深度学习实践实验八

PB22151796-莫环欣

- 0. 看看原始数据长什么样?
- 1. 把代码跑通（所有的图都是可以展示的，对于理解 Transformer 非常有用）(含 Debug 和非 Debug)
- 2 试试预训练模型 pretrain/pretrain_model.pt， 与你自己训的对比一下, 实验报告训练集使用 train.txt，验证集使用 dev_mini.txt，展示 dev_mini.txt 中的所有翻译结果对比
- 3. 看一看建立的英文词典和中文词典长什么样，思考一下为什么要有 "UNK" "BOS" "EOS"
- 4. word embedding 后的词长什么样?
- 5. 思考一下，为什么要加 positional_embedding
- 6. 将 encoder layer 的个数设为 4，将 decoder layer 的个数设为 5，展示训练后的 dev_mini.txt 中的所有翻译结果（实验报告训练集使用 train.txt，验证集使用 dev_mini.txt）
- 7.修改课程中的代码，使其完成中英翻译（原代码是英中翻译） ， 展示训练后的 dev_mini.txt 中的所有中英翻译结果（实验报告训练集使用 train.txt，验证集使用 dev_mini.txt）

0. 看看原始数据长什么样?

每一行都分为两部分，一句英文后接一句中文，中间以`\t`分隔（但为什么里面是繁体，刚跑出来第一眼我还以为乱码了）

```
Anyone can do that.	任何人都可以做到。
How about another piece of cake?	要不要再來一塊蛋糕？
She married him.	她嫁给了他。
I don't like learning irregular verbs.	我不喜欢学习不规则动词。
It's a whole new ball game for me.	這對我來說是個全新的球類遊戲。
```

1. 把代码跑通（所有的图都是可以展示的，对于理解 Transformer 非常有用）(含 Debug 和非 Debug)

首先是`DEBUG`配置的，每轮训练后会评估损失，完成训练后会调用`evaluate`对句子进行翻译。这里可以看到在`DEBUG`参数下模型基本无法完成正常的翻译任务，仅作为运行正确性测试。

```
>>>>>>> start train
Epoch 0 Batch: 0 Loss: 7.3113 Tokens per Sec: 4.26s
>>>>> Evaluate
<<<<< Evaluate loss: 7.27
>>>>> current best loss: 7.267736434936523
Epoch 1 Batch: 0 Loss: 7.2172 Tokens per Sec: 4.82s
>>>>> Evaluate
<<<<< Evaluate loss: 7.06
>>>>> current best loss: 7.0575666427612305
<<<<<<< finished train, cost 5.2203 seconds
>>>>>>> start evaluate

BOS how long did you live there ? EOS
BOS 你 住 在 那 裡 多 久 了 ? EOS
translation: 。

BOS it would take me too much time to UNK to you why it 's not going to work . EOS
BOS 给 你 解 UNK 这 为 什 么 行 不 通 要 花 很 多 时 间 。 EOS
translation: 。 。

BOS she put the UNK on the table . EOS
BOS 她 把 UNK UNK 放 在 桌 上 。 EOS
translation: 。 。 。 。 。

BOS i try not to think about it . EOS
BOS 我 試 著 不 去 想 了 。 EOS
translation: 。

BOS tom was there UNK , but not UNK . EOS
BOS 汤 姆 人 在 心 不 在 。 EOS
translation: 。

BOS UNK and boston are sister UNK . EOS
BOS 京 都 和 波 士 UNK 是 姐 妹 城 市 。 EOS
translation: 。 。

BOS please keep this UNK . EOS
BOS 請 保 UNK 這 個 秘 密 。 EOS
translation: 。

BOS i try not to think about it . EOS
BOS 我 試 著 不 去 想 了 。 EOS
translation: 。

BOS any UNK , if it is UNK , is UNK . EOS
BOS 任 何 情 UNK ， 只 要 它 是 真 UNK 的 ， 就 说 明 它 是 发 自 内 心 的 自 然 流 露 。 EOS
translation: 。

BOS UNK , what are you doing here ? EOS
BOS UNK ， 你 在 這 做 什 麼 ？ EOS
translation:
<<<<<<< finished evaluate, cost 3.8000 seconds
```

接着是非`DEBUG`配置的，流程与上面一样，只是增大了训练轮次并调整了嵌入等参数，模型基本可以正常运行了（虽然翻译结果依然很抽象）

```
>>>>>>> start train
Epoch 0 Batch: 0 Loss: 8.1274 Tokens per Sec: 1.37s
Epoch 0 Batch: 50 Loss: 7.1896 Tokens per Sec: 1.71s
Epoch 0 Batch: 100 Loss: 6.6656 Tokens per Sec: 1.65s
Epoch 0 Batch: 150 Loss: 5.7028 Tokens per Sec: 1.52s
Epoch 0 Batch: 200 Loss: 5.3229 Tokens per Sec: 1.52s
>>>>> Evaluate
Epoch 0 Batch: 0 Loss: 5.0825 Tokens per Sec: 1.69s
<<<<< Evaluate loss: 5.08
>>>>> current best loss: 5.081843852996826
Epoch 1 Batch: 0 Loss: 5.3792 Tokens per Sec: 1.46s
Epoch 1 Batch: 50 Loss: 4.9535 Tokens per Sec: 1.54s
Epoch 1 Batch: 100 Loss: 5.1623 Tokens per Sec: 1.57s
Epoch 1 Batch: 150 Loss: 4.6092 Tokens per Sec: 1.52s
Epoch 1 Batch: 200 Loss: 4.5219 Tokens per Sec: 1.52s
>>>>> Evaluate
Epoch 1 Batch: 0 Loss: 4.3276 Tokens per Sec: 1.75s
<<<<< Evaluate loss: 4.28
>>>>> current best loss: 4.276881694793701
Epoch 2 Batch: 0 Loss: 4.6479 Tokens per Sec: 1.49s
Epoch 2 Batch: 50 Loss: 4.0655 Tokens per Sec: 1.51s
Epoch 2 Batch: 100 Loss: 4.5078 Tokens per Sec: 1.51s
Epoch 2 Batch: 150 Loss: 3.9226 Tokens per Sec: 1.51s
Epoch 2 Batch: 200 Loss: 3.8663 Tokens per Sec: 1.50s
>>>>> Evaluate
Epoch 2 Batch: 0 Loss: 3.7474 Tokens per Sec: 1.68s
<<<<< Evaluate loss: 3.68
>>>>> current best loss: 3.6814794540405273
Epoch 3 Batch: 0 Loss: 4.3759 Tokens per Sec: 1.47s
Epoch 3 Batch: 50 Loss: 3.4908 Tokens per Sec: 1.52s
Epoch 3 Batch: 100 Loss: 4.0773 Tokens per Sec: 1.51s
Epoch 3 Batch: 150 Loss: 3.4677 Tokens per Sec: 1.53s
Epoch 3 Batch: 200 Loss: 3.4014 Tokens per Sec: 1.50s
>>>>> Evaluate
Epoch 3 Batch: 0 Loss: 3.2804 Tokens per Sec: 1.67s
<<<<< Evaluate loss: 3.18
>>>>> current best loss: 3.1798243522644043
Epoch 4 Batch: 0 Loss: 3.7337 Tokens per Sec: 1.49s
Epoch 4 Batch: 50 Loss: 3.0131 Tokens per Sec: 1.51s
Epoch 4 Batch: 100 Loss: 3.6760 Tokens per Sec: 1.49s
Epoch 4 Batch: 150 Loss: 3.0339 Tokens per Sec: 1.53s
Epoch 4 Batch: 200 Loss: 2.9347 Tokens per Sec: 1.50s
>>>>> Evaluate
Epoch 4 Batch: 0 Loss: 2.9550 Tokens per Sec: 1.70s
<<<<< Evaluate loss: 2.82
>>>>> current best loss: 2.8212525844573975
Epoch 5 Batch: 0 Loss: 3.4666 Tokens per Sec: 1.48s
Epoch 5 Batch: 50 Loss: 2.6670 Tokens per Sec: 1.51s
Epoch 5 Batch: 100 Loss: 3.3747 Tokens per Sec: 1.49s
Epoch 5 Batch: 150 Loss: 2.6901 Tokens per Sec: 1.50s
Epoch 5 Batch: 200 Loss: 2.5462 Tokens per Sec: 1.49s
>>>>> Evaluate
Epoch 5 Batch: 0 Loss: 2.7515 Tokens per Sec: 1.70s
<<<<< Evaluate loss: 2.51
>>>>> current best loss: 2.5071005821228027
Epoch 6 Batch: 0 Loss: 3.0719 Tokens per Sec: 1.45s
Epoch 6 Batch: 50 Loss: 2.4414 Tokens per Sec: 1.51s
Epoch 6 Batch: 100 Loss: 3.1585 Tokens per Sec: 1.50s
Epoch 6 Batch: 150 Loss: 2.4324 Tokens per Sec: 1.52s
Epoch 6 Batch: 200 Loss: 2.2724 Tokens per Sec: 1.47s
>>>>> Evaluate
Epoch 6 Batch: 0 Loss: 2.4351 Tokens per Sec: 1.69s
<<<<< Evaluate loss: 2.28
>>>>> current best loss: 2.280832290649414
Epoch 7 Batch: 0 Loss: 2.7537 Tokens per Sec: 1.47s
Epoch 7 Batch: 50 Loss: 2.2157 Tokens per Sec: 1.51s
Epoch 7 Batch: 100 Loss: 2.9444 Tokens per Sec: 1.50s
```

```
Epoch 7 Batch: 150 Loss: 2.3608 Tokens per Sec: 1.48s
Epoch 7 Batch: 200 Loss: 2.0874 Tokens per Sec: 1.46s
>>>>> Evaluate
Epoch 7 Batch: 0 Loss: 2.3118 Tokens per Sec: 1.69s
<<<<< Evaluate loss: 2.10
>>>>> current best loss: 2.096855878829956
Epoch 8 Batch: 0 Loss: 2.5307 Tokens per Sec: 1.50s
Epoch 8 Batch: 50 Loss: 2.0776 Tokens per Sec: 1.51s
Epoch 8 Batch: 100 Loss: 2.8054 Tokens per Sec: 1.46s
Epoch 8 Batch: 150 Loss: 2.1444 Tokens per Sec: 1.51s
Epoch 8 Batch: 200 Loss: 1.9694 Tokens per Sec: 1.49s
>>>>> Evaluate
Epoch 8 Batch: 0 Loss: 2.1461 Tokens per Sec: 1.64s
<<<<< Evaluate loss: 1.95
>>>>> current best loss: 1.9517287015914917
Epoch 9 Batch: 0 Loss: 2.2824 Tokens per Sec: 1.50s
Epoch 9 Batch: 50 Loss: 1.8519 Tokens per Sec: 1.49s
Epoch 9 Batch: 100 Loss: 2.6436 Tokens per Sec: 1.49s
Epoch 9 Batch: 150 Loss: 2.0210 Tokens per Sec: 1.51s
Epoch 9 Batch: 200 Loss: 1.7938 Tokens per Sec: 1.49s
>>>>> Evaluate
Epoch 9 Batch: 0 Loss: 1.9590 Tokens per Sec: 1.71s
<<<<< Evaluate loss: 1.78
>>>>> current best loss: 1.7800700664520264
<<<<<<< finished train, cost 1159.2935 seconds
>>>>>>> start evaluate

BOS i 've changed my website 's layout . EOS
BOS 我 改 了 一 下 我 的 网 站 的 布 局 。 EOS
translation: 我 收 到 了 我 的 計 程 車 。

BOS he brought back several UNK . EOS
BOS 他 帶 回 了 一 些 紀 念 品 。 EOS
translation: 他 被 一 個 有 趣 的 外 套 。

BOS he is very angry . EOS
BOS 他 非 常 生 气 。 EOS
translation: 他 很 生 氣 。

BOS how 's the weather in new york ? EOS
BOS 紐 約 的 天 氣 如 何 ? EOS
translation: 纽 约 的 天 气 怎 么 ?

BOS let 's go to a movie . EOS
BOS 讓 我 們 去 看 電 影 。 EOS
translation: 讓 我 們 去 電 影 。

BOS come on ! give me a chance . EOS
BOS 来 嘛 ！ 给 我 个 机 会 。 EOS
translation: 要 把 我 的 提 议 放 在 一 下 一 個 小 時 。

BOS she is a good UNK . EOS
BOS 她 是 一 個 很 好 的 舞 者 。 EOS
translation: 她 是 一 個 好 的 人 。

BOS some of them seem to be too difficult . EOS
BOS 其 中 一 些 似 乎 太 难 了 。 EOS
translation: 他 們 似 乎 很 好 。

BOS my parents usually speak to each other in french , even though my mother is a native english speaker . EOS
BOS 我 父 母 通 常 用 法 语 对 话 ， 即 使 我 母 亲 的 母 语 是 英 语 。 EOS
translation: 我 父 母 通 常 在 法 國 人 投 資 中 文 。

BOS hurry up , or you will miss the train . EOS
BOS 快 点 ， 不 然 你 就 要 错 过 火 车 了 。 EOS
translation: 快 要 用 你 的 火 车 ， 很 好 。
<<<<<<< finished evaluate, cost 10.1476 seconds
```

2 试试预训练模型 pretrain/pretrain_model.pt， 与你自己训的对比一下, 实验报告训练集使用 train.txt，验证集使用 dev_mini.txt，展示 dev_mini.txt 中的所有翻译结果对比

> 其实可以直接用 SentenceTransformer 库和专门的预训练模型，效果会更好（而且如果这样安排，布置实验的时候还可以给文档然后要求调包完全实现一个，感觉比单纯跑通自建 transformer 会更好一些？）

本次训练出来的模型为`save\models\large_model.pt`，预训练模型为`pretrain\pretrain_model.pt`

加载模型后分别使用`evaluate`评估即可

```
# 加载验证集数据
DEV_MINI_FILE = "data/nmt/en-cn/dev_mini.txt"
data = PrepareData(TRAIN_FILE, DEV_MINI_FILE)

# 加载两个模型
pretrained_model = make_model(
    len(data.en_word_dict),
    len(data.cn_word_dict),
    LAYERS,
    D_MODEL,
    D_FF,
    H_NUM,
    DROPOUT,
)
pretrained_model.load_state_dict(
    torch.load("pretrain/pretrain_model.pt", map_location=DEVICE)
)
pretrained_model.to(DEVICE)
pretrained_model.eval()

trained_model = make_model(
    len(data.en_word_dict),
    len(data.cn_word_dict),
    LAYERS,
    D_MODEL,
    D_FF,
    H_NUM,
    DROPOUT,
)
trained_model.load_state_dict(
    torch.load("save/models/large_model.pt", map_location=DEVICE)
)
trained_model.to(DEVICE)
trained_model.eval()

# 使用 evaluate 函数测试预训练模型
print(">>>>> Evaluating Pretrained Model")
evaluate(data, pretrained_model)

# 使用 evaluate 函数测试自己训练的模型
print("\n>>>>> Evaluating Trained Model")
evaluate(data, trained_model)
```

下面是整理后的对比输出，可以看到两个模型的表现都比较抽象。预训练模型的表现相对来说会更好一些（例如第三个，可以把`magazine`翻译为杂志），但是对于非词表词汇（`UNK`）的预测能力极弱（或者说基本没有，毕竟模型比较小）

```
>>>>> Evaluating Pretrained Model

BOS how long did you live there ? EOS
BOS 你 住 在 那 裡 多 久 了 ? EOS
Pretrained: 你 在 那 裡 多 久 了 ?
Mine: 你 住 在 那 裡 多 久 ?

BOS it would take me too much time to explain to you why it 's not going to work . EOS
BOS 给 你 解 释 这 为 什 么 行 不 通 要 花 很 多 时 间 。 EOS
Pretrained: 我 能 为 你 为 什 么 时 候 来 。
Mine: 當 你 為 什 麼 都 可 以 使 用 你 為 什 麼 不 可 以 使 我 也 不 會 做 什 麼 事 。

BOS she put the magazine on the table . EOS
BOS 她 把 雜 誌 放 在 桌 上 。 EOS
Pretrained: 她 把 雜 誌 放 在 桌 上 。
Mine: 她 把 書 放 在 桌 子 上 。

BOS i try not to think about it . EOS
BOS 我 試 著 不 去 想 了 。 EOS
Pretrained: 我 不 试 着 考 虑 。
Mine: 我 不 試 著 。

BOS tom was there UNK , but not UNK . EOS
BOS 汤 姆 人 在 心 不 在 。 EOS
Pretrained: 汤 姆 没 有 从 没 那 么 糗 过 。
Mine: 汤 姆 在 那 里 。

BOS kyoto and boston are sister cities . EOS
BOS 京 都 和 波 士 顿 是 姐 妹 城 市 。 EOS
Pretrained: 京 都 和 波 士 顿 的 男 人 都 和 城 市 。
Mine: 京 都 和 京 都 和 京 都 的 城 市 。

BOS please keep this secret . EOS
BOS 請 保 守 這 個 秘 密 。 EOS
Pretrained: 請 保 守 秘 密 。
Mine: 请 保 密 。

BOS i try not to think about it . EOS
BOS 我 試 著 不 去 想 了 。 EOS
Pretrained: 我 不 试 着 考 虑 。
Mine: 我 不 試 著 。

BOS any UNK , if it is sincere , is UNK . EOS
BOS 任 何 情 绪 ， 只 要 它 是 真 诚 的 ， 就 说 明 它 是 发 自 内 心 的 自 然 流 露 。 EOS
Pretrained: 如 果 有 任 何 情 绪 。
Mine: 如 果 所 有 的 情 ， 是 不 是 这 样 的 任 何 人 都 是 。

BOS hey , what are you doing here ? EOS
BOS 嘿 ， 你 在 這 做 什 麼 ？ EOS
Pretrained: 嘿 ， 你 在 这 儿 干 嘛 ？
Mine: 嘿 ， 你 在 这 里 ？
```

3. 看一看建立的英文词典和中文词典长什么样，思考一下为什么要有 "UNK" "BOS" "EOS"

> 有一点需要说明：代码中定义了 PAD 填充但从未启用，而是在下面的两处方法中默认为了 UNK 的 0 ，并且在 build_dict 中又将填充定义为 PAD 这里将其进行了修正，代码正常运行（虽然说不改也可以跑，但这两个的意义是不一样的）

```
def seq_padding(X, padding=PAD)
```

```
batches.append(Batch(batch_en, batch_cn, pad=PAD))
```

词表中的`UNK`是`UnKown Token`的意思，所有未在词表中出现过的单词都会被标注为`UNK`（避免词表太过庞大，同时减少非常用词的影响（比如在情感二分类任务中按词频排序后建表））；

`BOS`与`EOS`分别是`Beginning of Sentence`和`End of Sentence`的意思，用于标识句子的起止位置，有助于模型理解句子结构

可以通过以下代码查看词表

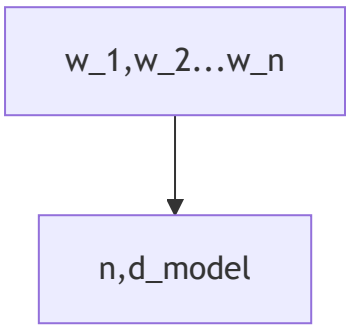
```
data = PrepareData(TRAIN_FILE, DEV_FILE)
src_vocab = len(data.en_word_dict) # 5493  英文词表
tgt_vocab = len(data.cn_word_dict) # 3194  中文词表
print(src_vocab, tgt_vocab)

print(f"src_vocab {data.en_word_dict}")
print(f"tgt_vocab {data.cn_word_dict}")
```

但是词表里有一些翻译感觉非常的迷惑，比如第九条将`a`翻译为`他`、第二十二条`me`翻译为`人`？？？

4. word embedding 后的词长什么样?

这是通过`Embeddings`类实现的，每个词在输入后会被处理为长度为`d_model`的向量，其中包含了词的语义信息




```
class Embeddings(nn.Module):
    def __init__(self, d_model, vocab):
        super(Embeddings, self).__init__()
        self.lut = nn.Embedding(vocab, d_model)
        self.d_model = d_model

    def forward(self, x):
        # return x's embedding vector（times math.sqrt(d_model)）
        return self.lut(x) * math.sqrt(self.d_model)
```

5. 思考一下，为什么要加 positional_embedding

`Transformer`模型的核心机制是自注意力机制，计算方式是基于输入序列中所有词之间的关系，本身对于词的位置信息不敏感，而文本翻译任务需要模型理解文本语义才能很好地进行——词的相对位置对于语义的影响显然非常大。

常用的包括正弦位置编码与旋转位置编码，下面是我在另一门课的实验报告内容

自注意力机制的核心是通过点积计算序列中每个位置的 `Query` 和 `Key` 的相似性，从而生成注意力权重，不关注序列中元素的顺序；位置编码的核心目的是**向模型注入序列中元素的位置信息**，使模型能够感知单词在句子中的相对或绝对位置，就比如说**正弦位置编码**的原理可以如下解释：

- 对于一个长度为 L、维度为 $d*\text{model}$ 的序列，为每个位置 i 生成一个 $d*\text{model}$ 维的向量
- 基于周期性，对偶数维度使用正弦函数、奇数使用余弦，公式为：

$$PE_{(i,2j)} = \sin\left(\frac{i}{10000^{2j/d_{\text{model}}}}\right)$$
$$PE_{(i,2j+1)} = \cos\left(\frac{i}{10000^{2j/d_{\text{model}}}}\right)$$

  - 其中 i 是位置（如第 i 个词），j 是维度索引
  - 这里的 `10000` 与样本数无关，其决定的是不同维度的周期性变化范围
  - 并且位置编码仅与样本长度有关，与样本数无关，而我们上面固定了样本长度，这里也就可以直接固定
- 不同位置的编码在高维空间中具有唯一性，且能保持相对位置关系（如位置 i 和 $i+k$ 的编码差异仅与 k 有关）
- 三角函数的平滑性使模型易于学习和泛化

基于其引入目的，这里将其放置在嵌入层与注意力层之间

对于注意力模块，单头注意力只能捕捉单一视角的依赖关系，而多头机制通过多个头的并行计算，能提取多维度的语义信息（如语法、语义、情感等），最终通过拼接或平均融合这些信息，帮助模型更好地决策

把输入序列 `x` 同时作为 `query` 、 `key` 和 `value`，能够让模型捕捉序列内部不同位置元素之间的相互依赖关系，对于序列中的每个位置，模型都会计算其与其他所有位置的相似度得分，从而确定该位置在生成输出时应该关注哪些其他位置的信息

对于模型的前向传递：

- 每次首先传入批次数量与每批大小 `[batch_size, seq_len]`，接着通过 `embedding` 模块为其附带上维度信息 `[batch_size, seq_len, d_model]`
- 随后经过位置编码层为输入序列附带上位置信息，接着再传入多头注意力层（这里直接调用 `nn.MultiheadAttention` ）
- 得到的张量每批取最后一层（包含所有时间步的信息，由于 mha 还会将序列信息聚合到时间步中，所以这里还附带了序列的全部上下文）输入到分类器（全连接层）中作为分类依据
- 最后再向调用者返回原始分数

除了文档中要求的基本正弦位置编码外，实现了`RoPE`，其关键是对 Q 和 K 向量进行**位置相关的旋转变换**，使相对位置信息嵌入到点积计算中。具体步骤如下：

将每个维度的实向量表示为复数的实部和虚部：

- 对于位置 n 和维度 $2m$（偶数维度），对应复数的虚部为 $\sin(\theta_{m,n})$；
- 维度 $2m+1$（奇数维度）对应实部为 $\cos(\theta_{m,n})$，其中频率 $\theta_{m,n} = n \cdot \omega_m$，$\omega_m = 1/10000^{2m/d_{\text{model}}}$ 是预定义的频率参数。

对于位置 n 和 $n+k$，相对位置为 k。RoPE 通过旋转矩阵对 Q 和 K 进行变换，使得：

- 当计算 Q_n 与 K_{n+k} 的点积时，结果隐式包含 k 的信息。
  数学上，对向量 $v = [v_1, v_2, v_3, v_4, \ldots, v_{d-1}, v_d]$（偶数维度，两两分组），位置 n 的旋转操作可表示为：

$$\text{rotate}(v, n) = \begin{bmatrix} v_1\cos(\theta_m) - v_2\sin(\theta_m) \\ v_1\sin(\theta_m) + v_2\cos(\theta_m) \\ v_3\cos(\theta_{m+1}) - v_4\sin(\theta_{m+1}) \\ v_3\sin(\theta_{m+1}) + v_4\cos(\theta_{m+1}) \\ \vdots \end{bmatrix},$$

  其中每组维度 $(2m, 2m+1)$ 对应频率 $\theta_m = n \cdot \omega_m$

6. 将 encoder layer 的个数设为 4，将 decoder layer 的个数设为 5，展示训练后的 dev_mini.txt 中的所有翻译结果（实验报告训练集使用 train.txt，验证集使用 dev_mini.txt）

这里需要修改`make_model`方法，将原先编码器与解码器的`N`替换掉即可

```
class Encoder(layer: EncoderLayer,N: int)
```

```
def make_model(src_vocab, tgt_vocab, N=6, d_model=512, d_ff=2048, h=8, dropout=0.1):
    encoder_layers = 4
    decoder_layers = 5
    c = copy.deepcopy
    #  Attention
    attn = MultiHeadedAttention(h, d_model).to(DEVICE)
    #  FeedForward
    ff = PositionwiseFeedForward(d_model, d_ff, dropout).to(DEVICE)
    #  Positional Encoding
    position = PositionalEncoding(d_model, dropout).to(DEVICE)
    #  Transformer
    model = Transformer(
        Encoder(
            EncoderLayer(d_model, c(attn), c(ff), dropout).to(DEVICE), encoder_layers
        ).to(DEVICE),
        Decoder(
            DecoderLayer(d_model, c(attn), c(attn), c(ff), dropout).to(DEVICE),
            decoder_layers,
        ).to(DEVICE),
        nn.Sequential(Embeddings(d_model, src_vocab).to(DEVICE), c(position)),
        nn.Sequential(Embeddings(d_model, tgt_vocab).to(DEVICE), c(position)),
        Generator(d_model, tgt_vocab),
    ).to(DEVICE)
```

训练后的结果为：

```
>>>>>>> start evaluate

BOS how long did you live there ? EOS
BOS 你 住 在 那 裡 多 久 了 ? EOS
translation: 你 住 在 那 裡 多 久 了 ?

BOS it would take me too much time to explain to you why it 's not going to work . EOS
BOS 给 你 解 释 这 为 什 么 行 不 通 要 花 很 多 时 间 。 EOS
translation: 它 給 我 無 解 決 的 時 候 要 的 理 解 決 你 什 麼 。

BOS she put the magazine on the table . EOS
BOS 她 把 雜 誌 放 在 桌 上 。 EOS
translation: 她 把 雜 誌 放 在 桌 上 。

BOS i try not to think about it . EOS
BOS 我 試 著 不 去 想 了 。 EOS
translation: 我 不 想 考 试 。

BOS tom was there UNK , but not UNK . EOS
BOS 汤 姆 人 在 心 不 在 。 EOS
translation: 汤 姆 是 没 有 人 。

BOS kyoto and boston are sister cities . EOS
BOS 京 都 和 波 士 顿 是 姐 妹 城 市 。 EOS
translation: 京 都 和 波 士 顿 是 姐 城 市 。

BOS please keep this secret . EOS
BOS 請 保 守 這 個 秘 密 。 EOS
translation: 請 保 守 這 個 秘 密 。

BOS i try not to think about it . EOS
BOS 我 試 著 不 去 想 了 。 EOS
translation: 我 不 想 考 试 。

BOS any UNK , if it is sincere , is UNK . EOS
BOS 任 何 情 绪 ， 只 要 它 是 真 诚 的 ， 就 说 明 它 是 发 自 内 心 的 自 然 流 露 。 EOS
translation: 有 任 何 情 绪 ， 有 ， 只 是 真 诚 的 ， 就 说 发 明 它 是 发 是 发 现 实 的 。

BOS hey , what are you doing here ? EOS
BOS 嘿 ， 你 在 這 做 什 麼 ? EOS
translation: 嘿 ， 你 在 这 儿 做 什 么 ?
<<<<<<< finished evaluate, cost 8.9078 seconds
```

7.修改课程中的代码，使其完成中英翻译（原代码是英中翻译），展示训练后的 dev_mini.txt 中的所有中英翻译结果（实验报告训练集使用 train.txt，验证集使用 dev_mini.txt）

此处只需要稍作修改，将源与目标进行转换即可

首先是`PrepareData`，加上一个`cn2en`参数，随后在其内部的`splitBatch`方法中修改`batch`的添加顺序

```
if not self.cn2en:
    batches.append(Batch(batch_en, batch_cn))
else:
    batches.append(Batch(batch_cn, batch_en))
```

随后在调用`evaluate`方法前将输入数据交换一下即可。

当然，改动最小的方法是仅为`PrepareData`加上参数，在`load_data`方法的最后根据参数情况决定返回的顺序，由于在本次实验代码中对两种语言的处理相同，因此仅需简单交换顺序即可实现任务翻转。这里选用这种方法

```
if not self.cn2en:
    return en, cn
else:
    return cn, en
```

```
CN2EN = True

# Step 1: Data Preprocessing
data = PrepareData(TRAIN_FILE, DEV_FILE, CN2EN)
```

效果如何暂且不论（数据集太小、模型比较简单且没有经过预训练），任务已经可以正常进行并正常输出内容

```
>>>>>>> start evaluate

BOS 汤 姆 人 在 心 不 在 。 EOS
BOS tom was there UNK , but not UNK . EOS
translation: tom is still at all .

BOS 任 何 情 绪 ， 只 要 它 是 真 诚 的 ， 就 说 明 它 是 发 自 内 心 的 自 然 流 露 。 EOS
BOS any UNK , if it is sincere , is UNK . EOS
translation: no matter in things are worse , but it was sincere is it .

BOS 她 把 雜 誌 放 在 桌 上 。 EOS
BOS she put the magazine on the table . EOS
translation: she put on the table .

BOS 你 住 在 那 裡 多 久 了 ? EOS
BOS how long did you live there ? EOS
translation: how long does you live there ?

BOS 京 都 和 波 士 頓 是 姐 妹 城 市 。 EOS
BOS kyoto and boston are sister cities . EOS
translation: kyoto and boston new boston are sister 's sister .

BOS 我 試 著 不 去 想 了 。 EOS
BOS i try not to think about it . EOS
translation: i tried to go .

BOS 子 承 父 业 。 EOS
BOS he UNK for his father . EOS
translation: he was afraid of father .

BOS 你 住 在 那 裡 多 久 了 ? EOS
BOS how long did you live there ? EOS
translation: how long does you live there ?

BOS 给 你 解 释 这 为 什 么 行 不 通 要 花 很 多 时 间 。 EOS
BOS it would take me too much time to explain to you why it 's not going to work . EOS
translation: you 'd better not be much time .

BOS 嘿 ， 你 在 這 做 什 麼 ？ EOS
BOS hey , what are you doing here ? EOS
translation: what 's your best idea ?
<<<<<<< finished evaluate, cost 9.9349 seconds
```

```
>>>>>>> start train
Epoch 0 Batch: 0 Loss: 13.7339 Tokens per Sec: 1.26s
Epoch 0 Batch: 50 Loss: 10.7593 Tokens per Sec: 1.57s
Epoch 0 Batch: 100 Loss: 9.7297 Tokens per Sec: 1.46s
Epoch 0 Batch: 150 Loss: 6.9621 Tokens per Sec: 1.39s
Epoch 0 Batch: 200 Loss: 5.1357 Tokens per Sec: 1.37s
>>>>> Evaluate
<<<<< Evaluate loss: 5.02
>>>>> current best loss: 5.016604423522949
Epoch 1 Batch: 0 Loss: 5.6970 Tokens per Sec: 1.41s
Epoch 1 Batch: 50 Loss: 5.1273 Tokens per Sec: 1.39s
Epoch 1 Batch: 100 Loss: 4.6686 Tokens per Sec: 1.38s
Epoch 1 Batch: 150 Loss: 4.4988 Tokens per Sec: 1.36s
Epoch 1 Batch: 200 Loss: 3.3664 Tokens per Sec: 0.67s
>>>>> Evaluate
<<<<< Evaluate loss: 3.79
>>>>> current best loss: 3.790665864944458
Epoch 2 Batch: 0 Loss: 4.6259 Tokens per Sec: 0.65s
Epoch 2 Batch: 50 Loss: 4.0877 Tokens per Sec: 0.67s
Epoch 2 Batch: 100 Loss: 3.7750 Tokens per Sec: 1.09s
Epoch 2 Batch: 150 Loss: 3.7414 Tokens per Sec: 1.27s
Epoch 2 Batch: 200 Loss: 2.8296 Tokens per Sec: 1.25s
>>>>> Evaluate
<<<<< Evaluate loss: 3.15
>>>>> current best loss: 3.1475038528442383
Epoch 3 Batch: 0 Loss: 4.0991 Tokens per Sec: 1.25s
Epoch 3 Batch: 50 Loss: 3.5601 Tokens per Sec: 1.36s
Epoch 3 Batch: 100 Loss: 3.3057 Tokens per Sec: 1.41s
Epoch 3 Batch: 150 Loss: 3.3516 Tokens per Sec: 1.43s
Epoch 3 Batch: 200 Loss: 2.3561 Tokens per Sec: 1.37s
>>>>> Evaluate
<<<<< Evaluate loss: 2.66
>>>>> current best loss: 2.658276081085205
Epoch 4 Batch: 0 Loss: 3.6511 Tokens per Sec: 1.43s
Epoch 4 Batch: 50 Loss: 3.2215 Tokens per Sec: 1.39s
Epoch 4 Batch: 100 Loss: 2.8592 Tokens per Sec: 1.39s
Epoch 4 Batch: 150 Loss: 2.8913 Tokens per Sec: 1.36s
Epoch 4 Batch: 200 Loss: 2.0212 Tokens per Sec: 1.38s
>>>>> Evaluate
<<<<< Evaluate loss: 2.18
>>>>> current best loss: 2.1802916526794434
Epoch 5 Batch: 0 Loss: 3.3002 Tokens per Sec: 1.40s
Epoch 5 Batch: 50 Loss: 2.7570 Tokens per Sec: 1.39s
Epoch 5 Batch: 100 Loss: 2.4930 Tokens per Sec: 1.38s
Epoch 5 Batch: 150 Loss: 2.5102 Tokens per Sec: 1.37s
Epoch 5 Batch: 200 Loss: 1.9634 Tokens per Sec: 1.34s
>>>>> Evaluate
<<<<< Evaluate loss: 1.69
>>>>> current best loss: 1.6917139291763306
Epoch 6 Batch: 0 Loss: 2.9535 Tokens per Sec: 1.36s
Epoch 6 Batch: 50 Loss: 2.4772 Tokens per Sec: 1.37s
Epoch 6 Batch: 100 Loss: 2.1682 Tokens per Sec: 1.38s
Epoch 6 Batch: 150 Loss: 2.2598 Tokens per Sec: 1.41s
Epoch 6 Batch: 200 Loss: 1.6043 Tokens per Sec: 1.39s
>>>>> Evaluate
<<<<< Evaluate loss: 1.43
>>>>> current best loss: 1.4309656620025635
Epoch 7 Batch: 0 Loss: 2.6941 Tokens per Sec: 1.44s
Epoch 7 Batch: 50 Loss: 2.2800 Tokens per Sec: 1.39s
Epoch 7 Batch: 100 Loss: 1.8968 Tokens per Sec: 1.38s
Epoch 7 Batch: 150 Loss: 2.0463 Tokens per Sec: 1.38s
Epoch 7 Batch: 200 Loss: 1.3255 Tokens per Sec: 1.36s
>>>>> Evaluate
<<<<< Evaluate loss: 1.38
>>>>> current best loss: 1.3774667978286743
Epoch 8 Batch: 0 Loss: 2.5342 Tokens per Sec: 1.42s
Epoch 8 Batch: 50 Loss: 2.1450 Tokens per Sec: 1.40s
```

```
Epoch 8 Batch: 100 Loss: 1.5988 Tokens per Sec: 1.38s
Epoch 8 Batch: 150 Loss: 1.9642 Tokens per Sec: 1.40s
Epoch 8 Batch: 200 Loss: 1.1750 Tokens per Sec: 1.34s
>>>>> Evaluate
<<<<< Evaluate loss: 1.11
>>>>> current best loss: 1.1052043437957764
Epoch 9 Batch: 0 Loss: 2.4406 Tokens per Sec: 1.41s
Epoch 9 Batch: 50 Loss: 1.9476 Tokens per Sec: 1.37s
Epoch 9 Batch: 100 Loss: 1.5137 Tokens per Sec: 1.38s
Epoch 9 Batch: 150 Loss: 1.7719 Tokens per Sec: 1.37s
Epoch 9 Batch: 200 Loss: 1.0388 Tokens per Sec: 1.37s
>>>>> Evaluate
<<<<< Evaluate loss: 0.92
>>>>> current best loss: 0.9183586835861206
<<<<<<< finished train, cost 970.5567 seconds
```