137784	4	These are delicious and healthy snacks! I with they were more affordable because they're really tasty and convenient. I purchased these because they're lower in sugar than many other brands and really enjoy them.
137785	5	Taste not to be believed. Buy a box for my office every week
137786	5	They are yummy!
137787	5	Oh so good.

137611 rows × 2 columns

In [17]:
```
for i in range (0,len(df_text['reviewText'])-1):
    if type(df_text['reviewText'].iloc[i])!= str:
        df_text['reviewText'].iloc[i] = str(df_text['reviewText'].iloc[i] )
```

```
C:\Users\blien\AppData\Local\Temp\ipykernel_10356\3079931999.py:3: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://p
andas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  df_text['reviewText'].iloc[i] = str(df_text['reviewText'].iloc[i] )
```

In [18]:
```
df_text.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 137788 entries, 0 to 137787
Data columns (total 2 columns):
 #   Column      Non-Null Count   Dtype 
---  ------      --------------   ----- 
 0   overall     137788 non-null  int32 
 1   reviewText  137788 non-null  object
dtypes: int32(1), object(1)
memory usage: 2.6+ MB
```

In [19]:
```
df_text[df_text['overall'] != 3]
```

Out[19]:

	overall	reviewText
0	4	I purchased this Saran premium plastic wrap after trying Reynolds press and seal wrap which I would never use again.. There is less static cling to this wrap than I remember. To me this is a good thing because it doesn't stick to its self .\n\nThis is my typical complaint with all plastic wraps. When trying to cut them they ball all up and are useless. However they have improved this. Now Saran clings to the bowl or plate you wish to cover.\n\nNow if only they could improve the cutters on the boxes so that the cutters actually cut and scissors weren't required would be better..
1	5	I am an avid cook and baker. Saran Premium Plastic Wrap is a staple in my pantry and the only plastic wrap I purchase. I have tried other brands including Glad and have consistently found Saran Wrap to be far superior.\n\nSaran Wrap is easy to use. It's cutting bar cuts the wrap smoothly and the end of the wrap is easy to remove from the roll, doesn't get all sticky and impossible to remove like on some other brands. Some of the comments mention that Saran Wrap does not cling, but I have never had this problem when using this wrap at room temperature, in the refrigerator, or in the microwave.\n\nKeeps food stuffs fresh and wonderful to use to separate layers of freshly baked cookies and brownies stored in containers in the freezer. I also use this to tightly wrap partially used fruits and vegetable like apples and avocadoes. Saran Wrap excels at keeping these partially used fruits and vegetables fresh with no browning. Another great Amazon Prime Pantry value.
2	5	Good wrap, keeping it in the fridge makes it easier to tear. Learned this trick from my sister.
3	4	I prefer Saran wrap over other brands. It doesn't cling as well to dishes, but it tangles less when pulling it out of the box.
4	5	Thanks
...	...	...
137783	5	great
137784	4	These are delicious and healthy snacks! I with they were more affordable because they're really tasty and convenient. I purchased these because they're lower in sugar than many other brands and really enjoy them.
137785	5	Taste not to be believed. Buy a box for my office every week
137786	5	They are yummy!
137787	5	Oh so good.

128679 rows × 2 columns

In [20]:
```
def label(i):
    return 1 if i >= 4 else 0
df_text['label'] = df_text['overall'].apply(label)
df_text.head(10)
```

Out[20]:

	overall	reviewText	label
0	4	I purchased this Saran premium plastic wrap after trying Reynolds press and seal wrap which I would never use again.. There is less static cling to this wrap than I remember. To me this is a good thing because it doesn't stick to its self .\n\nThis is my typical complaint with all plastic wraps. When trying to cut them they ball all up and are useless. However they have improved this. Now Saran clings to the bowl or plate you wish to cover.\n\nNow if only they could improve the cutters on the boxes so that the cutters actually cut and scissors weren't required would be better..	1
1	5	I am an avid cook and baker. Saran Premium Plastic Wrap is a staple in my pantry and the only plastic wrap I purchase. I have tried other brands including Glad and have consistently found Saran Wrap to be far superior.\n\nSaran Wrap is easy to use. It's cutting bar cuts the wrap smoothly and the end of the wrap is easy to remove from the roll, doesn't get all sticky and impossible to remove like on some other brands. Some of the comments mention that Saran Wrap does not cling, but I have never had this problem when using this wrap at room temperature, in the refrigerator, or in the microwave.\n\nKeeps food stuffs fresh and wonderful to use to separate layers of freshly baked cookies and brownies stored in containers in the freezer. I also use this to tightly wrap partially used fruits and vegetable like apples and avocadoes. Saran Wrap excels at keeping these partially used fruits and vegetables fresh with no browning. Another great Amazon Prime Pantry value.	1
2	5	Good wrap, keeping it in the fridge makes it easier to tear. Learned this trick from my sister.	1
3	4	I prefer Saran wrap over other brands. It doesn't cling as well to dishes, but it tangles less when pulling it out of the box.	1
4	5	Thanks	1
5	5	really good	1
6	4	Nice product, not a lot on the roll.	1
7	5	Great product.	1
8	4	When one can''t find the right lid, use this wrap. It stays in place and keeps food fresh. I use it to wrap sandwiches as well. I will purchase it again. Thanks for keeping it in stock/	1
9	5	good	1

In [21]:
```
df_final = df_text.drop(['overall'], axis = 1)
df_final.shape
```

Out[21]:
```
(137788, 2)
```

In [22]:
```
positive_df = df_final[df_final['label'] == 1].sample(n=len(df_final[df_final
negative_df = df_final[df_final['label'] == 0]
print(positive_df.shape, negative_df.shape)
```

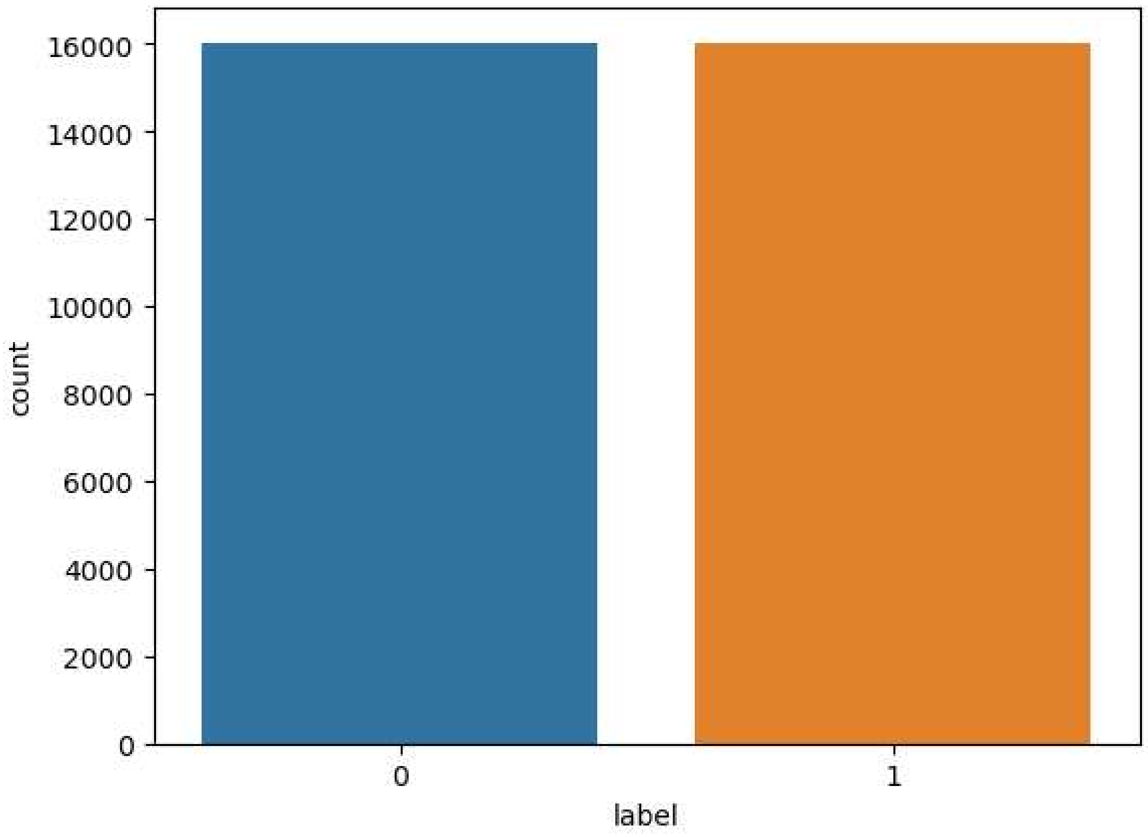
```
(16024, 2) (16024, 2)
```

In [23]:
```
review_df = positive_df.append(negative_df).reset_index(drop = True)
```

```
C:\Users\blien\AppData\Local\Temp\ipykernel_10356\2440028597.py:1: FutureWa
rning: The frame.append method is deprecated and will be removed from panda
s in a future version. Use pandas.concat instead.
  review_df = positive_df.append(negative_df).reset_index(drop = True)
```

In [24]:
```
sns.countplot(x= 'label', data = review_df)
```

Out[24]:
```
<AxesSubplot:xlabel='label', ylabel='count'>
```



In [25]:
```
review_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32048 entries, 0 to 32047
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   reviewText  32048 non-null  object
 1   label       32048 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 500.9+ KB
```

In [26]:
```
review_df['reviewText']
```

Out[26]:
```
0
Great tasting cereal bar. love it, new twist to a cereal bar
1
Great detergent!!  i workout a lot and this is the only detergent that gets
my clothes to smell good again.
2                                                         My mother swears b
y these things, I don't know what she uses it for, I think everything but I
never see her using it but I can always see that the sponge thing is gettin
g smaller.
3
BEEN USING FOR YEARS! THEY REALLY DON'T TASTE BAD AT ALL! BOIL WATER AND YO
UR SIDE DISH IS DONE! GREAT PRICE (SAME AS IN STORE)......FAST DELIVERY. I
RECOMMEND.
4
works good

...
32043
Tasted stale
32044
These were okay.  They seemed to have been the crunchiest cookies ever.  I
felt like I could break a tooth at any minute.
32045                               Meh. Seems like a cheap knockoff made b
y a big corporate brand. They're fine if this is what you want. Not great,
not bad. Good taste. Good quality. Arrived in good shape. Would not buy aga
in, though.
32046
Too much "cane syrup" or what-ever they used to sweeten this. I bought beca
use I thought there was no added sugar. I have my own sweeteners at home al
ready.
32047    This was my 1st time trying these and it is the last. They were ve
ry dry and the taste was terrible. It has no real raspberry flavor that I c
ould taste. I gave it one star because the price was not to bad for this ty
pe of food.
Name: reviewText, Length: 32048, dtype: object
```

In [27]:
```
#identify vocabualry size
tokenizer = Tokenizer()
tokenizer.fit_on_texts(review_df['reviewText'])
print("vocabulary size: ", len(tokenizer.word_index) + 1)
```

```
vocabulary size:  17853
```

In [107]:
```
#word embedding length
max_seq_embed = int(round(np.sqrt(np.sqrt(vocab_size)), 0))
```

In [108]:
```
max_seq_embed
```

Out[108]:
```
11
```

In [113]:
```
#max sequence length
review_length = []

for review in review_df.reviewText:
    review_length.append(len(review.split(' ')))

max_length = int(round(np.mean(review_length), 0))
print("Max length: ", max_length)
```

```
Max length:  25
```

In [28]:
```
stop_words = stopwords.words('english')
```

In [118]:
```
def preprocess_text(sen):
    # Removing html tags
    sentence = remove_tags(sen)

    # Remove punctuations and numbers
    sentence = re.sub('[^a-zA-Z]', ' ', sentence)

    # Single character removal
    sentence = re.sub(r"\s+[a-zA-Z]\s+", ' ', sentence)

    # Removing multiple spaces
    sentence = re.sub(r'\s+', ' ', sentence)

    #lower case
    sentence = sentence.lower()

    #tokenization
    sentence = nltk.word_tokenize(sentence)


    #lemmatize
    lemma = nltk.WordNetLemmatizer()
    sentence = [lemma.lemmatize(word) for word in sentence]

    #remove stop words
    sentence = [word for word in sentence if not word in stop_words]

    return sentence
```

In [119]:
```
TAG_RE = re.compile(r'<[^>]+>')

def remove_tags(text):
    return TAG_RE.sub('', text)
```

In [120]:
```
X = []
sentences = list(review_df['reviewText'])
for sen in sentences:
    X.append(preprocess_text(sen))
```

```
In [121]:  X[0:3]

Out[121]: [['great',
            'tasting',
            'cereal',
            'bar',
            'love',
            'new',
            'twist',
            'cereal',
            'bar'],
           ['great',
            'detergent',
            'workout',
            'lot',
            'detergent',
            'get',
            'clothes',
            'smell',
            'good'],
           ['mother',
            'swears',
            'thing',
            'know',
            'us',
            'think',
            'everything',
            'never',
            'see',
            'using',
            'always',
            'see',
            'sponge',
            'thing',
            'getting',
            'smaller']]
```

In [33]:
```
print(stop_words)
```
```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you'r
e", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves',
'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'i
t', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselve
s', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'th
ose', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'ha
s', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and',
'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'fo
r', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'be
fore', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out',
'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here',
'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'fe
w', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'o
wn', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just',
'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 'v
e', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't",
'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't",
'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "n
eedn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'were
n', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

In [41]:
```
y = review_df['label']
```

In [51]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, rar
```

In [52]:
```
y_train = pd.Series(y_train)
y_test = pd.Series(y_test)
X_train = pd.Series(X_train)
X_test = pd.Series(X_test)
```

In [53]:
```
X_train.shape
```
Out[53]:
```
(25638,)
```

In [54]:
```
X_test.shape
```
Out[54]:
```
(6410,)
```

In [55]:
```
from keras.utils.np_utils import to_categorical
y_train = to_categorical(y_train, num_classes = 2)
y_test = to_categorical(y_test, num_classes = 2)
```

In [56]:
```
vocab_size = 15000
oov_tok = "<oov>"
embedding_dim = 16
max_length = 50
trunc_type = 'post'
padding_type = 'post'
```

In [57]:
```python
X_train = [str(item) for item in X_train]
X_train = [item for item in X_train if not isinstance(item, int)]
```

In [58]:
```python
tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(X_train)
word_index = tokenizer.word_index
print(word_index)
```

```
g'": 72, "'amazon'": 73, "'delicious'": 74, "'enough'": 75, "'easy'": 76,
"'pantry'": 77, "'thought'": 78, "'tea'": 79, "'feel'": 80, "'coffee'": 8
1, "'know'": 82, "'scent'": 83, "'could'": 84, "'though'": 85, "'keep'":
86, "'first'": 87, "'never'": 88, "'made'": 89, "'kid'": 90, "'hard'": 9
1, "'cereal'": 92, "'pretty'": 93, "'expected'": 94, "'drink'": 95, "'wan
t'": 96, "'bottle'": 97, "'got'": 98, "'perfect'": 99, "'long'": 100, "'s
ure'": 101, "'le'": 102, "'item'": 103, "'year'": 104, "'hair'": 105, "'m
any'": 106, "'okay'": 107, "'order'": 108, "'peanut'": 109, "'quality'":
110, "'add'": 111, "'purchase'": 112, "'mix'": 113, "'regular'": 114, "'t
wo'": 115, "'bar'": 116, "'tasted'": 117, "'kind'": 118, "'hand'": 119,
"'salt'": 120, "'skin'": 121, "'star'": 122, "'give'": 123, "'without'":
124, "'nothing'": 125, "'texture'": 126, "'cracker'": 127, "'strong'": 12
8, "'come'": 129, "'buying'": 130, "'package'": 131, "'last'": 132, "'sau
ce'": 133, "'ingredient'": 134, "'recommend'": 135, "'back'": 136, "'ol
d'": 137, "'nut'": 138, "'since'": 139, "'found'": 140, "'however'": 141,
"'soft'": 142, "'rice'": 143, "'different'": 144, "'right'": 145, "'chees
e'": 146, "'put'": 147, "'prefer'": 148, "'probably'": 149, "'take'": 15
0, "'big'": 151, "'cup'": 152, "'tasting'": 153, "'fine'": 154, "'cook
y'": 155, "'every'": 156, "'see'": 157, "'pack'": 158, "'prime'": 159,
"'definitely'": 160, "'maybe'": 161, "'arrived'": 162, "'anything'": 163,
```

In [83]:
```python
#apply padding train data
sequences_train = tokenizer.texts_to_sequences(X_train)
padded_train = pad_sequences(sequences_train, maxlen = max_length, padding =

#padding test data
sequences_test = tokenizer.texts_to_sequences(X_test)
padded_test = pad_sequences(sequences_test, maxlen = max_length, padding = pa
```

In [61]:
```python
import sys
```

In [62]:
```python
#display padding sequence
np.set_printoptions(threshold=sys.maxsize)
padded_train[1]
```

Out[62]:
```
array([ 64, 352,  90,  39,  88, 130,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0])
```

In [84]:
```python
#convert all padding to array
train_pad = np.array(padded_train)
train_label = np.array(y_train)
test_pad = np.array(padded_test)
test_label = np.array(y_test)
```

In [85]:
```
pd.DataFrame(train_pad).to_csv("training_padded.csv")
```

In [86]:
```
pd.DataFrame(test_pad).to_csv("test_padded.csv")
```

In [87]:
```
pd.DataFrame(train_label).to_csv("training_label.csv")
```

In [88]:
```
pd.DataFrame(test_label).to_csv("test_label.csv")
```

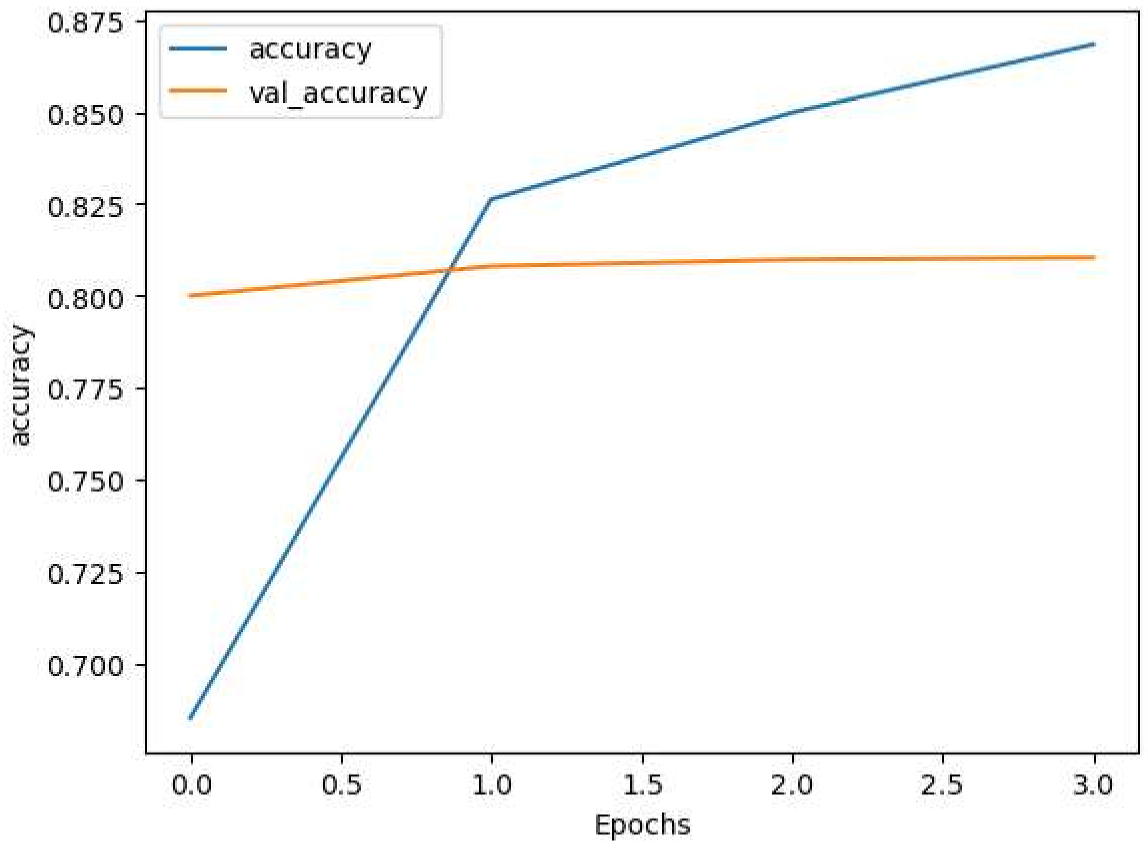
In [101]:

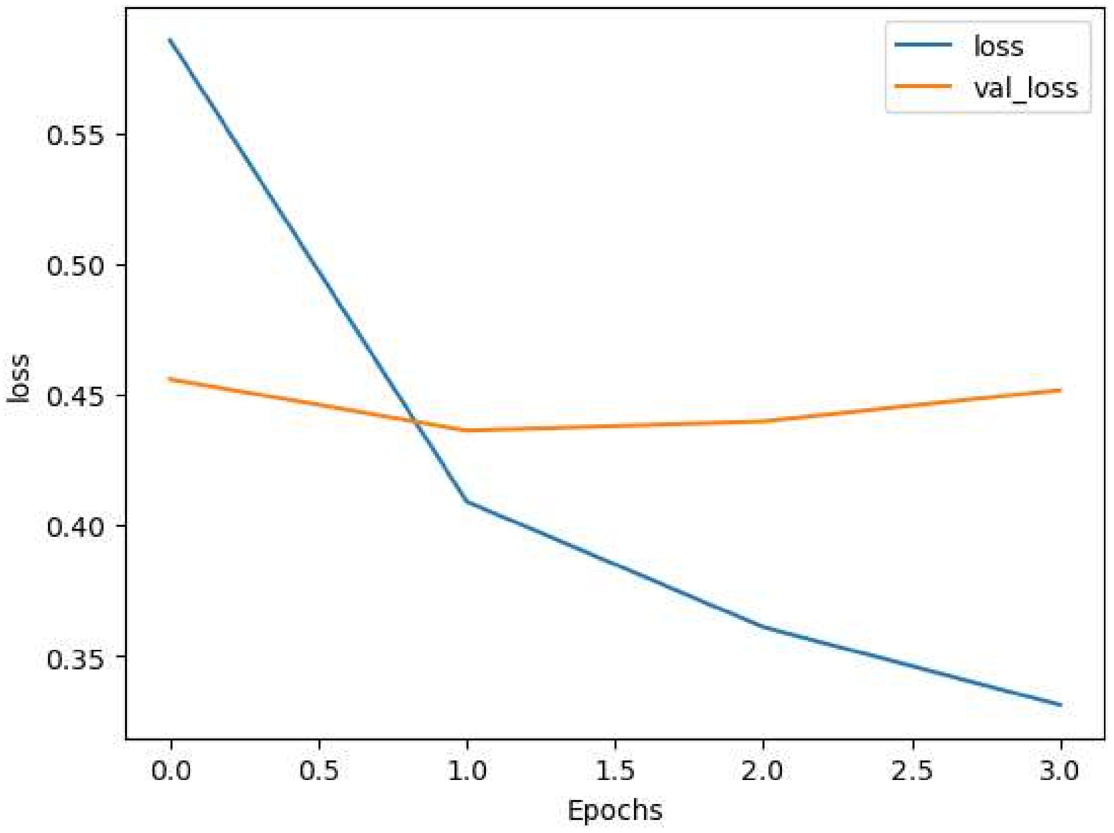
```
#Sentiment Analysis

#set parameters
activation = 'softmax'
loss = 'binary_crossentropy'
optimizer = 'adam'

num_epochs = 30

import tensorflow as tf
import keras
from keras.callbacks import ModelCheckpoint, EarlyStopping
early_stopping_monitor = EarlyStopping(patience = 2)

#define callback
callback = tf.keras.callbacks.EarlyStopping(monitor = 'loss', patience = 3)

#build neural netowrk model
from keras.layers import Dense

model = tf.keras.Sequential([tf.keras.layers.Embedding(vocab_size, embedding_

from sklearn import metrics

model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['a
model.summary()

history = model.fit(train_pad, train_label, epochs = num_epochs, batch_size =
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, 50, 16)            240000

 global_average_pooling1d_3   (None, 16)               0
 (GlobalAveragePooling1D)

 dense_9 (Dense)             (None, 100)               1700

 dense_10 (Dense)            (None, 50)                5050

 dense_11 (Dense)            (None, 2)                 102

=================================================================
Total params: 246,852
Trainable params: 246,852
Non-trainable params: 0
_________________________________________________________________
Epoch 1/30
359/359 [==============================] - 3s 6ms/step - loss: 0.5853 - a
ccuracy: 0.6851 - val_loss: 0.4558 - val_accuracy: 0.7999
Epoch 2/30
359/359 [==============================] - 2s 5ms/step - loss: 0.4090 - a
ccuracy: 0.8262 - val_loss: 0.4362 - val_accuracy: 0.8080
Epoch 3/30
```

```
359/359 [==============================] - 2s 5ms/step - loss: 0.3610 - a
ccuracy: 0.8499 - val_loss: 0.4397 - val_accuracy: 0.8098
Epoch 4/30
359/359 [==============================] - 2s 5ms/step - loss: 0.3313 - a
ccuracy: 0.8683 - val_loss: 0.4516 - val_accuracy: 0.8103
```

In [102]:
```
def plot_graphs(history, string):
    plt.plot(history.history[string])
    plt.plot(history.history['val_'+string])
    plt.xlabel("Epochs")
    plt.ylabel(string)
    plt.legend([string, 'val_'+string])
    plt.show()
plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
```





In [91]:
```
test_pad.shape
```
Out[91]:
```
(6410, 50)
```

In [92]:
```
test_label.shape
```
Out[92]:
```
(6410, 2)
```

In [93]:
```
score = model.evaluate(test_pad, test_label, verbose = 0)
print(f'Test loss: {score[0]} / Test accuracy: {score[1]}')
```
```
Test loss: 0.47823673486709595 / Test accuracy: 0.7984399199485779
```

In [95]:
```
model.save('SentimentAnalysisModel.h5')

my_model = tf.keras.models.load_model('SentimentAnalysisModel.h5')
```

In [96]:
```
predict = my_model.predict(test_pad)
```
```
201/201 [==============================] - 1s 2ms/step
```

In [100]:

```
i = 24

print("Predicted review:", X_test[i], "\n")
print("Predicted:", "Negative" if predict[i][0] >= 0.5 else "Positive", "revi
print("Actual: ", "Negative" if y_test[i][1] == 0 else "Positive", "review")
```

```
Predicted review: ['fantastic', 'snack', 'pack', 'could', 'probably', 'us
e', 'couple', 'cracker', 'overall', 'great', 'get', 'ice', 'cold', 'back',
'fridge', 'perfect', 'snack']

Predicted: Positive review
Actual:  Positive review
```