In [1]:
```python
import random
random.seed(0)
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```python
import pandas as pd
data=pd.read_table('C:\\Users\\uppugunduru venkat\\OneDrive\\Documents\\Documents\\framess.html\\amazon_alexa.tsv')
data
```

Out[2]:

	rating	date	variation	verified_reviews	feedback
0	5	31-Jul-18	Charcoal Fabric	Love my Echo!	1
1	5	31-Jul-18	Charcoal Fabric	Loved it!	1
2	4	31-Jul-18	Walnut Finish	Sometimes while playing a game, you can answer...	1
3	5	31-Jul-18	Charcoal Fabric	I have had a lot of fun with this thing. My 4 ...	1
4	5	31-Jul-18	Charcoal Fabric	Music	1
...	...	...	...	...	...
3145	5	30-Jul-18	Black Dot	Perfect for kids, adults and everyone in betwe...	1
3146	5	30-Jul-18	Black Dot	Listening to music, searching locations, check...	1
3147	5	30-Jul-18	Black Dot	I do love these things, i have them running my...	1
3148	5	30-Jul-18	White Dot	Only complaint I have is that the sound qualit...	1
3149	4	29-Jul-18	Black Dot	Good	1

3150 rows × 5 columns

In [3]:
```python
data = data[['variation','verified_reviews']]
```

In [4]:
```python
import re
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

data = data[data.verified_reviews != "Neutral"]
data['verified_reviews'] = data['verified_reviews'].apply(lambda x: x.lower())
data['verified_reviews'] = data['verified_reviews'].apply((lambda x: re.sub('[^a-zA-z0-9\s]','',x)))

print(data[ data['verified_reviews'] == 'Positive'].size)
print(data[ data['verified_reviews'] == 'Negative'].size)

for idx,row in data.iterrows():
    row[0] = row[0].replace('rt',' ')

vocabSize = 2000
tokenizer = Tokenizer(num_words=vocabSize, split=' ')
tokenizer.fit_on_texts(data['verified_reviews'].values)
X = tokenizer.texts_to_sequences(data['verified_reviews'].values)
X = pad_sequences(X)
```

```
0
0
```

In [5]:
```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM, SpatialDropout1D

embed_dim = 128
lstm_out = 196

model = Sequential()
model.add(Embedding(vocabSize, embed_dim,input_length = X.shape[1]))
model.add(SpatialDropout1D(0.4))
model.add(Dense(2,activation='softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 473, 128)          256000
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 473, 128)          0
_________________________________________________________________
dense (Dense)                (None, 473, 2)            258
=================================================================
Total params: 256,258
Trainable params: 256,258
Non-trainable params: 0
_________________________________________________________________
None
```

In [6]:
```python
from sklearn.model_selection import train_test_split

Y = pd.get_dummies(data['verified_reviews']).values
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.05, random_state = 42)
print(X_train.shape,Y_train.shape)
print(X_test.shape,Y_test.shape)
```

```
(2992, 473) (2992, 2242)
(158, 473) (158, 2242)
```

In [7]:
```python
batch_size = 32
model.fit(X_train, Y_train, epochs = -1, batch_size=batch_size, verbose = 2)
```

Out[7]:
```
<tensorflow.python.keras.callbacks.History at 0x22d41b9f940>
```

In [8]:
```python
import numpy as np

pos_cnt, neg_cnt, pos_correct, neg_correct = 0, 0, 0, 0

for x in range(len(X_test)):

    result = model.predict(X_test[x].reshape(1,X_test.shape[1]),batch_size=1,verbose = 2)[0]

    if np.argmax(result) == np.argmax(Y_test[x]):
        if np.argmax(Y_test[x]) == 0:
            neg_correct += 1
        else:
            pos_correct += 1

    if np.argmax(Y_test[x]) == 0:
        neg_cnt += 1
    else:
        pos_cnt += 1

print("pos_acc", pos_correct/pos_cnt*100, "%")
print("neg_acc", neg_correct/neg_cnt*100, "%")
```

```
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
pos_acc 2.5316455696202533 %
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-8-e2e7f618a692> in <module>
     19
     20 print("pos_acc", pos_correct/pos_cnt*100, "%")
---> 21 print("neg_acc", neg_correct/neg_cnt*100, "%")

ZeroDivisionError: division by zero
```

In [9]:
```python
twt = ['data']

twt = tokenizer.texts_to_sequences(twt)

twt = pad_sequences(twt, maxlen=28, dtype='int32', value=0)
print(twt)
sentiment = model.predict(twt,batch_size=1,verbose = 2)[0]
if(np.argmax(sentiment) == 0):
    print("negative")
elif (np.argmax(sentiment) == 1):
    print("positive")
```

```
[[   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 1498]]
WARNING:tensorflow:Model was constructed with shape (None, 473) for input Tensor("embedding_input:0", shape=(None, 473), dtype=float32), but it was called on an input with incompat
ible shape (1, 28).
1/1 - 0s
positive
```

In [ ]: