In [1]:

```python
import tensorflow as tf
import string
import requests
```

In [2]:

```python
response=requests.get("http://www.gutenberg.org/cache/epub/5200/pg5200.txt")
```

In [3]:

```python
response.text[:1500]
```

Out[3]:

```
'\ufeffThe Project Gutenberg EBook of Metamorphosis, by Franz Kafka\r\nTranslated by David Wyllie.\r\n\r\nThis eBook is for the use of anyone anywhere at no cost and with\r\nalmost no restrictions whatsoever.  You may copy it, give it away or\r\nre-use it under the terms of the Project Gutenberg License included\r\nwith this eBook or online at www.gutenberg.org\r\n\r\n** This is a COPYRIGHTED Project Gutenberg eBook, Details Below **\r\n**     Please follow the copyright guidelines in this file.     **\r\n\r\n\r\nTitle: Metamorphosis\r\n\r\nAuthor: Franz Kafka\r\n\r\nTranslator: David Wyllie\r\n\r\nRelease Date: August 16, 2005 [EBook #5200]\r\nFirst posted: May 13, 2002\r\nLast updated: May 20, 2012\r\n\r\nLanguage: English\r\n\r\n\r\n*** START OF THIS PROJECT GUTENBERG EBOOK METAMORPHOSIS ***\r\n\r\n\r\n\r\n\r\n\r\nCopyright (C) 2002 David Wyllie.\r\n\r\n\r\n\r\n\r\n\r\n  Metamorphosis\r\n  Franz Kafka\r\n\r\nTranslated by David Wyllie\r\n\r\n\r\n\r\nI\r\n\r\n\r\nOne morning, when Gregor Samsa woke from troubled dreams, he found\r\nhimself transformed in his bed into a horrible vermin.  He lay on\r\nhis armour-like back, and if he lifted his head a little he could\r\nsee his brown belly, slightly domed and divided by arches into stiff\r\nsections.  The bedding was hardly able to cover it and seemed ready\r\nto slide off any moment.  His many legs, pitifully thin compared\r\nwith the size of the rest of him, waved about helplessly as he\r\nlooked.\r\n\r\n"What\'s happened to me?" he thought.  It wasn\'t a dream.  His room,\r\na proper human room although a little too small, lay peacefully\r\nbetwee'
```

In [4]:

```python
data = response.text.split('\n')
data[0]
```

Out[4]: `'\ufeffThe Project Gutenberg EBook of Metamorphosis, by Franz Kafka\r'`

In [5]:

```python
data = data[253:]
data[0]
```

Out[5]: `'away from the bed, bend down with the load and then be patient and\r'`

In [6]:

```python
len(data)
```

Out[6]: 2110

In [7]:

```python
data = " ".join(data)
data[:1000]
```

Out[7]:

```
'away from the bed, bend down with the load and then be patient and\r careful as he swang over onto the floor, where, hopefully, the\r little legs would find a use.  Should he really call for help\r though, even apart from the fact that all the doors were locked?\r Despite all the difficulty he was in, he could not suppress a smile\r at this thought.\r \r After a while he had already moved so far across that it would have\r been hard for him to keep his balance if he rocked too hard.  The\r time was now ten past seven and he would have to make a final\r decision very soon.  Then there was a ring at the door of the flat.\r "That\'ll be someone from work", he said to himself, and froze very\r still, although his little legs only became all the more lively as\r they danced around.  For a moment everything remained quiet.\r "They\'re not opening the door", Gregor said to himself, caught in\r some nonsensical hope.  But then of course, the maid\'s firm steps\r went to the door as ever and opened it.  Gregor on'
```

In [8]:

```python
def clean_text(doc):
 tokens = doc.split()
 table = str.maketrans('', '', string.punctuation)
 tokens = [w.translate(table) for w in tokens]
 tokens = [word for word in tokens if word.isalpha()]
 tokens = [word.lower() for word in tokens]
 return tokens
tokens = clean_text(data)
print(tokens[:50])
```

```
['away', 'from', 'the', 'bed', 'bend', 'down', 'with', 'the', 'load', 'and', 'then', 'be', 'patient', 'and', 'careful', 'as', 'he', 'swang', 'over', 'onto', 'the', 'floor', 'where', 'hopefully', 'the', 'little', 'legs', 'would', 'find', 'a', 'use', 'should', 'he', 'really', 'call', 'for', 'help', 'though', 'even', 'apart', 'from', 'the', 'fact', 'that', 'all', 'the', 'doors', 'were', 'locked', 'despite']
```

In [9]:

```python
len(tokens)
```

Out[9]: 22607

In [10]:

```python
length = 50 + 1
lines = []

for i in range(length, len(tokens)):
  seq = tokens[i-length:i]
  line = ' '.join(seq)
  lines.append(line)
  if i > 200000:
   break

print(len(lines))
```

22556

In [11]:

```python
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

In [12]:

```python
tokenizer = Tokenizer()
tokenizer.fit_on_texts(lines)
sequences = tokenizer.texts_to_sequences(lines)
```

In [13]:

```python
sequences = np.array(sequences)
X, y = sequences[:, :-1], sequences[:,-1]
X[0]
```

Out[13]:

```
array([ 103,   29,    1,  245, 2883,   98,   14,    1, 1435,    3,   48,
         30,  618,    3,  756,   13,    6, 1434,  107,  165,    1,  149,
         86, 2880,    1,   78,  225,   21,  530,   12,  156,  193,    6,
        142,  754,   17,  180,  116,   49, 1433,   29,    1,  753,   11,
         26,    1,  455,   58,  617,  329])
```

In [14]:

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

In [15]:

```python
y = to_categorical(y, num_classes=vocab_size)
```

In [16]:

```python
seq_length = X.shape[1]
seq_length
```

Out[16]: 50

In [17]:

```python
model = Sequential()
model.add(Embedding(vocab_size, 50, input_length=seq_length))
model.add(LSTM(100, return_sequences=True))
model.add(LSTM(100))
model.add(Dense(100, activation='relu'))
model.add(Dense(vocab_size, activation='softmax'))
```

In [18]:

```python
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 50, 50)	144250
lstm (LSTM)	(None, 50, 100)	60400
lstm_1 (LSTM)	(None, 100)	80400
dense (Dense)	(None, 100)	10100
dense_1 (Dense)	(None, 2885)	291385

Total params: 586,535
Trainable params: 586,535
Non-trainable params: 0

In [19]:

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

In [20]:

```python
model.fit(X, y, batch_size = 256, epochs = 100)
```

```
Epoch 1/100
89/89 [==============================] - 36s 403ms/step - loss: 6.6534 - accuracy: 0.0464
Epoch 2/100
89/89 [==============================] - 33s 370ms/step - loss: 6.1880 - accuracy: 0.0540
Epoch 3/100
89/89 [==============================] - 32s 365ms/step - loss: 6.1551 - accuracy: 0.0540
Epoch 4/100
89/89 [==============================] - 33s 365ms/step - loss: 6.0518 - accuracy: 0.0540
Epoch 5/100
89/89 [==============================] - 33s 368ms/step - loss: 5.9690 - accuracy: 0.0541
Epoch 6/100
89/89 [==============================] - 33s 367ms/step - loss: 5.8392 - accuracy: 0.0593
Epoch 7/100
89/89 [==============================] - 32s 359ms/step - loss: 5.7397 - accuracy: 0.0664
Epoch 8/100
89/89 [==============================] - 33s 365ms/step - loss: 5.6532 - accuracy: 0.0732
Epoch 9/100
89/89 [==============================] - 34s 377ms/step - loss: 5.5703 - accuracy: 0.0771
Epoch 10/100
89/89 [==============================] - 35s 391ms/step - loss: 5.4973 - accuracy: 0.0825
Epoch 11/100
89/89 [==============================] - 32s 354ms/step - loss: 5.4175 - accuracy: 0.0884
Epoch 12/100
89/89 [==============================] - 31s 352ms/step - loss: 5.3321 - accuracy: 0.0932
Epoch 13/100
89/89 [==============================] - 32s 363ms/step - loss: 5.2377 - accuracy: 0.1031
Epoch 14/100
89/89 [==============================] - 30s 334ms/step - loss: 5.1354 - accuracy: 0.1085
Epoch 15/100
89/89 [==============================] - 35s 396ms/step - loss: 5.0173 - accuracy: 0.1135
Epoch 16/100
89/89 [==============================] - 20s 226ms/step - loss: 4.9329 - accuracy: 0.1174
Epoch 17/100
89/89 [==============================] - 17s 190ms/step - loss: 4.8586 - accuracy: 0.1217
Epoch 18/100
89/89 [==============================] - 17s 191ms/step - loss: 4.7850 - accuracy: 0.1283
Epoch 19/100
89/89 [==============================] - 17s 189ms/step - loss: 4.7261 - accuracy: 0.1329
Epoch 20/100
89/89 [==============================] - 17s 190ms/step - loss: 4.6663 - accuracy: 0.1348
Epoch 21/100
89/89 [==============================] - 20s 222ms/step - loss: 4.6148 - accuracy: 0.1386
Epoch 22/100
89/89 [==============================] - 23s 257ms/step - loss: 4.5633 - accuracy: 0.1439
Epoch 23/100
89/89 [==============================] - 24s 269ms/step - loss: 4.5158 - accuracy: 0.1440
Epoch 24/100
89/89 [==============================] - 25s 284ms/step - loss: 4.4733 - accuracy: 0.1471
Epoch 25/100
89/89 [==============================] - 20s 227ms/step - loss: 4.4291 - accuracy: 0.1498
Epoch 26/100
89/89 [==============================] - 18s 197ms/step - loss: 4.3900 - accuracy: 0.1509
Epoch 27/100
89/89 [==============================] - 18s 199ms/step - loss: 4.3473 - accuracy: 0.1543
Epoch 28/100
89/89 [==============================] - 18s 204ms/step - loss: 4.3087 - accuracy: 0.1562
Epoch 29/100
89/89 [==============================] - 18s 199ms/step - loss: 4.2722 - accuracy: 0.1586
Epoch 30/100
89/89 [==============================] - 20s 227ms/step - loss: 4.2290 - accuracy: 0.1612
Epoch 31/100
89/89 [==============================] - 18s 202ms/step - loss: 4.1910 - accuracy: 0.1649
Epoch 32/100
89/89 [==============================] - 17s 194ms/step - loss: 4.1578 - accuracy: 0.1657
Epoch 33/100
89/89 [==============================] - 19s 219ms/step - loss: 4.1212 - accuracy: 0.1683
Epoch 34/100
89/89 [==============================] - 19s 213ms/step - loss: 4.0861 - accuracy: 0.1716
Epoch 35/100
89/89 [==============================] - 19s 209ms/step - loss: 4.0489 - accuracy: 0.1755
Epoch 36/100
89/89 [==============================] - 20s 225ms/step - loss: 4.0141 - accuracy: 0.1778
Epoch 37/100
89/89 [==============================] - 19s 208ms/step - loss: 3.9785 - accuracy: 0.1799
Epoch 38/100
89/89 [==============================] - 17s 196ms/step - loss: 3.9523 - accuracy: 0.1819
Epoch 39/100
89/89 [==============================] - 18s 199ms/step - loss: 3.9115 - accuracy: 0.1860
Epoch 40/100
89/89 [==============================] - 18s 199ms/step - loss: 3.8771 - accuracy: 0.1895
Epoch 41/100
89/89 [==============================] - 18s 208ms/step - loss: 3.8506 - accuracy: 0.1915
Epoch 42/100
89/89 [==============================] - 22s 253ms/step - loss: 3.8134 - accuracy: 0.1946
Epoch 43/100
89/89 [==============================] - 19s 210ms/step - loss: 3.8287 - accuracy: 0.1950
Epoch 44/100
89/89 [==============================] - 19s 210ms/step - loss: 3.8398 - accuracy: 0.1949
Epoch 45/100
89/89 [==============================] - 18s 206ms/step - loss: 3.7607 - accuracy: 0.2007
Epoch 46/100
89/89 [==============================] - 21s 238ms/step - loss: 3.7078 - accuracy: 0.2077
Epoch 47/100
89/89 [==============================] - 26s 288ms/step - loss: 3.6642 - accuracy: 0.2117
Epoch 48/100
89/89 [==============================] - 20s 227ms/step - loss: 3.6294 - accuracy: 0.2151
Epoch 49/100
89/89 [==============================] - 19s 211ms/step - loss: 3.6000 - accuracy: 0.2182
Epoch 50/100
89/89 [==============================] - 18s 205ms/step - loss: 3.5772 - accuracy: 0.2213
Epoch 51/100
89/89 [==============================] - 24s 271ms/step - loss: 3.5445 - accuracy: 0.2248
Epoch 52/100
89/89 [==============================] - 20s 226ms/step - loss: 3.5105 - accuracy: 0.2303
Epoch 53/100
89/89 [==============================] - 17s 196ms/step - loss: 3.4758 - accuracy: 0.2355
Epoch 54/100
89/89 [==============================] - 19s 214ms/step - loss: 3.4474 - accuracy: 0.2397
Epoch 55/100
89/89 [==============================] - 18s 198ms/step - loss: 3.4203 - accuracy: 0.2406
Epoch 56/100
89/89 [==============================] - 19s 214ms/step - loss: 3.3891 - accuracy: 0.2469
Epoch 57/100
89/89 [==============================] - 18s 207ms/step - loss: 3.3894 - accuracy: 0.2483
Epoch 58/100
89/89 [==============================] - 18s 207ms/step - loss: 3.3703 - accuracy: 0.2497
Epoch 59/100
89/89 [==============================] - 18s 202ms/step - loss: 3.3278 - accuracy: 0.2546
Epoch 60/100
89/89 [==============================] - 18s 203ms/step - loss: 3.2881 - accuracy: 0.2622
Epoch 61/100
89/89 [==============================] - 18s 199ms/step - loss: 3.2594 - accuracy: 0.2657
Epoch 62/100
89/89 [==============================] - 18s 201ms/step - loss: 3.2934 - accuracy: 0.2627
Epoch 63/100
89/89 [==============================] - 17s 195ms/step - loss: 3.2281 - accuracy: 0.2725
Epoch 64/100
89/89 [==============================] - 17s 194ms/step - loss: 3.1821 - accuracy: 0.2787
Epoch 65/100
89/89 [==============================] - 17s 193ms/step - loss: 3.1514 - accuracy: 0.2828
Epoch 66/100
89/89 [==============================] - 18s 197ms/step - loss: 3.1249 - accuracy: 0.2900
Epoch 67/100
89/89 [==============================] - 21s 235ms/step - loss: 3.0996 - accuracy: 0.2897
Epoch 68/100
89/89 [==============================] - 23s 260ms/step - loss: 3.0731 - accuracy: 0.2966
Epoch 69/100
89/89 [==============================] - 20s 230ms/step - loss: 3.0453 - accuracy: 0.3000
Epoch 70/100
89/89 [==============================] - 19s 212ms/step - loss: 3.0191 - accuracy: 0.3035
Epoch 71/100
89/89 [==============================] - 21s 239ms/step - loss: 3.0113 - accuracy: 0.3059
Epoch 72/100
89/89 [==============================] - 19s 212ms/step - loss: 2.9867 - accuracy: 0.3119
Epoch 73/100
89/89 [==============================] - 19s 217ms/step - loss: 2.9721 - accuracy: 0.3150
Epoch 74/100
89/89 [==============================] - 19s 215ms/step - loss: 2.9364 - accuracy: 0.3151
Epoch 75/100
89/89 [==============================] - 19s 215ms/step - loss: 2.9092 - accuracy: 0.3223
Epoch 76/100
89/89 [==============================] - 19s 212ms/step - loss: 2.8794 - accuracy: 0.3279
Epoch 77/100
89/89 [==============================] - 22s 245ms/step - loss: 2.8554 - accuracy: 0.3324
Epoch 78/100
89/89 [==============================] - 19s 211ms/step - loss: 2.8243 - accuracy: 0.3403
Epoch 79/100
89/89 [==============================] - 18s 207ms/step - loss: 2.8022 - accuracy: 0.3425
Epoch 80/100
89/89 [==============================] - 19s 213ms/step - loss: 2.7815 - accuracy: 0.3491
Epoch 81/100
89/89 [==============================] - 19s 212ms/step - loss: 2.7616 - accuracy: 0.3514
Epoch 82/100
89/89 [==============================] - 20s 227ms/step - loss: 2.7357 - accuracy: 0.3542
Epoch 83/100
89/89 [==============================] - 22s 246ms/step - loss: 2.7153 - accuracy: 0.3560
Epoch 84/100
89/89 [==============================] - 19s 214ms/step - loss: 2.6952 - accuracy: 0.3615
Epoch 85/100
89/89 [==============================] - 19s 219ms/step - loss: 2.6790 - accuracy: 0.3665
Epoch 86/100
89/89 [==============================] - 18s 207ms/step - loss: 2.6547 - accuracy: 0.3705
Epoch 87/100
89/89 [==============================] - 18s 205ms/step - loss: 2.6396 - accuracy: 0.3710
Epoch 88/100
89/89 [==============================] - 19s 212ms/step - loss: 2.6334 - accuracy: 0.3736
Epoch 89/100
89/89 [==============================] - 19s 212ms/step - loss: 2.6175 - accuracy: 0.3774
Epoch 90/100
89/89 [==============================] - 21s 230ms/step - loss: 2.5821 - accuracy: 0.3840
Epoch 91/100
89/89 [==============================] - 19s 209ms/step - loss: 2.5634 - accuracy: 0.3866
Epoch 92/100
89/89 [==============================] - 21s 236ms/step - loss: 2.5403 - accuracy: 0.3917
Epoch 93/100
89/89 [==============================] - 19s 216ms/step - loss: 2.5179 - accuracy: 0.3935
Epoch 94/100
89/89 [==============================] - 18s 207ms/step - loss: 2.4990 - accuracy: 0.4013
Epoch 95/100
89/89 [==============================] - 20s 223ms/step - loss: 2.4983 - accuracy: 0.3993
Epoch 96/100
89/89 [==============================] - 24s 273ms/step - loss: 2.4890 - accuracy: 0.3998
Epoch 97/100
89/89 [==============================] - 20s 220ms/step - loss: 2.4698 - accuracy: 0.4014
Epoch 98/100
89/89 [==============================] - 20s 220ms/step - loss: 2.4488 - accuracy: 0.4043
Epoch 99/100
89/89 [==============================] - 20s 220ms/step - loss: 2.4207 - accuracy: 0.4138
Epoch 100/100
89/89 [==============================] - 20s 225ms/step - loss: 2.4076 - accuracy: 0.4151
```

Out[20]:

```
<tensorflow.python.keras.callbacks.History at 0x21c13c94e20>
```

In [21]:

```python
seed_text=lines[12343]
seed_text
```

Out[21]:

```
'condition seemed serious enough to remind even his father that gregor despite his current sad and revolting form was a family member who could not be treated as an enemy on the contrary as a family there was a duty to swallow any revulsion for him and to be patient just'
```

In [26]:

```python
def generate_text_seq(model, tokenizer, text_seq_length, seed_text, n_words):
    text = []

    for _ in range(n_words):
        encoded = tokenizer.texts_to_sequences([seed_text])[0]
        encoded = pad_sequences([encoded], maxlen = text_seq_length, truncating='pre')

        y_predict = model.predict_classes(encoded)

        predicted_word = ''
        for word, index in tokenizer.word_index.items():
            if index == y_predict:
                predicted_word = word
                break
        seed_text = seed_text + ' ' + predicted_word
        text.append(predicted_word)
    return ' '.join(text)
```

In [27]:

```python
generate_text_seq(model, tokenizer, seq_length, seed_text, 100)
```

Out[27]:

```
'to be patient easy back closer that would threatened that he would have to be heard home the kitchen locked himself hidden as he was not a loss but the room he isnt we cant see it out of the door and listening him and the chief clerk but was still a minor discourtesy and the rooms gentleman in the endless holiday i appear immediate notice with the same conclusion for the time being blossoming as all playing he had poured their door on him and then they answered answered with confirmation of him and were very times to transport'
```

In [ ]:

In [ ]: