William_Wordsworth_and_Robert_Frost

April 28, 2025

1 use LSTM neural networks (Long-Short-Term Memory)

1.1 in order to tech our computer to write Poems like William Wordsworth and Robert Frost

[ ]:
```
# Importing necessery packages
import random
import numpy as np
import tensorflow as tf
```

```
2025-04-28 02:05:22.644251: I tensorflow/core/util/port.cc:153] oneDNN custom
operations are on. You may see slightly different numerical results due to
floating-point round-off errors from different computation orders. To turn them
off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-04-28 02:05:22.683546: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:467] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
E0000 00:00:1745786122.715715   13795 cuda_dnn.cc:8579] Unable to register cuDNN
factory: Attempting to register factory for plugin cuDNN when one has already
been registered
E0000 00:00:1745786122.725460   13795 cuda_blas.cc:1407] Unable to register
cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has
already been registered
W0000 00:00:1745786122.748125   13795 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1745786122.748158   13795 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1745786122.748161   13795 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1745786122.748163   13795 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
2025-04-28 02:05:22.755807: I tensorflow/core/platform/cpu_feature_guard.cc:210]
```

```
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other
operations, rebuild TensorFlow with the appropriate compiler flags.
```

Data Source : https://www.kaggle.com/datasets/charunisa/english-poems-dataset?resource=download

[ ]:
```
# Loading the text file
filepath = '/home/abhisek/Project/poems.txt'
text = open(filepath, 'rb').read().decode(encoding='utf-8')
```

[4]:
```
text = open(filepath, 'rb').read().decode(encoding='utf-8').lower()
```

[5]:
```
print(f"Total number of characters in the text: {len(text)}")
```

```
Total number of characters in the text: 75752
```

[ ]:
```
# SEQ_LENGTH = how many charecters will be used to predict the next character
SEQ_LENGTH = 40

# STEP_SIZE = how many characters we want to shift to next sequence
STEP_SIZE = 3


# Creating empty list of sentences and next characters
sentences = []
next_char = []
```

[ ]:
```
# We iterate through the whole text and gather all sentences and their next␣
 ↪character.
# This is the training data for our neural network.
# Now we just need to convert it into a numerical format.

for i in range(0, len(text) - SEQ_LENGTH, STEP_SIZE):
    sentences.append(text[i: i + SEQ_LENGTH])
    next_char.append(text[i + SEQ_LENGTH])
```

[ ]:
```
# sorting the characters
characters = sorted(set(text))
```

[ ]:
```
# creating two dictionaries from characters to index and from index to␣
 ↪characters
char_to_index = dict((c, i) for i, c in enumerate(characters))
index_to_char= dict((i, c) for i, c in enumerate(characters))
```

[13]:
```
x = np.zeros((len(sentences), SEQ_LENGTH,
              len(characters)), dtype= bool)
y = np.zeros((len(sentences),
```

```
            len(characters)), dtype= bool)

for i, satz in enumerate(sentences):
    for t, char in enumerate(satz):
        x[i, t, char_to_index[char]] = 1
    y[i, char_to_index[next_char[i]]] = 1
```

1.2 Building Recurrent Neural Network

[ ]:
```
# Importing necessery packages
import random
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.layers import Activation, Dense, LSTM
```

We will use Sequential for our model, Activation, Dense and LSTM for our layers and RMSprop for optimization during the compilation of our model.

[ ]:
```
model = Sequential()
model.add(LSTM(128,
               input_shape=(SEQ_LENGTH,
                            len(characters)))) # The inputs immediately flow␣
 ↪into our LSTM layer with 128 neurons
# Our input shape is the length of a sentence times the amount of characters.
model.add(Dense(len(characters))) #This layer is followed by a Dense hidden␣
 ↪layer, which just increases complexity
model.add(Activation('softmax')) # In the end we use the Softmax activation␣
 ↪function in order to make our results add up to one. This gives us the␣
 ↪probability for each character.
```

```
2025-04-28 11:41:53.487656: E
external/local_xla/xla/stream_executor/cuda/cuda_platform.cc:51] failed call to
cuInit: INTERNAL: CUDA error: Failed call to cuInit: UNKNOWN ERROR (303)
/home/abhisek/anaconda3/lib/python3.12/site-
packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

We'll now compile and train the model for four epochs using a batch size of 256, meaning the model will iterate through the entire training data four times.

[ ]:
```
model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(learning_rate=0.01))

model.fit(x, y, batch_size=256, epochs=4)
```

```
model.save('/home/abhisek/Project/poemgenerator.keras')
```

```
Epoch 1/4
99/99             15s 135ms/step -
loss: 1.5171
Epoch 2/4
99/99             14s 143ms/step -
loss: 1.3949
Epoch 3/4
99/99             14s 144ms/step -
loss: 1.3069
Epoch 4/4
99/99             15s 156ms/step -
loss: 1.2125
```

[ ]:
```
model = tf.keras.models.load_model('poemgenerator.keras')
```

[ ]:
```
# Additional functions to make our script generate some reasonable text
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)
```

This function samples a character from the prediction output based on a 'temperature' parameter. Higher temperatures lead to more random (less likely) character choices, while lower temperatures result in more predictable (more likely) selections.

Generating Text

[22]:
```
def generate_text(length, temperature):
    start_index = random.randint(0, len(text) - SEQ_LENGTH - 1)
    generated = ''
    sentence = text[start_index: start_index + SEQ_LENGTH]
    generated += sentence
    for i in range(length):
        x_predictions = np.zeros((1, SEQ_LENGTH, len(characters)))
        for t, char in enumerate(sentence):
            x_predictions[0, t, char_to_index[char]] = 1

        predictions = model.predict(x_predictions, verbose=0)[0]
        next_index = sample(predictions,
                            temperature)
        next_character = index_to_char[next_index]

        generated += next_character
```

```
        sentence = sentence[1:] + next_character
    return generated
```

[ ]:
```
# Output
print(generate_text(300, 0.2))
print(generate_text(300, 0.4))
print(generate_text(300, 0.5))
print(generate_text(300, 0.6))
print(generate_text(300, 0.7))
print(generate_text(300, 0.8))
```

```
and washing dishes after them-from doing his wild reart.

i have be garden you the light,
and string with the starling fan he stop
to seen and blowers in the stars,
and the moundas strengm:
there sand of the manthis would be one off seen and beart,
where the starling fan the starniss,
of ling, and stopted from the starf
suchord
w more fair:
dull would he be of soul with the flack.
and stan in its go dound be been withone.

i have be in provised on the starl
to so mene far in on the light be ond my hervenow.
i gone of the births some bake when it shouse of the star all oft,
and strown both a more to the grien sone.
of the given at reast of all our sead.

after sunset, sir,
when it is light and beary there,
and it i sook to see and blowers.
i dound at like a stop and make on the star
the warse and bodk to busent meantwryes,
but we bight to should be then are them some.

there strees ingred to be bears to sun
the hearth spores of stownt
when they caunt of stantime of a mistorar.
h
o could pass by
a sight so touching in birts
and it a gust seem nor to sun mear.

you gain-dook howle, and boid is than sull be bed
```

```
our her is gow a wind them once to busher arape.

whowe they is its ground on them.
what is mome of come in ming.

when we doon to the mints of the blow.

i fen with the may so must fan one where
oving soul.
--and often, trifling with the ellace of them of main daghn stowe
but it seld on thy off of the liel of them owner ofrisen.

i hould thes sead with the fon one the there's deate me.

when there singal, and some rigged.
i dound enough to be chanted they sweet
somets not borad bood. the earth-shereno.
ind strings in the
in that there
he sat and waited till he will,
be loving come all you he had wild then sing
the winds to sumpre in the light.
look there, was ho bush ampleep and flowers,
one headen fight
so feell, is its a rand
as in our bees, if fash and so we roors,
nor the fornts, of flalling aten of fond
ow hervedanood,
and i sould the dilla
```

[27]:
```python
# First, generate all outputs
outputs = []
outputs.append(generate_text(300, 0.2))
outputs.append(generate_text(300, 0.4))
outputs.append(generate_text(300, 0.5))
outputs.append(generate_text(300, 0.6))
outputs.append(generate_text(300, 0.7))
outputs.append(generate_text(300, 0.8))

# Now, save them into a text file
output_path = '/home/abhisek/Project/output.txt'
with open(output_path, 'w', encoding='utf-8') as f:
    for i, text in enumerate(outputs):
        f.write(f"Output for temperature {0.2 + 0.2 * i}:\n")
        f.write(text)
        f.write("\n\n" + "-"*50 + "\n\n")  # separator between outputs
```