第四次实践课作业-Transformer中英翻译

罗浩铭 PB21030838

题目

修改课程中的代码，使其完成中英翻译（原代码是英中翻译） 同时，将encoder layer的个数设为4，将decoder layer的个数设为5 展示训练后的中英翻译结果

实验内容

实现中英翻译

进行如下修改使得原来的模型变为中英翻译模型：

在`splitBatch`函数中，将`batches.append(Batch(batch_en, batch_cn))`修改为`batches.append(Batch(batch_cn, batch_en))`，这将使得训练集和测试集中的source与target对换，使得训练与测试过程都适配中英翻译。

对换`src_vocab`与`tgt_vocab`如下，这将使模型维度变为适配中英翻译的维度：

```
src_vocab = len(data.cn_word_dict)
tgt_vocab = len(data.en_word_dict)
```

修改`evaluate`函数，使evaluate过程适配中英翻译。具体修改如下：

- 将模型输入数据改为中文：`src = torch.from_numpy(np.array(data.dev_cn[i])).long().to(DEVICE)`
- 输出符号改为英文：`sym = data.en_index_dict[out[0, j].item()]`
- `greedy_decode`函数的`start_symbol`参数改为`data.en_word_dict["BOS"]`（这条其实不影响正确性，主要影响严谨性）
- 调整输出显示的格式，将输出调回输入、预期输出、真实输出的顺序（这条

其实不影响正确性，主要影响严谨性与美观性）

再进行少量注释修改。

完成后的中英翻译测试结果如下：

```
BOS 我 改 了 一 下 我 的 网 站 的 布 局 。 EOS
BOS i 've changed my website 's layout . EOS
translation: i changed the store before it 's play .

BOS 他 帶 回 了 一 些 紀 念 品 。 EOS
BOS he brought back several UNK . EOS
translation: he took a little longer .

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

BOS 紐 約 的 天 氣 如 何 ? EOS
BOS how 's the weather in new york ? EOS
translation: what is the weather like ?

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

BOS 来 嘛 ！ 给 我 个 机 会 。 EOS
BOS come on ! give me a chance . EOS
translation: let me take a telephone .

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

BOS 其 中 一 些 似 乎 太 难 了 。 EOS
BOS some of them seem to be too difficult . EOS
translation: it 's difficult to be very much .

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

BOS 快 点 ， 不 然 你 就 要 错 过 火 车 了 。 EOS
BOS hurry up , or you will miss the train . EOS
translation: you have n't look at the train .
<<<<<<< finished evaluate, cost 7.0191 seconds
```

修改encoder与decoder的个数

我们维持API规范来进行修改

先修改配置信息：

```
N_ENCODER = 4
N_DECODER = 5
```

同时修改`make_model`函数：
先修改函数的参数`N`为`n_encoder, n_decoder`：

```
def make_model(src_vocab, tgt_vocab, n_encoder = N_ENCODER,
      n_decoder = N_DECODER, d_model=512, d_ff=2048, h=8,
dropout=0.1):
```

再将上述参数用于`Encoder`与`Decoder`的调用：

```
Encoder(EncoderLayer(d_model, c(attn), c(ff),
          dropout).to(DEVICE), n_encoder).to(DEVICE)
Decoder(DecoderLayer(d_model, c(attn), c(attn),
           c(ff), dropout).to(DEVICE), n_decoder).to(DEVICE)
```

再按上述API修改`make_model`的调用：

```
# Step 2: Init model
model = make_model(
      src_vocab,
      tgt_vocab,
      N_ENCODER,
      N_DECODER,
      D_MODEL,
      D_FF,
      H_NUM,
      DROPOUT
)
```

由此完成修改。

debug

DEVICE类型判断条件错误

源代码中存在如下的判断语句：

```
if DEVICE == "cuda":
```

但由于`DEVICE = torch.device("cuda")`，因此`DEVICE`的类型

是`torch.device`，而不是`str`，因此上述判断语句永远为`False`，与预期值`True`不符。

因此，需要将上述判断语句修改为：

```
if DEVICE.__str__() == "cuda":
```

附：练习

0. 看看原始数据长什么样?

略

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)

略

1.5 试试预训练模型 save/models/large_model.pt， 与你自己训的对比一下

Baseline翻译测试结果如下：

```
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 用 了 我 的 火 車 站 要 花 了 。

BOS he brought back several UNK . EOS
BOS 他 帶 回 了 一 些 紀 念 品 。 EOS
translation: 他 按 下 了 一 些 大 的 多 少 。

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

BOS how 's the weather in new york ? EOS
BOS 紐 約 的 天 氣 如 何 ? EOS
translation: 纽 约 的 火 車 怎 麼 樣 ?

BOS let 's go to a movie . EOS
BOS 讓 我 們 去 看 電 影 。 EOS
translation: 讓 我 們 一 個 電 影 。

BOS come on ! give me a chance . EOS
BOS 来 嘛 ！ 给 我 个 机 会 。 EOS
translation: 你 在 一 點 上 了 我 一 個 好 的 時 間 。

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 。

BOS some of them seem to be too difficult . EOS
BOS 其 中 一 些 似 乎 太 难 了 。 EOS
translation: 他 們 看 起 来 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 親 法 語 言 ， 但 我 父 母 語 言 成 為 我 父 母 語 言 老 師 。

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 点 你 会 错 过 火 车 。
<<<<<<< finished evaluate, cost 7.1324 seconds
```

预训练模型翻译测试结果如下：

```
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 改 了 我 的 收 音 机 器 。

BOS he brought back several UNK . EOS
BOS 他 帶 回 了 一 些 紀 念 品 。 EOS
translation: 他 到 了 按 鈕 。

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

BOS how 's the weather in new york ? EOS
BOS 紐 約 的 天 氣 如 何 ? EOS
translation: 紐 約 天 氣 怎 麼 長 椅 子 ?

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

BOS come on ! give me a chance . EOS
BOS 来 嘛 ! 给 我 个 机 会 。 EOS
translation: 來 給 我 一 點 了 機 會 。

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

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 對 它 來 說 很 難 。

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 妈 通 常 和 我 妈 法 语 是 一 名 的 母 语 说 英 语 。

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 點 , 你 會 錯 過 一 下 火 車 。
<<<<<<< finished evaluate, cost 8.9434 seconds
```

二者的翻译效果不相上下。

不过，Baseline的best-loss是1.788，预训练模型的best-loss是0.834，预训练模型的loss更低。

2. 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"

建立的英文词典和中文词典如下：

5277, 'bowl': 5278, 'murmur': 5279, 'brook': 5280, 'lulls': 5281, 'tragic': 5282, 'overnight': 5283, 'necessity': 5284, 'religious': 5285, 'abate': 5286, 'senator': 5287, 'censured': 5288, 'congressional': 5289, 'ethics': 5290, 'exists': 5291, 'pochi': 5292, 'span': 5293, 'butterfly': 5294, 'candlelight': 5295, 'mastering': 5296, 'influenced': 5297, 'delays': 5298, 'commanded': 5299, 'fork': 5300, 'umpire': 5301, 'dollar': 5302, 'homes': 5303, 'cardboard': 5304, 'pollen': 5305, 'birmingham': 5306, 'platform': 5307, 'justice': 5308, 'prevail': 5309, 'p.m.': 5310, 'sadly': 5311, 'appeal': 5312, 'madrid': 5313, 'household': 5314, 'patriotism': 5315, 'remote': 5316, 'left-handed': 5317, 'physician': 5318, 'worm': 5319, 'shirtless': 5320, 'awfully': 5321, 'shed': 5322, 'cap': 5323, 'astronaut': 5324, 'toyota': 5325, 'ford': 5326, 'easy-going': 5327, 'fate': 5328, 'imposing': 5329, 'beliefs': 5330, 'cramped': 5331, 'weigh': 5332, 'female': 5333, 'capitals': 5334, 'publication': 5335, 'timed': 5336, 'coincide': 5337, 'seventy': 5338, 'misses': 5339, 'ambiguities': 5340, 'senses': 5341, 'slamming': 5342, 'thoughtful': 5343, 'microwave': 5344, 'comics': 5345, 'toothpaste': 5346, 'engagement': 5347, 'madman': 5348, 'frequent': 5349, 'progressing': 5350, 'abstained': 5351, 'invite': 5352, 'shoebox': 5353, 'capable': 5354, '40': 5355, 'scarf': 5356, 'interrupted': 5357, 'jet': 5358, 'landed': 5359, 'fourth': 5360, 'rats': 5361, 'simply': 5362, 'divine': 5363, 'spilt': 5364, 'wasting': 5365, 'underwater': 5366, 'survivors': 5367, 'topic': 5368, 'poker': 5369, 'drum': 5370, 'self-service': 5371, 'in-laws': 5372, 'bridges': 5373, 'ten-thirty': 5374, 'narrowly': 5375, 'handing': 5376, 'blocked': 5377, 'landslide': 5378, 'fireplace': 5379, 'developed': 5380, 'tool': 5381, 'drugstore': 5382, 'brazen': 5383, 'defiance': 5384, 'perform': 5385, 'educated': 5386, 'stroganoff': 5387, 'containers': 5388, 'airtight': 5389, 'universities': 5390, 'fast-moving': 5391, 'policemen': 5392, 'intruded': 5393, '1,600': 5394, 'retiring': 5395, '19th': 5396, 'loaves': 5397, 'alice': 5398, 'bible': 5399, 'priest': 5400, 'timetable': 5401, 'unlock': 5402, 'bribes': 5403, 'highway': 5404, '58': 5405, 'admired': 5406, 'non-smoking': 5407, 'violated': 5408, 'sour': 5409, 'stocks': 5410, 'germanic': 5411, 'lad': 5412, 'stack': 5413, 'political': 5414, 'career': 5415, 'capture': 5416, 'over-sleeping': 5417, 'spared': 5418, 'pains': 5419, 'annoyed': 5420, 'noises': 5421, 'rugby': 5422, 'differs': 5423, 'washington': 5424, 'drowsy': 5425, 'awarded': 5426, 'online': 5427, 'utterly': 5428, 'joints': 5429, 'extremes': 5430, 'satisfactory': 5431, 'accomplished': 5432, 'deposit': 5433, 'extreme': 5434, 'remind': 5435, 'nagasaki': 5436, 'youngsters': 5437, 'thirty-one': 5438, 'liking': 5439, 'strict': 5440, 'catching': 5441, 'cycling': 5442, 'short-sleeved': 5443, 'shirts': 5444, 'sailing': 5445, 'weekends': 5446, 'confidential': 5447, 'conceal': 5448, 'disappointment': 5449, 'proceed': 5450, 'agenda': 5451, '56': 5452, 'russell': 5453, 'welfare': 5454, 'sewage': 5455, 'pollutes': 5456, 'loser': 5457, 'statues': 5458, 'stopover': 5459, 'warrior': 5460, 'weakness': 5461, 'stage': 5462, 'roller': 5463, 'skates': 5464, 'occurred': 5465, 'deciding': 5466, 'sleeve': 5467, 'accomplish': 5468, 'spilling': 5469, 'phrase': 5470, 'wealthy': 5471, 'modest': 5472, 'hawk': 5473, 'mouse': 5474, 'ceases': 5475, 'parrot': 5476, 'pet': 5477, 'cupcakes': 5478, 'ties': 5479, 'boarding': 5480, 'execution': 5481, 'eleventh': 5482, 'artist': 5483, 'calculated': 5484, 'opposition': 5485, 'springs': 5486, 'schoolroom': 5487, 'absence': 5488, 'fonder': 5489, 'field': 5490, 'educational': 5491, 'foster': 5492, 'UNK': 0, 'PAD': 1}
{'BOS': 2, 'EOS': 3, ' ': 4, '我': 5, '的': 6, '了': 7, '你': 8, '他': 9, '是': 10, '一': 11, '在': 12, '不': 13, '?': 14, '有': 15, '她': 16, '這': 17, '很': 18, '姆': 19, ',': 20, '要': 21, '人': 22, '天': 23, '們': 24, '個': 25, '去': 26, '到': 27, '这': 28, '汤': 29, '上': 30, '们': 31, '个': 32, '什': 33, '好': 34, '想': 35, '那': 36, '嗎': 37, '子': 38, '能': 39, '么': 40, '看': 41, '得': 42, '做': 43, '會': 44, '多': 45, '下': 46, '生': 47, '吗': 48, '以': 49, '裡': 50, '大': 51, '来': 52, '可': 53, '麼': 54, '会': 55, '道': 56, '都': 57, '来': 58, '没': 59, '喜': 60, '把': 61, '家': 62, '知': 63, '?': 64, '事': 65, '就': 66, '起': 67, '说': 68, '本': 69, '作': 70, '沒': 71, '时': 72, '为': 73, '常': 74, '里': 75, '車': 76, '為': 77, '意': 78, '打': 79, '年': 80, '和': 81, '時': 82, '地': 83, '自': 84, '說': 85, '吃': 86, '用': 87, '成': 88, '出': 89, '工': 90, '日': 91, '些': 92, '法': 93, '孩': 94, '給': 95, '學': 96, '明': 97, '晚': 98, '點': 99, '前': 100, '今': 101, '最': 102, '英': 103, '開': 104, '小': 105, '中': 106, '信': 107, '請': 108, '学': 109, '回': 110, '哪': 111, '候': 112, '行': 113, '真': 114, '所': 115, '歡': 116, '房': 117, '對': 118, '告': 119, '它': 120, '跟': 121, '给': 122, '過': 123, '点': 124, '被': 125, '比': 126, '车': 127, '每': 128, '語': 129, '正': 130, '昨': 131, '早': 132, '何': 133, '欢': 134, '書': 135, '相': 136, '果': 137, '过': 138, '父': 139, '高': 140, '必': 141, '太': 142, '着': 143, '快': 144, '对': 145, '儿': 146, '心': 147, '公': 148, '您': 149, '如': 150, '再': 151, '手': 152, '开': 153, '已': 154, '住': 155, '该': 156, '非': 157, '美': 158, '湯': 159, '球': 160, '任': 161, '定': 162, '丽': 163, '感': 164, '該': 165, '0': 166, '花': 167, '十': 168, '走': 169, '聞': 170, 'o': 171, '怎': 172, '名': 173, '西': 174, '玛': 175, '女': 176, '让': 177, '望': 178, '讓': 179, '經': 180, '友': 181, '老': 182, '面': 183, '间': 184, '语': 185, '話': 186, '发': 187, '需': 188, '完': 189, '见': 190, '著': 191, '问': 192, '话': 193, '火': 194, '次': 195, '應': 196, '親': 197, '只': 198, '男': 199, '母': 200, '力': 201, '书': 202, '見': 203, '己': 204, '也': 205, '现': 206, '期': 207, '電': 208, '問': 209, '物': 210, '认': 211, '朋': 212, '加': 213, '了': 214, '还': 215, '士': 216, '從': 217, '三': 218, '之': 219, '同': 220, '!': 221, '照': 222, '現': 223, '音': 224, 'm': 225, '样': 226, '受': 227, '找': 228, '新': 229, '外': 230, '方': 231, '国': 232, '理': 233, '分': 234, '機': 235, '少': 236, '但': 237, '等': 238, '水': 239, '雨': 240, '放': 241, '从': 242, '發': 243, '始': 244, '字': 245, '更': 246, '解': 247, '校': 248, '國': 249, '游': 250, '東': 251, '情': 252, '希': 253, '星': 254, '長': 255, '吧': 256, '全': 257, '像': 258, '幫': 259, '觉': 260, '经': 261, '認': 262, '应': 263, '聽': 264, '直': 265, '後': 266, '狗': 267, '待': 268, '电': 269, '重': 270, '於': 271, '蕉': 272, '别': 273, '餐': 274, '睡': 275, '喝': 276, '请': 277, '午': 278, '文': 279, '1': 280, '服': 281, '玩': 282, '钱': 283, '月': 284,

其为python的`dict`对象，key为词，value为词的编号。其中，`UNK`表示未知词，编号为0；`PAD`表示填充词，编号为1；其余词为数据集分词后出现过的词，按出现频率从高到低编号为2~N。两个词典中，`BOS`表示句子的开始，编号均为2；`EOS`表示句子的结束，编号均为3。

`UNK`、`PAD`、`BOS`、`EOS`的作用如下：

- `UNK`：未知词，当输入不在训练集里时，模型会遇到未知词时，这时需要将其替换为`UNK`，以使得模型可以正常运行。
- `PAD`：填充词，句子长度小于模型接收的最大token长度时，用于填充剩余部分。
- `BOS`：句子开始，`BOS`用于提示模型开始翻译一个句子。
- `EOS`：句子结束，`EOS`用于提示模型结束翻译一个句子。

3. word embedding后的词长什么样?

word embedding由`Embeddings`类实现，代码如下：

```
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_mode
l))
        return self.lut(x) * math.sqrt(self.d_model)
```

它使用了一个`nn.Embedding`对象，类似于查找表，输入每个词的编号即可得到每个该词的embedding，其为一个`d_model = D_MODEL = 256`维的向量。

因此，word embedding后每个词都是一个`d_model = D_MODEL = 256`维的向量。

4. 思考一下，为什么要加positional_embedding

这是为了让模型能够获得词的位置信息。因为Transformer架构中输入的各个位置是同时进行处理，在微结构上都是对等的，互相之间没有顺序关系，所以需要额外的方式来让模型学习到词的位置信息。

附：修改的代码

实验中进行的所有代码修改如下：

```
@@ -64,7 +64,8 @@
64  64
65  65  if DEBUG:
66  66      EPOCHS = 2
67      -   LAYERS = 3
        67  +   N_ENCODER = 3
        68  +   N_DECODER = 3
68  69      H_NUM = 8
69  70      D_MODEL = 128
70  71      D_FF = 256
@@ -75,7 +76,8 @@
75  76      SAVE_FILE = 'save/models/model.pt'
76  77  else:
77  78      EPOCHS = 10
78      -   LAYERS = 6
        79  +   N_ENCODER = 4
        80  +   N_DECODER = 5
79  81      H_NUM = 8
80  82      D_MODEL = 256
81  83      D_FF = 1024
@@ -201,7 +203,7 @@ def splitBatch(self, en, cn, batch_size, shuffle=True):
201 203              # paddings: batch, batch_size, batch_MaxLength
202 204              batch_cn = seq_padding(batch_cn)
203 205              batch_en = seq_padding(batch_en)
204     -           batches.append(Batch(batch_en, batch_cn))
        206 +           batches.append(Batch(batch_cn, batch_en))
205 207              #!!! 'Batch' Class is called here but defined in later section.
206 208          return batches
207 209
```

```
510     - def make_model(src_vocab, tgt_vocab, N=6, d_model=512, d_ff=2048, h=8, dropout=0.1):
        512 + def make_model(src_vocab, tgt_vocab, n_encoder = N_ENCODER,
        513 +     n_decoder = N_DECODER, d_model=512, d_ff=2048, h=8, dropout=0.1):
511 514      c = copy.deepcopy
512 515      #  Attention
513 516      attn = MultiHeadedAttention(h, d_model).to(DEVICE)
@@ -518,9 +521,9 @@ def make_model(src_vocab, tgt_vocab, N=6, d_model=512, d_ff=2048, h=8, dropout=0
518 521      #  Transformer
519 522      model = Transformer(
520 523          Encoder(EncoderLayer(d_model, c(attn), c(ff),
521     -                            dropout).to(DEVICE), N).to(DEVICE),
        524 +                            dropout).to(DEVICE), n_encoder).to(DEVICE),
522 525          Decoder(DecoderLayer(d_model, c(attn), c(attn),
523     -                            c(ff), dropout).to(DEVICE), N).to(DEVICE),
        526 +                            c(ff), dropout).to(DEVICE), n_decoder).to(DEVICE),
524 527          nn.Sequential(Embeddings(d_model, src_vocab).to(DEVICE), c(position)),
525 528          nn.Sequential(Embeddings(d_model, tgt_vocab).to(DEVICE), c(position)),
526 529          Generator(d_model, tgt_vocab)).to(DEVICE)
@@ -699,8 +702,8 @@ def train(data, model, criterion, optimizer):
699 702
700 703  # Step 1: Data Preprocessing
701 704  data = PrepareData(TRAIN_FILE, DEV_FILE)
702     - src_vocab = len(data.en_word_dict)
703     - tgt_vocab = len(data.cn_word_dict)
        705 + src_vocab = len(data.cn_word_dict)
        706 + tgt_vocab = len(data.en_word_dict)
704 707  print(src_vocab, tgt_vocab)
705 708
706 709  print(f"src_vocab {src_vocab}")
```

```
@@ -713,7 +716,8 @@ def train(data, model, criterion, optimizer):
713 716  model = make_model(
714 717      src_vocab,
715 718      tgt_vocab,
716     -   LAYERS,
        719 +   N_ENCODER,
        720 +   N_DECODER,
717 721      D_MODEL,
718 722      D_FF,
719 723      H_NUM,
```

```
//1  //5          for i in np.random.randint(len(data.dev_en), size=10):
772               -         # Print English sentence
773               -         en_sent = " ".join([data.en_index_dict[w] for w in data.dev_en[i]])
774               -         print("\n" + en_sent)
775               -
776               -         # Print Target Chinese sentence
      776         +         # Print Chinese sentence
777   777                   cn_sent = " ".join([data.cn_index_dict[w] for w in data.dev_cn[i]])
778               -         print("".join(cn_sent))
      778         +         print("\n" + "".join(cn_sent))
      779         +
      780         +         # Print Target English sentence
      781         +         en_sent = " ".join([data.en_index_dict[w] for w in data.dev_en[i]])
      782         +         print(en_sent)
779   783
780               -         # conver English to tensor
781               -         src = torch.from_numpy(np.array(data.dev_en[i])).long().to(DEVICE)
      784         +         # conver Chinese to tensor
      785         +         src = torch.from_numpy(np.array(data.dev_cn[i])).long().to(DEVICE)
782   786                   src = src.unsqueeze(0)
783   787                   # set attention mask
784   788                   src_mask = (src != 0).unsqueeze(-2)
785   789                   # apply model to decode, make prediction
786   790                   out = greedy_decode(
787               -             model, src, src_mask, max_len=MAX_LENGTH, start_symbol=data.cn_word_dict["BOS"])
      791         +             model, src, src_mask, max_len=MAX_LENGTH, start_symbol=data.en_word_dict["BOS"])
788   792                   # save all in the translation list
789   793                   translation = []
790   794                   # convert id to Chinese, skip 'BOS' 0.
791   795                   # 遍历翻译输出字符的下标（注意：跳过开始符"BOS"的索引 0）
792   796                   for j in range(1, out.size(1)):
793               -             sym = data.cn_index_dict[out[0, j].item()]
      797         +             sym = data.en_index_dict[out[0, j].item()]
794   798                       if sym != 'EOS':
```