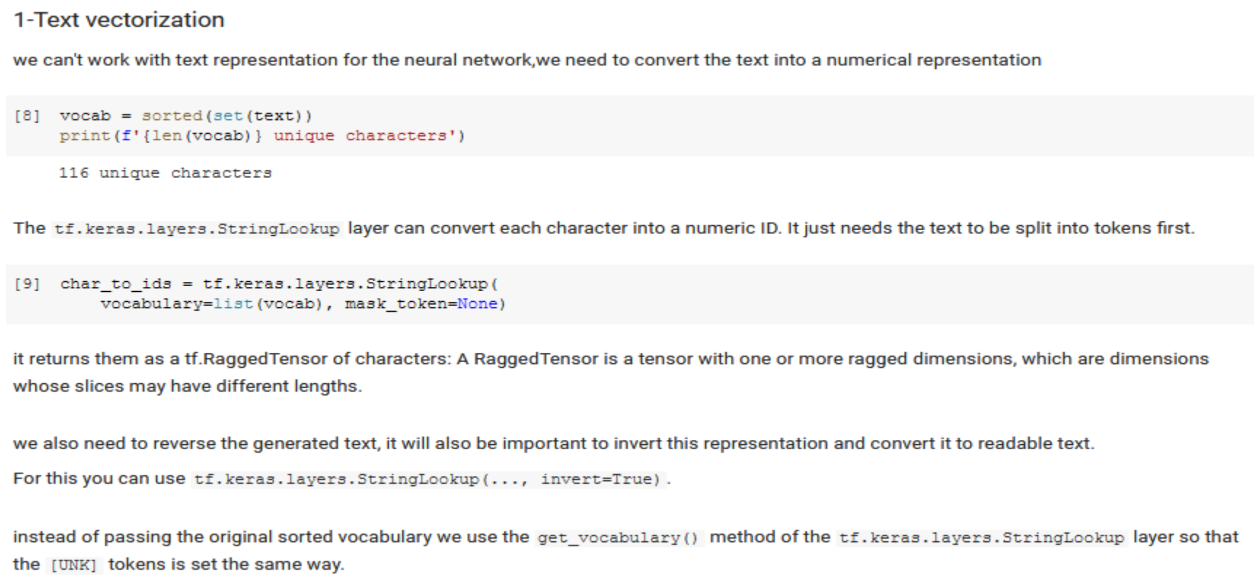
Problem formulation:
Our goal Is to develop a language model that is able to generate text similar to the work of the greek philosopher plato, The Republic.

1- Organize the text into input:

According to the writing from Andrej Karpathy's The Unreasonable Effectiveness of Recurrent Neural Networks, a character level model can produce longer text generation by calling the model repeatedly, this method reduces the need to accurately design a word-level language model and how long the input sequences should be.
This comes at the cost of some grammatical mistakes, as the model has not learned the meaning of words but it can follow the writing structure and language writing skills of the writer which is crucial in works of philosophy.



1-Text vectorization

we can't work with text representation for the neural network,we need to convert the text into a numerical representation

```
[8]  vocab = sorted(set(text))
     print(f'{len(vocab)} unique characters')

     116 unique characters
```

The `tf.keras.layers.StringLookup` layer can convert each character into a numeric ID. It just needs the text to be split into tokens first.

```
[9]  char_to_ids = tf.keras.layers.StringLookup(
         vocabulary=list(vocab), mask_token=None)
```

it returns them as a tf.RaggedTensor of characters: A RaggedTensor is a tensor with one or more ragged dimensions, which are dimensions whose slices may have different lengths.

we also need to reverse the generated text, it will also be important to invert this representation and convert it to readable text.

For this you can use `tf.keras.layers.StringLookup(..., invert=True)`.

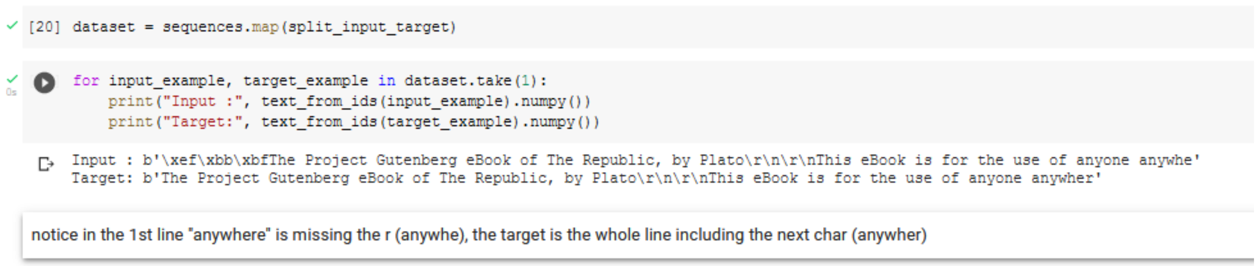
instead of passing the original sorted vocabulary we use the `get_vocabulary()` method of the `tf.keras.layers.StringLookup` layer so that the `[UNK]` tokens is set the same way.

Then We split the text into (input & label) :

- **input** is the current letter
- **label** is the next letter

Next, we divide the text into example sequences. Containing seq_length characters from the text, The required input length was 50 words. We considered an average char count of 5,

so the final input sequence size is 5(chars)*50(sequence) = 250 character , the corresponding targets contain the same length of text, except shifted one character to the right.



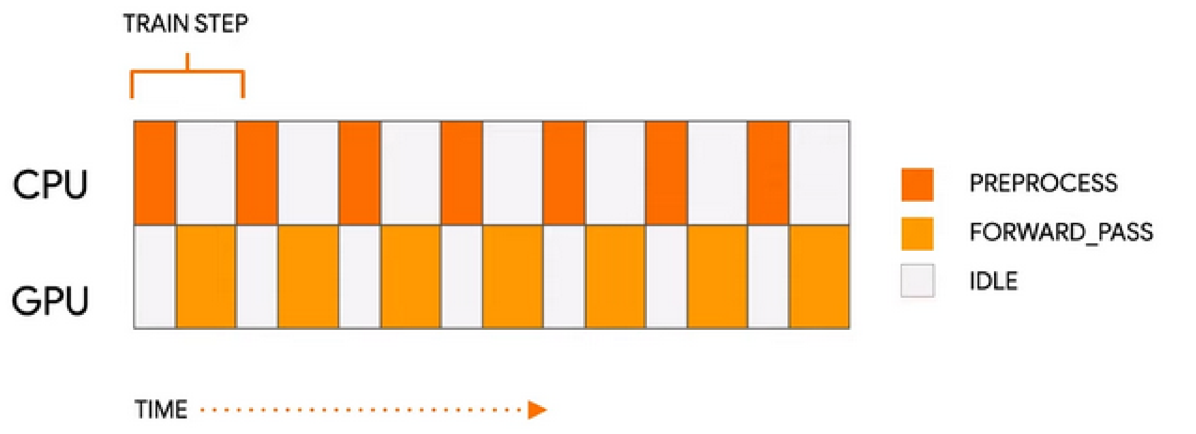
```
[20] dataset = sequences.map(split_input_target)
```

```
for input_example, target_example in dataset.take(1):
    print("Input :", text_from_ids(input_example).numpy())
    print("Target:", text_from_ids(target_example).numpy())
```

```
Input : b'\xef\xbb\xbfThe Project Gutenberg eBook of The Republic, by Plato\r\n\r\nThis eBook is for the use of anyone anywhe'
Target: b'The Project Gutenberg eBook of The Republic, by Plato\r\n\r\nThis eBook is for the use of anyone anywher'
```

notice in the 1st line "anywhere" is missing the r (anywhe), the target is the whole line including the next char (anywher)

Then we further optimized the performance by using prefetch :



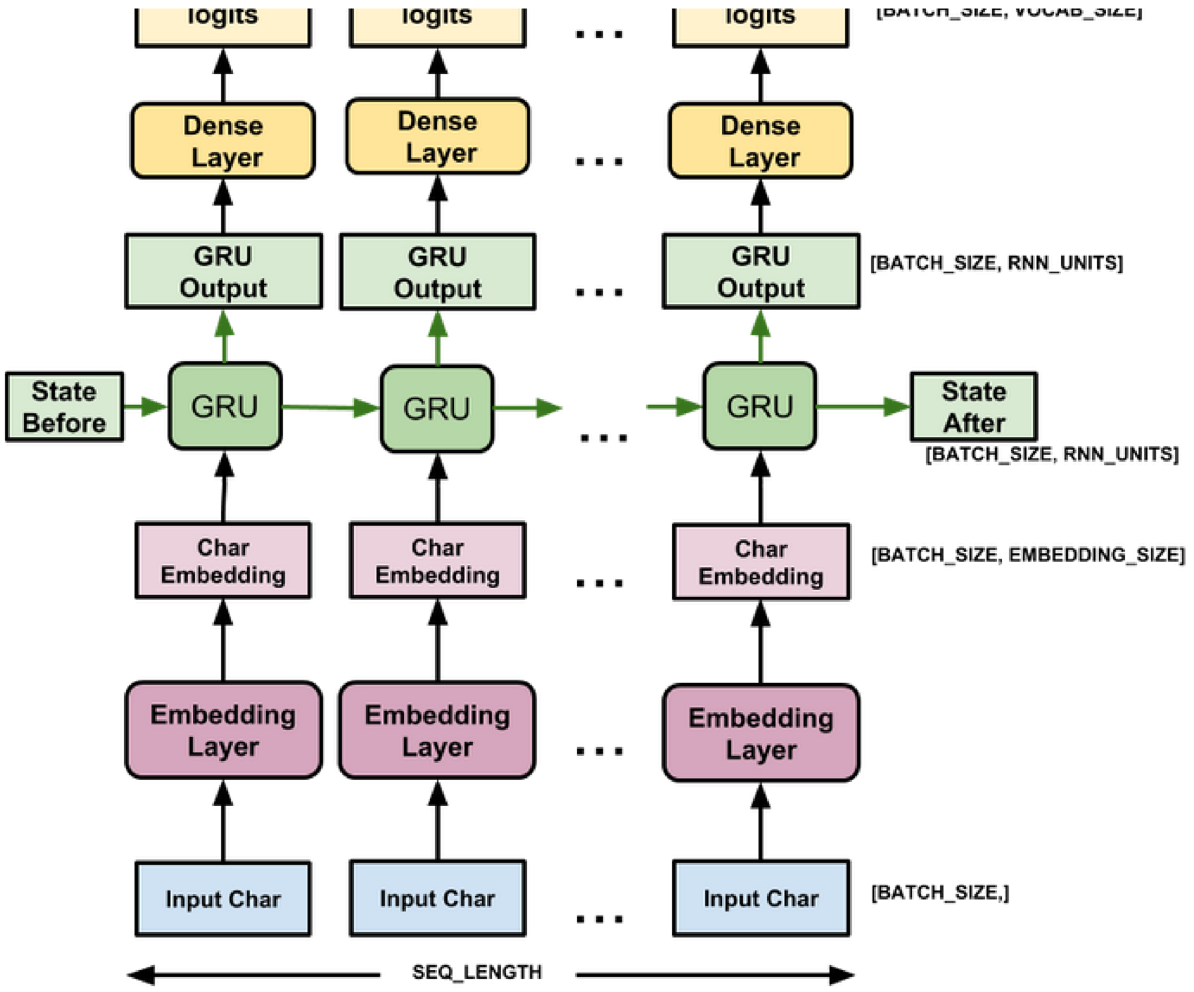
https://www.youtube.com/watch?v=GVShIlh3_yE #official TF youtube channel

normally when we are using preprocessing layer, the preprocessing happens on the CPU, meanwhile the gpu sits there doing nothing waiting the next batch, to optimize this we are using prefetch

Prefetching overlaps the preprocessing and model execution of a training step. While the model is executing training step s, the input pipeline is reading the data for step s+1. Doing so reduces the step time to the maximum (as opposed to the sum) of the training and the time it takes to extract the data.

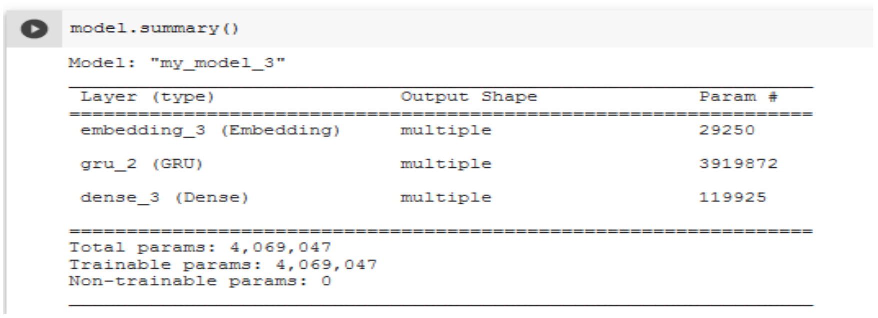
2- model training

Our approach was not to understand the meaning of the words but the writing method of the writer so in each step the model will take an input sequence and the time step and try to predict the next char



This model has only three layers:

- tf.keras.layers.Embedding: The input layer. A trainable lookup table that will map each character-ID to a vector with embedding_dim dimensions;
- tf.keras.layers.GRU: A type of RNN with size units=rnn_units (for our 1st model we chose GRU but we can also choose LSTM here.)
- tf.keras.layers.Dense: The fully connected layer, with vocab_size outputs. It outputs one logit for each character in the vocabulary. These are the log-likelihood of each character according to the model.

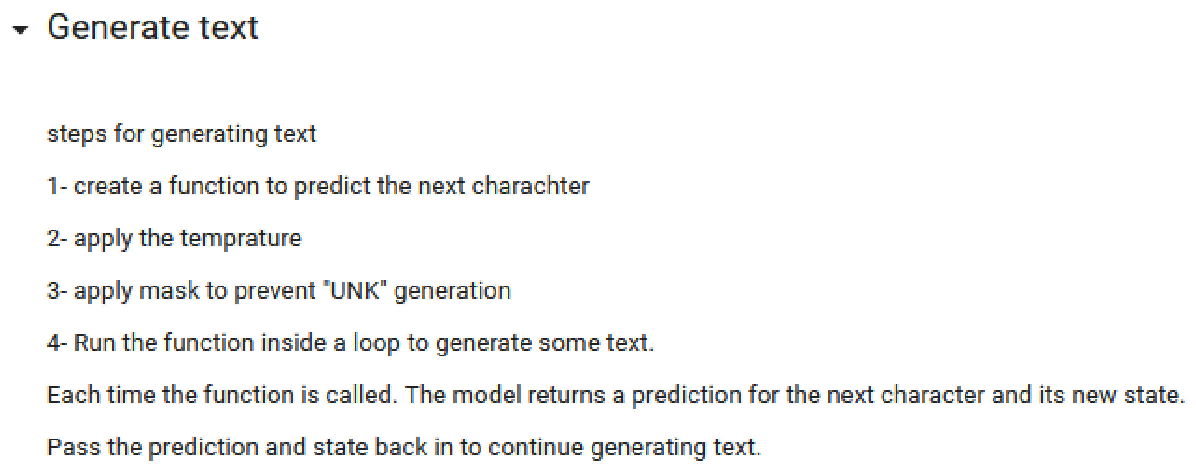


```
model.summary()
```

```
Model: "my_model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     multiple                  29250

 gru_2 (GRU)                 multiple                  3919872

 dense_3 (Dense)             multiple                  119925

=================================================================
Total params: 4,069,047
Trainable params: 4,069,047
Non-trainable params: 0
_________________________________________________________________
```

We trained the model for 20 epochs until it started saturating at 0.7 loss

3- text generation



Generate text

steps for generating text

1- create a function to predict the next charachter

2- apply the temprature

3- apply mask to prevent "UNK" generation

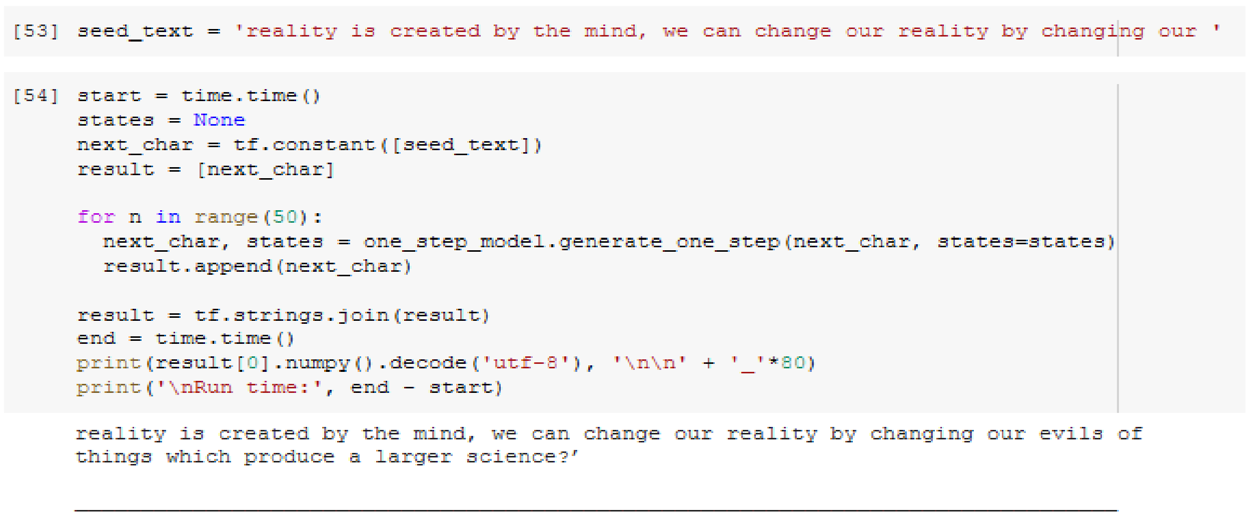
4- Run the function inside a loop to generate some text.

Each time the function is called. The model returns a prediction for the next character and its new state.

Pass the prediction and state back in to continue generating text.

4- results:

to test the model understanding of plato's writing i chose one of my favourite quotes of his as a seed



```
[53] seed_text = 'reality is created by the mind, we can change our reality by changing our '
```

```
[54] start = time.time()
     states = None
     next_char = tf.constant([seed_text])
     result = [next_char]

     for n in range(50):
       next_char, states = one_step_model.generate_one_step(next_char, states=states)
       result.append(next_char)

     result = tf.strings.join(result)
     end = time.time()
     print(result[0].numpy().decode('utf-8'), '\n\n' + '_'*80)
     print('\nRun time:', end - start)
```

```
reality is created by the mind, we can change our reality by changing our evils of
things which produce a larger science?'

________________________________________________________________________________
```

5-Comments:

After generating multiple quotes we kept getting a lot of fun results that contains a lot of mistakes but surprisingly containing good meaning and you can really tell that it's a work of a philosopher

"reality is created by the mind, we can change our reality by changing our mind" - plato

vs

"Reality is created by the mind, we can change our reality by changing our evils of things which produce a larger science?" -our rnn model

there is still room for improvements which may include :

1- **training for more epoches**

2-**adding more LSTM layers**

3-**adjust the temperature parameter to generate more or less random predictions.**

4-**using pretrained transformer models (GPT3- huggingface pipeline)**
https://keras.io/examples/generative/text_generation_with_miniature_gpt/

6- references :

https://keras.io/examples/generative/lstm_character_level_text_generation/

https://www.tensorflow.org/text/tutorials/text_generation

https://www.youtube.com/watch?v=GVShIIh3_yE #official TF youtube channel

https://www.tensorflow.org/guide/keras/custom_layers_and_models