Div: B
Roll No.: 422004
Gr. No.: 21810939
Name of Student: Aakanksha Bhondve
subject: Advanced Machine Learning

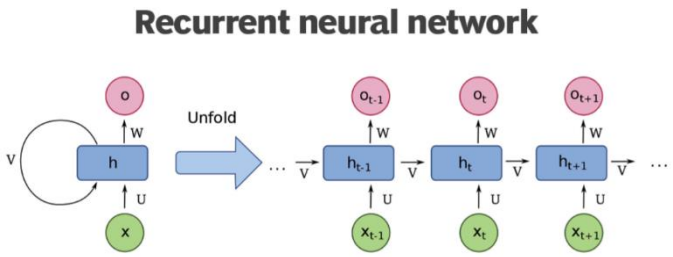
Assignment No. 5
Title: Implement Recurrent Neural Network for Sentiment Analysis.

Theory:

RNN:
RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer. An RNN can handle sequential data, accepting the current input data, and previously received inputs. RNNs can memorize previous inputs due to their internal memory.

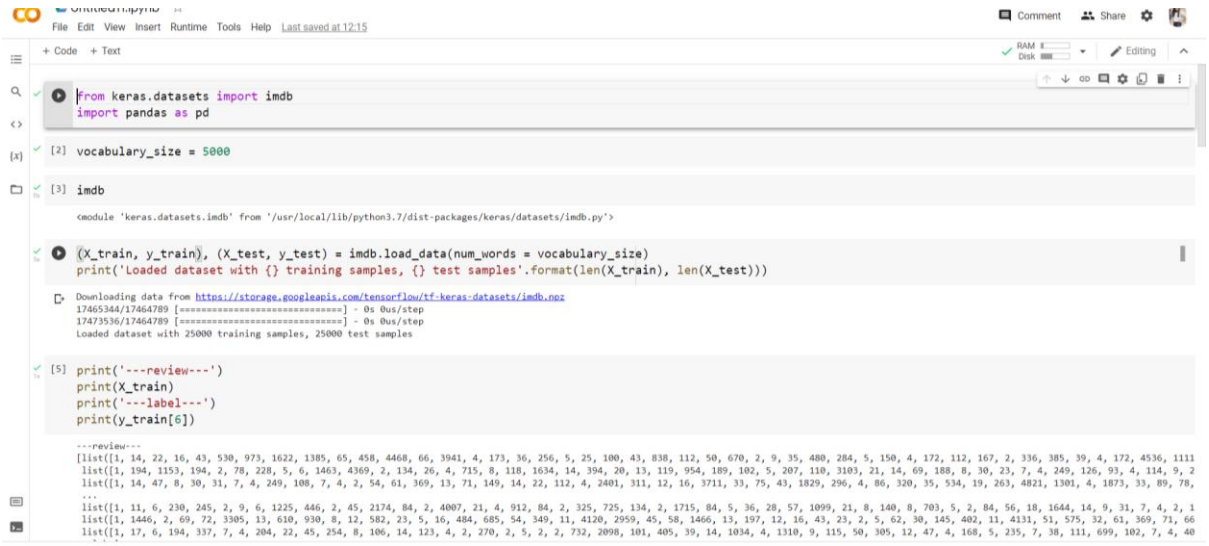
Working of RNN:

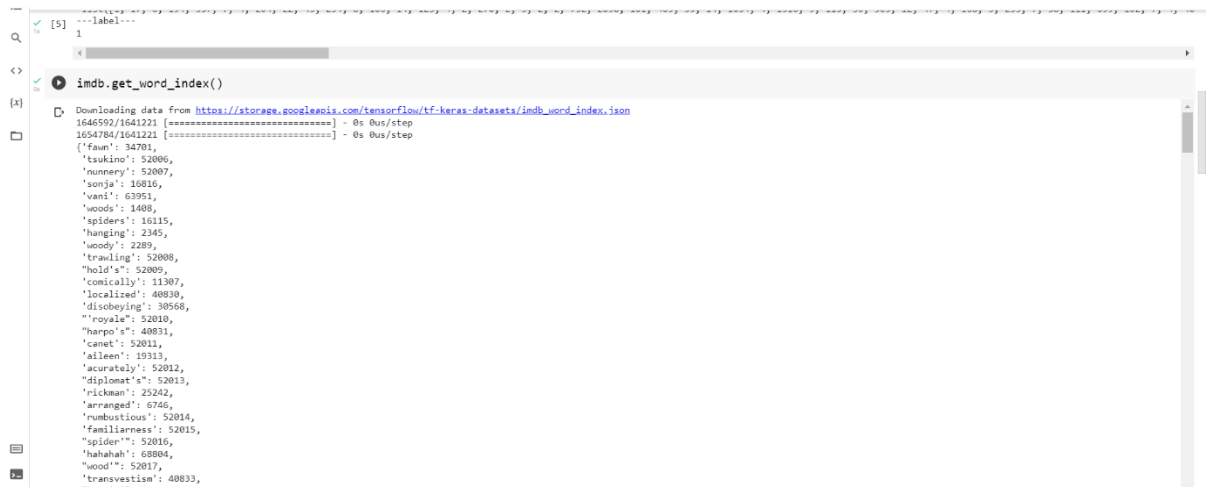
● In Recurrent Neural networks, the information cycles through a loop to the middle hidden layer.
● The input layer 'x' takes in the input to the neural network and processes it and passes it onto the middle layer.
● The middle layer 'h' can consist of multiple hidden layers, each with its own activation functions and weights and biases. If you have a neural network where the various parameters of different hidden layers are not affected by the previous layer, ie: the neural network does not have memory, then you can use a recurrent neural network.
● The Recurrent Neural Network will standardize the different activation functions and weights and biases so that each hidden layer has the same parameters. Then, instead of creating multiple hidden layers, it will create one and loop over it as many times as required.

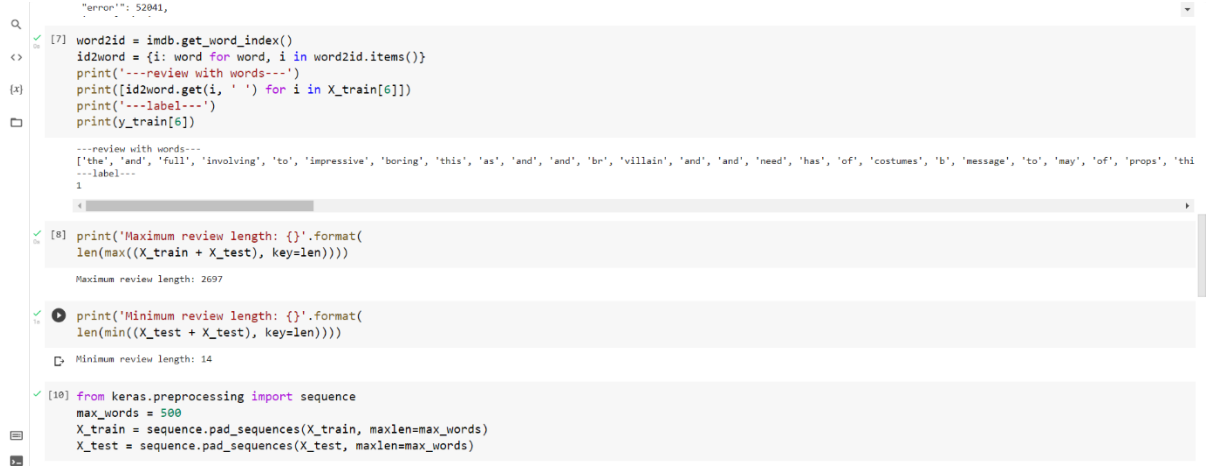


Implementation link:
https://colab.research.google.com/drive/118t3fawgrOjLC5_I5JPkDTB3isGv_NpT

Code & Output:


File Edit View Insert Runtime Tools Help Last saved at 12:15
Comment
Share
+ Code
+ Text
RAM
Disk
Editing
from keras.datasets import imdb
import pandas as pd
[2] vocabulary_size = 5000
[3] imdb
<module 'keras.datasets.imdb' from '/usr/local/lib/python3.7/dist-packages/keras/datasets/imdb.py'>
(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(X_test)))
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17465344/17464789 [==============================] - 0s 0us/step
17473536/17464789 [==============================] - 0s 0us/step
Loaded dataset with 25000 training samples, 25000 test samples
[5] print('---review---')
print(X_train)
print('---label---')
print(y_train[6])
---review---
[list([1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43, 838, 112, 50, 670, 2, 9, 35, 480, 284, 5, 150, 4, 172, 112, 167, 2, 336, 385, 39, 4, 172, 4536, 1111
list([1, 194, 1153, 194, 2, 78, 228, 5, 6, 1463, 4369, 2, 134, 26, 4, 715, 8, 118, 1634, 14, 394, 20, 13, 119, 954, 189, 102, 5, 207, 110, 3103, 21, 14, 69, 188, 8, 30, 23, 7, 4, 249, 126, 93, 4, 114, 9, 2
list([1, 14, 47, 8, 30, 31, 7, 4, 249, 108, 7, 4, 2, 54, 61, 369, 13, 71, 149, 14, 22, 112, 4, 2401, 311, 12, 16, 3711, 33, 75, 43, 1829, 296, 4, 86, 320, 35, 534, 19, 263, 4821, 1301, 4, 1873, 33, 89, 78,
...
list([1, 11, 6, 230, 245, 2, 9, 6, 1225, 446, 2, 45, 2174, 84, 2, 4007, 21, 4, 912, 84, 2, 325, 725, 134, 2, 1715, 84, 5, 36, 28, 57, 1099, 21, 8, 140, 8, 703, 5, 2, 84, 56, 18, 1644, 14, 9, 31, 7, 4, 2, 1
list([1, 1446, 2, 69, 72, 3305, 13, 610, 930, 8, 12, 582, 23, 5, 16, 484, 685, 54, 349, 11, 4120, 2959, 45, 58, 1466, 13, 197, 12, 16, 43, 23, 2, 5, 62, 30, 145, 402, 11, 4131, 51, 575, 32, 61, 369, 71, 66
list([1, 17, 6, 194, 337, 7, 4, 204, 22, 45, 254, 8, 106, 14, 123, 4, 2, 270, 2, 5, 2, 2, 732, 2098, 101, 405, 39, 14, 1034, 4, 1310, 9, 115, 50, 305, 12, 47, 4, 168, 5, 235, 7, 38, 111, 699, 102, 7, 4, 40


[5] ---label---
1
imdb.get_word_index()
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb_word_index.json
1646592/1641221 [==============================] - 0s 0us/step
1654784/1641221 [==============================] - 0s 0us/step
{'fawn': 34701,
'tsukino': 52006,
'nunnery': 52007,
'sonja': 16816,
'vani': 63951,
'woods': 1408,
'spiders': 16115,
'hanging': 2345,
'woody': 2289,
'trawling': 52008,
"hold's": 52009,
'comically': 11307,
'localized': 40830,
'disobeying': 30568,
"'royale": 52010,
"harpo's": 40831,
'canet': 52011,
'aileen': 19313,
'acurately': 52012,
"diplomat's": 52013,
'rickman': 25242,
'arranged': 6746,
'rumbustious': 52014,
'familiarness': 52015,
"spider'": 52016,
'hahahah': 68804,
"wood'": 52017,
'transvestism': 40833,


"error'": 52041,
[7] word2id = imdb.get_word_index()
id2word = {i: word for word, i in word2id.items()}
print('---review with words---')
print([id2word.get(i, ' ') for i in X_train[6]])
print('---label---')
print(y_train[6])
---review with words---
['the', 'and', 'full', 'involving', 'to', 'impressive', 'boring', 'this', 'as', 'and', 'and', 'br', 'villain', 'and', 'and', 'need', 'has', 'of', 'costumes', 'b', 'message', 'to', 'may', 'of', 'props', 'thi
---label---
1
[8] 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
[10] from keras.preprocessing import sequence
max_words = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_words)
X_test = sequence.pad_sequences(X_test, maxlen=max_words)

```
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, 500, 32)           160000

 lstm (LSTM)                 (None, 100)               53200

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 213,301
Trainable params: 213,301
Non-trainable params: 0
_________________________________________________________________
None
```

```
_________________________________________________________________
None
```

```
[12] model.compile(loss='binary_crossentropy',
                   optimizer='adam',
                   metrics=['accuracy'])
```

```
[13] batch_size = 64
num_epochs = 3
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, epochs=num_epochs)
```

```
Epoch 1/3
390/390 [==============================] - 221s 561ms/step - loss: 0.4635 - accuracy: 0.7653 - val_loss: 0.2910 - val_accuracy: 0.9219
Epoch 2/3
390/390 [==============================] - 217s 556ms/step - loss: 0.3164 - accuracy: 0.8714 - val_loss: 0.2588 - val_accuracy: 0.9219
Epoch 3/3
390/390 [==============================] - 217s 557ms/step - loss: 0.2409 - accuracy: 0.9066 - val_loss: 0.2202 - val_accuracy: 0.9219
<keras.callbacks.History at 0x7f77fd7d1fd0>
```

```
scores = model.evaluate(X_test, y_test, verbose=0)
print('Test accuracy:', scores[1])
```

```
Test accuracy: 0.8769999742507935
```