**COIMBATORE INSTITUTE OF TECHNOLOGY, COIMBATORE – 641 014**

**(Government Aided Autonomous Institution affiliated to Anna University)**

**COIMBATORE-641014**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**B.E, VI SEMESTER**

**ADA LOVELACE DAY HACKATHON**

****

**AI BASED POETRY GENERATION(SONNET)**

**SUBMITTED BY:**

**K.J.BHASKER A.RAGAVENTHRAN**

**(**[**1804012ece@cit.edu.in**](mailto:1804012ece@cit.edu.in)**)** [**(1904205ece@cit.edu.in**](mailto:1904205ece@cit.edu.in)**)**

**N.SUBHIKSHA R.SIVASANDHIYA**

**(**[**1804055ece@cit.edu.in**](mailto:1804055ece@cit.edu.in)**) (**[**1804053ece@cit.edu.in**](mailto:1804053ece@cit.edu.in)**)**

**Department of Electronics and Communication Engineering**

**PROBLEM STATEMENT:**

To build and train an AI model with the given dataset to generate a poem(sonet).

**CONTENTS:**

* **SOLUTION CONCEPT**
* **SETUP**
* **READING AND ALIGNING THE DATASET**
* **PROCESSING THE DATA**
* **PREDICTION PROCESS**
* **CRETE TRAINING AND TARGETS**
* **ALLOCATING BATCHES**
* **MODEL**

1. **TRAIN THE MODEL**
2. **EXECUTE THE TRAINING**
3. **GENERATING POEM**

* **API INTERFACINGS**
* **FUTURE ASPECTS**
* **CONCLUSION**

**SOLUTION CONCEPT:**

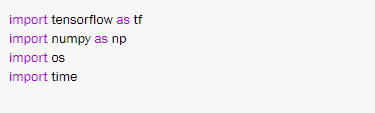
We build this model with RNN concept, to predict the words for the given starting string and generate a poem (14 lines) with approximate sentence meanings.

While some of the sentences are grammatical, most do not make sense. The model has not learned the meaning of words, but considers:

* The model is character-based. When training started, the model did not know how to spell an English word, or that words were even a unit of text.
* The output resembles a play—blocks of text with any input string.

**SETUP:**

Install and import the following library file for this model



**READING AND ALIGNING THE DATASET:**

I used my Google drive as my cloud storage and downloaded the dataset for my model ,I attached text document of training dataset with it in the name of ‘test.txt’

You can download it and copy the path name in the place of my path location



Go Through the dataset should be aligned properly before feeding into the training process to avoid logical errors in the output.



**PROCESSING THE DATA:**

Check the unique characters in the dataset and victories the text. Before training, we need to map strings to a numerical representation. Create two lookup tables: one mapping characters to numbers, and another for numbers to characters.



Now we have an integer representation for each character.

**PREDICTION PROCESS:**

Given a character, or a sequence of characters, what is the most probable next character? This is the task we're training the model to perform. The input to the model will be a sequence of characters, and we train the model to predict the output, the following character at each time step. Since RNNs maintain an internal state that depends on the previously seen elements, given all the characters computed until this moment, what is the next character?

### **CREATE TRAINING AND TARGETS:**

Next divide the text into example sequences. Each input sequence will contain “seq\_length” characters from the text. For each input sequence, the corresponding targets contain the same length of text, except shifted one character to the right. So break the text into chunks of “seq\_length+1”. For example, say “seq\_length” is 4 and our text is "Hello". The input sequence would be "Hell", and the target sequence "ello".To do this first use the “tf.Data.dataset.from\_tensor\_slices ”function to convert the text vector into a stream of character indices.

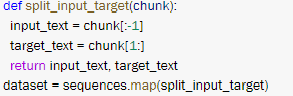


**ALLOCATING BATCHES:**

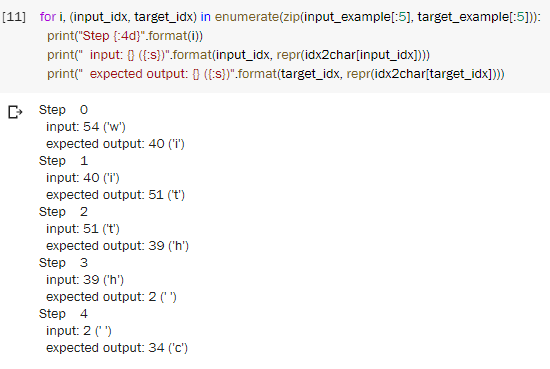
The batch method lets us easily convert these individual characters to sequences of the desired size.



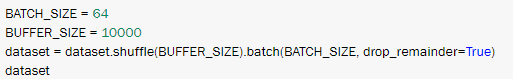
For each sequence, duplicate and shift it to form the input and target text by using the map method to apply a simple function to each batch



Each index of these vectors are processed as one time step. For the input at time step 0, the model receives the index for "F" and tries to predict the index for "i" as the next character. At the next time step, it does the same thing but the RNN considers the previous step context in addition to the current input character. to check that in the program we make print that indexes.



We used tf.data to split the text into manageable sequences. But before feeding this data into the model, we need to shuffle the data and pack it into batches.



**MODEL:**

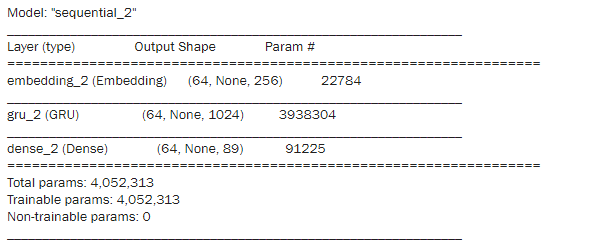
Use tf.keras.sequential to define the model. For this simple example three layers are used to define our model:

* tf.keras.embedding: The input layer. A trainable lookup table that will map the numbers of each character to a vector with embedding\_dim dimensions;
* tf.keras.GRU: A type of RNN with size units=(You can also use a LSTM layer here.)
* tf.keras.layers.dense: The output layer, with vacob\_size outputs
* For each character the model looks up the embedding, runs the GRU one time step with the embedding as input, and applies the dense layer to generate log its predicting the log-likelihood of the next character:

Now run the model to see that it behaves as expected;

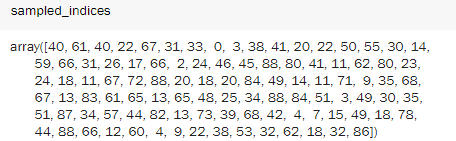


Also look on to the model summary :



Try it for the first example in the batch, it will give the prediction text as array,



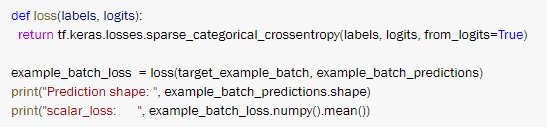


**TRAIN THE MODEL:**

To decode the array to get predictive text output



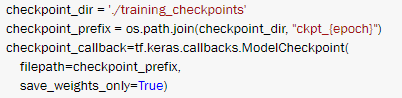
At this point the problem can be treated as a standard classification problem. Given the previous RNN state, and the input this time step, predict the class of the next character.Attach an optimizer, and a loss function



Configure the training procedure using the tf.keras.Model.compile method. We'll use tf.keras.optimizers.Adam with default arguments and the loss function.

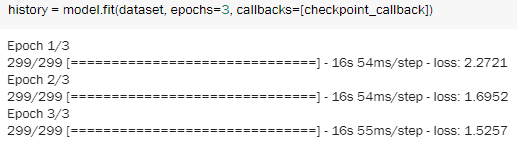


Use a tf.keras.callbacks.ModelCheckpoint to ensure that checkpoints are saved during training:



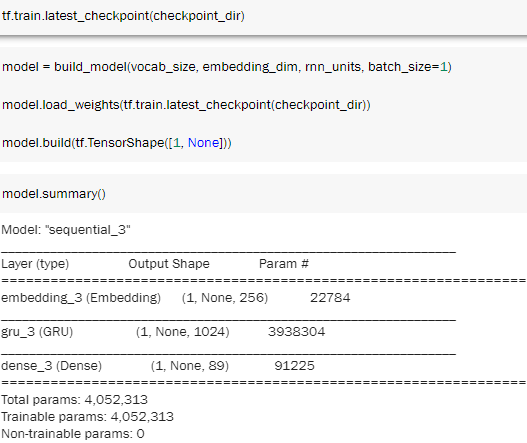
**EXECUTE THE TRAINING:**

Here I used 3 epochs for my training to get a good trained model ,according to your GPU and dataset you can modify it.

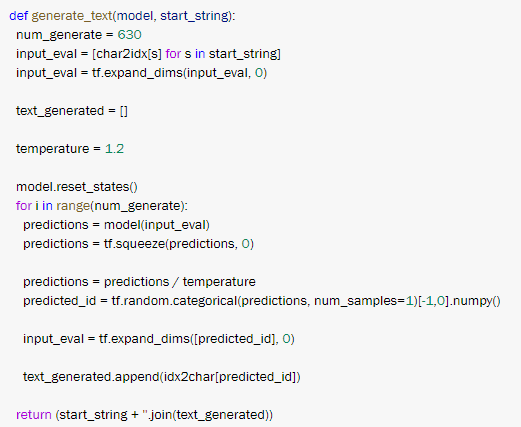


**GENERATING POEM:**

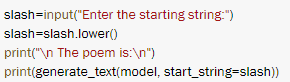
To run the model with a different batch\_size, we need to rebuild the model and restore the weights from the checkpoint.Restore the latest checkpoint,and take the model summary once again, because now the model is trained

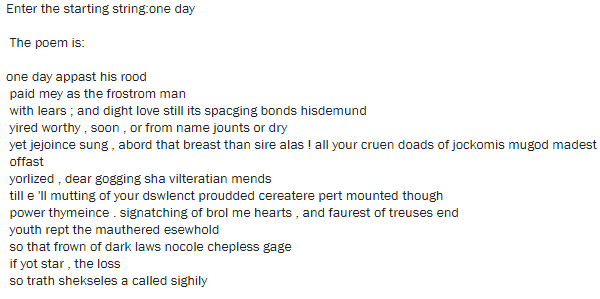


You can also experiment with a different start string, or try adding another RNN layer to improve the model's accuracy, or adjusting the temperature parameter to generate more or less random predictions. Now call the function to generate a poem,



Now the function are defined and the model is trained write a snippet to get the starting string as the input from the compiler and print the generated poem



And an example of generated poem :

**API INTERFACING:**

Additionally we interfaced an API(telegram)for better user environment and user friendly communication, to make it we referred “tensorflow.telegram”you can create a bot by using BOT FATHER in the telegram and copy your bot token

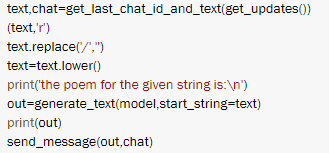
You can replace your token in this code to run this code on your telegram bot, for which you should import some additional library function like



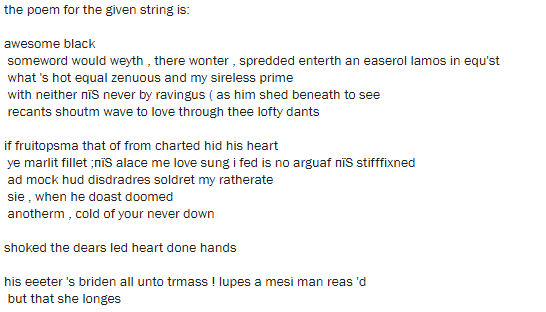
Define the function to your aspects,make the url with the token code



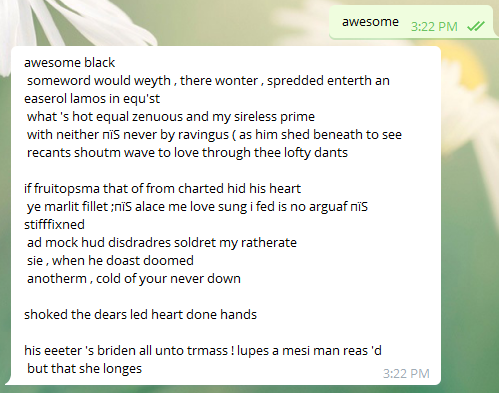
The main program of telegram is:



Run this code to check whether it works on our telegram:



THE OUTPUT I GOT IN TELEGRAM WINDOW:



**FUTURE ASPECTS:**

If we upload our model in the cloud we can access our code at anywhere anytime with the help of an local or server hoist

**CONCLUSION:**

Thus the poem is generated using the model which is trained by the given dataset ,you can increase the efficiency of the model by increasing the training dataset, or increasing the epoch value.