**ICP-6:**

**Name: Amarnadha Reddy Ankireddypalli**

**ID: 16314889**

**Email: aa9f3@umsystem.edu**

**Description:**

Use a different data and use the model provided in ICP6 to perform Text generation. You must make 4 changes (for example adding LSTM 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.

**Objective:**

1. Successfully executing the code and making 4 changes in the model.
2. Using a new and good dataset
3. Providing the logical explanation of the changes that you made to model and over all code quality
4. Wiki Report quality, video explanation

**Implementation (with screenshots):**

1. Import required layers and models from tensorflow and keras libraries.

Graphical user interface, text, application

Description automatically generated

1. Read the data. Here I'm using the .txt file of my favorite book 'The Alchemist'.

Text

Description automatically generated

1. There are 86 unique characters in the text file including punctuations.

Graphical user interface, text, application

Description automatically generated

1. Prior to training, we must translate strings to numerical representations. Make two lookup tables: one that maps characters to numbers and another that maps numbers to characters.

Text

Description automatically generated

1. Use the tf.data.Dataset.from\_tensor\_slices function to convert the text vector into a stream of character indices.

Graphical user interface, text, application, email

Description automatically generated

1. The batch method lets us easily convert these individual characters to sequences of the desired size.

Text

Description automatically generated

1. For each sequence, duplicate and shift it to form the input and target text by using the map method to apply a simple function to each batch:

Graphical user interface, text, application, email

Description automatically generated

1. Before feeding this data into the model, we need to shuffle the data and pack it into batches.

Graphical user interface, text

Description automatically generated

1. To define the model, use tf.keras.Sequential. Three layers are utilized to define our model in this simple example:

* The input layer is called embedding. A trainable lookup table that will map each character's numbers to a vector with embedding dimensions.
* GRU: A kind of RNN with size units equal to rnn\_units.
* Dense: The output layer, with outputs of vocab size.

Graphical user interface, text, application

Description automatically generated

1. Check the shape of the model and the summary of the model is as follows:

Table

Description automatically generated

1. Its time to train the model.

* The problem may now be regarded as a normal classification problem.
* Predict the class of the next character based on the previous RNN state and the input this time step.
* Include an optimizer and a loss function. Here, I’m using optimizer ‘adam’.
* We need to specify the from logits flag since our model returns logits.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

1. Fit the model with 10 epochs. Here, the model did not perform well with the loss is 1.68 and an accuracy of 50.7%. Time to generate the text.

Table

Description automatically generated

1. Restore the latest checkpoint To keep this prediction step simple, use a batch size of 1.

Table

Description automatically generated

1. The following code block generates the text:

Text

Description automatically generated

1. Here, the generated text is full of misspelt words and unordered punctuations. Overall, the text doesn’t make any sense and unreadable. Time to tune the hyperparameters.

Text

Description automatically generated

1. Hence, I'll try using a layer of LSTM instead of GRU, with a couple of activation functions to tune it furthermore.

* Here I used tanh and sigmoid functions as activation and recurrent\_activation functions as they go well with classification data.
* In addition to this, I’ve changed the optimizer Adam’s learning rate to 0.01 to give it some pace.

Text

Description automatically generated

Table

Description automatically generated

1. Fitting the model here, with 25 epochs. It did pretty decent here with a loss of 1.24 and an accuracy of 62.08%. Let’s generate the text.

Table

Description automatically generated

Table

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. Here, the model did a decent job in generating valid text with accuracy of 62%. To tune it furthermore.

* I'll add one more layer of LSTM with a couple of activation functions
* I’m dropping out 20% after each layer of LSTM i.e., Dropout(0.2).
* Adam’s learning rate back to its default.
* Decreasing batch size to 64 and increasing epochs to 50.

Text

Description automatically generated with medium confidence

1. Summary of the model:

Table

Description automatically generated

1. Fit the model with 50 epochs. This time, the model did an excellent job with an accuracy of 93% and the loss (0.3) is very less here. Time to generate the text.

Table

Description automatically generated

Table

Description automatically generated

1. This time, the model did an excellent job in generating the exact text with an accuracy of 93%. However, there exists some misspelt words.

Text

Description automatically generated

**Difficulties faced:** In tuning the right metrics for LSTM.

**Video Link:** <https://youtu.be/cHCg8yu0ptU>

**Conclusion:**

From this ICP6, I’ve learnt:

* How to deal with text data, converting char into vectors.
* How to tune-in the parameters in a right way, different approaches for different cases.
* How to generate text using LSTM, GRU by adding their activation functions.