In [1]:
```
# LSTM Project
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM
from keras.optimizers import RMSprop
import numpy as np
import sys
```

In [2]:
```
filename = 'plain.txt'
raw_text = open(filename,'r', encoding='utf-8').read()
raw_text = raw_text.lower()
raw_text[0:1000]
```

Out[2]:
```
'the project gutenberg ebook of the jungle book\n    \nthis ebook is for the us
e of anyone anywhere in the united states and\nmost other parts of the world at
no cost and with almost no restrictions\nwhatsoever. you may copy it, give it a
way or re-use it under the terms\nof the project gutenberg license included wit
h this ebook or online\nat www.gutenberg.org. if you are not located in the uni
ted states,\nyou will have to check the laws of the country where you are locat
ed\nbefore using this ebook.\n\ntitle: the jungle book\n\nauthor: rudyard kipli
ng\n\nrelease date: january 16, 2006 [ebook #236]\n                most recentl
y updated: may 1, 2023\n\nlanguage: english\n\n\n\n*** start of the project gut
enberg ebook the jungle book ***\n\n\n\n\nthe jungle book\n\nby rudyard kipling
\n\n\n\ncontents\n\n     mowgli's brothers\n     hunting-song of the seeonee pa
ck\n     kaa's hunting\n     road-song of the bandar-log\n     "tiger! tige
r!"\n     mowgli's song\n     the white seal\n     lukannon\n     "rikki-tikki
-tavi"\n     darzee'
```

In [3]:
```
# # Text Cleaning
# import re
# corpus = []
# for i in range(len(raw_text)):
#     rp = re.sub('[^a-zA-Z]', " ",raw_text[i])
#     rp = rp.lower()
#     rp = rp.split()
#     rp = " ".join(rp)
#     corpus.append(rp)
```

In [4]:
```
# Remove Numbers
raw_text = ''.join(i for i in raw_text if not i.isdigit())

# Total Character
chars = sorted(list(set(raw_text)))

char_to_int = dict((c,i) for i,c in enumerate(chars))

int_to_char = dict((i,c) for i,c in enumerate(chars))
```

In [5]:
```
n_chars = len(raw_text)
n_vocab = len(chars)
print('Total Character in Text, Corpus Length',n_chars)
print('Total Vocabulary',n_vocab)
```

```
Total Character in Text, Corpus Length 296196
Total Vocabulary 55
```

In [6]:
```
seq_length = 80  # length of each i/p sequence
step = 12 # instead of moving 1 letter at time try skipping few
sentence = [] # Xvalues
```

```
next_chars =[] # Y_Values

for i in range(0,n_chars -seq_length ,step):
    sentence.append(raw_text[i:i+seq_length])  # Sentence in
    next_chars.append(raw_text[i + seq_length]) # Sentence out
n_patterns = len(sentence)
print("Number of sequence ",n_patterns)
```

```
Number of sequence  24677
```

In [7]:
```
# from sklearn.feature_extraction.text import CountVectorizer
# cv = CountVectorizer()
# X = cv.fit_transform(sentence,seq_length).toarray()
# X.shape
```

In [8]:
```
X = np.zeros((len(sentence), seq_length, n_vocab), dtype=np.bool_)
y = np.zeros((len(sentence), n_vocab), dtype=np.bool_)
for i, sentences in enumerate(sentence):
    for j, char in enumerate(sentences):
        X[i,j, char_to_int[char]]=1
    y[i,char_to_int[next_chars[i]]] = 1
print(X.shape)
print(y.shape)
```

```
(24677, 80, 55)
(24677, 55)
```

In [13]:
```
# yaha tumhara seq_length, n_vocab already defined hone chahiye

def create_model(optimizer="rmsprop", learning_rate=0.001, dropout_rate=0.3):
    if optimizer == "adam":
        opt = Adam(learning_rate=learning_rate)
    else:
        opt = RMSprop(learning_rate=learning_rate)

    model = Sequential()
    model.add(LSTM(129, input_shape=(seq_length, n_vocab), return_sequences=True
    model.add(Dropout(dropout_rate))
    model.add(LSTM(130))
    model.add(Dropout(dropout_rate))
    model.add(Dense(n_vocab, activation='softmax'))

    model.compile(loss='categorical_crossentropy', optimizer=opt)
    return model
```

In [ ]:
```
# # define the checkpoint
# # !pip install scikeras
# !pip install --upgrade "scikit-learn>=1.4.1post1" scikeras



from keras.callbacks import ModelCheckpoint

from scikeras.wrappers import KerasClassifier

from sklearn.model_selection import GridSearchCV
from tensorflow.keras.optimizers import Adam, RMSprop


filepath="saved_weights/saved_weights-{epoch:02d}-{loss:.4f}.keras"
```

```
checkpoint = ModelCheckpoint(filepath,monitor='loss', verbose=1, save_best_only=

callbacks_list = [checkpoint]

# 2. Estimator for GridSearchCV
estimator = KerasClassifier(model=create_model, verbose=0)

# 3. Improved param_grid
param_grid = {
    'batch_size': [32, 64, 128,150],
    'epochs': [20, 50,150],
    'optimizer': ['adam', 'rmsprop']
}

grid = GridSearchCV(estimator=estimator,
                    param_grid=param_grid,
                    n_jobs=-1,
                    cv=5)

# 4. Fit with checkpoint callback
history = grid.fit(X, y, callbacks=callbacks_list)

print("Best params:", grid.best_params_)
print("Best score :", grid.best_score_)
# Fit the model

# history = model.fit(X, y,
#           batch_size=128,
#           epochs=100,
#           callbacks=callbacks_list)


best_model = grid.best_estimator_.model_
best_model.save("my_saved_weights_jungle_book_best.h5")
```

In [ ]:
```
from matplotlib import pyplot as plt
#plot the training and validation accuracy and loss at each epoch
loss = history.history['loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.title('Training loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

In [ ]:
```
def sample(preds):
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds)
    exp_preds = np.exp(preds) #exp of log (x), isn't this same as x??
    preds = exp_preds / np.sum(exp_preds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)
```

In [ ]:
```
#Prediction
# load the network weights
filename = "my_saved_weights_jungle_book_50epochs.h5"
model.load_weights(filename)
```

```
#Pick a random sentence from the text as seed.
start_index = random.randint(0, n_chars - seq_length - 1)

#Initiate generated text and keep adding new predictions and print them out
generated = ''
sentence = raw_text[start_index: start_index + seq_length]
generated += sentence

print('----- Seed for our text prediction: "' + sentence + '"')
#sys.stdout.write(generated)


for i in range(400):   # Number of characters including spaces
    x_pred = np.zeros((1, seq_length, n_vocab))
    for t, char in enumerate(sentence):
        x_pred[0, t, char_to_int[char]] = 1.

    preds = model.predict(x_pred, verbose=0)[0]
    next_index = sample(preds)
    next_char = int_to_char[next_index]

    generated += next_char
    sentence = sentence[1:] + next_char

    sys.stdout.write(next_char)
    sys.stdout.flush()
print()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: