Punctuation Restoration

Approach:

1. The input, and output formats are as follows: Given text without punctuations, for each word, the model has to predict which punctuation is likely to follow the word.

   Hi - ,
   How - None
   are - None
   you - ?
2. We need information from both sides of a word to restore punctuations in the text. A simple way to include information from both sides of a word is bi-directional LSTM. Then the next step is to predict the probabilities of punctuations. This can be done using the Linear layer.

   Architecture:

```
def forward(self,inputs,lengths):
    print(f"Inputs: {inputs.shape}")
    embeds=self.embeddings(inputs)

    packed_input = pack_padded_sequence(embeds, lengths, batch_first=True,enforce_sorted=False)
    packed_output, _ = self.bilstm(packed_input)
    output, _ = pad_packed_sequence(packed_output, batch_first=True)

    prob=self.fc(output)
    return prob
```

Steps followed:

1. Generate Vocabulary using given training data. Here, all the unique words are stored in a dictionary.
2. Divided the data into train, validation, test and declaring data loaders.
3. Defining hyperparameters for the model. Then start training.

Results:

- This architecture did not achieve good results as the model was not able to predict any punctuations. (predicts ‘none’ for every word.) The possible reason could be the architecture is too simple that it could not understand the positions of punctuation.

```
              precision    recall  f1-score   support

           .       0.00      0.00      0.00      4346
           ,       0.00      0.00      0.00      3375
           ?       0.00      0.00      0.00       447
        None       0.97      1.00      0.99    269395

    accuracy                           0.97    277563
   macro avg       0.24      0.25      0.25    277563
weighted avg       0.94      0.97      0.96    277563

Epoch 0                 Training Loss: 0.8714914294806394           Validation Loss: 0.7790489345788956
0.8714914294806394
Number of batches 44
```

	precision	recall	f1-score	support
.	0.00	0.25	0.01	1742
,	0.04	0.19	0.07	1429
?	0.01	0.07	0.01	163
None	0.90	0.09	0.16	125129
accuracy			0.09	128463
macro avg	0.24	0.15	0.06	128463
weighted avg	0.88	0.09	0.16	128463

Test Loss: 1.382284470966884

Challenges faced:

1. It took some time to figure out the input, and output formats. At first, It looked like translation (i.e., unpunctuated text to punctuated text), and started searching for translation techniques. It took some time to think of mechanisms where we could get variable length output (i.e., output length independent of input). Then, realized that there's a simple way of formatting inputs and outputs. I.e, for each word, predict whether there's any punctuation following it.
2. After going through some research papers, found different ways of achieving the goal but because of short and limited time, restricted myself to implementing a simple model. The papers mentioned solutions using transformers, adding attention to RNNs. It was a little difficult to properly understand those architectures because of the time limit.
3. In the proposed solution, the issue comes because of having a high number of samples with 'None' output. It's essential to find a way to balance the classes.