In [1]:
```
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))
```

```
Downloading data from https://s3.amazonaws.com/text-datasets/nietzsche.txt (https://s3.amazonaws.com/text-datasets/nietzsche.txt)
600901/600901 [==============================] - 0s 1us/step
Corpus length: 600901
```

In [3]:
```
maxlen = 60
step = 3
sentences = []
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))
chars = sorted(list(set(text)))
print('Unique characters:', len(chars))
char_indices = dict((char, chars.index(char)) for char in chars)
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: 200281
Unique characters: 59
Vectorization...

C:\Users\Kate\AppData\Local\Temp/ipykernel_11760/321955488.py:13: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To silence this warning, use `bool` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations)
  x = np.zeros((len(sentences), maxlen, len(chars)), dtype=np.bool)
C:\Users\Kate\AppData\Local\Temp/ipykernel_11760/321955488.py:14: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To silence this warning, use `bool` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations)
  y = np.zeros((len(sentences), len(chars)), dtype=np.bool)
```

In [4]:
```
from keras import layers
model = keras.models.Sequential()
model.add(layers.LSTM(128, input_shape=(maxlen, len(chars))))
model.add(layers.Dense(len(chars), activation='softmax'))
```

In [5]:
```
optimizer = keras.optimizers.RMSprop(lr=0.01)
model.compile(loss='categorical_crossentropy', optimizer=optimizer)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\keras\optimizers\optimizer_v2\rmsprop.py:135: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(RMSprop, self).__init__(name, **kwargs)
```

In [6]:
```
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 [7]:
```
import random
import sys
for epoch in range(1, 60):
    print('epoch', epoch)
    model.fit(x, y, batch_size=128, epochs=1)
    start_index = random.randint(0, len(text) - maxlen - 1)
    generated_text = text[start_index: start_index + maxlen]
    print('--- Generating with seed: "' + generated_text + '"')
    for temperature in [0.2, 0.5, 1.0, 1.2]:
        print('------ temperature:', temperature)
        sys.stdout.write(generated_text)
    for i in range(400):
        sampled = np.zeros((1, maxlen, len(chars)))
        for t, char in enumerate(generated_text):
            sampled[0, t, char_indices[char]] = 1.
        preds = model.predict(sampled, verbose=0)[0]
        next_index = sample(preds, temperature)
        next_char = chars[next_index]
        generated_text += next_char
        generated_text = generated_text[1:]
        sys.stdout.write(next_char)
```

```
--> 947       return self._stateless_fn(*args, **kwds)  # pylint: disable=not-callable
    948     elif self._stateful_fn is not None:
    949       # Release the lock early so that multiple threads can perform the call

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py in __call__(self, *args, **kwargs)
   2451       (graph_function,
   2452        filtered_flat_args) = self._maybe_define_function(args, kwargs)
-> 2453     return graph_function._call_flat(
   2454         filtered_flat_args, captured_inputs=graph_function.captured_inputs)  # pylint: disable=protected-access
   2455

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py in _call_flat(self, args, captured_inputs, can
cellation_manager)
   1858         and executing_eagerly):
   1859       # No tape is watching; skip to running the function.
-> 1860       return self._build_call_outputs(self._inference_function.call(
   1861           ctx, args, cancellation_manager=cancellation_manager))
   1862     forward_backward = self._select_forward_and_backward_functions(

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\eager\function.py in call(self, ctx, args, cancellation_manager)
```

In [ ]:

In [ ]: