Using section 8.1 in Deep Learning with Python as a guide, implement an LSTM text generator. Train the model on the Enron corpus or a text source of your choice. Save the model and generate 20 examples to the results directory of dsc650/assignments/assignment11/.

In [ ]:
```
from tensorflow import keras
import numpy as np
path = keras.utils.get_file(
 'nietzsche.txt',
 origin='https://s3.amazonaws.com/text-datasets/nietzsche.txt')
text = open(path).read().lower()
print('Corpus length:', len(text))
```

```
Corpus length: 600893
```

In [ ]:
```
# Length of extracted character sequences
maxlen = 60
# We sample a new sequence every `step` characters
step = 3
# This holds our extracted sequences
sentences = []
# This holds the targets (the follow-up characters)
next_chars = []
for i in range(0, len(text) - maxlen, step):
     sentences.append(text[i: i + maxlen])
     next_chars.append(text[i + maxlen])
print('Number of sequences:', len(sentences))
# List of unique characters in the corpus
chars = sorted(list(set(text)))
print('Unique characters:', len(chars))
# Dictionary mapping unique characters to their index in `chars`
char_indices = dict((char, chars.index(char)) for char in chars)
# Next, one-hot encode the characters into binary arrays.
print('Vectorization...')
x = np.zeros((len(sentences), maxlen, len(chars)), dtype=np.bool)
y = np.zeros((len(sentences), len(chars)), dtype=np.bool)
for i, sentence in enumerate(sentences):
     for t, char in enumerate(sentence):
          x[i, t, char_indices[char]] = 1
          y[i, char_indices[next_chars[i]]] = 1
```

```
Number of sequences: 200278
Unique characters: 57
Vectorization...
```

In [ ]:
```
model = keras.models.Sequential()
model.add(keras.layers.LSTM(128, input_shape=(maxlen, len(chars))))
model.add(keras.layers.Dense(len(chars), activation='softmax'))
```

In [ ]:
```
optimizer = keras.optimizers.RMSprop(lr=0.01)
model.compile(loss='categorical_crossentropy', optimizer=optimizer)
```

In [ ]:
```
def sample(preds, temperature=1.0):
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds) / temperature
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)
```

In [ ]:
```
import random
import sys
for epoch in range(1, 40):
    print('epoch', epoch)
    # Fit the model for 1 epoch on the available training data
    model.fit(x, y,
              batch_size=128,
              epochs=1)
    # Select a text seed at random
```

```
epoch 1
1565/1565 [==============================] - 138s 87ms/step - loss: 2.2
620
epoch 2
1565/1565 [==============================] - 135s 86ms/step - loss: 1.6
160
epoch 3
1565/1565 [==============================] - 133s 85ms/step - loss: 1.5
273
epoch 4
1565/1565 [==============================] - 127s 81ms/step - loss: 1.4
818
epoch 5
1565/1565 [==============================] - 130s 83ms/step - loss: 1.4
518
epoch 6
1565/1565 [==============================] - 129s 82ms/step - loss: 1.4
301
epoch 7
1565/1565 [==============================] - 130s 83ms/step - loss: 1.4
128
epoch 8
1565/1565 [==============================] - 129s 82ms/step - loss: 1.3
981
epoch 9
1565/1565 [==============================] - 129s 83ms/step - loss: 1.3
858
epoch 10
1565/1565 [==============================] - 131s 84ms/step - loss: 1.3
754
epoch 11
1565/1565 [==============================] - 129s 83ms/step - loss: 1.3
658
epoch 12
1565/1565 [==============================] - 130s 83ms/step - loss: 1.3
572
epoch 13
1565/1565 [==============================] - 133s 85ms/step - loss: 1.3
497
epoch 14
1565/1565 [==============================] - 130s 83ms/step - loss: 1.3
430
epoch 15
1565/1565 [==============================] - 131s 84ms/step - loss: 1.3
361
epoch 16
1565/1565 [==============================] - 131s 84ms/step - loss: 1.3
314
epoch 17
1565/1565 [==============================] - 129s 82ms/step - loss: 1.3
272
epoch 18
1565/1565 [==============================] - 133s 85ms/step - loss: 1.3
194
epoch 19
1565/1565 [==============================] - 130s 83ms/step - loss: 1.3
153
```

```
epoch 20
1565/1565 [==============================] - 129s 82ms/step - loss: 1.3
111
epoch 21
1565/1565 [==============================] - 129s 82ms/step - loss: 1.3
084
epoch 22
1565/1565 [==============================] - 132s 84ms/step - loss: 1.3
030
epoch 23
1565/1565 [==============================] - 132s 84ms/step - loss: 1.3
013
epoch 24
1565/1565 [==============================] - 131s 84ms/step - loss: 1.2
956
epoch 25
1565/1565 [==============================] - 132s 84ms/step - loss: 1.2
933
epoch 26
1565/1565 [==============================] - 133s 85ms/step - loss: 1.2
905
epoch 27
1565/1565 [==============================] - 133s 85ms/step - loss: 1.2
876
epoch 28
1565/1565 [==============================] - 258s 165ms/step - loss: 1.
2858
epoch 29
1565/1565 [==============================] - 252s 161ms/step - loss: 1.
2818
epoch 30
1565/1565 [==============================] - 247s 158ms/step - loss: 1.
2804
epoch 31
1565/1565 [==============================] - 246s 157ms/step - loss: 1.
2792
epoch 32
1565/1565 [==============================] - 264s 169ms/step - loss: 1.
2760
epoch 33
1565/1565 [==============================] - 234s 150ms/step - loss: 1.
2731
epoch 34
1565/1565 [==============================] - 264s 168ms/step - loss: 1.
2720
epoch 35
1565/1565 [==============================] - 256s 164ms/step - loss: 1.
2688
epoch 36
1565/1565 [==============================] - 254s 162ms/step - loss: 1.
2658
epoch 37
1565/1565 [==============================] - 157s 100ms/step - loss: 1.
2641
epoch 38
1565/1565 [==============================] - 135s 86ms/step - loss: 1.2
621
```

```
epoch 39
1565/1565 [==============================] - 131s 84ms/step - loss: 1.2
596
```

In [ ]:
```
model.save("/Users/muduo/Documents/GitHub/dsc650/dsc650/assignments/assi
gnment11/LSTMtextgenmodel")
```

```
2021-11-19 23:33:08.319174: W tensorflow/python/util/util.cc:348] Sets
are not currently considered sequences, but this may change in the futu
re, so consider avoiding using them.
WARNING:absl:Found untraced functions such as lstm_cell_1_layer_call_an
d_return_conditional_losses, lstm_cell_1_layer_call_fn, lstm_cell_1_lay
er_call_fn, lstm_cell_1_layer_call_and_return_conditional_losses, lstm_
cell_1_layer_call_and_return_conditional_losses while saving (showing 5
of 5). These functions will not be directly callable after loading.
WARNING:absl:Found untraced functions such as lstm_cell_1_layer_call_an
d_return_conditional_losses, lstm_cell_1_layer_call_fn, lstm_cell_1_lay
er_call_fn, lstm_cell_1_layer_call_and_return_conditional_losses, lstm_
cell_1_layer_call_and_return_conditional_losses while saving (showing 5
of 5). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: /Users/muduo/Documents/GitHub/dsc65
0/dsc650/assignments/assignment11/LSTMtextgenmodel/assets

INFO:tensorflow:Assets written to: /Users/muduo/Documents/GitHub/dsc65
0/dsc650/assignments/assignment11/LSTMtextgenmodel/assets
```

In [ ]:
```
from tensorflow import keras
load = keras.models.load_model("/Users/muduo/Documents/GitHub/dsc650/dsc
650/assignments/assignment11/LSTMtextgenmodel")
```

In [ ]:
```
import random
import sys
for n in range(1, 21):
    # Select a text seed at random
    start_index = random.randint(0, len(text) - maxlen - 1)
    generated_text = text[start_index: start_index + maxlen]
    seed_text = generated_text
    with open("results/"+str(n)+".txt", "a") as a:
        a.write(f"--- Generating with seed: {seed_text}\n")
        for temperature in [0.5, 1.2]:
            final_text = seed_text
            # We generate 200 characters
            for i in range(200):
                sampled = np.zeros((1, maxlen, len(chars)))
                for t, char in enumerate(generated_text):
                    sampled[0, t, char_indices[char]] = 1.
                preds = load.predict(sampled, verbose=0)[0]
                next_index = sample(preds, temperature)
                next_char = chars[next_index]
                generated_text += next_char
                final_text += next_char
                generated_text = generated_text[1:]
            a.write(f"Temperature: {temperature} \n")
            a.write(final_text)
            a.write("\n"*2)
```

```
/var/folders/1y/5m3skjbn33ggq1vw0_jc_wcc0000gn/T/ipykernel_43156/168472
4676.py:3: RuntimeWarning: divide by zero encountered in log
  preds = np.log(preds) / temperature
```

In [ ]: