In [1]:
```
import tensorflow as tf
import string
import requests
```

In [2]:
```
response=requests.get("http://www.gutenberg.org/cache/epub/5200/pg5200.txt")
```

In [4]:
```
response.text[:1000]
```

Out[4]:
```
'\ufeffThe Project Gutenberg EBook of Metamorphosis, by Franz Kafka\r\nTranslat
ed by David Wyllie.\r\n\r\nThis eBook is for the use of anyone anywhere at no c
ost and with\r\nalmost no restrictions whatsoever.  You may copy it, give it aw
ay 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 COPYRIGHTE
D Project Gutenberg eBook, Details Below **\r\n**     Please follow the copyrig
ht guidelines in this file.     **\r\n\r\n\r\nTitle: Metamorphosis\r\n\r\nAutho
r: 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\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 Wylli
e\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.'
```

In [5]:
```
data = response.text.split('\n')
data[0]
```

Out[5]:
```
'\ufeffThe Project Gutenberg EBook of Metamorphosis, by Franz Kafka\r'
```

In [7]:
```
data = data[300:]
data[0]
```

Out[7]:
```
'movement, drew his foot from the living room, and rushed forward in\r'
```

In [8]:
```
len(data)
```

Out[8]:
```
1810
```

In [9]:
```
data = " ".join(data)
data[:1000]
```

Out[9]:
```
'movement, drew his foot from the living room, and rushed forward in\r a panic.
In the hall, he stretched his right hand far out towards\r the stairway as if o
ut there, there were some supernatural force\r waiting to save him.\r \r Gregor
realised that it was out of the question to let the chief\r clerk go away in th
is mood if his position in the firm was not to be\r put into extreme danger.  T
hat was something his parents did not\r understand very well; over the years, t
hey had become convinced that\r this job would provide for Gregor for his entir
e life, and besides,\r they had so much to worry about at present that they had
lost sight\r of any thought for the future.  Gregor, though, did think about th
e\r future.  The chief clerk had to be held back, calmed down, convinced\r and
finally won over; the future of Gregor and his family depended\r on it! If only
his sister were here! She was clever; she was already\r in tears while Gregor w
as still lying peacefully on his back.  And\r the chief clerk wa'
```

In [10]:
```
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])
```

```
['movement', 'drew', 'his', 'foot', 'from', 'the', 'living', 'room', 'and', 'ru
shed', 'forward', 'in', 'a', 'panic', 'in', 'the', 'hall', 'he', 'stretched',
'his', 'right', 'hand', 'far', 'out', 'towards', 'the', 'stairway', 'as', 'if',
'out', 'there', 'there', 'were', 'some', 'supernatural', 'force', 'waiting', 't
o', 'save', 'him', 'gregor', 'realised', 'that', 'it', 'was', 'out', 'of', 'th
e', 'question', 'to']
```

In [11]:
```
len(tokens)
```

Out[11]:
```
19140
```

In [13]:
```
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))
```

```
19089
```

Build LSTM Model and Prepare X and y

In [14]:
```
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 [15]:
```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(lines)
sequences = tokenizer.texts_to_sequences(lines)
```

In [16]:
```
sequences = np.array(sequences)
X, y = sequences[:, :-1], sequences[:,-1]
X[0]
```

Out[16]:
```
array([ 888,  887,    5,  452,   29,    1,  165,   27,    3,  451,  291,
          7,   12, 1282,    7,    1,  886,    6,  663,    5,  206,  144,
        344,   38,  343,    1, 1281,   14,   32,   38,   42,   42,   58,
         75, 2655,  531,  885,    2,  661,   21,   19,  342,   11,   10,
          8,   38,    4,    1,  447,    2])
```

In [17]:
```
vocab_size = len(tokenizer.word_index) + 1
```

In [18]:
```
y = to_categorical(y, num_classes=vocab_size)
```

In [19]:
```
seq_length = X.shape[1]
seq_length
```

Out[19]:
```
50
```

LSTM Model

New Section

In [20]:
```
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 [21]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 50, 50)            132950
_________________________________________________________________
lstm (LSTM)                  (None, 50, 100)           60400
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               80400
_________________________________________________________________
dense (Dense)                (None, 100)               10100
_________________________________________________________________
dense_1 (Dense)              (None, 2659)              268559
=================================================================
Total params: 552,409
Trainable params: 552,409
Non-trainable params: 0
_________________________________________________________________
```

In [22]:
```
model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['
```

In [24]:
```
model.fit(X, y, batch_size = 256, epochs = 20)
```

```
Epoch 1/20
75/75 [==============================] - 33s 435ms/step - loss: 6.2934 - accu
racy: 0.0530
Epoch 2/20
75/75 [==============================] - 33s 438ms/step - loss: 6.1567 - accu
racy: 0.0534
Epoch 3/20
75/75 [==============================] - 32s 431ms/step - loss: 6.1341 - accu
racy: 0.0534
Epoch 4/20
75/75 [==============================] - 32s 432ms/step - loss: 6.0245 - accu
racy: 0.0534
Epoch 5/20
75/75 [==============================] - 33s 439ms/step - loss: 5.9400 - accu
racy: 0.0534
Epoch 6/20
75/75 [==============================] - 34s 447ms/step - loss: 5.8209 - accu
racy: 0.0587
Epoch 7/20
75/75 [==============================] - 33s 442ms/step - loss: 5.7115 - accu
racy: 0.0657
Epoch 8/20
75/75 [==============================] - 33s 443ms/step - loss: 5.6256 - accu
racy: 0.0685
Epoch 9/20
75/75 [==============================] - 33s 446ms/step - loss: 5.5516 - accu
racy: 0.0723
Epoch 10/20
75/75 [==============================] - 33s 439ms/step - loss: 5.4882 - accu
racy: 0.0767
Epoch 11/20
75/75 [==============================] - 33s 445ms/step - loss: 5.4193 - accu
racy: 0.0800
Epoch 12/20
75/75 [==============================] - 33s 444ms/step - loss: 5.3467 - accu
racy: 0.0861
Epoch 13/20
75/75 [==============================] - 33s 444ms/step - loss: 5.2714 - accu
racy: 0.0920
Epoch 14/20
75/75 [==============================] - 33s 442ms/step - loss: 5.1985 - accu
racy: 0.0933
Epoch 15/20
75/75 [==============================] - 33s 435ms/step - loss: 5.2202 - accu
racy: 0.0945
Epoch 16/20
75/75 [==============================] - 33s 439ms/step - loss: 5.1782 - accu
racy: 0.0950
Epoch 17/20
75/75 [==============================] - 33s 438ms/step - loss: 5.0366 - accu
racy: 0.1026
Epoch 18/20
75/75 [==============================] - 33s 436ms/step - loss: 4.9320 - accu
```

```
racy: 0.1090
Epoch 19/20
75/75 [==============================] - 33s 439ms/step - loss: 4.8597 - accu
racy: 0.1121
Epoch 20/20
75/75 [==============================] - 33s 438ms/step - loss: 4.7953 - accu
racy: 0.1171
```

Out[24]:
```
<tensorflow.python.keras.callbacks.History at 0x7f8aacb69ad0>
```

In [25]:
```
seed_text=lines[12343]
seed_text
```

Out[25]:
```
'disappointed that they had had enough of the whole performance and it was only
now out of politeness that they allowed their peace to be disturbed it was espe
cially unnerving the way they all blew the smoke from their cigarettes upwards
from their mouth and noses yet gregors sister was playing'
```

In [27]:
```
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 [30]:
```
generate_text_seq(model, tokenizer, seq_length, seed_text, 100)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/sequentia
l.py:455: UserWarning: `model.predict_classes()` is deprecated and will be remo
ved after 2021-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-
1)`,   if your model does multi-class classification   (e.g. if it uses a `soft
max` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if
your model does binary classification   (e.g. if it uses a `sigmoid` last-layer
activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

Out[30]:
```
'the
'
```

In [ ]:
```
```