Implementation of Sentiment analysis by building a Recurrent Neural Network

Keras has a built-in IMDb movie reviews dataset that here used

```
from keras.datasets import imdb
```

```
vocabulary_size = 7000
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words = vocabulary_size)
```

```
print('Loaded dataset with {} training samples, {} test samples'.format(len(X_train), len(

     Loaded dataset with 25000 training samples, 25000 test samples
```

```
#A sample review and its label
print('--------------------Review-------------------------')
print(X_train[6])
print('---------------------label-------------------------')
print(y_train[6])

     --------------------Review-------------------------
     [1, 6740, 365, 1234, 5, 1156, 354, 11, 14, 5327, 6638, 7, 1016, 2, 5940, 356, 44, 4,
     ---------------------label-------------------------
     1
```

```
#Map word IDs back to words
word_to_id = imdb.get_word_index()
id_to_word = {i: word for word, i in word_to_id.items()}
print('---review with words---')
print([id_to_word.get(i, ' ') for i in X_train[6]])
print('---label---')
print(y_train[6])

     ---review with words---
     ['the', 'boiled', 'full', 'involving', 'to', 'impressive', 'boring', 'this', 'as', 'r
     ---label---
     1
```

```
#Maximum review length and minimum review length
print('Maximum review length: {}'.format(len(max((X_train + X_test), key=len))))

     Maximum review length: 2697
```

```
print('Minimum review length: {}'.format(len(min((X_test + X_test), key=len))))

     Minimum review length: 14
```

```
#Pad sequences
from keras.preprocessing import sequence

max_words = 10000
X_train = sequence.pad_sequences(X_train, maxlen=max_words)
X_test = sequence.pad_sequences(X_test, maxlen=max_words)
```

```
#An RNN model for sentiment analysis
from keras import Sequential
from keras.layers import Embedding, LSTM, Dense, Dropout

embedding_size=32
model=Sequential()
model.add(Embedding(vocabulary_size, embedding_size, input_length=max_words))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))

print(model.summary())

     Model: "sequential"
     _________________________________________________________________
      Layer (type)                Output Shape              Param #
     =================================================================
      embedding (Embedding)       (None, 10000, 32)         224000

      lstm (LSTM)                 (None, 100)               53200

      dense (Dense)               (None, 1)                 101

     =================================================================
     Total params: 277,301
     Trainable params: 277,301
     Non-trainable params: 0
     _________________________________________________________________
     None
```

```
#Train and evaluate our model
model.compile(loss='binary_crossentropy',
             optimizer='adam',
             metrics=['accuracy'])
```

```
batch_size = 128
epochs = 1

X_valid, y_valid = X_train[:batch_size], y_train[:batch_size]
X_train2, y_train2 = X_train[batch_size:], y_train[batch_size:]

model.fit(X_train2, y_train2, validation_data=(X_valid, y_valid), batch_size=batch_size, e

     195/195 [==============================] - 5564s 28s/step - loss: 0.5316 - accuracy:
```

```
<keras.callbacks.History at 0x7f02446db950>
```

```
scores = model.evaluate(X_test, y_test)
print('Test accuracy:', scores[1])
```

```
782/782 [==============================] - 1417s 2s/step - loss: 0.3729 - accuracy: 0
Test accuracy: 0.8446000218391418
```

✓ 22m 58s completed at 12:19