Importing the libraries

In [2]:
```python
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.keras.optimizers import Adam
import pickle
import numpy as np
import os
```

In [3]:
```python
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen   Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving ip for rnn.txt to ip for rnn.txt
```

Loading and Pre-Processing the Data

In [4]:
```python
# Now, Let's Load the data.
file = open("ip for rnn.txt", "r", encoding = "utf8")
# we opened the file in reading mode.

#Let's Pre-Process the data.
#Pre-Processng methods depends on our data.

# store file in list
lines = []
for i in file:
    lines.append(i)

# Convert list to string
data = ""
for i in lines:
  data = ' '. join(lines)

#Depending on our data these steps are needed to mae our data suitable for processing.
#replace unnecessary stuff with space
data = data.replace('\n', '').replace('\r', '').replace('\ufeff', '').replace('“','').replace('”','')  #new line, carriage return, unicode character --> replace by space

#remove unnecessary spaces
data = data.split()
data = ' '.join(data)
data[:500]
```

Out[4]:
```
'The Project Gutenberg eBook of Pride and Prejudice, by Jane Austen This eBook is for the use of anyone anywhere in the United States and most other parts of the world at no cost a
nd with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gut
enberg.org. If you are not located in the United States, you will have to check the laws of the country where you are located before using th'
```

In [5]:
```python
len(data)
```

Out[5]:
```
698483
```

Tokenizer:

In [6]:
```python
tokenizer = Tokenizer()
tokenizer.fit_on_texts([data])

# saving the tokenizer for predict function
pickle.dump(tokenizer, open('token.pkl', 'wb'))

sequence_data = tokenizer.texts_to_sequences([data])[0]
sequence_data[:15]
```

Out[6]:
```
[1, 176, 157, 916, 3, 321, 4, 1174, 30, 72, 2535, 41, 916, 23, 21]
```

In [7]:
```python
len(sequence_data)
```

Out[7]:
```
125317
```

In [8]:
```python
vocab_size = len(tokenizer.word_index) + 1
print(vocab_size)
```

```
7036
```

Here we are using 3 words to predict the 4th word.

In [9]:
```python
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:  125314
```

Out[9]:
```
array([[   1,  176,  157,  916],
       [ 176,  157,  916,    3],
       [ 157,  916,    3,  321],
       [ 916,    3,  321,    4],
       [   3,  321,    4, 1174],
       [ 321,    4, 1174,   30],
       [   4, 1174,   30,   72],
       [1174,   30,   72, 2535],
       [  30,   72, 2535,   41],
       [  72, 2535,   41,  916]])
```

Above array each element has 4 elements the first 3 are the input and the last word is the output.

In [10]:
```python
# Now as we got to kow what are the i/p and o/p are so let's separate them.
X = []
y = []

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

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

In [11]:
```python
print("Input Data: ", X[:10])
print("Output Response: ", y[:10])
```

```
Input Data:  [[   1  176  157]
 [ 176  157  916]
 [ 157  916    3]
 [ 916    3  321]
 [   3  321    4]
 [ 321    4 1174]
 [   4 1174   30]
 [1174   30   72]
 [  30   72 2535]
 [  72 2535   41]]
Output Response:  [ 916    3  321    4 1174   30   72 2535   41  916]
```

In [12]:
```python
#Now we will be converting our Class Vectors into some binary Class Matrix Later we will be using Loss function as Categorical Loss entropy. This requires our Interger values to be
y = to_categorical(y, num_classes=vocab_size)
y[:5]
```

Out[12]:
```
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)
```

Now Let's Create the Model

In [13]:
```python
#we will be using Sequential for our model RNN
model = Sequential()
model.add(Embedding(vocab_size, 10, input_length=3))
#input_length means we will be using 3 words to predict next word.
model.add(LSTM(1000, return_sequences=True))
model.add(LSTM(1000))
#Two LSTM layers are there
model.add(Dense(1000, activation="relu"))
model.add(Dense(vocab_size, activation="softmax"))
```

In [14]:
```python
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 3, 10)             70360
_________________________________________________________________
lstm (LSTM)                  (None, 3, 1000)           4044000
_________________________________________________________________
lstm_1 (LSTM)                (None, 1000)              8004000
_________________________________________________________________
dense (Dense)                (None, 1000)              1001000
_________________________________________________________________
dense_1 (Dense)              (None, 7036)              7043036
=================================================================
Total params: 20,162,396
Trainable params: 20,162,396
Non-trainable params: 0
_________________________________________________________________
```

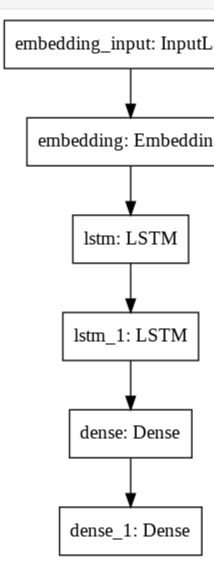
Ploting Our Model

In [15]:
```python
#Just Take a look at Our Model

from tensorflow import keras
from keras.utils.vis_utils import plot_model

keras.utils.plot_model(model, to_file='plot.png', show_layer_names=True)
```

Out[15]:



Building Our Model

In [16]:
```python
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=70, batch_size=64, callbacks=[checkpoint])
```

```
Epoch 1/70
1959/1959 [==============================] - 38s 15ms/step - loss: 6.2294

Epoch 00001: loss improved from inf to 6.22942, saving model to next_words.h5
Epoch 2/70
1959/1959 [==============================] - 30s 15ms/step - loss: 5.5943

Epoch 00002: loss improved from 6.22942 to 5.59425, saving model to next_words.h5
Epoch 3/70
1959/1959 [==============================] - 30s 15ms/step - loss: 5.2643

Epoch 00003: loss improved from 5.59425 to 5.26425, saving model to next_words.h5
Epoch 4/70
1959/1959 [==============================] - 30s 15ms/step - loss: 5.0368

Epoch 00004: loss improved from 5.26425 to 5.03678, saving model to next_words.h5
Epoch 5/70
1959/1959 [==============================] - 30s 15ms/step - loss: 4.8360

Epoch 00005: loss improved from 5.03678 to 4.83603, saving model to next_words.h5
Epoch 6/70
1959/1959 [==============================] - 30s 15ms/step - loss: 4.6383

Epoch 00006: loss improved from 4.83603 to 4.63827, saving model to next_words.h5
Epoch 7/70
1959/1959 [==============================] - 30s 15ms/step - loss: 4.4390

Epoch 00007: loss improved from 4.63827 to 4.43897, saving model to next_words.h5
Epoch 8/70
1959/1959 [==============================] - 30s 15ms/step - loss: 4.2460

Epoch 00008: loss improved from 4.43897 to 4.24598, saving model to next_words.h5
Epoch 9/70
1959/1959 [==============================] - 30s 15ms/step - loss: 4.0521

Epoch 00009: loss improved from 4.24598 to 4.05209, saving model to next_words.h5
Epoch 10/70
1959/1959 [==============================] - 30s 15ms/step - loss: 3.8508

Epoch 00010: loss improved from 4.05209 to 3.85077, saving model to next_words.h5
Epoch 11/70
1959/1959 [==============================] - 30s 15ms/step - loss: 3.6498

Epoch 00011: loss improved from 3.85077 to 3.64976, saving model to next_words.h5
Epoch 12/70
1959/1959 [==============================] - 30s 15ms/step - loss: 3.4436

Epoch 00012: loss improved from 3.64976 to 3.44359, saving model to next_words.h5
Epoch 13/70
1959/1959 [==============================] - 30s 15ms/step - loss: 3.2351

Epoch 00013: loss improved from 3.44359 to 3.23513, saving model to next_words.h5
Epoch 14/70
1959/1959 [==============================] - 30s 15ms/step - loss: 3.0253

Epoch 00014: loss improved from 3.23513 to 3.02528, saving model to next_words.h5
Epoch 15/70
1959/1959 [==============================] - 30s 15ms/step - loss: 2.8119

Epoch 00015: loss improved from 3.02528 to 2.81192, saving model to next_words.h5
Epoch 16/70
1959/1959 [==============================] - 30s 15ms/step - loss: 2.5897

Epoch 00016: loss improved from 2.81192 to 2.58974, saving model to next_words.h5
Epoch 17/70
1959/1959 [==============================] - 30s 15ms/step - loss: 2.3721

Epoch 00017: loss improved from 2.58974 to 2.37210, saving model to next_words.h5
Epoch 18/70
1959/1959 [==============================] - 30s 15ms/step - loss: 2.1531

Epoch 00018: loss improved from 2.37210 to 2.15306, saving model to next_words.h5
Epoch 19/70
1959/1959 [==============================] - 30s 15ms/step - loss: 1.9439

Epoch 00019: loss improved from 2.15306 to 1.94393, saving model to next_words.h5
Epoch 20/70
1959/1959 [==============================] - 30s 16ms/step - loss: 1.7422

Epoch 00020: loss improved from 1.94393 to 1.74222, saving model to next_words.h5
Epoch 21/70
1959/1959 [==============================] - 30s 15ms/step - loss: 1.5599

Epoch 00021: loss improved from 1.74222 to 1.55992, saving model to next_words.h5
Epoch 22/70
1959/1959 [==============================] - 30s 15ms/step - loss: 1.3994

Epoch 00022: loss improved from 1.55992 to 1.39941, saving model to next_words.h5
Epoch 23/70
1959/1959 [==============================] - 30s 15ms/step - loss: 1.2620

Epoch 00023: loss improved from 1.39941 to 1.26204, saving model to next_words.h5
Epoch 24/70
1959/1959 [==============================] - 30s 15ms/step - loss: 1.1464

Epoch 00024: loss improved from 1.26204 to 1.14640, saving model to next_words.h5
Epoch 25/70
1959/1959 [==============================] - 30s 15ms/step - loss: 1.0492

Epoch 00025: loss improved from 1.14640 to 1.04921, saving model to next_words.h5
Epoch 26/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.9733

Epoch 00026: loss improved from 1.04921 to 0.97328, saving model to next_words.h5
Epoch 27/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.9054

Epoch 00027: loss improved from 0.97328 to 0.90537, saving model to next_words.h5
Epoch 28/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.8547

Epoch 00028: loss improved from 0.90537 to 0.85473, saving model to next_words.h5
Epoch 29/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.8096

Epoch 00029: loss improved from 0.85473 to 0.80957, saving model to next_words.h5
Epoch 30/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.7745

Epoch 00030: loss improved from 0.80957 to 0.77445, saving model to next_words.h5
Epoch 31/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.7423

Epoch 00031: loss improved from 0.77445 to 0.74230, saving model to next_words.h5
Epoch 32/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.7175

Epoch 00032: loss improved from 0.74230 to 0.71750, saving model to next_words.h5
Epoch 33/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.6969

Epoch 00033: loss improved from 0.71750 to 0.69693, saving model to next_words.h5
Epoch 34/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.6715

Epoch 00034: loss improved from 0.69693 to 0.67148, saving model to next_words.h5
Epoch 35/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.6574

Epoch 00035: loss improved from 0.67148 to 0.65738, saving model to next_words.h5
Epoch 36/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.6427

Epoch 00036: loss improved from 0.65738 to 0.64275, saving model to next_words.h5
Epoch 37/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.6259

Epoch 00037: loss improved from 0.64275 to 0.62590, saving model to next_words.h5
Epoch 38/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.6145

Epoch 00038: loss improved from 0.62590 to 0.61449, saving model to next_words.h5
Epoch 39/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.6042

Epoch 00039: loss improved from 0.61449 to 0.60418, saving model to next_words.h5
Epoch 40/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.5946

Epoch 00040: loss improved from 0.60418 to 0.59463, saving model to next_words.h5
Epoch 41/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.5807

Epoch 00041: loss improved from 0.59463 to 0.58068, saving model to next_words.h5
Epoch 42/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.5711

Epoch 00042: loss improved from 0.58068 to 0.57108, saving model to next_words.h5
Epoch 43/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.5639

Epoch 00043: loss improved from 0.57108 to 0.56392, saving model to next_words.h5
Epoch 44/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.5542

Epoch 00044: loss improved from 0.56392 to 0.55424, saving model to next_words.h5
Epoch 45/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.5482

Epoch 00045: loss improved from 0.55424 to 0.54818, saving model to next_words.h5
Epoch 46/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.5383

Epoch 00046: loss improved from 0.54818 to 0.53834, saving model to next_words.h5
Epoch 47/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.5349

Epoch 00047: loss improved from 0.53834 to 0.53493, saving model to next_words.h5
Epoch 48/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.5288

Epoch 00048: loss improved from 0.53493 to 0.52876, saving model to next_words.h5
Epoch 49/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.5195

Epoch 00049: loss improved from 0.52876 to 0.51948, saving model to next_words.h5
Epoch 50/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.5180

Epoch 00050: loss improved from 0.51948 to 0.51797, saving model to next_words.h5
Epoch 51/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.5104

Epoch 00051: loss improved from 0.51797 to 0.51039, saving model to next_words.h5
Epoch 52/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.5051

Epoch 00052: loss improved from 0.51039 to 0.50507, saving model to next_words.h5
Epoch 53/70
1959/1959 [==============================] - 30s 16ms/step - loss: 0.4987

Epoch 00053: loss improved from 0.50507 to 0.49873, saving model to next_words.h5
Epoch 54/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.4974

Epoch 00054: loss improved from 0.49873 to 0.49745, saving model to next_words.h5
Epoch 55/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.4908

Epoch 00055: loss improved from 0.49745 to 0.49083, saving model to next_words.h5
Epoch 56/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.4861

Epoch 00056: loss improved from 0.49083 to 0.48613, saving model to next_words.h5
Epoch 57/70
1959/1959 [==============================] - 30s 16ms/step - loss: 0.4826

Epoch 00057: loss improved from 0.48613 to 0.48259, saving model to next_words.h5
Epoch 58/70
1959/1959 [==============================] - 30s 16ms/step - loss: 0.4804

Epoch 00058: loss improved from 0.48259 to 0.48044, saving model to next_words.h5
Epoch 59/70
1959/1959 [==============================] - 30s 16ms/step - loss: 0.4711

Epoch 00059: loss improved from 0.48044 to 0.47111, saving model to next_words.h5
Epoch 60/70
1959/1959 [==============================] - 30s 16ms/step - loss: 0.4707

Epoch 00060: loss improved from 0.47111 to 0.47073, saving model to next_words.h5
Epoch 61/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.4662

Epoch 00061: loss improved from 0.47073 to 0.46620, saving model to next_words.h5
Epoch 62/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.4713

Epoch 00062: loss did not improve from 0.46620
Epoch 63/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.4584

Epoch 00063: loss improved from 0.46620 to 0.45838, saving model to next_words.h5
Epoch 64/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.4590

Epoch 00064: loss did not improve from 0.45838
Epoch 65/70
1959/1959 [==============================] - 30s 16ms/step - loss: 0.4519

Epoch 00065: loss improved from 0.45838 to 0.45192, saving model to next_words.h5
Epoch 66/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.4520

Epoch 00066: loss did not improve from 0.45192
Epoch 67/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.4509

Epoch 00067: loss improved from 0.45192 to 0.45086, saving model to next_words.h5
Epoch 68/70
1959/1959 [==============================] - 31s 16ms/step - loss: 0.4506

Epoch 00068: loss improved from 0.45086 to 0.45061, saving model to next_words.h5
Epoch 69/70
1959/1959 [==============================] - 31s 16ms/step - loss: 0.4400

Epoch 00069: loss improved from 0.45061 to 0.44001, saving model to next_words.h5
Epoch 70/70
1959/1959 [==============================] - 30s 15ms/step - loss: 0.4417

Epoch 00070: loss did not improve from 0.44001
```

Out[16]:
```
<tensorflow.python.keras.callbacks.History at 0x7f1b6950cc10>
```

Prediction:

In [17]:
```python
from tensorflow.keras.models import load_model
import numpy as np
import pickle

# Load the model and tokenizer
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
```

In [18]:
```python
while(True):
  text = input("Enter your line: ")

  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 occurred: ",e)
        continue
```

```
Enter your line: after the melancholy scene so lately
['scene', 'so', 'lately']
gone
Enter your line: 0
Execution completed.....
```