Deep Learning Lab #4 (Fall 2021)

Lab Objective:

In this assignment, you will need to implement a seq2seq encoder-decoder network with recurrent units for English spelling correction.

Rules:

(1) This assignment should be done individually. Plagiarism is strictly prohibited.
(2) You can only use Numpy and other Python standard library. Only PyTorch are allowed in this lab.
(3) You should add comments throughout your implementation for easy understanding.
(4) Write a report in the end of the Jupyter Notebook to detail your procedures and discussions.

Submission:

(1) Please write your code on Jupyter notebook.
(2) Pack the .ipynb to .zip, and submit .zip to E3. Please name as "Lab4_YourStudentID.zip".

Deadline: 2021/12/27 (Mon.) 23:55

Requirements:

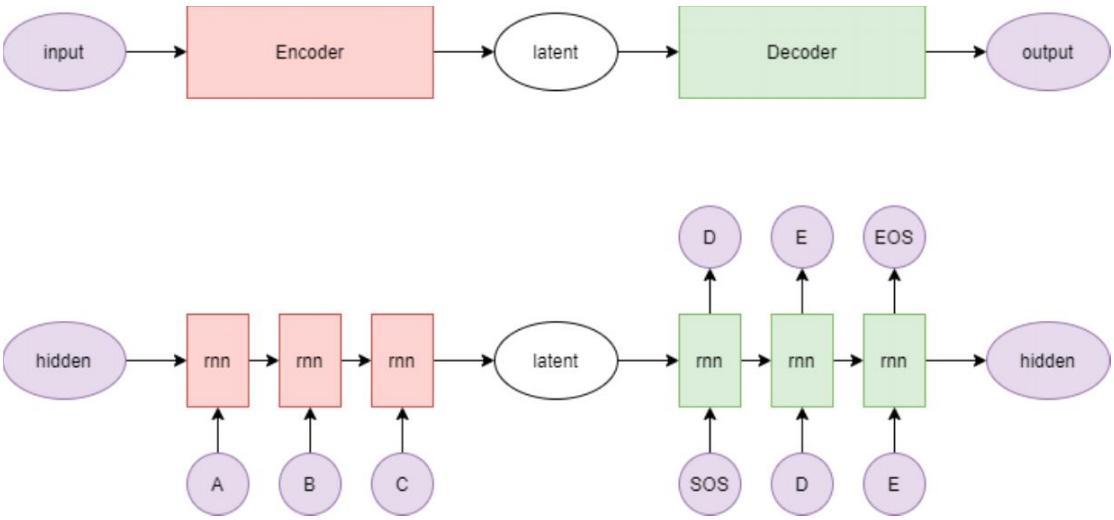
(1) Implement a seq2seq model: select either track.
    I. Track 1: Write your own code!
    II. Track 2: Use sample code (sample.py)
        - Modify encoder, decoder, and training functions.
        - Implement the evaluation function and the dataloader.
        - Note: This sample code is provided for those who have problem constructing their own project. The model structure in sample code may not bring out an optimal result. If you wish to improve the performance, we strongly recommend Track 1 for you!
(2) Plot the CrossEntropy training loss and BLEU-4 testing score curves during training.
(3) Output the correction results from test.json and new_test.json

Descriptions:

(1) Dataset

Train your model with train.json, and test your model with both test.json and new_test.json. Details of the dataset can be found in readme.txt.

(2) Seq2seq architecture



Each character in a word can be regarded as a vector. One simple approach to convert a character to a vector is encoding a character to a number and adopting Embedding Layer (see more information in [1]). In the decoder part, you will first feed the hidden output from the encoder and a <start of string> token to the decoder and stop generation process until you receive a <end of string> token or reach the last token of the target word (the token should also be <end of string>).

(3) Word encoding/embedding function

Since we cannot directly feed 'characters' into the model, so encoding/embedding techniques are required here. The simplest way is to encode each word into one-hot vector. However, one-hot vector is unable to represent the **relation between words** which is very important to NLP, and **padding** is also needed because of the uneven length of words. Therefore, you can further encode words with Embedding function that can be regarded as a trainable projection matrix or lookup table. Embedding function can not only compress feature dimension but also building connection between different words. If you prefer using embeddings instead of one-hot encoding, you can look up word2vec [3], Glove embedding [4], or other tools.

(4) Teacher forcing

In the course, we have talked about teacher forcing technique, which feeds the correct target $y^{(t-1)}$ into $h^{(t)}$ during training. Thus, in this part, you will need to implement teacher forcing technique. Furthermore, to enhance the

robustness of the model, we can do the word dropout to weaken the decoder by randomly replacing the input character tokens with the unknown token (defined by yourself). This forces the model only relying on the latent vector z to make predictions.

(5) Other implementation details

I. The encoder and decoder must be implemented by LSTM, otherwise you will get no point on your report.

II. Evaluate your model performance with BLEU-4 score. You may use NLTK toolkit to help you get this score (see reference [2]).

(6) Parameters settings

(For reference only. Try different settings and find out which combinations is better!)

I. Loss function: nn.CrossEntropyLoss()

II. Optimizer: SGD.

III. LSTM hidden size: 256 or 512

IV. Learning rate: 0.05

(7) Output examples

English spelling correction with BLEU-4 score

(test.json and new_test.json)

```
============================
input:  oportunity
target: opportunity
pred:   opportunity
============================
input:  parenthasis
target: parenthesis
pred:   parenthesis
============================
input:  recetion
target: recession
pred:   recession
============================
input:  scadual
target: schedule
pred:   schedule
BLEU-4 score:0.8030
```

Assignment Evaluation:

(1) Code & model performances (60%)

(2) Report (40%)

Reference:

[1] Seq2seq reference code:

I. https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

II. https://github.com/pytorch/tutorials/blob/master/intermediate_source/seq2seq_translation_tutorial.py

[2] Natural Language Toolkit: https://www.nltk.org/

[3] Distributed Representations of Words and Phrases and their Compositionality

[4] Glove: Global Vectors for Word Representation

[5] Sequence to Sequence Learning with Neural Networks

Please contact TA if you have any questions.