CSE402 Assignment #2: Neural Machine Translation

Due Date: May 31st, Saturday, 11:59 PM KST.
In this assignment, you will build a neural machine translation with attention mechanism based on LSTM, using PyTorch.

1. Sequence to Sequence LSTM Model (125 points)

A Sequence-to-Sequence model generates an output sequence $[y_1, \ldots, y_m]$ given an input sequence $[x_1, \ldots, x_n]$. Each token in the output sequence can be expressed as a conditional probability that depends on the previously generated output tokens and the input sequence. The output sequence can be represented as follows:

$$p(y, .., y_m | x_1, ..., x_n) = \prod_{t=1}^{m} p(y_t | \mathbf{x}, y_1, ..., y_{t-1})$$
$$\mathbf{x} = \text{encoder}(x_1, ...x_n)$$

where $\mathbf{x}$ denotes the representation of the input sequence processed by the model's encoder.

LSTM uses two types of representations:1) the *cell state*, which records the internal state of the LSTM cell from the previous time step, and 2) the *hidden state*, which represents the output corresponding to the current token.

To enhance the representational power of the encoder, a *bidirectional LSTM* is employed to capture information in both directions: from past to future and from future to past. On the other hand, the decoder generates tokens sequentially without access to future words; therefore, it uses a *unidirectional* LSTM.

In attention-based neural machine translation, while the exact specification of the decoder's hidden states may not be critically important, we aim to initialize the decoder's hidden and cell states using those of the encoder, likely following the original non-attentive neural machine translation. To this end, we initialize the decoder's hidden and cell states by projecting the bidirectional representations from the encoder into the decoder's space. To be more specific, suppose that $\overrightarrow{\mathbf{h}}_t^{enc} \in \mathbb{R}^d$ and $\overleftarrow{\mathbf{h}}_t^{enc} \in \mathbb{R}^d$ are the forward and backward hidden states of the encoder at the t-th timestep. Similarly, $\overrightarrow{\mathbf{c}}_t^{enc} \in \mathbb{R}^d$ and $\overleftarrow{\mathbf{c}}_t^{enc} \in \mathbb{R}^d$ are the corresponding cell states of the forward and backward encoders, respectively. The initial hidden and cell states, referred to as $\mathbf{h}_0^{dec}$ and $\mathbf{c}_0^{dec}$, respectively, are defined as follows:

$$\begin{aligned}
\mathbf{h}_0^{dec} &= \mathbf{W}_h \cdot \left[\overrightarrow{\mathbf{h}}_T^{enc}, \overleftarrow{\mathbf{h}}_T^{enc}\right] + \mathbf{b}_h && (1)\\
\mathbf{c}_0 &= \mathbf{W}_c \left[\overrightarrow{\mathbf{c}}_T^{enc}, \overleftarrow{\mathbf{c}}_T^{enc}\right] + \mathbf{b}_c && (2)\\
\mathbf{h}_1^{dec}, \mathbf{c}_1^{dec} &= \text{LSTM}^{dec}(y_0, \mathbf{h}_0^{dec}, \mathbf{c}_0) && (3)
\end{aligned}$$

where T is the number of the input sentence in the encoder, $\mathbf{h}_t^{dec}$ and $\mathbf{c}_t^{dec}$ refers to the hidden and cell states of the decoder at t-th timestep, and y_0 refers to the start token for the decoder.

In neural machine translation, Luong et al. (2015) further explored the attention mechanism initially proposed by Bahdanau et al. (2015), comparing global and local attention approaches, as illustrated in Figure 1.

Furthermore, Luong et al. (2015) proposed the *input feeding* method, which incorporates the previous attention results by injecting the attentive output from the previous time step into the embedding layer, as in Figure 2.

Please first read the following papers for the LSTM-based Sequence-to-Sequence model:

1. Neural Machine Translation by Jointly Learning to Align and Translate
2. Effective Approaches to Attention-based Neural Machine Translation

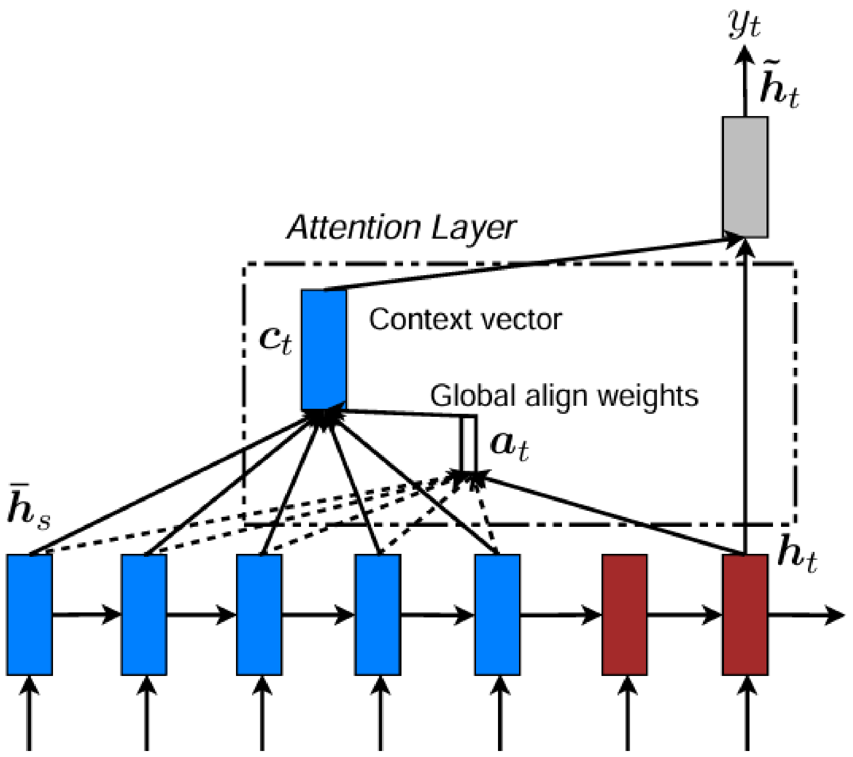

Figure 1: Attention Mechanism

(a) (100 points) We provide a jupyter notebook file (`seq2seq.ipynb`) that implements a version of the LSTM-based Sequence-to-Sequence model You are required to complete the missing codes by filling up the parts in the following modules.

- `GlobalAttention.forward`
- `LocalAttention.forward`
- `LocalAttention.gather local context`
- `Encoder.forward`
- `Seq2Seq.forward`
- `Seq2Seq.generate`

You don't need to modify other parts of the notebook files, except for settings and configurations based on your environment.

Once you correctly add the notebook file, you should see the following results.

Listing 1: Outputs after running nb_classifier.ipynb

```
    ...
Seq2Seq(
  (embedding): Embedding(65002, 512, padding_idx=65000)
  (encoder): Encoder(
    (encoder): LSTM(512, 512, num_layers=2, batch_first=True, dropout=0.1, bidirectional=
    ↪True)
    (h_dec_proj): Linear(in_features=1024, out_features=512, bias=True)
    (c_dec_proj): Linear(in_features=1024, out_features=512, bias=True)
  )
```

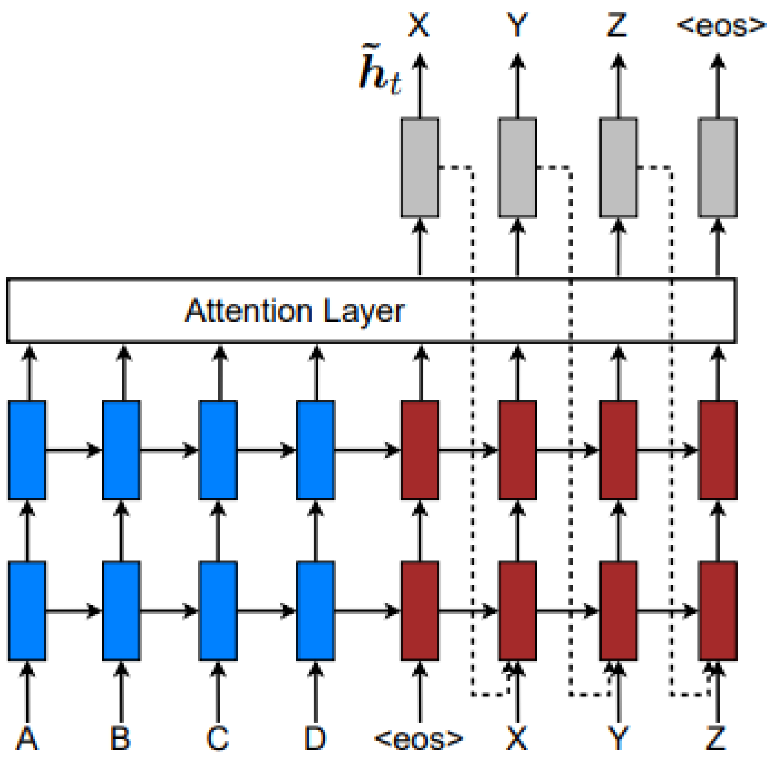

Figure 2: Input Feeding

```
  (decoder): Decoder(
    (decoder): LSTM(512, 512, num_layers=2, batch_first=True, dropout=0.1)
    (attention): GlobalAttention(
      (query_proj): Linear(in_features=512, out_features=512, bias=False)
      (key_proj): Linear(in_features=1024, out_features=512, bias=False)
      (value_proj): Linear(in_features=1024, out_features=512, bias=False)
      (output_proj): Linear(in_features=512, out_features=512, bias=False)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (lm_head): Linear(in_features=512, out_features=65002, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
)
Epoch 1/3:   0%|          | 72/18750 [02:01<8:52:25,  1.71s/it, loss: 9.2253 lr: 1.00e
    ↪-04]
...
```

(b) (25 points) Please present the effect of using the *input feeding* method by comparing the runs with and without it. Provide the comparison results and briefly explain the reason behind the observed difference.

Submission Instructions

You shall submit this assignment on Blackboard by uploading a single file `assignment2.zip`, which consists of two submission files – one for coding questions and another for "report", as follows:

1. `assignment2_coding.zip`: Run the `collect_submission.sh` script to produce your `assignment2_coding.zi` file, and Upload your `assignment2_coding.zip` file. Please note that it is your responsibility to ensure that your code is runnable. If the code does not run, no points will be awarded.

2. `assignment2_report.pdf`: Upload your written reports for the problem (b).

References

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Y. Bengio & Y. LeCun (Eds.), *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings.* Retrieved from `http://arxiv.org/abs/1409.0473`

Luong, T., Pham, H., & Manning, C. D. (2015, September). Effective approaches to attention-based neural machine translation. In L. Màrquez, C. Callison-Burch, & J. Su (Eds.), *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 1412–1421). Lisbon, Portugal: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/D15-1166/` doi: 10.18653/v1/D15-1166