Assignment-1 : Machine Translation – Igpay Atinlay

CSE 291(G) - Wintern 2019

General Instructions to be kept in mind:

1. You are allowed to use a library of your choice for this task, consisting of but not limited to, TensorFlow, PyTorch and Keras.
2. Starter code has been provided in an iPython notebook. Please try to keep your edits limited to this file **only**.
3. Obviously there are multiple ways to code this out and the starter code is just one of the many different ways, you are free to make edits to the function definitions and/or code to suit yourself.
4. There are multiple hints provided in the starter code so it's recommended that you go through the notebook first before you make any edits. This would also be helpful in understanding the overall flow of the code.
5. The submission consists of the edited iPython Notebook and a report (1-2 pages) consisting of answers to questions and the details/insights obtained from your experiments.
6. This assignment has to be done **individually** abiding by the Academic Integrity Policy which has already been relayed to you by the instructor and is included on the syllabus.

1 Pig Latin

"Itontronay Engineersway ockray ethay eefray orldway"

In this assignment we will train a Character RNN model to convert English words to Pig Latin. Although, it's a somewhat deterministic problem, we still would study the potential of RNNs in finding and learning patterns present in language.

Pig Latin is not real language, but a playful way to communicate in "secret." Some general rules for Pig Latin are (read the rest of them at Wikipedia):

1. For words that begin with a consonant, all the letters before the first vowel are removed and placed at the end of the word. Also, an extra *ay* is concatenated at the end. Ex - `pig` gets converted to `igpay`
2. For words that begin with a vowel, the word remains unchanged and an extra *way* is concatenated at the end. Ex - `amazing` gets converted to `amazingway`

To translate a sentence of words to Pig Latin, we apply the rules to each of the words individually.

2 Data and Preprocessing

The dataset has been provided to you in the file "**pig_latin_data.txt**". It consists of 6511 pairs of English and Pig Latin words, consisting of only lowercase english letters and the dash (-) symbol.

You don't have to worry about reading and pre-processing the data from the file. It has already been done for you in the starter code provided and a 80%/20% train/val split has also been established.

The vocablury for this particular task would then consist of the 26 lowercase letters, the dash symbol and the "start of sequence" `<SOS>` and "end of sequence" `<EOS>` symbols, making it a total of 29.

In order to save the headache of variable length sequences in a batch, we would only consider pairs of inputs and targets having equal length across the whole batch which too has already been implemented in the code for you.

3 Model Description

We will be applying an Encoder-Decoder framework for this particular problem as in case with most Machine Translation problems. Specifically, the Encoder takes in a sequence of letters as input and find a hidden representation of the same at each time-step. The last hidden state would be fed at the first step of the decoder which would then produce the output probability distribution of the vocabulary at each time step given the first input as the start symbol.

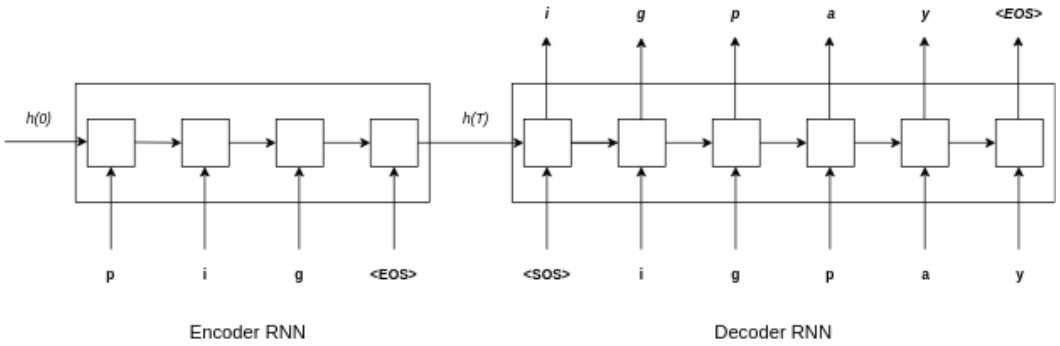

Figure 1: Training example of `pig` as input and target as `igpay`

3.1 GRU [2 points]

The basic unit of an RNN is the cell which processes the input and the hidden state at each time step. As discussed in the class, a GRU cell is a candidate for an RNN cell along with LSTM and a Vanilla cell.

Mathematically, a GRU cell takes in two inputs, a hidden state from the previous time-step $h(t)$ and the input $x(t)$ at the current time step t.

$$z(t) = \sigma(W_z \cdot [h(t-1), x(t)]) \tag{1}$$
$$r(t) = \sigma(W_r \cdot [h(t-1), x(t)]) \tag{2}$$
$$\widehat{h}(t) = tanh(W \cdot [r(t) * h(t-1), x(t)] \tag{3}$$
$$h(t) = (1 - z(t)) * h(t-1) + z(t) * \widehat{h}(t) \tag{4}$$

Question 1. Fill in your code for the MyGRUCell class, the basic structure for which has been provided in the starter code.

3.2 Encoder Architecture [2 points]

As described above, an Encoder is comprised of many GRU cells which process the input sequence.

Question 2. Fill in your code for the Encoder class and write it's forward pass. Note that since you would be passing a sequence of symbol indices to the encoder, you would need to embed them. For this embedding (a vector having size equal to that of the vocablury), you may use the functionality provided by the library that you are using.

3.3 Decoder Architecture [2 points]

The decoder is also comprised of multiple GRU cells. Also note that the first hidden state of the decoder is the output of the last time-step of the encoder. The first input to the decoder is always the start symbol. At each time-step you try to output the next character which should come in the sequence and therefore, the output targets are shifted by one. This method of training with ground-truth inputs as targets is called **Teacher Forcing**.

More formally, "Teacher forcing works by using the actual or expected output from the training dataset at the current time step y(t) as input in the next time step x(t+1), rather than the output generated by the network". Refer Figure 1.

Question 3. Fill in your code for the Decoder class and write it's forward pass. You will also need character embeddings here in addition to an activation function for outputs generated at each time step.

3.4 Training [2 points]

A function to the train the model has already been provided. It loops for n epochs, and in each epoch, it processes the inputs in batches. Needless to say but as mentioned above, the input sequence length (say m) and target sequence length (say n) are the same for all samples in a batch. *Note : m is not necessarily equal to n*

You are also required to define a loss criterion for the outputs of the decoder.

Question 4. Fill in the remaining code for the train function. A high level flow would comprise of generating the input and target batch tensors, feeding the input tensors to the encoder, obtaining the last hidden state of the encoder, feeding it as the first hidden state to the decoder and subsequently the target characters. Refer the Figure 1 to get a more clear picture.

3.5 Evaluation/Inference [1 point]

In the inference stage, you first pass the English language characters at each time-step of the encoder and obtain the hidden representation. This hidden state is fed into decoder at the first time step along with the start symbol. The output (here output refers to the index of the symbol corresponding to the max value of the probability distribution at the current time step) of this is fed as the input symbol for the next time step.

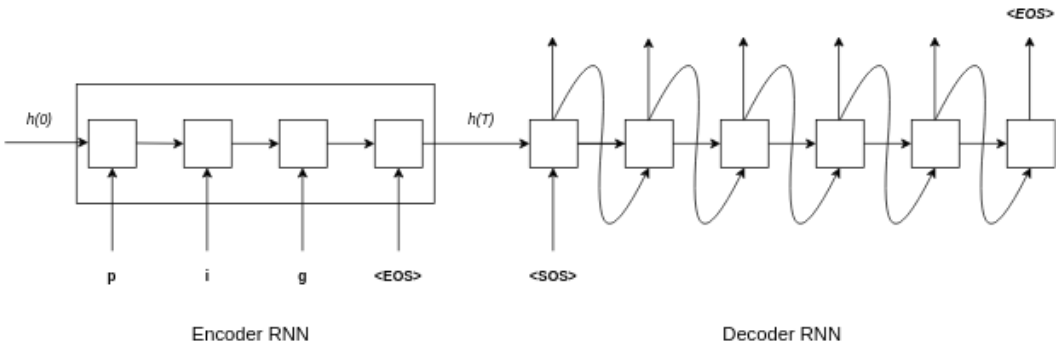

Figure 2: Inference Example of `pig`

Question 5. Complete the *evaluation* and *translate* functions which should be pretty straight-forward once you are done with Question 4.

4 Experiments [8 points]

You are required to run experiments for different batch size and hidden layer size and report the accuracy of your model for each of those cases.

Question 6. Report the hyper-parameters and the loss plots for your best model and give examples of cases where the model succeeds and the cases where it fails.

Also, write briefly about what worked/didn't work? How can the model be improved further? Can Bi-Directional GRU work better for this case? Run a few experiments using a Bi-Directional Encoder and report the results.