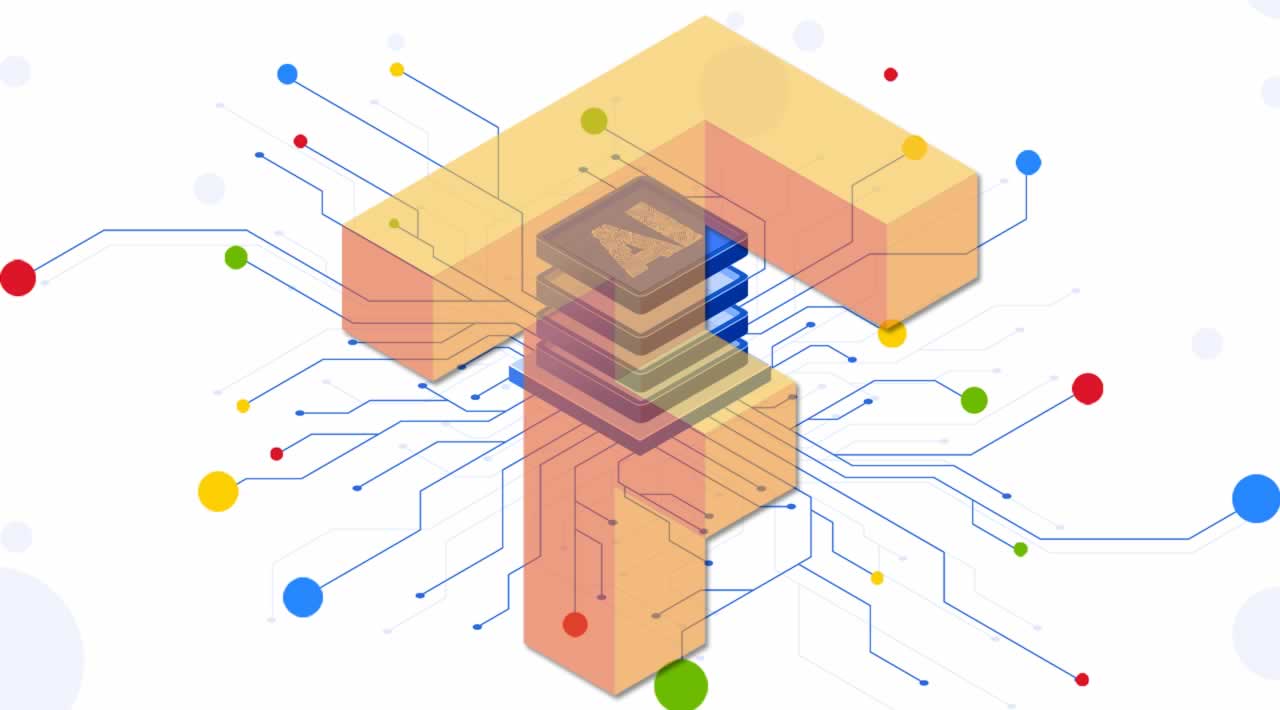
ENGLISH POEM GENERATOR

* A REPORT

For Value Added Course from 2nd to 7th December 2019



**Narayanan SL**

III CSE A

Mepco Schlenk Engineering College (Autonomous), Sivakasi.

# INTRODUCTION

The project titled English Poem Generator focuses on generating English poems when given with a starting sequence. It was trained from a dataset consisting of English poems. This was accomplished using machine learning framework called tensorflow.

# DATASET

The dataset was obtained using a random collection of English poems neatly organised into a text document with each poem being separated by a new line and with proper indentations and punctuation using in poetic constructs. The dataset consists of 18,120 words which can be split into 101,071 recognizable individual characters. The poems in the dataset consisted of words from authors such as Shakespeare, Elizabeth Barett Browning, Owens, John Milton, Rudyard Kipling, Robert Burns to name a few.

# ARCHITECTURE

The model generated for this task consists of the following layers.

Model Type: Sequential with 1024 RNN units and 512 embedding dimensions.

Layer 1: Embedding (input) - size of the total distinct words in the dataset with embedding dimensions.

Layer 2: GRU with 1024 gate units with initialiser glorot uniform.

Layer 3: GRU with 1024 gate units with initialiser glorot uniform.

Layer 4: GRU with 512 gate units with initialiser he uniform.

Layer 5: Dense layer with size of the total distinct words in the dataset.

The above architecture is trained with a batch size of 64 and the training loss is found to be 0.0075 for the best epoch using the sparse categorical cross entropy function.

# 

# SAMPLE GENERATIONS

**1)**

Death, My childhood's idol is waiting for me.

For none returns from those quiet shores,

Who confessed her when she died.

The setting is all of rubies red,

And pearls which a Peri might have kept.

For each ruby there my heart hath bled:

For each pearl my eyes have wept.

So, the truth of the skies,

Since my Love died for my daigle's peace to pray;

For, are the glory of a narrower heart than mine!

Yet it shall be: thou shalt thou see me sair to see thee weipe.

**2)**

Journey there alone. All the care!

I remember well, too well, dear bread,

Of bones your tables gave me store,

Your missed him worth gives thee!

A dream, the banner, and the field, Glory and Greece about us see;

The Spartan borne up in me and a thousand griefs, shall wake,

And pine at them beyond tomorrow.

Away! away! for I will fly to thee, Not charioted by Bacchus and cold.

**3)**

Autumn friends, Now feels in some own I wandered for the sound of strain

The gold me gone the color, Day after where I am there!

O, tell me before the day I die.

The building round I hear,

And the roaring of the whole of its fantasies,

And brings life near in utter despair,

The beck grows wider, the hands down to rest,

Who forbids our complaint.

Morning from the same flowers in the rest.

# CODE

from \_\_future\_\_ import absolute\_import, division, print\_function, unicode\_literals

import tensorflow as tf

tf.compat.v1.enable\_eager\_execution()

import numpy as np

import os

import time

from google.colab import drive

drive.mount('/content/drive')

path\_to\_file = 'drive/My Drive/text\_generation/pg16786dataset.txt'

text = open(path\_to\_file, 'rb').read().decode(encoding='utf-8')

vocab = sorted(set(text))

# Creating a mapping from unique characters to indices

char2idx = {u:i for i, u in enumerate(vocab)}

idx2char = np.array(vocab)

text\_as\_int = np.array([char2idx[c] for c in text])

# The maximum length sentence we want for a single input in characters

seq\_length = 100

examples\_per\_epoch = len(text)//(seq\_length+1)

# Create training examples / targets

char\_dataset = tf.data.Dataset.from\_tensor\_slices(text\_as\_int)

sequences = char\_dataset.batch(seq\_length+1, drop\_remainder=True)

def split\_input\_target(chunk):

input\_text = chunk[:-1]

target\_text = chunk[1:]

return input\_text, target\_text

dataset = sequences.map(split\_input\_target)

BATCH\_SIZE = 64

BUFFER\_SIZE = 10000

dataset = dataset.shuffle(BUFFER\_SIZE).batch(BATCH\_SIZE, drop\_remainder=True)

vocab\_size = len(vocab)

embedding\_dim = 512

rnn\_units = 1024

def build\_model(vocab\_size, embedding\_dim, rnn\_units, batch\_size):

model = tf.keras.Sequential([

tf.keras.layers.Embedding(vocab\_size, embedding\_dim,

batch\_input\_shape=[batch\_size, None]),

tf.keras.layers.GRU(1024,return\_sequences=True,

stateful=True,

recurrent\_initializer='glorot\_uniform'),

tf.keras.layers.GRU(1024,return\_sequences=True,

stateful=True,

recurrent\_initializer='glorot\_uniform'),

tf.keras.layers.GRU(512,return\_sequences=True,

stateful=True,

recurrent\_initializer='he\_uniform'),

tf.keras.layers.Dense(vocab\_size)

])

return model

model = build\_model(

vocab\_size = len(vocab),

embedding\_dim=embedding\_dim,

rnn\_units=rnn\_units,

batch\_size=BATCH\_SIZE)

def loss(labels, logits):

return tf.keras.losses.sparse\_categorical\_crossentropy(labels, logits, from\_logits=True)

model.compile(optimizer='adam', loss=loss)

checkpoint\_dir = './drive/My Drive/text\_generation\_1/training\_checkpoints'

checkpoint\_prefix = os.path.join(checkpoint\_dir, "ckpt\_{epoch}")

checkpoint\_callback=tf.keras.callbacks.ModelCheckpoint(

filepath=checkpoint\_prefix,

save\_weights\_only=True)

EPOCHS=50

history = model.fit(dataset, epochs=EPOCHS, callbacks=[checkpoint\_callback])

tf.train.latest\_checkpoint(checkpoint\_dir)

model = build\_model(vocab\_size, embedding\_dim, rnn\_units, batch\_size=1)

model.load\_weights(tf.train.latest\_checkpoint(checkpoint\_dir))

model.build(tf.TensorShape([1, None]))

def generate\_text(model, start\_string,num\_generate,temperature):

input\_eval = [char2idx[s] for s in start\_string]

input\_eval = tf.expand\_dims(input\_eval, 0)

text\_generated = []

model.reset\_states()

for i in range(num\_generate):

predictions = model(input\_eval)

predictions = tf.squeeze(predictions, 0)

predictions = predictions / temperature

predicted\_id = tf.random.categorical(predictions, num\_samples=1)[-1,0].numpy()

input\_eval = tf.expand\_dims([predicted\_id], 0)

text\_generated.append(idx2char[predicted\_id])

return (start\_string + ''.join(text\_generated))

generation=generate\_text(model, start\_string=u"Autumn",num\_generate=500,temperature=0.1)

last\_stop\_index=0

for i in range(len(generation)-1,0,-1):

if(generation[i]=='.' or generation[i]=='?'):

last\_stop\_index=i+1

break

if(i==0):

last\_stop\_index=len(generation)-1

poem=generation[0:last\_stop\_index]

print(poem)

# CONCLUSION

Thus the English poem generator was constructed using tensorflow and it achieved a reasonable output when analysed by human readers in terms of relevancy and meaning.

# REFERENCES

1. https://www.tensorflow.org/tutorials/text/text\_generation - Text Generation using Tensorflow.