CS5542 Big Data Apps and Analytics

In Class Programming –6 Report
(Jongkook Son)

Project Overview:

Use a different data and use the model provided in ICP6 to perform Text generation. You must make 4 changes (for example adding more layers to model, changing hyperparameters etc) in the source code. Report your findings in detail.
Note: please indicate in your reports which 4 changes you made in the source code and why in your opinion these changes are logical.

Requirements/Task(s):

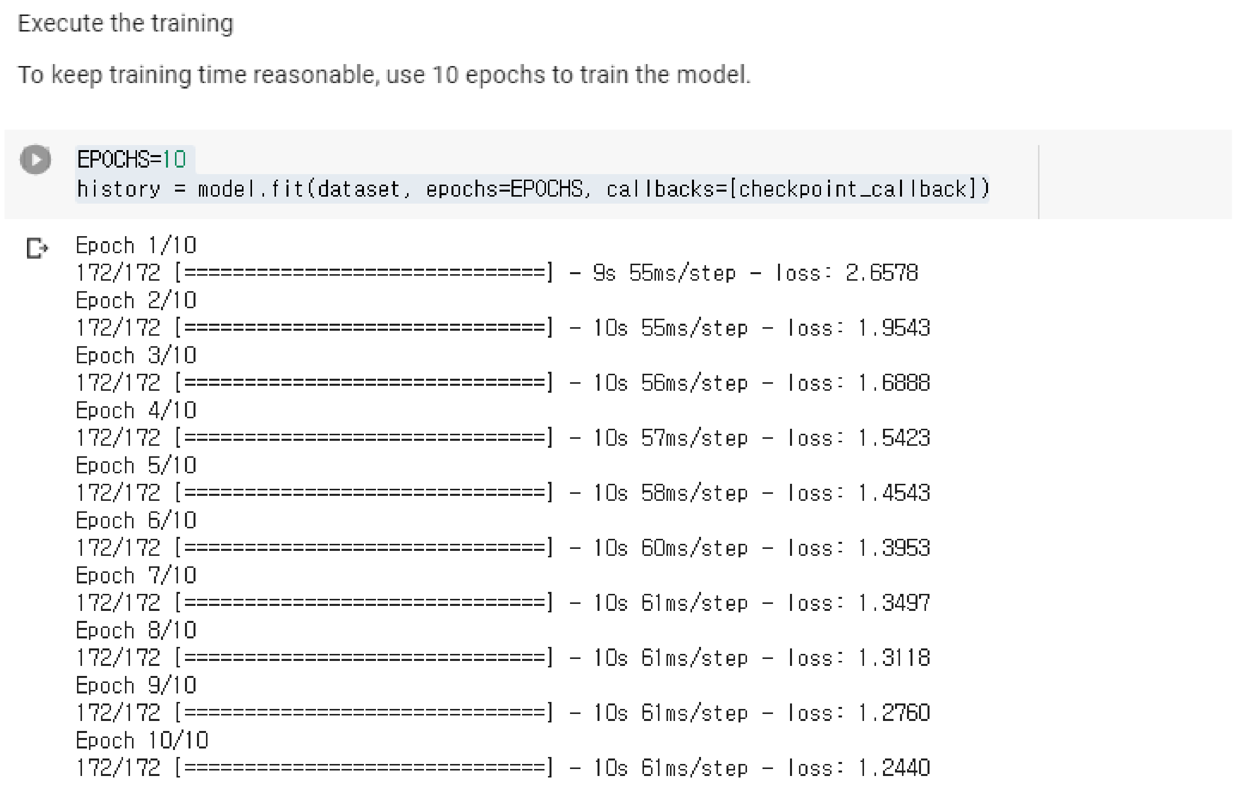
1) Successfully executing the code and making 4 changes in the model (75 points)
2) Using a new and good dataset (5 points)
3) Providing the logical explanation of the changes that you made to model and over all code quality (10 points)
4) Pdf Report quality, video explanation (10 points)

What I learned in ICP:

Overall, I could have learned The basic structure of CNN and how it is applied to generate text by doing this ICP. Text processing is a little different from previous model. We created a mapping from unique characters to indices and assign the maximum length sentence we want for a single input in characters. In the lecture model was built on GRU layer. But I changed it to LSTM in my model. By doing this, I could get some insight about the common and the difference between two models. Finally I learned I should configure the checkpoints and make it as a callback to work and restore this checkpoint in this CNN model.

ICP description what was the task you were performing and Screen shots that shows the successful execution of each required step of your code

The model's loss and result before change

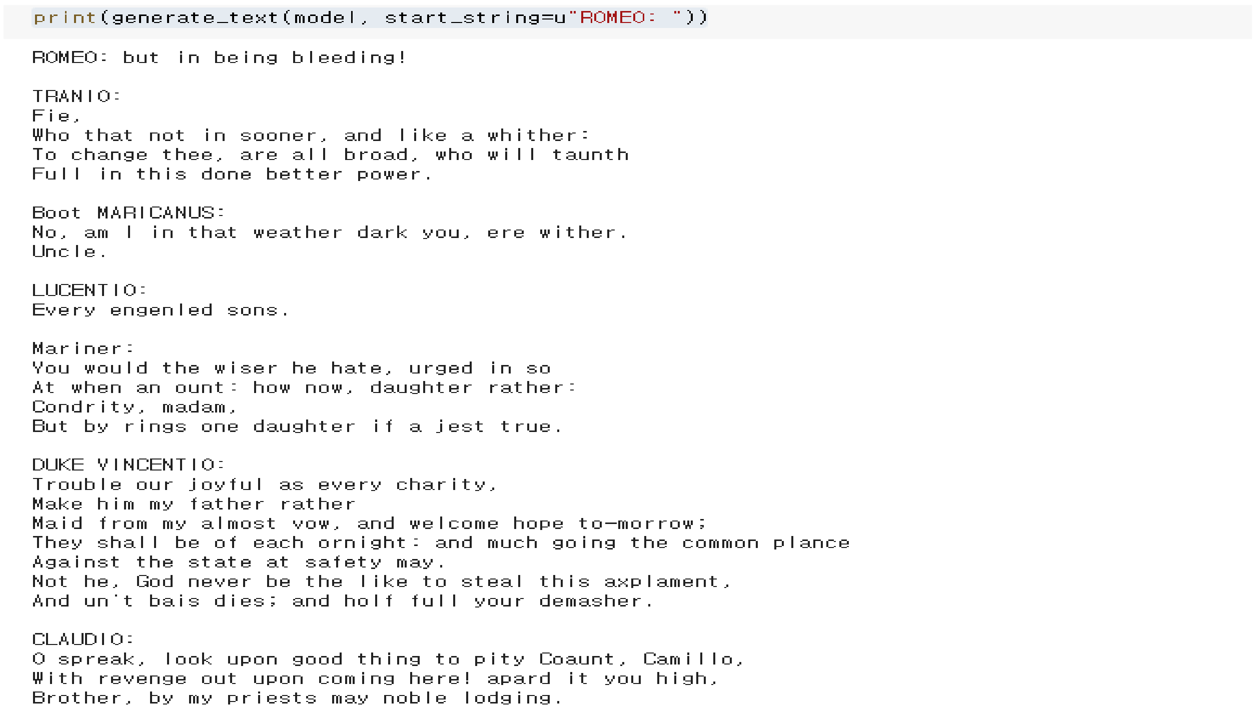


Execute the training

To keep training time reasonable, use 10 epochs to train the model.

```
EPOCHS=10
history = model.fit(dataset, epochs=EPOCHS, callbacks=[checkpoint_callback])
```

```
Epoch 1/10
172/172 [==============================] - 9s 55ms/step - loss: 2.6578
Epoch 2/10
172/172 [==============================] - 10s 55ms/step - loss: 1.9543
Epoch 3/10
172/172 [==============================] - 10s 56ms/step - loss: 1.6888
Epoch 4/10
172/172 [==============================] - 10s 57ms/step - loss: 1.5423
Epoch 5/10
172/172 [==============================] - 10s 58ms/step - loss: 1.4543
Epoch 6/10
172/172 [==============================] - 10s 60ms/step - loss: 1.3953
Epoch 7/10
172/172 [==============================] - 10s 61ms/step - loss: 1.3497
Epoch 8/10
172/172 [==============================] - 10s 61ms/step - loss: 1.3118
Epoch 9/10
172/172 [==============================] - 10s 61ms/step - loss: 1.2760
Epoch 10/10
172/172 [==============================] - 10s 61ms/step - loss: 1.2440
```



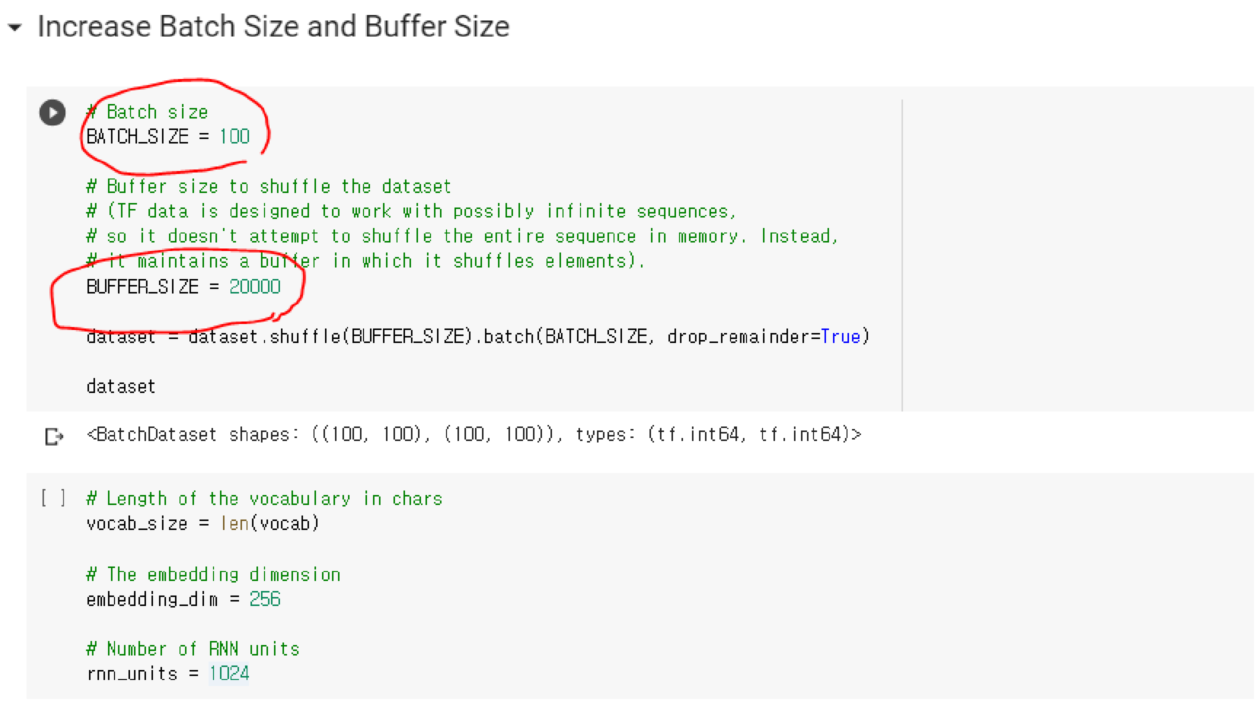
```
print(generate_text(model, start_string=u"ROMEO: "))
```

```
ROMEO: but in being bleeding!

TRANIO:
Fie,
Who that not in sooner, and like a whither:
To change thee, are all broad, who will taunth
Full in this done better power.

Boot MARICANUS:
No, am I in that weather dark you, ere wither.
Uncle.

LUCENTIO:
Every engenled sons.

Mariner:
You would the wiser he hate, urged in so
At when an ount: how now, daughter rather:
Condrity, madam,
But by rings one daughter if a jest true.

DUKE VINCENTIO:
Trouble our joyful as every charity,
Make him my father rather
Maid from my almost vow, and welcome hope to-morrow;
They shall be of each ornight: and much going the common plance
Against the state at safety may.
Not he, God never be the like to steal this axplament,
And un't bais dies; and holf full your demasher.

CLAUDIO:
O spreak, look upon good thing to pity Coaunt, Camillo,
With revenge out upon coming here! apard it you high,
Brother, by my priests may noble lodging.
```

Use a different data and use the model provided in icp 6 source code

Changing 4 hyper parameters

1. Change the Batch Size and Buffer size


Increase Batch Size and Buffer Size

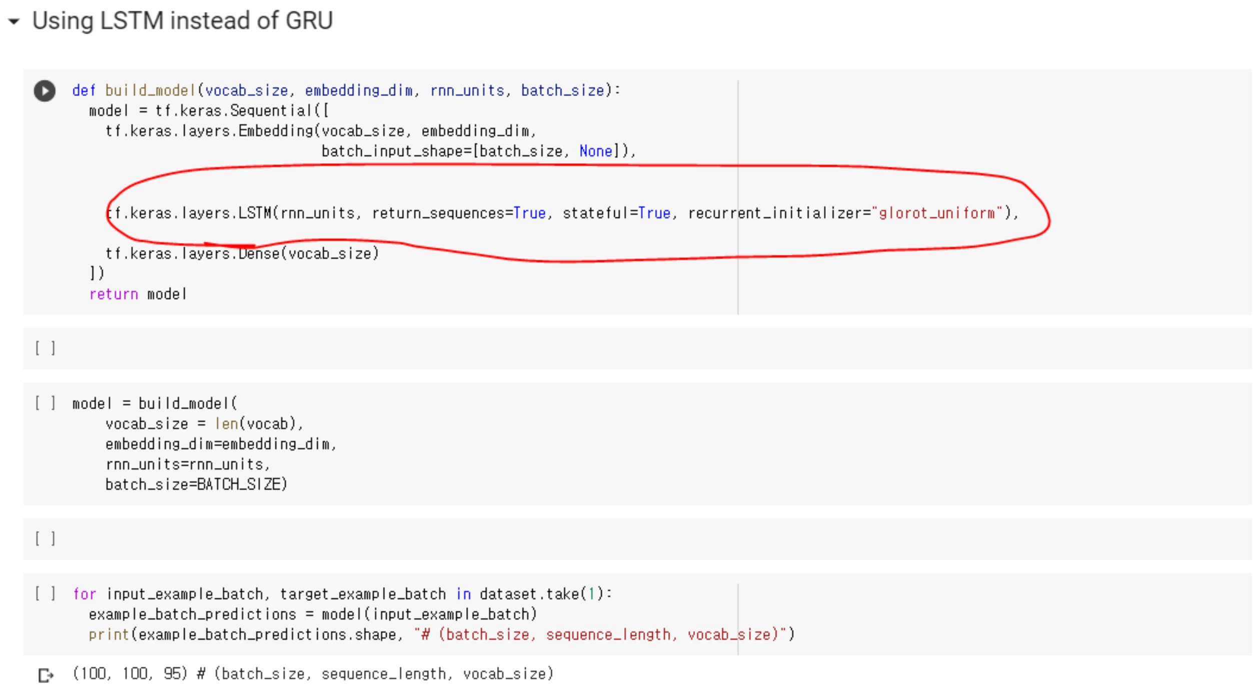
```
# Batch size
BATCH_SIZE = 100

# Buffer size to shuffle the dataset
# (TF data is designed to work with possibly infinite sequences,
# so it doesn't attempt to shuffle the entire sequence in memory. Instead,
# it maintains a buffer in which it shuffles elements).
BUFFER_SIZE = 20000

dataset = dataset.shuffle(BUFFER_SIZE).batch(BATCH_SIZE, drop_remainder=True)

dataset
```

```
<BatchDataset shapes: ((100, 100), (100, 100)), types: (tf.int64, tf.int64)>
```

```
# Length of the vocabulary in chars
vocab_size = len(vocab)

# The embedding dimension
embedding_dim = 256

# Number of RNN units
rnn_units = 1024
```

⇨ **I thought that the model accuracy could be improved by increasing Batch size. Because With a large batch size, I can get more “accurate” gradients because now you are optimizing the loss simultaneously over a larger set of datas. However this is not always true depends on the conditions in model. Also I increased the number of buffer size because I want to get more shuffled data**

2. Change GRU to LSTM Layer



▾ Using LSTM instead of GRU

```
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.LSTM(rnn_units, return_sequences=True, stateful=True, recurrent_initializer="glorot_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)
```

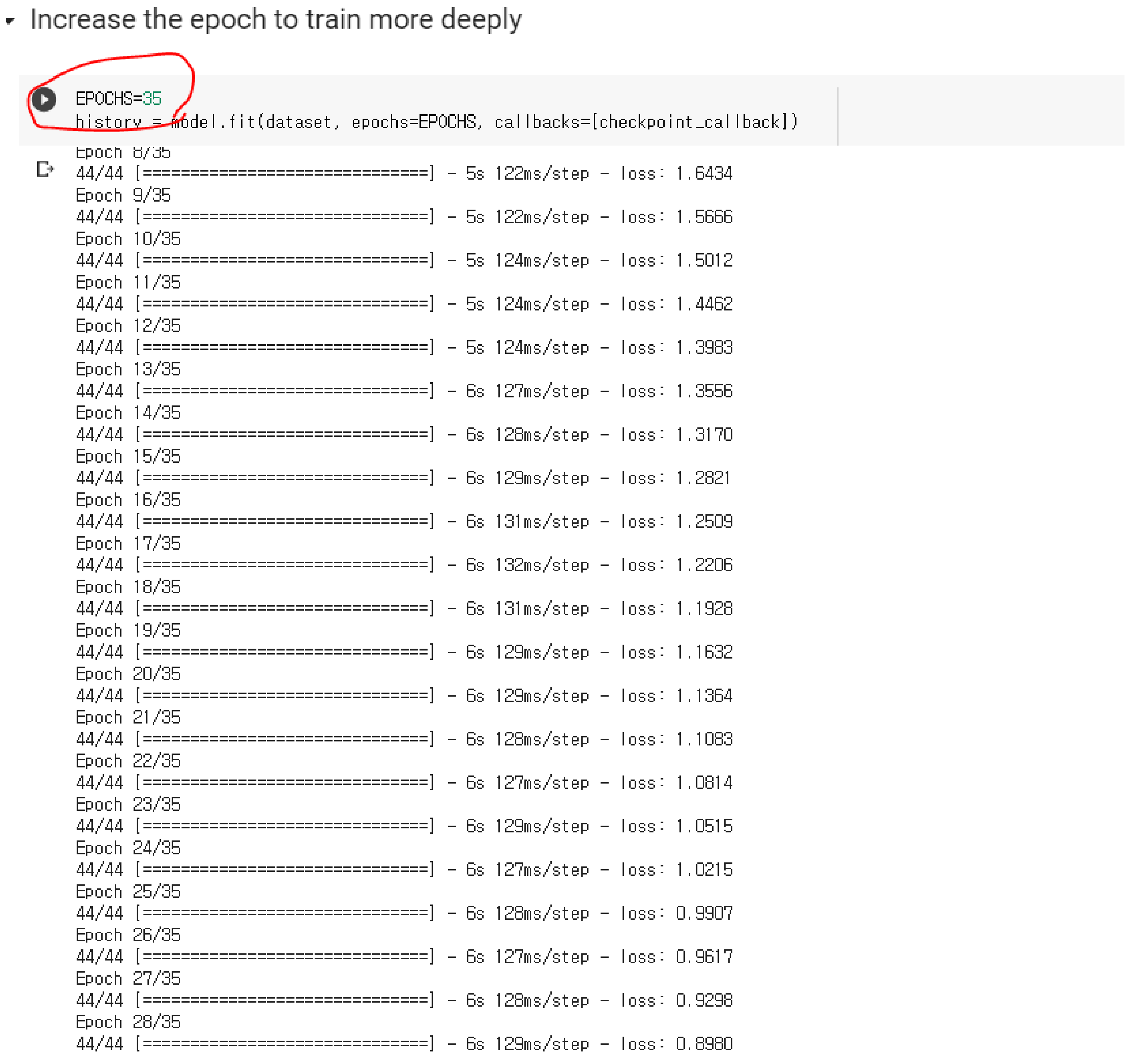
```
[ ]
```

```
[ ] for input_example_batch, target_example_batch in dataset.take(1):
      example_batch_predictions = model(input_example_batch)
      print(example_batch_predictions.shape, "# (batch_size, sequence_length, vocab_size)")
```

```
(100, 100, 95) # (batch_size, sequence_length, vocab_size)
```

⇨ **I changed GRU to LSTM layer. Basically there is no big difference between these two models. However there are some difference GRUs are simpler and thus easier to modify, for example adding new gates in case of additional input to the network. It's just less code in general. LSTMs should in theory remember longer sequences than GRUs and outperform them in tasks requiring modeling long-distance relations. So I changed it to LSTM to see it is better.**

3. Increase the epoch number



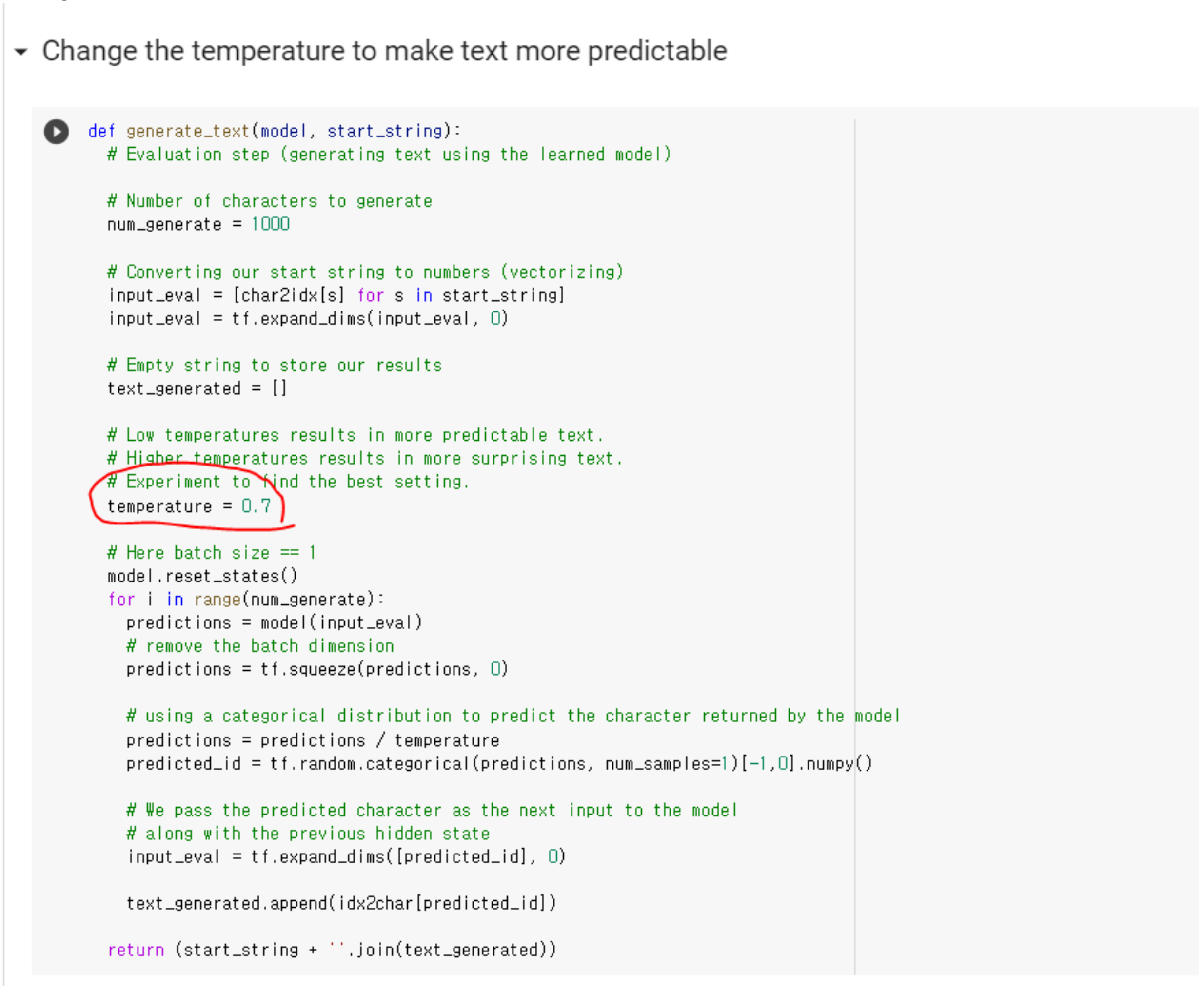
Increase the epoch to train more deeply

```
EPOCHS=35
history = model.fit(dataset, epochs=EPOCHS, callbacks=[checkpoint_callback])
```

```
Epoch 8/35
44/44 [==============================] - 5s 122ms/step - loss: 1.6434
Epoch 9/35
44/44 [==============================] - 5s 122ms/step - loss: 1.5666
Epoch 10/35
44/44 [==============================] - 5s 124ms/step - loss: 1.5012
Epoch 11/35
44/44 [==============================] - 5s 124ms/step - loss: 1.4462
Epoch 12/35
44/44 [==============================] - 5s 124ms/step - loss: 1.3983
Epoch 13/35
44/44 [==============================] - 6s 127ms/step - loss: 1.3556
Epoch 14/35
44/44 [==============================] - 6s 128ms/step - loss: 1.3170
Epoch 15/35
44/44 [==============================] - 6s 129ms/step - loss: 1.2821
Epoch 16/35
44/44 [==============================] - 6s 131ms/step - loss: 1.2509
Epoch 17/35
44/44 [==============================] - 6s 132ms/step - loss: 1.2206
Epoch 18/35
44/44 [==============================] - 6s 131ms/step - loss: 1.1928
Epoch 19/35
44/44 [==============================] - 6s 129ms/step - loss: 1.1632
Epoch 20/35
44/44 [==============================] - 6s 129ms/step - loss: 1.1364
Epoch 21/35
44/44 [==============================] - 6s 128ms/step - loss: 1.1083
Epoch 22/35
44/44 [==============================] - 6s 127ms/step - loss: 1.0814
Epoch 23/35
44/44 [==============================] - 6s 129ms/step - loss: 1.0515
Epoch 24/35
44/44 [==============================] - 6s 127ms/step - loss: 1.0215
Epoch 25/35
44/44 [==============================] - 6s 128ms/step - loss: 0.9907
Epoch 26/35
44/44 [==============================] - 6s 127ms/step - loss: 0.9617
Epoch 27/35
44/44 [==============================] - 6s 128ms/step - loss: 0.9298
Epoch 28/35
44/44 [==============================] - 6s 129ms/step - loss: 0.8980
```

⇨ **Increase the epoch 10 to 35 Because I want to get more accurate result from my model. I used a GPU runtime So it does not take much time. I could have increased epoch number more than 35. But I was afraid that there would be overfitting for the training data. So I just limit it at 35.**

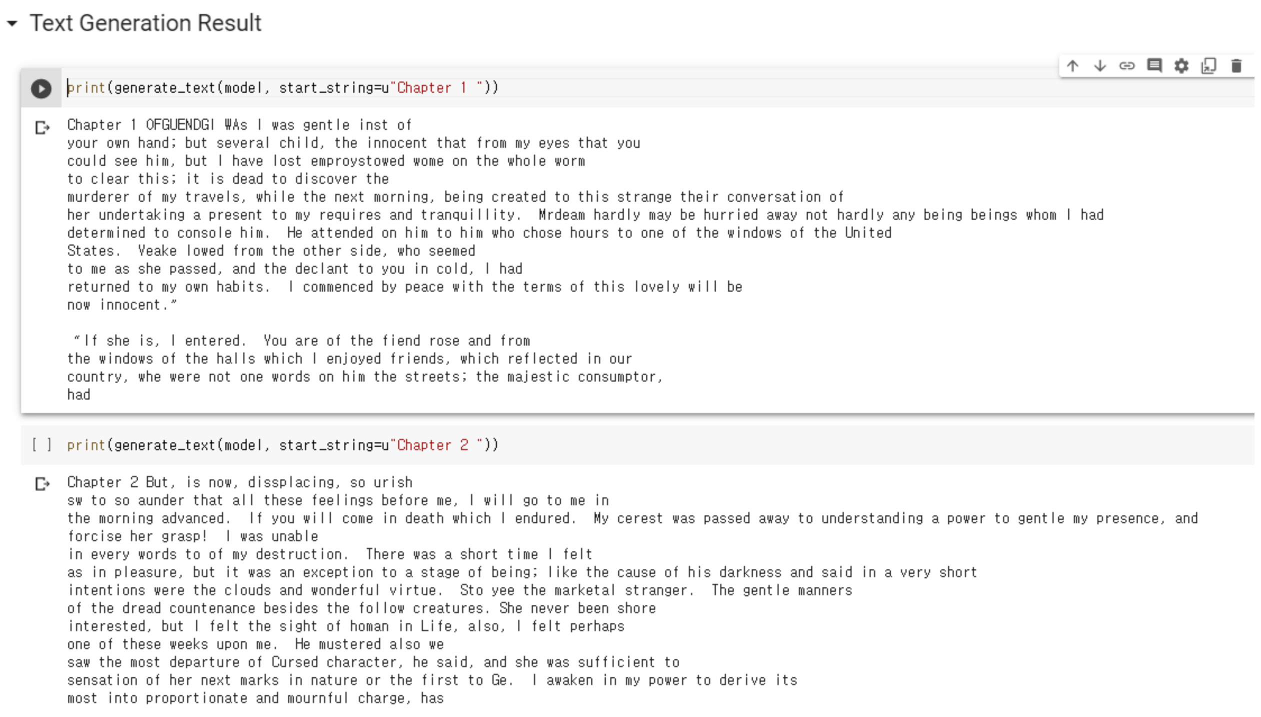
4. **Change the temperature number**



```
def generate_text(model, start_string):
  # Evaluation step (generating text using the learned model)

  # Number of characters to generate
  num_generate = 1000

  # Converting our start string to numbers (vectorizing)
  input_eval = [char2idx[s] for s in start_string]
  input_eval = tf.expand_dims(input_eval, 0)

  # Empty string to store our results
  text_generated = []

  # Low temperatures results in more predictable text.
  # Higher temperatures results in more surprising text.
  # Experiment to find the best setting.
  temperature = 0.7

  # Here batch size == 1
  model.reset_states()
  for i in range(num_generate):
    predictions = model(input_eval)
    # remove the batch dimension
    predictions = tf.squeeze(predictions, 0)

    # using a categorical distribution to predict the character returned by the model
    predictions = predictions / temperature
    predicted_id = tf.random.categorical(predictions, num_samples=1)[-1,0].numpy()

    # We pass the predicted character as the next input to the model
    # along with the previous hidden state
    input_eval = tf.expand_dims([predicted_id], 0)

    text_generated.append(idx2char[predicted_id])

  return (start_string + ''.join(text_generated))
```

⇨ **Increase the epoch 10 to 13. I want to improve more accuracy by increasing accuracy But I found that If I set epoch more than 13 the validation accuracy tend to decrease So I limit the epoch number to 13**

Result of the text generation



Text Generation Result

```
print(generate_text(model, start_string=u"Chapter 1 "))
```

```
Chapter 1 OFGUENDGI WAs I was gentle inst of
your own hand; but several child, the innocent that from my eyes that you
could see him, but I have lost emproystowed wome on the whole worm
to clear this; it is dead to discover the
murderer of my travels, while the next morning, being created to this strange their conversation of
her undertaking a present to my requires and tranquillity.  Mrdeam hardly may be hurried away not hardly any being beings whom I had
determined to console him.  He attended on him to him who chose hours to one of the windows of the United
States.  Veake lowed from the other side, who seemed
to me as she passed, and the declant to you in cold, I had
returned to my own habits.  I commenced by peace with the terms of this lovely will be
now innocent."

 "If she is, I entered.  You are of the fiend rose and from
the windows of the halls which I enjoyed friends, which reflected in our
country, whe were not one words on him the streets; the majestic consumptor,
had
```

```
[ ] print(generate_text(model, start_string=u"Chapter 2 "))
```

```
Chapter 2 But, is now, dissplacing, so urish
sw to so aunder that all these feelings before me, I will go to me in
the morning advanced.  If you will come in death which I endured.  My cerest was passed away to understanding a power to gentle my presence, and
forcise her grasp!  I was unable
in every words to of my destruction.  There was a short time I felt
as in pleasure, but it was an exception to a stage of being; like the cause of his darkness and said in a very short
intentions were the clouds and wonderful virtue.  Sto yee the marketal stranger.  The gentle manners
of the dread countenance besides the follow creatures. She never been shore
interested, but I felt the sight of homan in Life, also, I felt perhaps
one of these weeks upon me.  He mustered also we
saw the most departure of Cursed character, he said, and she was sufficient to
sensation of her next marks in nature or the first to Ge.  I awaken in my power to derive its
most into proportionate and mournful charge, has
```

- **I downloaded random text from internet and build a model and generate text from it. Obviously, There are some misspelling and phrases that does not make sense. But in overall, it is readable and can understand some part of the text.**

Challenges that I faced:

The most difficult challenge that I faced was it was hard to understand the structure of CNN model. So I kept look the documentation of the CNN model and tried to understand it. Finally I could get some insight about this concept and grasp it.

Video link

https://www.youtube.com/watch?v=o2WFhcQusME