LSTM on Amazon reviews

In [1]:

```
import sqlite3
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Load Data

In [43]:

```
con =sqlite3.connect(r'C:\Users\Friend\AI\AI_datasets\Amazon\cleaned_database.sqlite')
filtered_data  = pd.read_sql_query('SELECT * FROM Reviews WHERE Score != 3',con)
filtered_data = filtered_data.drop('index',axis = 1)
filtered_data['Score'] = filtered_data['Score'].map(lambda x: 1 if x == 'positive' else 0)
filtered_data = filtered_data.sort_values('Time')
```

In [81]:

```
data = filtered_data.head(30000)
data.columns
```

Out[81]:

```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text',
       'CleanedText'],
      dtype='object')
```

In [82]:

```
from sklearn import cross_validation

X_train, X_test, y_train, y_test = cross_validation.train_test_split(data['CleanedText'], data['Score']
, test_size=0.3, random_state=0)
print(X_train.shape,y_train.shape,X_test.shape,y_test.shape)
```

```
(21000,) (21000,) (9000,) (9000,)
```

Convert Data into IMDB format

In [83]:

```
from sklearn.feature_extraction.text import CountVectorizer

vec = CountVectorizer().fit(X_train)
bag_of_words = count_vect.transform(X_train)
```

In [84]:

```
sum_words = bag_of_words.sum(axis=0)
words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
words_freq_sort = sorted(words_freq, key = lambda x: x[1], reverse=True)
```

In [85]:

```
words_order = [word for (word,idx) in words_freq_sort]
check = words_order[0:4999]
```

In [86]:

```
train_vectors = []
for each_review in X_train:
    rank_vector = [((check.index(word))+1) if word in check else  0 for word in each_review.split()]
    train_vectors.append(rank_vector)
```

In [87]:

```
test_vectors = []
for each_review in X_test:
    rank_vector = [((check.index(word))+1) if word in check else  0 for word in each_review.split()]
    test_vectors.append(rank_vector)
```

In [88]:

```
from keras.preprocessing import sequence
max_review_length = 600
X_train = sequence.pad_sequences(train_vectors, maxlen=max_review_length)
X_test = sequence.pad_sequences(test_vectors, maxlen=max_review_length)
```

LSTM Model

In [89]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
```

In [90]:

```
top_words = 5000
```

Single Layer

In [105]:

```
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length=max_review_length))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_13 (Embedding)     (None, 600, 32)           160000
_________________________________________________________________
lstm_21 (LSTM)               (None, 100)               53200
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 101
=================================================================
Total params: 213,301
Trainable params: 213,301
Non-trainable params: 0
_________________________________________________________________
None
```

In [106]:

```
model.fit(X_train, y_train, nb_epoch=10, batch_size=64)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
C:\Users\Friend\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: The `nb_epoch` argume
nt in `fit` has been renamed `epochs`.
  """Entry point for launching an IPython kernel.
```

```
Epoch 1/10
21000/21000 [==============================] - 331s 16ms/step - loss: 0.2755 - acc: 0.9033
Epoch 2/10
21000/21000 [==============================] - 334s 16ms/step - loss: 0.1756 - acc: 0.9339
Epoch 3/10
21000/21000 [==============================] - 353s 17ms/step - loss: 0.1443 - acc: 0.9469
Epoch 4/10
21000/21000 [==============================] - 369s 18ms/step - loss: 0.1245 - acc: 0.9550
Epoch 5/10
21000/21000 [==============================] - 493s 23ms/step - loss: 0.1138 - acc: 0.9602
Epoch 6/10
21000/21000 [==============================] - 592s 28ms/step - loss: 0.1048 - acc: 0.9641
Epoch 7/10
21000/21000 [==============================] - 474s 23ms/step - loss: 0.0956 - acc: 0.9678
Epoch 8/10
21000/21000 [==============================] - 369s 18ms/step - loss: 0.0851 - acc: 0.9722
Epoch 9/10
21000/21000 [==============================] - 368s 18ms/step - loss: 0.0762 - acc: 0.9759
Epoch 10/10
21000/21000 [==============================] - 370s 18ms/step - loss: 0.0675 - acc: 0.9790
Accuracy: 92.47%
```

Double LSTM Layer

In [103]:

```
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length=max_review_length))
model.add(LSTM(100,return_sequences=True))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_12 (Embedding)     (None, 600, 32)           160000
_________________________________________________________________
lstm_19 (LSTM)               (None, 600, 100)          53200
_________________________________________________________________
lstm_20 (LSTM)               (None, 100)               80400
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 101
=================================================================
Total params: 293,701
Trainable params: 293,701
Non-trainable params: 0
_________________________________________________________________
None
```

In [104]:

```
model.fit(X_train, y_train, nb_epoch=10, batch_size=64)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
C:\Users\Friend\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: The `nb_epoch` argume
nt in `fit` has been renamed `epochs`.
```

```
  """Entry point for launching an IPython kernel.
```

```
Epoch 1/10
21000/21000 [==============================] - 847s 40ms/step - loss: 0.2655 - acc: 0.9077
Epoch 2/10
21000/21000 [==============================] - 880s 42ms/step - loss: 0.1626 - acc: 0.9390
Epoch 3/10
21000/21000 [==============================] - 868s 41ms/step - loss: 0.1365 - acc: 0.9498
Epoch 4/10
21000/21000 [==============================] - 911s 43ms/step - loss: 0.1238 - acc: 0.9554
Epoch 5/10
21000/21000 [==============================] - 602s 29ms/step - loss: 0.1065 - acc: 0.9626
Epoch 6/10
21000/21000 [==============================] - 603s 29ms/step - loss: 0.0942 - acc: 0.9678
Epoch 7/10
21000/21000 [==============================] - 599s 29ms/step - loss: 0.0790 - acc: 0.9739
Epoch 8/10
21000/21000 [==============================] - 600s 29ms/step - loss: 0.0678 - acc: 0.9786
Epoch 9/10
21000/21000 [==============================] - 626s 30ms/step - loss: 0.0560 - acc: 0.9832
Epoch 10/10
21000/21000 [==============================] - 625s 30ms/step - loss: 0.0520 - acc: 0.9841
Accuracy: 91.63%
```