```python
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

import os

import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.models import Sequential
#from tensorflow.keras.utils import to_categorical
from tensorflow.python.keras.utils.np_utils import to_categorical
from tensorflow.keras.optimizers import Adam

import pickle
```

```python
from google.colab import files
uploaded = files.upload()
```

Choose Files pride_and_prejudice.txt

- **pride_and_prejudice.txt**(text/plain) - 772448 bytes, last modified: 11/1/2023 - 100% done

```
Saving pride_and_prejudice.txt to pride_and_prejudice.txt
```

```python
file = open("/content/pride_and_prejudice.txt","r",encoding="utf8")
```

```python
lines = []

for i in file:
    lines.append(i)
```

```python
lines
```

```
 'minutes with him in his library. He had entertained hopes of being\n',
 'admitted to a sight of the young ladies, of whose beauty he had heard\n',
 'much; but he saw only the father. The ladies were somewhat more\n',
 'fortunate, for they had the advantage of ascertaining, from an upper\n',
 'window, that he wore a blue coat and rode a black horse.\n',
 '\n',
 'An invitation to dinner was soon afterwards despatched; and already had\n',
 'Mrs. Bennet planned the courses that were to do credit to her\n',
 'housekeeping, when an answer arrived which deferred it all. Mr. Bingley\n',
 'was obliged to be in town the following day, and consequently unable to\n',
 'accept the honour of their invitation, etc. Mrs. Bennet was quite\n',
 'disconcerted. She could not imagine what business he could have in town\n',
 'so soon after his arrival in Hertfordshire; and she began to fear that\n',
 'he might always be flying about from one place to another, and never\n',
 'settled at Netherfield as he ought to be. Lady Lucas quieted her fears a\n',
 'little by starting the idea of his\n',
 '\n',
 '[Illustration:\n',
 '\n',
 '     “When the Party entered”\n',
 '\n',
 '[_Copyright 1894 by George Allen._]]\n',
 ...]
```

```python
data = ""

for i in lines:
    data = ' '.join(lines)
```

```python
data
```

```
'\ufeffThe Project Gutenberg eBook of Pride and Prejudice\n     \n This ebook is for the use of anyone anywhere in the United States and
ost other parts of the world at no cost and with almost no restrictions\n whatsoever. You may copy it, give it away or re-use it under t
erms\n of the Project Gutenberg License included with this ebook or online\n at www.gutenberg.org. If you are not located in the United
es,\n you will have to check the laws of the country where you are located\n before using this eBook.\n \n Title: Pride and Prejudice\n
\n Author: Jane Austen\n \n Release date: June 1, 1998 [eBook #1342]\n                 Most recently updated: April 14, 2023\n \n Langua
English\n \n Credits: Chuck Greif and the Online Distributed Proofreading Team at http://www.pgdp.net (This file was produced from image
ailable at The Internet Archive)\n \n \n *** START OF THE PROJECT GUTENBERG EBOOK PRIDE AND PREJUDICE ***\n \n \n \n \n
[Illustration:…'
```

```python
data = data.replace('\n','').replace('\r','').replace('\ufeff','').replace('"','').replace('"','
```

```python
data = data.split()
data = ' '.join(data)
data[:500]
```

```
'The Project Gutenberg eBook of Pride and Prejudice This ebook is for the use of anyone anywhere in the United States and most other par
f the world at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Pro
Gutenberg License included with this ebook or online at www.gutenberg.org. If you are not located in the United States, you will have to
ck the laws of the country where you are located before using this eBook. Title:'
```

```python
len(data)
```

```
737518
```

```python
tokenizer = Tokenizer()
tokenizer.fit_on_texts([data])

pickle.dump(tokenizer, open('token.pkl','wb'))

sequence_data = tokenizer.texts_to_sequences([data])[0]

sequence_data[:15]
```

```
[1, 189, 451, 1029, 3, 304, 4, 975, 42, 1029, 24, 22, 1, 515, 3]
```

```
len(sequence_data)
```

```
132957
```

```
vocab_size = len(tokenizer.word_index) + 1
print(vocab_size)
```

```
7560
```

```
sequences = []

for i in range(3, len(sequence_data)):
    words = sequence_data[i-3:i+1]
    sequences.append(words)

print("The Length of sequences are : ", len(sequences))
sequences = np.array(sequences)
sequences[:10]
```

```
The Length of sequences are :  132954
array([[   1,  189,  451, 1029],
       [ 189,  451, 1029,    3],
       [ 451, 1029,    3,  304],
       [1029,    3,  304,    4],
       [   3,  304,    4,  975],
       [ 304,    4,  975,   42],
       [   4,  975,   42, 1029],
       [ 975,   42, 1029,   24],
       [  42, 1029,   24,   22],
       [1029,   24,   22,    1]])
```

```
X = []
y = []

for i in sequences:
    X.append(i[0:3])
    y.append(i[3])

X = np.array(X)
y = np.array(y)
```

```
print("Data: ", X[:10])
print("Responce: ",y[:10])
```

```
Data:  [[   1  189  451]
 [ 189  451 1029]
 [ 451 1029    3]
 [1029    3  304]
 [   3  304    4]
 [ 304    4  975]
 [   4  975   42]
 [ 975   42 1029]
 [  42 1029   24]
 [1029   24   22]]
Responce:  [1029    3  304    4  975   42 1029   24   22    1]
```

```
y = to_categorical(y, num_classes=vocab_size)
y[:5]
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

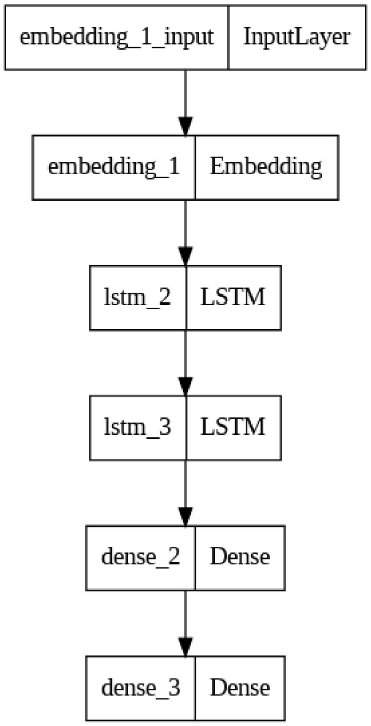
```
model = Sequential()
model.add(Embedding(vocab_size, 10,input_length=3))
model.add(LSTM(1000, return_sequences=True))
model.add(LSTM(1000))
model.add(Dense(1000, activation="relu"))
model.add(Dense(vocab_size, activation="softmax"))

model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_1 (Embedding)     (None, 3, 10)             75600     
                                                                 
 lstm_2 (LSTM)               (None, 3, 1000)           4044000   
                                                                 
 lstm_3 (LSTM)               (None, 1000)              8004000   
                                                                 
 dense_2 (Dense)             (None, 1000)              1001000   
                                                                 
 dense_3 (Dense)             (None, 7560)              7567560   
                                                                 
=================================================================
Total params: 20692160 (78.93 MB)
Trainable params: 20692160 (78.93 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
from tensorflow import keras
from keras.utils import plot_model

keras.utils.plot_model(model, to_file='plot.png', show_layer_n
```



```
!pip install keras-utils
```

```
Collecting keras-utils
  Downloading keras-utils-1.0.13.tar.gz (2.4 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: Keras>=2.1.5 in /usr/local/lib/python3.10/dist-packages (from keras-utils) (2.14.0)
Building wheels for collected packages: keras-utils
  Building wheel for keras-utils (setup.py) ... done
  Created wheel for keras-utils: filename=keras_utils-1.0.13-py3-none-any.whl size=2632 sha256=a7b1c99023824d3d43e341583a189744a2c95cec2
  Stored in directory: /root/.cache/pip/wheels/5c/c0/b3/0c332de4fd71f3733ea6d61697464b7ae4b2b5ff0300e6ca7a
Successfully built keras-utils
Installing collected packages: keras-utils
Successfully installed keras-utils-1.0.13
```

```
from keras.src.mixed_precision.loss_scale_optimizer import optimizer
from tensorflow.keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint("next_words.h5", monitor = 'loss', verbose=1, save_best_only = True
model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001))
model.fit(X,y,epochs=50, batch_size=60,callbacks=[checkpoint])
```

```
2216/2216 [==============================] - 37s 17ms/step - loss: 0.6374
Epoch 37/50
2213/2216 [============================>.] - ETA: 0s - loss: 0.6285
Epoch 37: loss improved from 0.63744 to 0.62864, saving model to next_words.h5
2216/2216 [==============================] - 37s 17ms/step - loss: 0.6286
Epoch 38/50
2216/2216 [==============================] - ETA: 0s - loss: 0.6127
Epoch 38: loss improved from 0.62864 to 0.61265, saving model to next_words.h5
2216/2216 [==============================] - 37s 17ms/step - loss: 0.6127
Epoch 39/50
2213/2216 [============================>.] - ETA: 0s - loss: 0.6029
Epoch 39: loss improved from 0.61265 to 0.60290, saving model to next_words.h5
2216/2216 [==============================] - 39s 17ms/step - loss: 0.6029
Epoch 40/50
2215/2216 [============================>.] - ETA: 0s - loss: 0.5912
Epoch 40: loss improved from 0.60290 to 0.59130, saving model to next_words.h5
2216/2216 [==============================] - 42s 19ms/step - loss: 0.5913
Epoch 41/50
2215/2216 [============================>.] - ETA: 0s - loss: 0.5824
Epoch 41: loss improved from 0.59130 to 0.58243, saving model to next_words.h5
2216/2216 [==============================] - 42s 19ms/step - loss: 0.5824
Epoch 42/50
2214/2216 [============================>.] - ETA: 0s - loss: 0.5734
Epoch 42: loss improved from 0.58243 to 0.57338, saving model to next_words.h5
2216/2216 [==============================] - 42s 19ms/step - loss: 0.5734
Epoch 43/50
2214/2216 [============================>.] - ETA: 0s - loss: 0.5663
Epoch 43: loss improved from 0.57338 to 0.56637, saving model to next_words.h5
2216/2216 [==============================] - 41s 19ms/step - loss: 0.5664
Epoch 44/50
2216/2216 [==============================] - ETA: 0s - loss: 0.5560
Epoch 44: loss improved from 0.56637 to 0.55603, saving model to next_words.h5
2216/2216 [==============================] - 42s 19ms/step - loss: 0.5560
Epoch 45/50
2214/2216 [============================>.] - ETA: 0s - loss: 0.5497
Epoch 45: loss improved from 0.55603 to 0.54978, saving model to next_words.h5
2216/2216 [==============================] - 43s 19ms/step - loss: 0.5498
Epoch 46/50
2216/2216 [==============================] - ETA: 0s - loss: 0.5425
Epoch 46: loss improved from 0.54978 to 0.54251, saving model to next_words.h5
2216/2216 [==============================] - 46s 21ms/step - loss: 0.5425
Epoch 47/50
2213/2216 [============================>.] - ETA: 0s - loss: 0.5360
Epoch 47: loss improved from 0.54251 to 0.53599, saving model to next_words.h5
2216/2216 [==============================] - 48s 21ms/step - loss: 0.5360
Epoch 48/50
2213/2216 [============================>.] - ETA: 0s - loss: 0.5274
Epoch 48: loss improved from 0.53599 to 0.52765, saving model to next_words.h5
2216/2216 [==============================] - 42s 19ms/step - loss: 0.5276
Epoch 49/50
2215/2216 [============================>.] - ETA: 0s - loss: 0.5268
Epoch 49: loss improved from 0.52765 to 0.52673, saving model to next_words.h5
2216/2216 [==============================] - 42s 19ms/step - loss: 0.5267
Epoch 50/50
2216/2216 [==============================] - ETA: 0s - loss: 0.5162
Epoch 50: loss improved from 0.52673 to 0.51619, saving model to next_words.h5
2216/2216 [==============================] - 42s 19ms/step - loss: 0.5162
<keras.src.callbacks.History at 0x7e50905929e0>
```

```
from tensorflow.keras.models import load_model
import numpy as np
import pickle
```

```
model=load_model('next_words.h5')
tokenizer = pickle.load(open('token.pkl','rb'))

```

```
def Predict_Next_Words(model,tokenizer, text):
    sequence = tokenizer.texts_to_sequences([text])
    sequence = np.array(sequence)
    preds = np.argmax(model.predict(sequence))
    predicted_word = ""

    for key,value in tokenizer.word_index.items():
        if value == preds:
            predicted_word = key
            break

    print(predicted_word)
    return predicted_word
```

```
while (True):
    text = input("Enter your text: ")

    if text == "0":
        print("Execution completed.....")
        break

    else:
        try:
            text = text.split(" ")
            text = text[-3:]
            print(text)

            Predict_Next_Words(model, tokenizer,text)


        except Exception as e:
            print("Error occured: ",e)
            continue
```

```
Enter your text: acknowledgment of all
['acknowledgment', 'of', 'all']
1/1 [==============================] - 1s 683ms/step
i
Enter your text: GUTENBERG EBOOK PRIDE
['GUTENBERG', 'EBOOK', 'PRIDE']
1/1 [==============================] - 0s 19ms/step
and
Enter your text: Most recently updated
['Most', 'recently', 'updated']
1/1 [==============================] - 0s 19ms/step
april
Enter your text: A Beginner’s Guide
['A', 'Beginner’s', 'Guide']
1/1 [==============================] - 1s 610ms/step
the
```

1

```
1/1 [==============================] - 0s 19ms/step
the
Enter your text: You Talk About
['You', 'Talk', 'About']
1/1 [==============================] - 0s 18ms/step
your
Enter your text: he is playing
['he', 'is', 'playing']
1/1 [==============================] - 0s 19ms/step
between
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-38-d75f3402ccb5> in <cell line: 1>()
      1 while (True):
----> 2     text = input("Enter your text: ")
      3 
      4     if text == "0":
      5         print("Execution completed.....")
```

1 frames

```
/usr/local/lib/python3.10/dist-packages/ipykernel/kernelbase.py in _input_request(self,
prompt, ident, parent, password)
    893             except KeyboardInterrupt:
```