Neural Machine Translation

CSC401/2511 A2 Tutorial 1

Outline

1. Typos in the starter code
2. Walkthrough of the assignment
   a. Overview
   b. Calculating BLEU scores
   c. Encoder
   d. DecoderWithoutAttention
   e. DecoderWithAttention
   f. DecoderWithMultiHeadAttention
   g. Putting it together: EncoderDecoder
   h. Training and testing loop
3. teach.cs with GPU: srun
4. Q&A

Most of the material is covered in the SMT lecture

Page number to the slides: [pXX]

Typos in the Starter Code (piazza @354)

1. EncoderDecoder.init_modules(...):

```
# …
# 3. You will need the following object attributes:
# … self.heads
```

2. BLEU_score(reference, hypothesis, n):

↓

```
BLEU_score(reference, candidate, n):
```

3. DecoderWithAttention.attend(...):

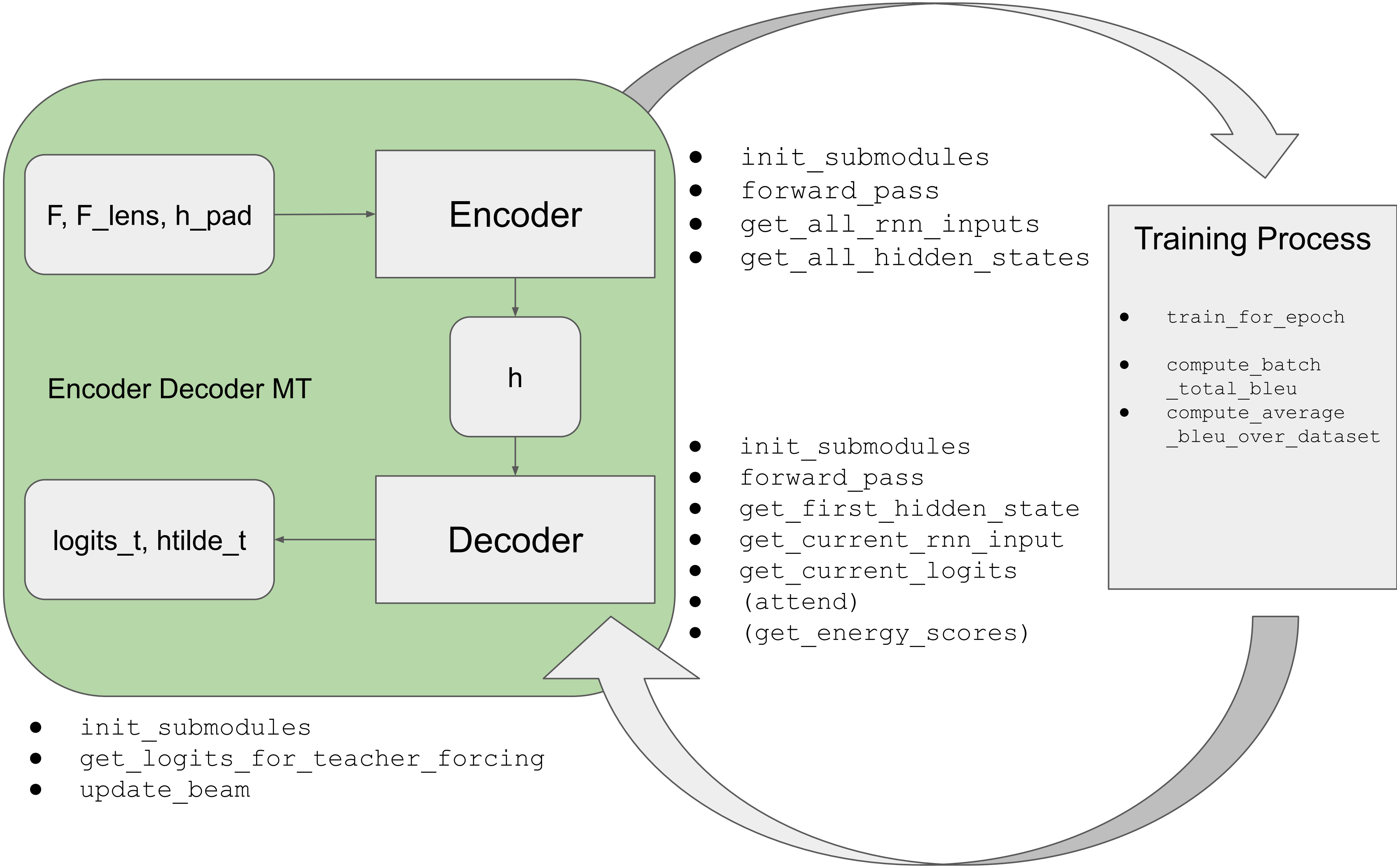
```
c_t : torch.FloatTensor
    A float tensor of shape ``(M, self.hidden_state_size)``.
The context vector c_t is the product of weights alpha_t and
h.
```

Overview: Canadian Hansards

- Data located at `/u/cs401/A2/data/Hansard/`
- You don’t need to download the data, just set these environment variables

```
export TRAIN=/h/u1/cs401/A2/data/Hansard/Training/
export TEST=/h/u1/cs401/A2/data/Hansard/Testing/
```

Overview: The Assignment



Encoder Decoder MT

F, F_lens, h_pad

Encoder

h

Decoder

logits_t, htilde_t

- `init_submodules`
- `forward_pass`
- `get_all_rnn_inputs`
- `get_all_hidden_states`

- `init_submodules`
- `forward_pass`
- `get_first_hidden_state`
- `get_current_rnn_input`
- `get_current_logits`
- `(attend)`
- `(get_energy_scores)`

Training Process

- `train_for_epoch`
- `compute_batch_total_bleu`
- `compute_average_bleu_over_dataset`

- `init_submodules`
- `get_logits_for_teacher_forcing`
- `update_beam`

Calculating BLEU scores

- `grouper`

  ```
  >>> grouper(['a', 'b', 'c'], 2)

  [['a', 'b'], ['b', 'c']]
  ```

- `n_gram_precision`
  - $\frac{C}{N}$
  - [p68]
- `brevity_penalty`
  - $\text{Brevity}_i = \frac{r_i}{c_i}$

    $$\text{BP} = \begin{cases} 1 & \text{brevity}_i < 1 \\ e^{1-\text{brevity}_i} & \text{brevity}_i \geq 1 \end{cases}$$
  - [p72]
- `BLEU_score`
  - $BLEU_c = BP_c \times (p_1 p_2 ... p_n)^{\frac{1}{n}}$
  - [p73]

- No capping
- Only 1 reference and 1 candidate
- Don’t include SOS and EOS

Encoder

Encoder

$$x_s = T_F(F_s)$$

$$h = f(x)$$

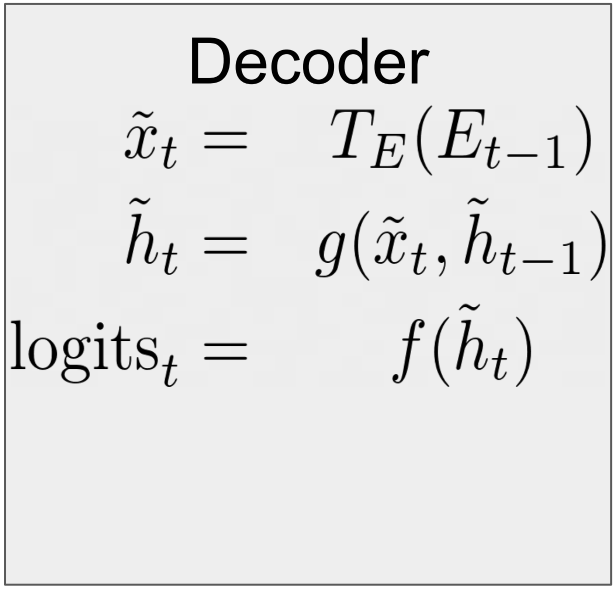
h: last hidden state of RNN [p39]

DecoderWithoutAttention

Decoder

$$
\begin{aligned}
\tilde{x}_t &= T_E(E_{t-1}) \\
\tilde{h}_t &= g(\tilde{x}_t, \tilde{h}_{t-1}) \\
\mathrm{logits}_t &= f(\tilde{h}_t)
\end{aligned}
$$

DecoderWithoutAttention



Decoder

$$\begin{aligned} \tilde{x}_t &= T_E(E_{t-1}) \\ \tilde{h}_t &= g(\tilde{x}_t, \tilde{h}_{t-1}) \\ \text{logits}_t &= f(\tilde{h}_t) \end{aligned}$$

- logits_t is the **un-normalized log probability**, which means the actual probability should be normalized by **softmax**, instead of average.
- `Pr_b(i) = softmax(logits_t[m])`
- [p40]

DecoderWithAttention

Decoder

$$
\begin{aligned}
\tilde{x}_t &= [T_E(E_{t-1}), c_{t-1}] \\
c_{t-1} &= \mathrm{Attend}(h_{t-1}, h_{1:s}) \\
&= \textstyle\sum_s \alpha_{t-1,s} h_s \\
\alpha_{t-1} &= \mathrm{softmax}(e_{t-1,1:s}, s) \\
e_{t-1,s} &= \text{cosine\_similarity}(\tilde{h}_{t-1}, h_s)
\end{aligned}
$$

DecoderWithAttention

Decoder

$$\begin{aligned}
\tilde{x}_t &= [T_E(E_{t-1}), c_{t-1}] \\
c_{t-1} &= \text{Attend}(h_{t-1}, h_{1:s}) \\
&= \textstyle\sum_s \alpha_{t-1,s} h_s \\
\alpha_{t-1} &= \text{softmax}(e_{t-1,1:s}, s) \\
e_{t-1,s} &= \text{cosine\_similarity}(\tilde{h}_{t-1}, h_s)
\end{aligned}$$

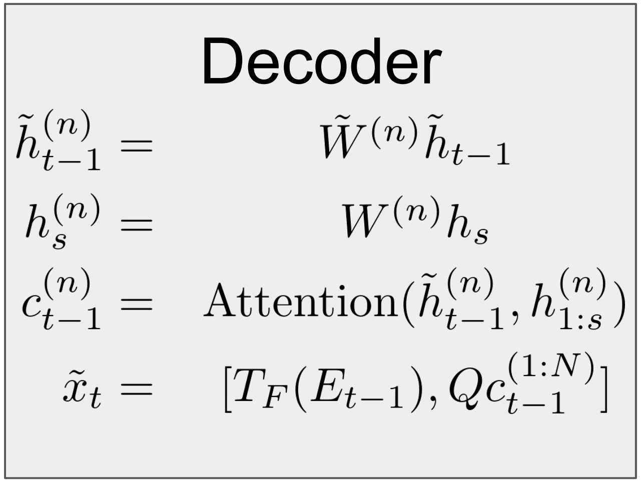
- The function to compute attention weights is provided to you `get_attention_weights`
- Be careful with t vs. t-1!
- For the assignment, we’ll use cosine similarity as the score function (1.4)
- [p43-45]

DecoderWithMultiHeadAttention

Decoder

$$\begin{aligned}
\tilde{h}_{t-1}^{(n)} &= \tilde{W}^{(n)} \tilde{h}_{t-1} \\
h_s^{(n)} &= W^{(n)} h_s \\
c_{t-1}^{(n)} &= \mathrm{Attention}(\tilde{h}_{t-1}^{(n)}, h_{1:s}^{(n)}) \\
\tilde{x}_t &= [T_F(E_{t-1}), Q c_{t-1}^{(1:N)}]
\end{aligned}$$

DecoderWithMultiHeadAttention

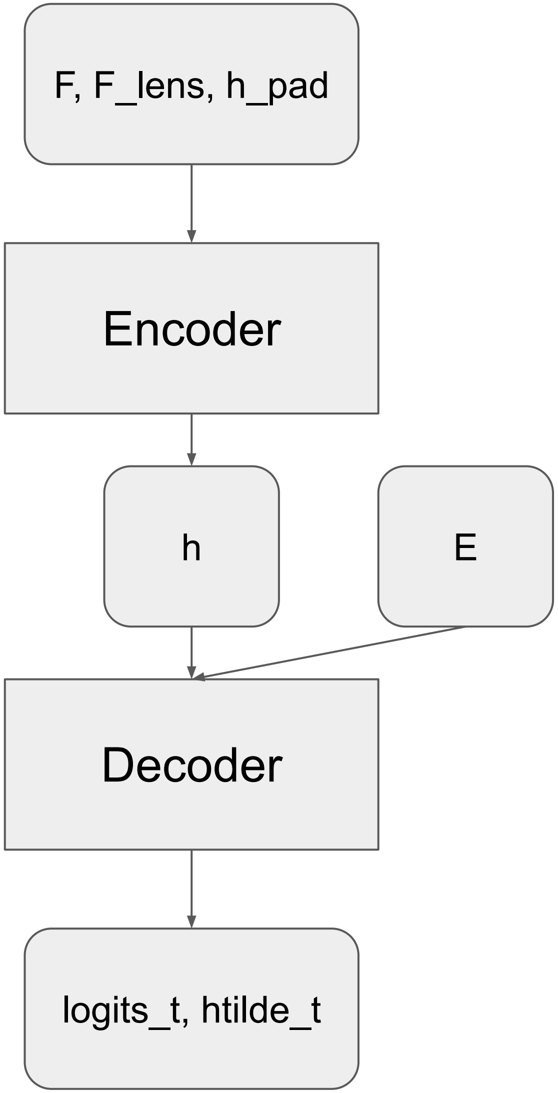


Decoder

$$\begin{aligned}\tilde{h}_{t-1}^{(n)} &= \tilde{W}^{(n)}\tilde{h}_{t-1}\\ h_s^{(n)} &= W^{(n)}h_s\\ c_{t-1}^{(n)} &= \mathrm{Attention}(\tilde{h}_{t-1}^{(n)}, h_{1:s}^{(n)})\\ \tilde{x}_t &= [T_F(E_{t-1}), Qc_{t-1}^{(1:N)}]\end{aligned}$$

- You don’t really need to slice the hidden weights
- Try starting with heads=1
- You also don’t need for loops!
- [p46]

Putting it together: EncoderDecoder



- `init_submodules`
- `get_logits_for_teacher_forcing`
    - Basic idea: replace y with E
    - [p42]
- `update_beam`
    - One step of the beam search
    - A greedy update function is provided to you, you can test the rest of the assignment by using the `--greedy` option
    - More beam search on next week’s tutorial by Zhewei
    - [p55-60]

Training and Testing Loop

- `train_for_epoch`
  - Follow the instructions in the docstring
  - Don’t forget to normalize loss!
  - tqdm: easy progress bar
- `compute_batch_total_bleu`
  - `a2_bleu_score.BLEU_score` for a batch of sentences
- `compute_average_bleu_over_dataset`
  - Calculate the average BLEU score of the given dataset
  - Use `compute_batch_total_bleu`

teach.cs with GPU: srun

- First make sure your code works in cpu mode! Debugging in CUDA mode is much more difficult

- Basic usage: `srun -p csc401 your_regular_command`
  - `srun -p csc2511` if you enrolled in CSC 2511
- Check current queue: `squeue -p csc401`
- Keep training after disconnecting: Use `screen` (A.3)