DL Lab3: seq2seq model

Lab Objective:

In this assignment, you will need to implement two seq2seq encoder-decoder networks for English spelling correction, they are LSTM & Transformer respectively.

Rules:

(1) This assignment should be done individually. Plagiarism is strictly prohibited. Once the T.A. finds plagiarism, you will receive a score of **0** on this assignment.

(2) **PyTorch** is the only framework allowed in this lab, beyond that you can only use numpy, matplotlib, nltk and other Python standard library.

(3) If this assignment format and files are not in accordance with the regulations, the assignment score will be **multiplied by 0.9**.

(4) If the assignment is missing or incomplete training for any item, the assignmentscore will be deducted proportionally to the incompleteness.

(5) If you submit your assignment late, your score will be **multiplied by 0.9 for each day of delay**.

Submission:

(1) Please submit your code on Jupyter notebook and the filename should be **A3_studentID_studentName_LSTM.ipynb & A3_studentID_studentName_Transformer.ipynb**.

(2) The report should be saved as **A3_studentID_studentName.pdf** and only be handed in **eight A4 pages at most**. You should explain all the implications of all the programs you write and post them in the report. Besides, you need to **compare LSTM and Transformer,** please explain the difference and present their effects in the report, but it is strictly forbidden to post the entire code.

(3) Upload the compressed file (.zip) of “the report and all program files” to the e3 platform. The file name is **A3_studentID_studentName.zip**.

Deadline: 2023/11/21 (Tue.) 05:20

Requirements:

(1) Track A. Implement a seq2seq model by yourself.

(2) Track B. Or Use sample code (sample.py)

- Modify encoder, decoder, and training functions.
- Implement the evaluation function and the dataset.

(3) Plot the **CrossEntropy training loss and BLEU-4 testing score curves** during training. And show the correction results from **test.json** and **new_test.json** respectively.

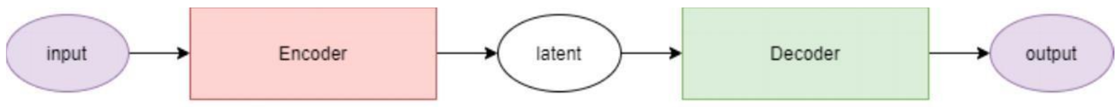
(4) (Optional parts) Compare performance changes due to different parameters and model structures and write them into reports (Bonus)

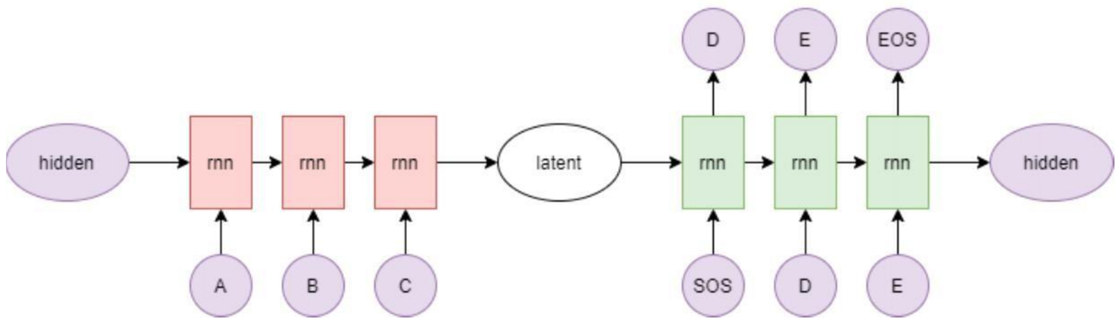
Descriptions:

(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>). Apart from the above mentioned, you may use some **<padding of string>** tokens to control the length of each sequence.

(3) Requirements for LSTM and Transformer respectively

I. LSTM: Implement LSTM by yourself or just use torch.nn.LSTM

II. Transformer: Implement Transformer by yourself or use torch.nn (TransformerEncoder, TransformerEncoderLayer, etc.) to construct your encoder-decoder Transformer **except torch.nn.Transformer**.

(4) 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. 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 [5], Glove embedding [6], or other tools.

(5) 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.

(6) Other implementation details

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

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

(7) Parameters settings

(For reference only. Try different settings and find out which combinations is better and compare the results into a report, will get your extra points)

I. Loss function: nn.CrossEntropyLoss()

II. Optimizer: SGD.

III. Learning rate: 0.05

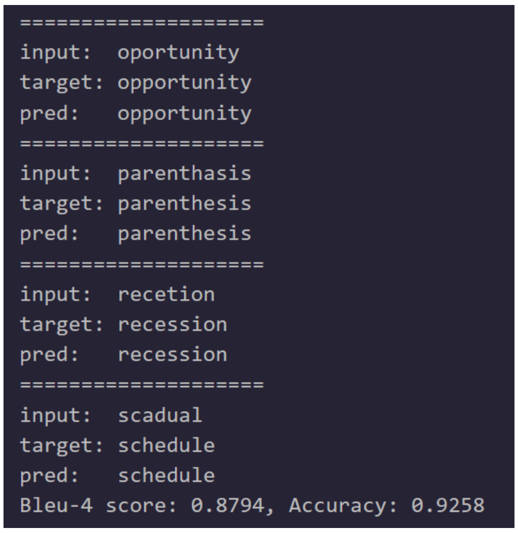
IV. Word embedding: 256

V. Hidden size: 512

(8) Output examples

English spelling correction with BLEU-4 score and Accuracy

(Evaluate test.json and new_test.json respectively)



```
====================
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: 0.8794, Accuracy: 0.9258
```

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] LSTM:

I. https://www.bioinf.jku.at/publications/older/2604.pdf

II. https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html

[3] Transformer:

I. https://arxiv.org/pdf/1706.03762.pdf

II. https://pytorch.org/docs/stable/nn.html#transformer-layers

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

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

[6] Glove: Global Vectors for Word Representation

[7] Sequence to Sequence Learning with Neural Networks

Please contact TA if you have any questions.

tumeng0302.ai11@nycu.edu.tw

bocheng.ai11@nycu.edu.tw