Lab03
Machine Translation

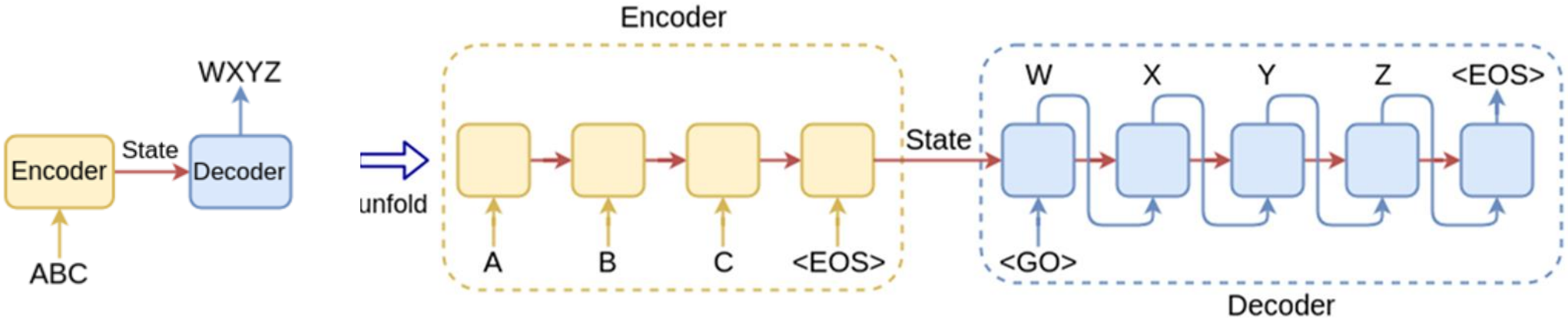
Dataset: English-Chinese Translation

- Tatoeba
  - https://tatoeba.org/zh-cn/
- XDailyDialog
  - https://github.com/liuzeming01/XDailyDialog
- The dataset contains daily-style conversation sentence pairs
- Select 50000 sentence pairs for training and validation
- Select 200 sentence pairs for testing

English	Chinese
Who are you looking for, Tom?	你在找誰,湯姆?
I was a student at that time.	我当时是学生。
What's your real goal?	你真正的目标是什么?
I'm at the beach.	我在海滩。
You can always count on Tom.	你可以永遠信賴湯姆。
...	...

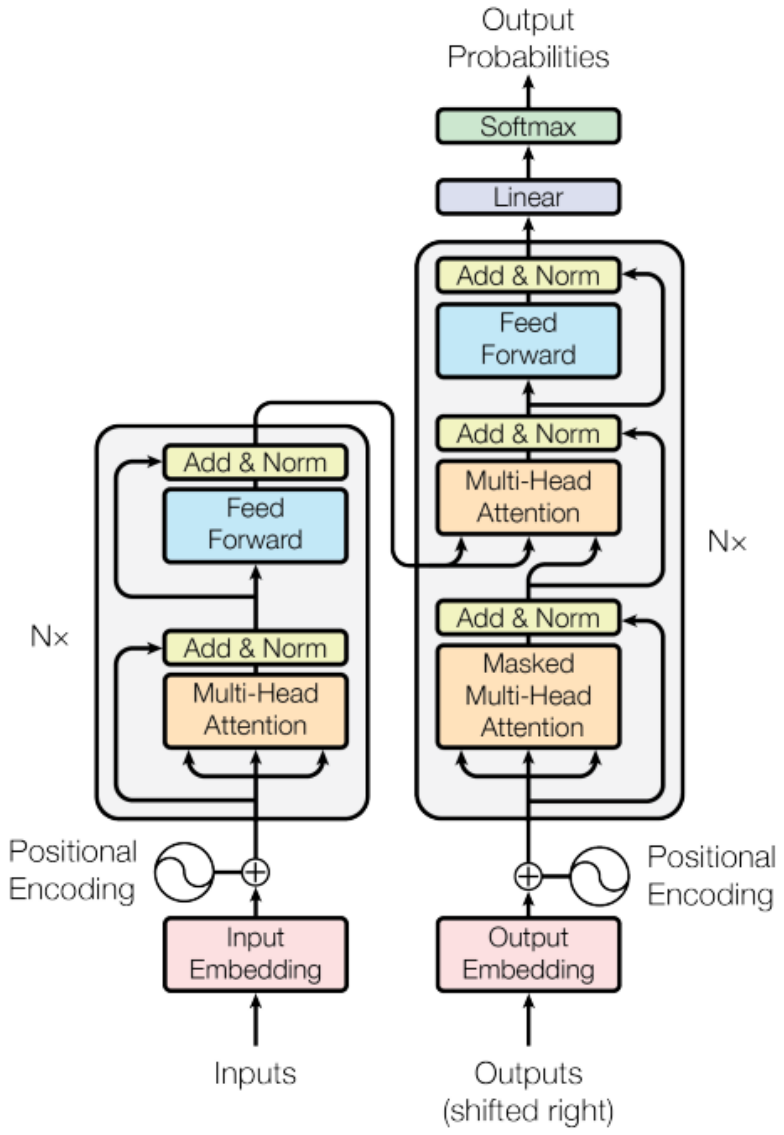
Seq2Seq Model

- Encoder-Decoder structure
- Applications: machine translation, chatbot…



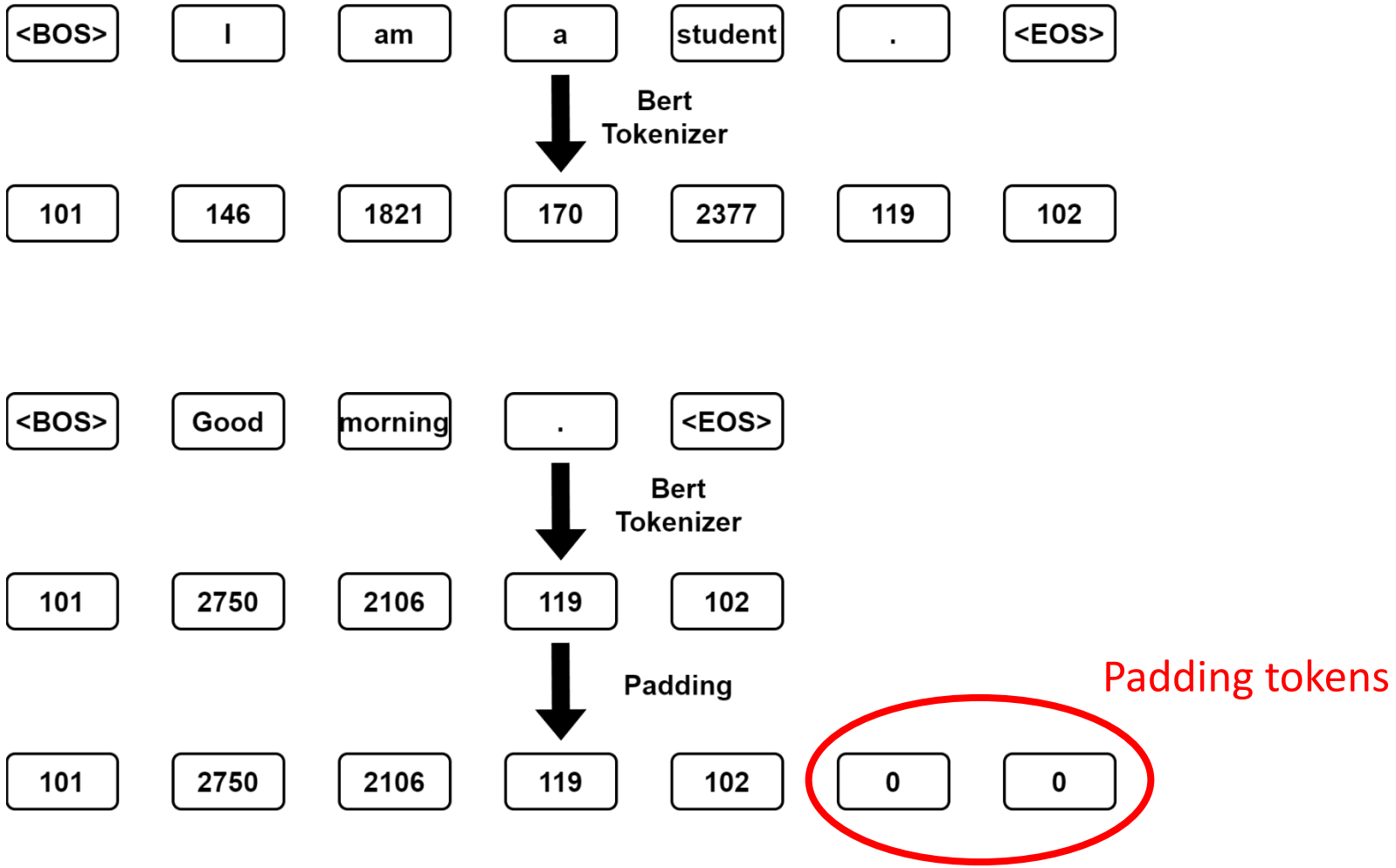
Transformer Model Architecture

- Embedding Layer
- Positional Encoding
- Encoder
  - Multi-head self attention
  - FFN
- Decoder
  - Masked multi-head self attention
  - Multi-head cross attention
  - FFN



Input Paddings

- Make the input sentence equal-length



Masks

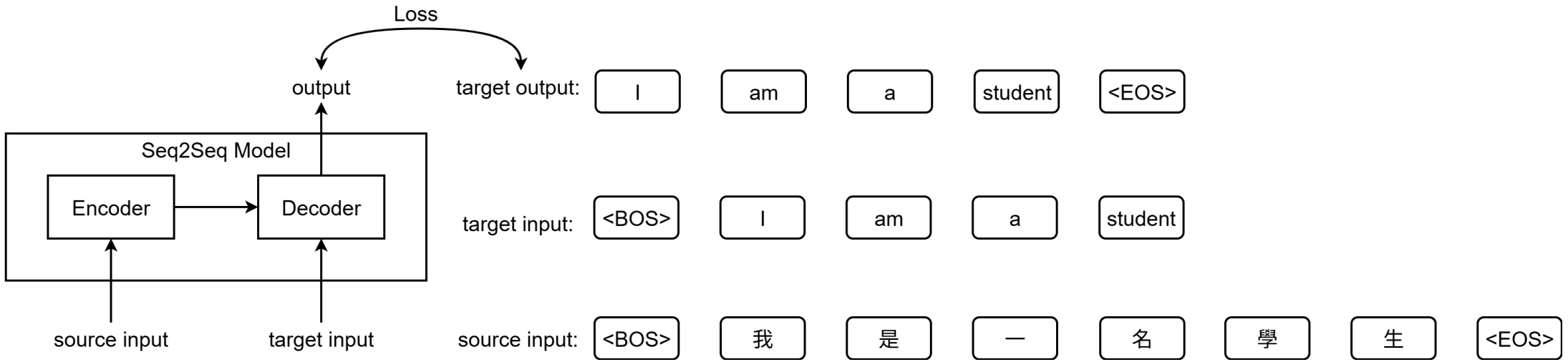
- Masking is directly applied in attention score
- Future mask
  - Used in decoder self attention
  - Prevent from attending to the future tokens
- Padding mask
  - Used in encoder, decoder self attention both
  - Prevent from attending to the padding tokens

	<BOS>	I	am	a	student
<BOS>	−∞	−∞	−∞	−∞	−∞
I		−∞	−∞	−∞	−∞
am			−∞	−∞	−∞
a				−∞	−∞
student					−∞

<BOS>	Good	morning	.	<EOS>	<PAD>	<PAD>
					−∞	−∞

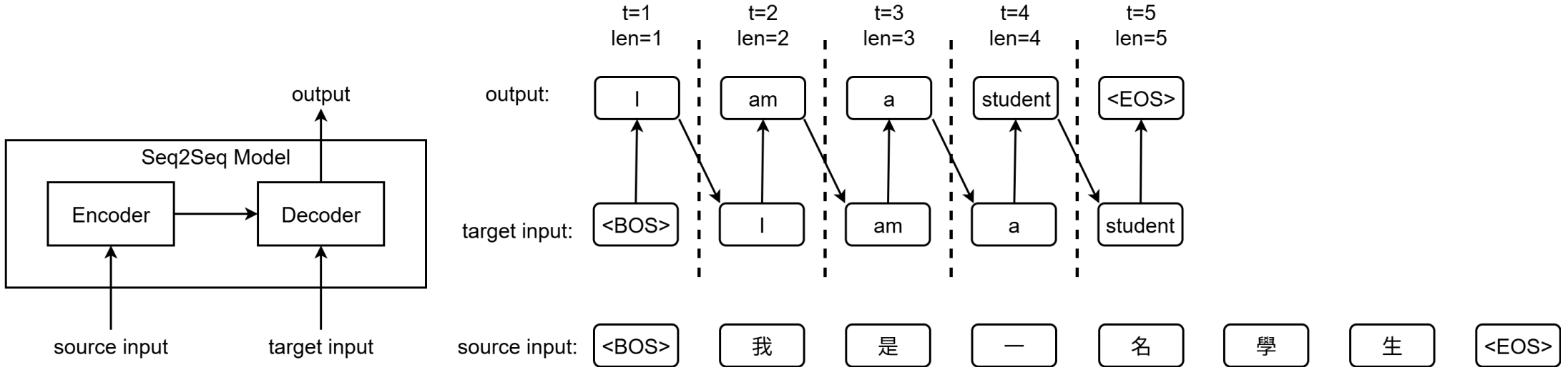
Sequence Generation Algorithm

- During training
  - Model input: source sentence, target sentence
  - Expected model output: target sentence with one token left shift
  - The decoder is trained to predict the next token



Sequence Generation Algorithm

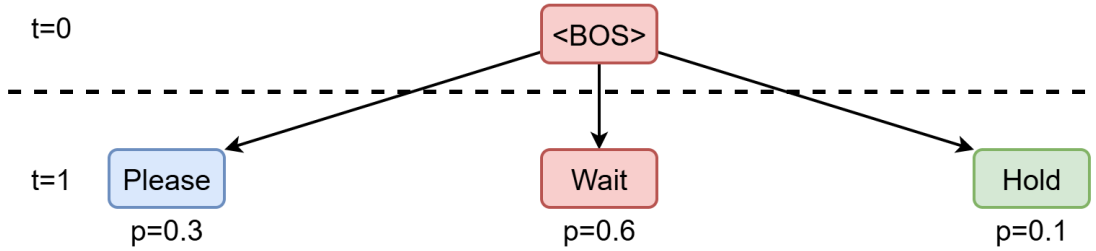
- During inference
  - Generate output tokens autoregressively (the output token depends on its previous output)
  - Greedy search: in every iteration, choose the token with the highest probability as output



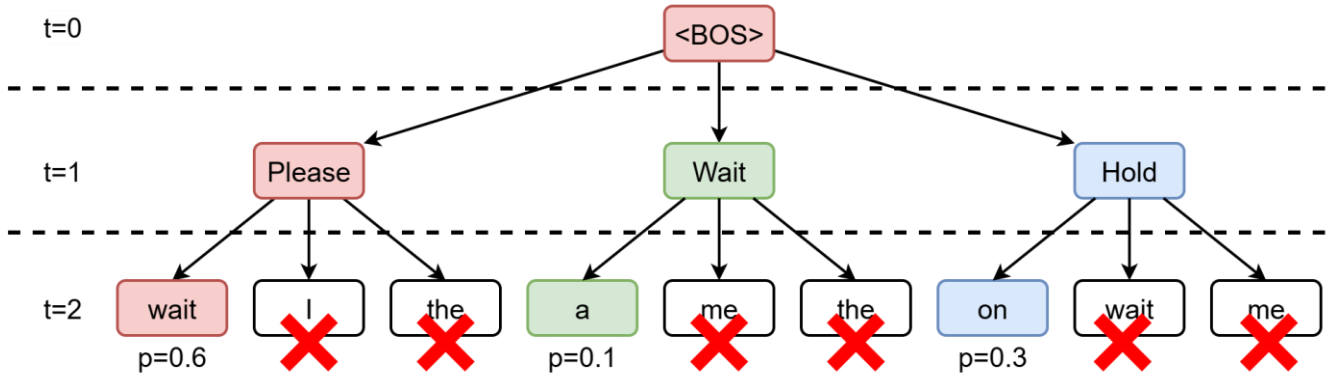
Sequence Generation Algorithm

- During inference
  - Beam search: keep a number of sentences as candidate (top-k highest probability)
  - Using beam search may improve the inference quality, but slow down the inference speed

An example with beam width = 3



Prune the search tree and keep 3 sentences with highest probability



Tasks in this lab

1. In “Lab03.ipynb” and “network.py”
   - Build vanilla transformer (from Attention is All You Need) by yourself
     - https://arxiv.org/abs/1706.03762
   - You should build the encoder, decoder by yourself
   - Inside the encoder/decoder, you should build the attention mechanism (multi-head attention) by yourself

(You can’t call the model directly with the command)

```
self.encoder_layer = nn.TransformerEncoderLayer(d_model, nhead, d_feedforward)
self.encoder = nn.TransformerEncoder(self.encoder_layer, num_enc_layer)
self.decoder_layer = nn.TransformerDecoderLayer(d_model, nhead, d_feedforward)
self.decoder = nn.TransformerDecoder(self.decoder_layer, num_dec_layer)

self.transformer = nn.Transformer(d_model, nhead, num_enc_layer, num_dec_layer, d_feedforward, dropout)

self.attention = nn.MultiheadAttention(emb_dim, nhead)
```

Tasks in this lab

1. In “Lab03.ipynb” and “network.py”
   - The parameter size of model should be less than **100M** !
   - Achieve at least **0.25** 1-gram BLEU score on testing dataset
   - Achieve at least **0.1** 2-gram BLEU score on testing dataset
   - The inference latency (program execution time) should be less than **200s**

Tasks in this lab

1. In “Lab03.ipynb” and “network.py”
   - You can NOT use the following models, TA will check your code!
     - torch.nn.TransformerEncoderLayer
     - torch.nn.TransformerEncoder
     - torch.nn.TransformerDecoderLayer
     - torch.nn.TransformerDecoder
     - torch.nn.Transformer
     - torch.nn.MultiheadAttention

Tasks in this lab

2. Write a report

- Required
  - The structure of Transformer you implement (ex. number of encoder layers, decoder layers)
  - Training strategy (ex. learning rate scheduler)
  - Anything you do to improve the performance.
  - Any difficulty you encounter

- **Plagiarism is forbidden !!!**
- **There should be comment in your code**
- **You should make sure your code is executable**

File List

- translation_train_data.json (50000 sentence pairs)
- translation_test_data.json (200 sentence pairs)
- util.py
  - Include all the used functions
    - BLEU score calculation
    - English/Chinese tokenizer
    - Definition of BOS/EOS/PAD ID
- run.py
  - Run inference of your model. TA will use this code to check parameter size of your model and evaluate performance
- network.py
  - TO-DO: finish Transformer model and translate function here
- Lab3.ipynb
  - TO-DO: alter the training setting and train the model

File List

- In “network.py”: Finish these parts

```python
def load_model(MODEL_PATH=None):
    EMB_SIZE = 1024
    NHEAD = 64
    FFN_HID_DIM = 2048
    NUM_ENCODER_LAYERS = 6
    NUM_DECODER_LAYERS = 6
    SRC_VOCAB_SIZE = tokenizer_chinese().vocab_size
    TGT_VOCAB_SIZE = tokenizer_english().vocab_size

    model = Seq2SeqNetwork(NUM_ENCODER_LAYERS, NUM_DECOD
    if MODEL_PATH is not None:
        model.load_state_dict(torch.load(MODEL_PATH))
    return model
```

```python
def translate(model: torch.nn.Module, src_sentence: str, input_tokenizer, output_tokenizer):
    model.eval()
    sentence = input_tokenizer.encode(src_sentence)
    sentence = torch.tensor(sentence).view(1, -1)
    num_tokens = sentence.shape[1]

    # TO-DO (beam search, greedy search, ...)
    #     sentence shape: (batch_size, seq_length)
    tgt_tokens =

    output_sentence = output_tokenizer.decode(tgt_tokens, skip_special_tokens=True)
    return output_sentence
```

```python
class MultiHeadAttention(nn.Module):
    def __init__(self):
        super().__init__()
        # TO-DO

    def forward(self):
        # TO-DO


class TransformerEncoderLayer(nn.Module):
    def __init__(self):
        super().__init__()
        # TO-DO

    def forward(self):
        # TO-DO


class TransformerDecoderLayer(nn.Module):
    def __init__(self):
        super().__init__()
        # TO-DO

    def forward(self):
        # TO-DO


class Transformer(nn.Module):
    def __init__(self):
        super().__init__()
        # TO-DO

    def forward(self):
        # TO-DO
```

File List

- In “run.py”
  - Input the following command in terminal to check the result
    - python run.py
    - python run.py model.ckpt translation_test_data.json
  - TA will run the above command to check your code and evaluate performance

All the requirement are fulfilled

```
(d12025f) d12025f_ta_4@pc414-71:~/lab3_TA$ python run.py model.ckpt translation_test_data.json
The parameter size of model is 22919.492 k
====================  PASS parameter size requirement  ====================
BLEU score (1-gram) =  0.3294061906635761
BLEU score (2-gram) =  0.18195724342018366
BLEU score (3-gram) =  0.10283244669437408
BLEU score (4-gram) =  0.06065125815570355
====================  PASS BLEU score requirement      ====================
execution time = 5.610s
====================  PASS execution time requirement  ====================
```

The BLEU score requirement FAILs

```
(d12025f) d12025f_ta_4@pc414-71:~/lab3_TA$ python run.py model.ckpt translation_test_data.json
The parameter size of model is 22919.492 k
====================  PASS parameter size requirement  ====================
BLEU score (1-gram) =  0.17155522231012582
BLEU score (2-gram) =  0.06659947019070386
BLEU score (3-gram) =  0.027821140252053737
BLEU score (4-gram) =  0.013770492970943451
====================  FAIL BLEU score requirement      ====================
execution time = 26.534s
====================  PASS execution time requirement  ====================
```

Score

- Four requirements are fulfilled (60%)
  - 1-gram BLEU score on testing data > **0.25**
  - 2-gram BLEU score on testing data > **0.1**
  - Parameter size < **100M**
  - Program execution time (on ED414 workstation or Google Colab) < **200s**
- Report (20%)
- Performance (20%)
  - Performance rank (1-gram BLEU score) on hidden testing data
  - No performance score if the above four requirements are NOT fulfilled

Reminder

- Submit Deadline : 2 week (2025/10/20 23:59 PM)
- Please compress a folder into a zip file named StudentID.zip (example: 313555555.zip) and upload it to new e3
- The folder should include:
  - Lab3.ipynb
  - network.py
  - model.ckpt
  - StudentID_report.pdf (example: 313555555_report.pdf)

HAVE FUN !!!