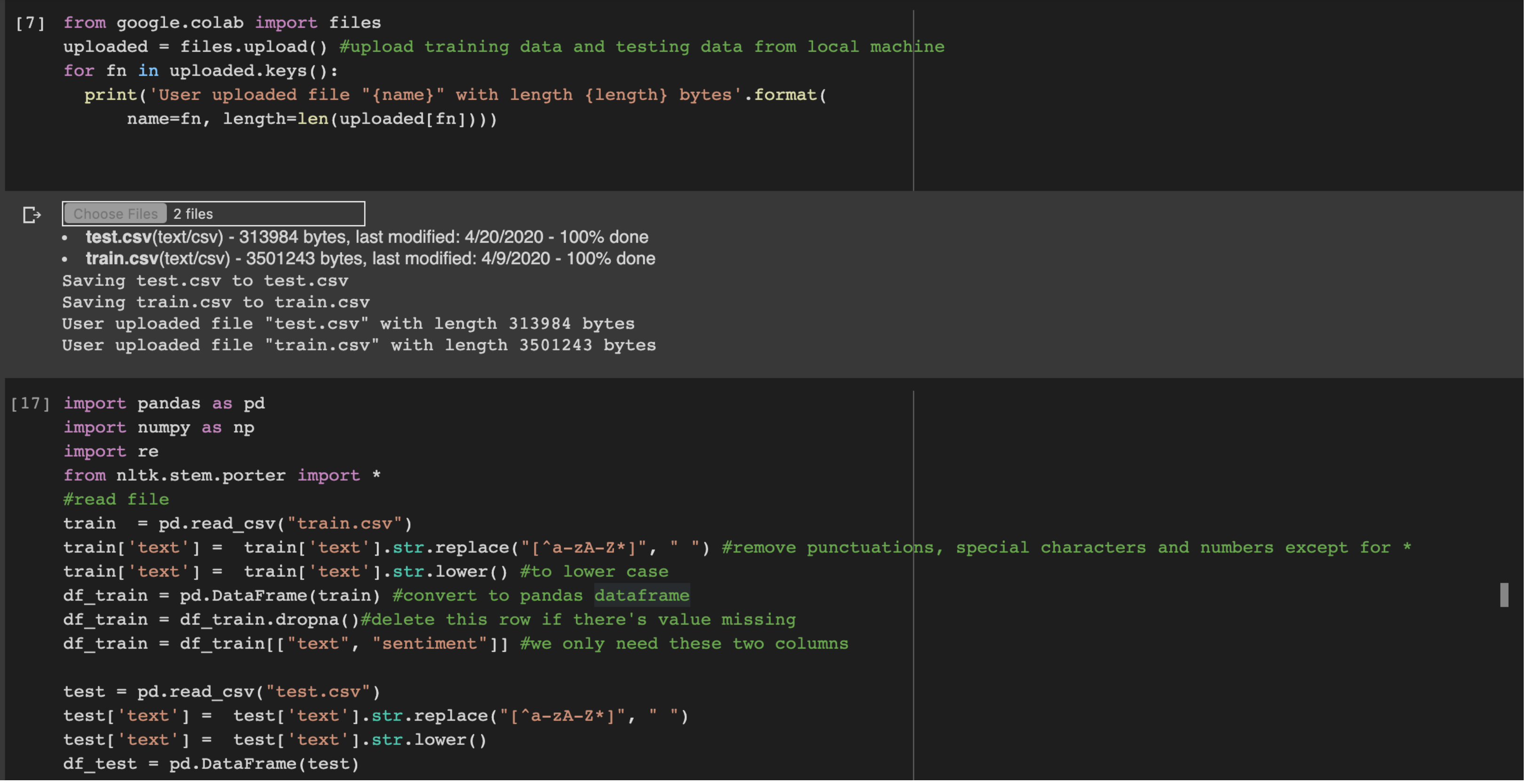


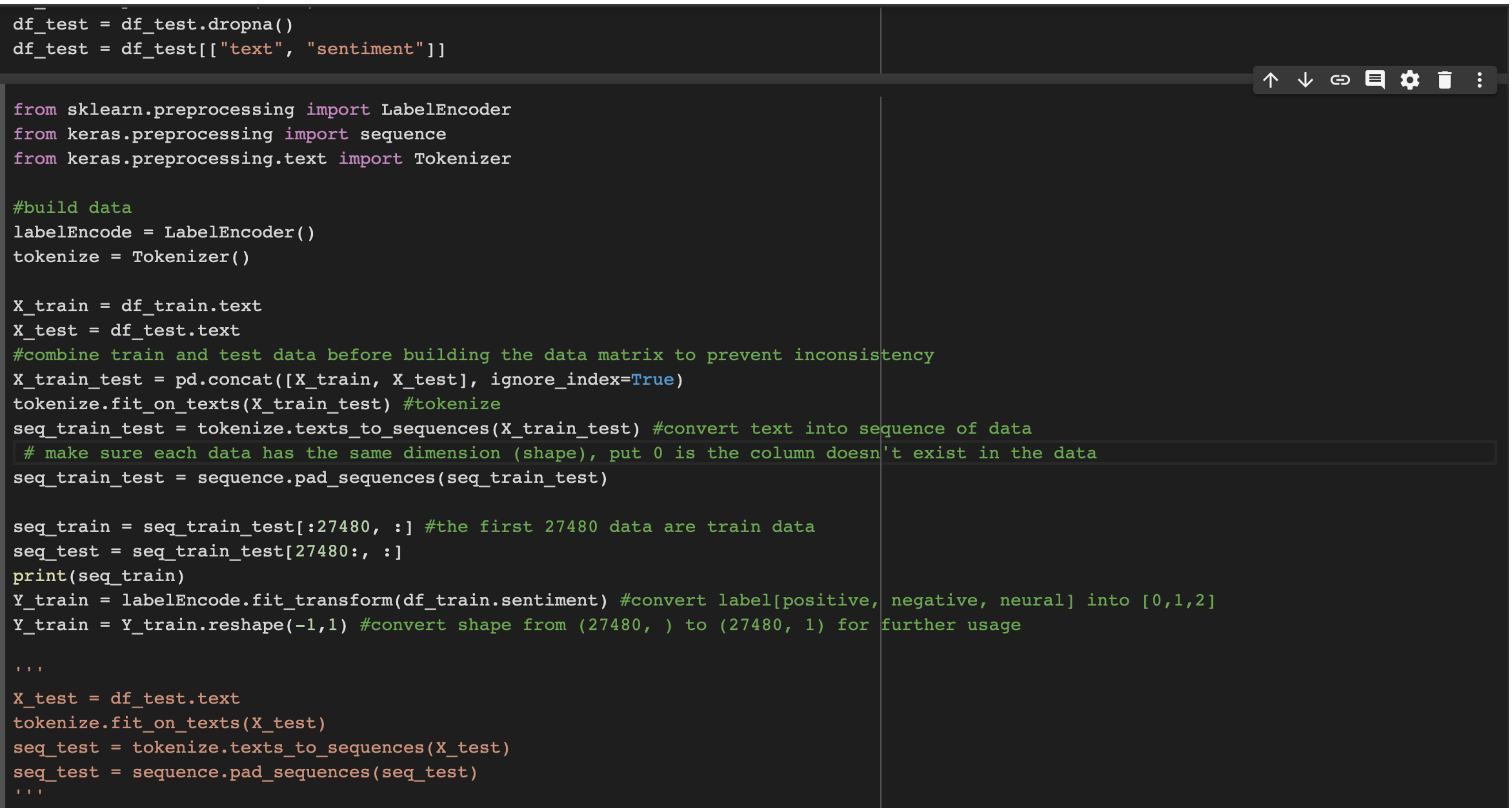
```
from google.colab import files
uploaded = files.upload() #upload training data and testing data from local machine
for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

- **test.csv**(text/csv) - 313984 bytes, last modified: 4/20/2020 - 100% done
- **train.csv**(text/csv) - 3501243 bytes, last modified: 4/9/2020 - 100% done

```
Saving test.csv to test.csv
Saving train.csv to train.csv
User uploaded file "test.csv" with length 313984 bytes
User uploaded file "train.csv" with length 3501243 bytes
```

```
import pandas as pd
import numpy as np
import re
from nltk.stem.porter import *
#read file
train  = pd.read_csv("train.csv")
train['text'] =  train['text'].str.replace("[^a-zA-Z*]", " ") #remove punctuations, special characters and numbers except for *
train['text'] =  train['text'].str.lower() #to lower case
df_train = pd.DataFrame(train) #convert to pandas dataframe
df_train = df_train.dropna()#delete this row if there's value missing
df_train = df_train[["text", "sentiment"]] #we only need these two columns

test = pd.read_csv("test.csv")
test['text'] =  test['text'].str.replace("[^a-zA-Z*]", " ")
test['text'] =  test['text'].str.lower()
df_test = pd.DataFrame(test)
```



```
df_test = df_test.dropna()
df_test = df_test[["text", "sentiment"]]
```

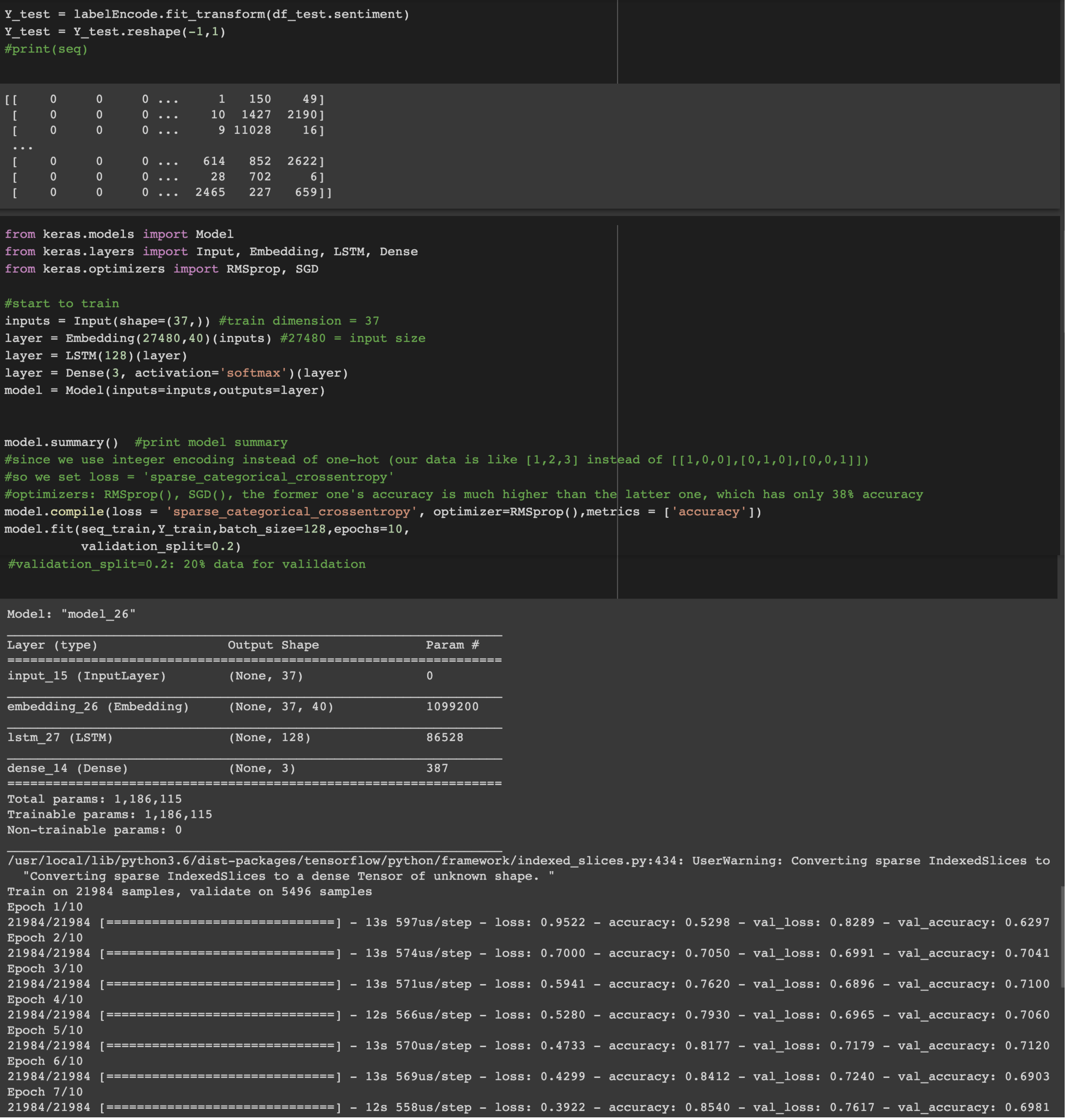
```
from sklearn.preprocessing import LabelEncoder
from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer

#build data
labelEncode = LabelEncoder()
tokenize = Tokenizer()

X_train = df_train.text
X_test = df_test.text
#combine train and test data before building the data matrix to prevent inconsistency
X_train_test = pd.concat([X_train, X_test], ignore_index=True)
tokenize.fit_on_texts(X_train_test) #tokenize
seq_train_test = tokenize.texts_to_sequences(X_train_test) #convert text into sequence of data
 # make sure each data has the same dimension (shape), put 0 is the column doesn't exist in the data
seq_train_test = sequence.pad_sequences(seq_train_test)

seq_train = seq_train_test[:27480, :] #the first 27480 data are train data
seq_test = seq_train_test[27480:, :]
print(seq_train)
Y_train = labelEncode.fit_transform(df_train.sentiment) #convert label[positive, negative, neural] into [0,1,2]
Y_train = Y_train.reshape(-1,1) #convert shape from (27480, ) to (27480, 1) for further usage

'''
X_test = df_test.text
tokenize.fit_on_texts(X_test)
seq_test = tokenize.texts_to_sequences(X_test)
seq_test = sequence.pad_sequences(seq_test)
'''
```



```
Y_test = labelEncode.fit_transform(df_test.sentiment)
Y_test = Y_test.reshape(-1,1)
#print(seq)
```

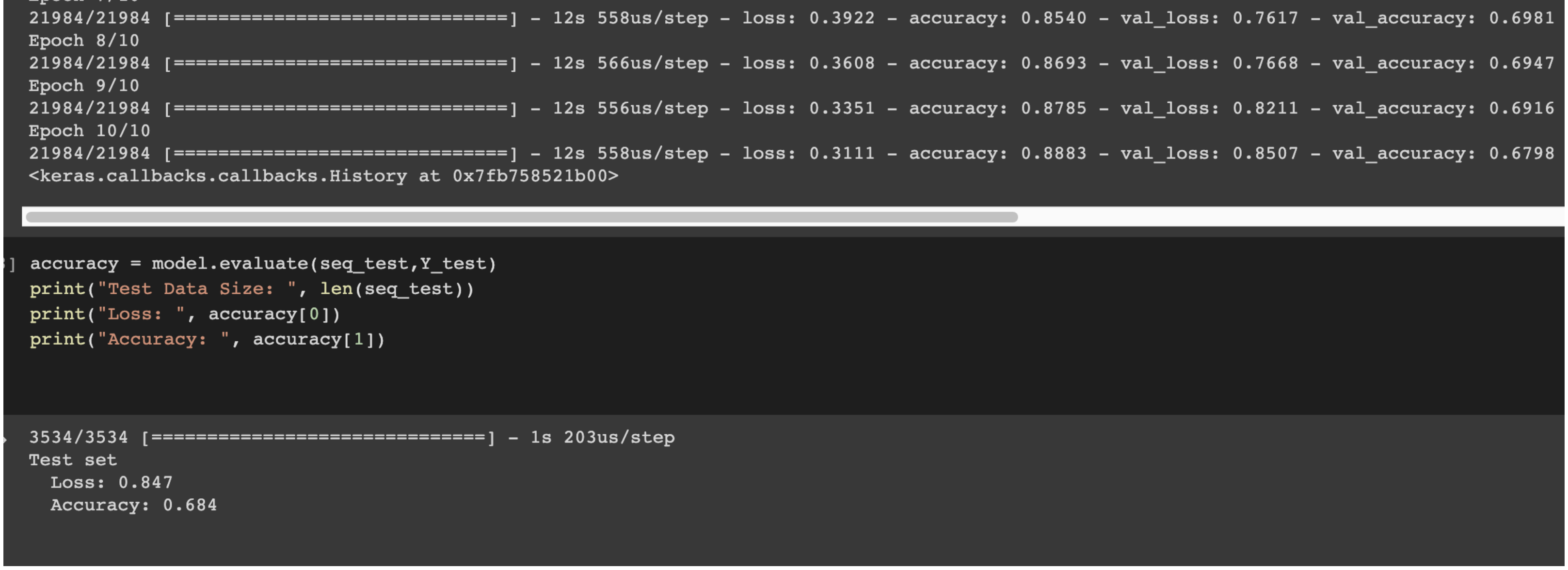
```
[[    0     0     0 ...     1   150    49]
 [    0     0     0 ...    10  1427  2190]
 [    0     0     0 ...     9 11028    16]
 ...
 [    0     0     0 ...   614   852  2622]
 [    0     0     0 ...    28   702     6]
 [    0     0     0 ...  2465   227   659]]
```

```
from keras.models import Model
from keras.layers import Input, Embedding, LSTM, Dense
from keras.optimizers import RMSprop, SGD

#start to train
inputs = Input(shape=(37,)) #train dimension = 37
layer = Embedding(27480,40)(inputs) #27480 = input size
layer = LSTM(128)(layer)
layer = Dense(3, activation='softmax')(layer)
model = Model(inputs=inputs,outputs=layer)


model.summary()  #print model summary
#since we use integer encoding instead of one-hot (our data is like [1,2,3] instead of [[1,0,0],[0,1,0],[0,0,1]])
#so we set loss = 'sparse_categorical_crossentropy'
#optimizers: RMSprop(), SGD(), the former one's accuracy is much higher than the latter one, which has only 38% accuracy
model.compile(loss = 'sparse_categorical_crossentropy', optimizer=RMSprop(),metrics = ['accuracy'])
model.fit(seq_train,Y_train,batch_size=128,epochs=10,
          validation_split=0.2)
 #validation_split=0.2: 20% data for valildation
```

```
Model: "model_26"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_15 (InputLayer)        (None, 37)                0         
_________________________________________________________________
embedding_26 (Embedding)     (None, 37, 40)            1099200   
_________________________________________________________________
lstm_27 (LSTM)               (None, 128)               86528     
_________________________________________________________________
dense_14 (Dense)             (None, 3)                 387       
=================================================================
Total params: 1,186,115
Trainable params: 1,186,115
Non-trainable params: 0
_________________________________________________________________
/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/indexed_slices.py:434: UserWarning: Converting sparse IndexedSlices to
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
Train on 21984 samples, validate on 5496 samples
Epoch 1/10
21984/21984 [==============================] - 13s 597us/step - loss: 0.9522 - accuracy: 0.5298 - val_loss: 0.8289 - val_accuracy: 0.6297
Epoch 2/10
21984/21984 [==============================] - 13s 574us/step - loss: 0.7000 - accuracy: 0.7050 - val_loss: 0.6991 - val_accuracy: 0.7041
Epoch 3/10
21984/21984 [==============================] - 13s 571us/step - loss: 0.5941 - accuracy: 0.7620 - val_loss: 0.6896 - val_accuracy: 0.7100
Epoch 4/10
21984/21984 [==============================] - 12s 566us/step - loss: 0.5280 - accuracy: 0.7930 - val_loss: 0.6965 - val_accuracy: 0.7060
Epoch 5/10
21984/21984 [==============================] - 13s 570us/step - loss: 0.4733 - accuracy: 0.8177 - val_loss: 0.7179 - val_accuracy: 0.7120
Epoch 6/10
21984/21984 [==============================] - 13s 569us/step - loss: 0.4299 - accuracy: 0.8412 - val_loss: 0.7240 - val_accuracy: 0.6903
Epoch 7/10
21984/21984 [==============================] - 12s 558us/step - loss: 0.3922 - accuracy: 0.8540 - val_loss: 0.7617 - val_accuracy: 0.6981
```



```
Epoch 7/10
21984/21984 [==============================] - 12s 558us/step - loss: 0.3922 - accuracy: 0.8540 - val_loss: 0.7617 - val_accuracy: 0.6981
Epoch 8/10
21984/21984 [==============================] - 12s 566us/step - loss: 0.3608 - accuracy: 0.8693 - val_loss: 0.7668 - val_accuracy: 0.6947
Epoch 9/10
21984/21984 [==============================] - 12s 556us/step - loss: 0.3351 - accuracy: 0.8785 - val_loss: 0.8211 - val_accuracy: 0.6916
Epoch 10/10
21984/21984 [==============================] - 12s 558us/step - loss: 0.3111 - accuracy: 0.8883 - val_loss: 0.8507 - val_accuracy: 0.6798
<keras.callbacks.callbacks.History at 0x7fb758521b00>
```

```
accuracy = model.evaluate(seq_test,Y_test)
print("Test Data Size: ", len(seq_test))
print("Loss: ", accuracy[0])
print("Accuracy: ", accuracy[1])
```

```
3534/3534 [==============================] - 1s 203us/step
Test set
  Loss: 0.847
  Accuracy: 0.684
```