#!/usr/bin/env python  
# coding: utf-8  
  
# In[29]:  
  
  
import pandas as pd  
import numpy as np  
import re  
from nltk.stem import WordNetLemmatizer  
from nltk.corpus import stopwords  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import MinMaxScaler  
from tensorflow.keras import layers  
from tensorflow.keras import models  
from tensorflow.keras.preprocessing.text import Tokenizer  
from tensorflow.keras.preprocessing.sequence import pad\_sequences  
from tensorflow.keras.utils import to\_categorical  
lemaoi = WordNetLemmatizer()  
  
  
# In[15]:  
  
  
def word\_sequence(n\_most\_words, x, max\_len):  
 tokeniz = Tokenizer(num\_words=n\_most\_words)  
 tokeniz.fit\_on\_texts(x.values)  
 sequen = tokeniz.texts\_to\_sequences(x.values)  
 word\_index = tokeniz.word\_index  
 X = pad\_sequences(sequen, maxlen=max\_len)  
 return X, word\_index  
  
  
# In[2]:  
  
  
def under\_sample(raw):  
 posi = raw\_data.loc[raw\_data['sentiment']=='Positive'][:100]  
 Neu = raw\_data.loc[raw\_data['sentiment']=='Neutral'][:100]  
 Nega = raw\_data.loc[raw\_data['sentiment']=='Negative']  
 traning\_data = pd.concat([posi, Neu, Nega])  
 traning\_data.reset\_index(inplace=True)  
 traning\_data['sentiment'].replace({'Positive':1,'Negative':-1,'Neutral':0}, inplace=True)  
 return traning\_data  
  
  
# In[3]:  
  
  
def over\_sample(raw):  
 posi = raw\_data.loc[raw\_data['sentiment']=='Positive'][:1000]  
 Neu = raw\_data.loc[raw\_data['sentiment']=='Neutral']  
 Nega = raw\_data.loc[raw\_data['sentiment']=='Negative']  
 Neu\_new = pd.concat([Neu,Neu,Neu,Neu,Neu,Neu], axis=0)  
 Nega\_new = pd.concat([Nega,Nega,Nega,Nega,Nega,Nega,Nega,Nega,Nega,Nega])  
 traning\_data = pd.concat([posi, Neu\_new, Nega\_new])  
 traning\_data.reset\_index(inplace=True)  
 traning\_data['sentiment'].replace({'Positive':1,'Negative':-1,'Neutral':0}, inplace=True)  
 return traning\_data  
   
  
  
# In[4]:  
  
  
def pre\_processing(x):  
 x = x.lower()  
 x = re.sub('[^a-z0-9 ]','',x)  
 result = []  
 for i in x.split():  
 if i in stopwords.words('english'):  
 pass  
 else:  
 if len(i) <3:  
 pass  
 else:  
 result.append(lemaoi.lemmatize(i))  
 res = ' '.join(result)  
 res = re.sub('\d{2,}\w+','',res)  
 return res  
  
  
# In[6]:  
  
  
raw\_data = pd.read\_csv('train\_data.csv')  
raw\_data.head(5)  
  
  
# In[7]:  
  
  
train\_frame = over\_sample(raw\_data)  
train\_frame.columns  
  
  
# In[9]:  
  
  
train\_frame['data'] = train\_frame['reviews.text'] + ' '+ train\_frame['reviews.title']  
data\_to\_train = train\_frame.loc[:,['data','sentiment']]  
data\_to\_train.head(5)  
  
  
# In[26]:  
  
  
data\_to\_train['sentiment'].value\_counts()  
  
  
# In[11]:  
  
  
data\_to\_train.dropna(axis=0, inplace=True)  
  
  
# In[14]:  
  
  
data\_to\_train['cleaned\_data'] = data\_to\_train['data'].apply(lambda x: pre\_processing(x))  
data\_to\_train  
  
  
# In[30]:  
  
  
Y = to\_categorical(data\_to\_train['sentiment'], num\_classes=3)  
Y  
  
  
# In[31]:  
  
  
vocab = 15000  
max\_len\_for\_sen = 130  
Embedding = 50  
  
  
# In[32]:  
  
  
X, word\_i = word\_sequence(vocab,data\_to\_train['cleaned\_data'],max\_len\_for\_sen)  
  
  
# In[18]:  
  
  
X  
  
  
# In[40]:  
  
  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(X,Y,test\_size=0.3, random\_state=42)  
print(x\_train.shape)  
print(x\_test.shape)  
print(y\_train.shape)  
print(y\_test.shape)  
  
  
# In[35]:  
  
  
amazon = models.Sequential()  
amazon.add(layers.Embedding(vocab, 40, input\_length=max\_len\_for\_sen))  
amazon.add(layers.SpatialDropout1D(0.2))  
amazon.add(layers.LSTM(100))  
amazon.add(layers.Dense(3, activation='sigmoid'))  
print(amazon.summary())  
  
  
# In[37]:  
  
  
amazon.compile(optimizer='adam',loss='categorical\_crossentropy',metrics=['acc'])  
  
  
# In[39]:  
  
  
amazon.fit(x\_train,y\_train,batch\_size=50,epochs=20)  
  
  
# In[41]:  
  
  
amazon.evaluate(x\_test,y\_test)  
  
  
# In[42]:  
  
  
amazon.save('amazon.h5')  
  
  
# In[43]:  
  
  
test\_cr = pd.read\_csv('test\_data.csv')  
  
  
# In[44]:  
  
  
test\_cr.dropna(axis=0, inplace=True)  
  
  
# In[46]:  
  
  
test\_cr['data'] = test\_cr['reviews.text'] + ' '+ test\_cr['reviews.title']  
  
  
# In[47]:  
  
  
test\_cr['cleaned\_data'] = test\_cr['data'].apply(lambda x: pre\_processing(x))  
test\_cr  
  
  
# In[48]:  
  
  
y\_test\_cr,wi= word\_sequence(vocab,test\_cr['cleaned\_data'],max\_len\_for\_sen)  
  
  
# In[51]:  
  
  
y\_pred = amazon.predict\_classes(y\_test\_cr)  
  
  
# In[52]:  
  
  
y\_pred  
  
  
# In[ ]: