记录我踩过的坑

1. 有关转码错误

   本实验所使用的数据集是我随便在网上搜到的翻译数据集（英文法文互译）

   刚开始还是想做英文中文互译，但是考虑到中文的构词和英文不同（对中文来说tokenizer是个问题，例“北京大学”如何断词？“北京/大学”or“北京大学”？但这种现象不会再英语和法语中出现，为了方便还是选择了这两种外文。。。）

   数据集的格式如下（英文法文同行）：

   ```
   Go! Va!
   ```

   为了之后模型的数据提取，我做了一个预处理（prework.py），思路很简单就是把一行中的两句分开放在不同的文件中

   ```
   # en.txt
   Go!
   # ft.txt
   Va!
   ```

   但是我在分句的时候发现了如下现象

   ```
   ['Va!', 'Cours\u202f!', 'Courez\u202f!',
   ```

   查询资料后发现\u202f属于CSS的编码方式（代表窄的无中断空格），显然我不能让他出现在word_dict中！如何删掉呢？

   尝试：*# -*- coding: utf-8 -*-*（没什么用）

   后来找资料看到：

新知识get：import re用来处理正则表达式

```
re.sub允许使用函数对匹配项的替换进行复杂的处理。
sub()方法提供一个替换值，可以是字符串或一个函数，和一个要被处理的字符串
```

```
sentences[1] = re.sub(u'\u202f', "", sentences[1])
```

虽然就一行代码，但我还是耗了会时间，刚开始想转码，后来气急败坏想直接删除，但又不想写遍历（感觉有点蠢，还是想搜api保证执行速度以及代码美观。。。）

2. 有关RNN

   使用自己不熟悉的api的时候一定要先看文档！！！

   自己想当然的认为RNN的参数是（batch_size, sentence_len, class_num），但其实：

Inputs: input, h_0

- **input**: tensor of shape (L, N, H_{in}) when `batch_first=False` or (N, L, H_{in}) when `batch_first=True` containing the features of the input sequence. The input can also be a packed variable length sequence. See `torch.nn.utils.rnn.pack_padded_sequence()` or `torch.nn.utils.rnn.pack_sequence()` for details.
- **h_0**: tensor of shape $(D * \text{num\_layers}, N, H_{out})$ containing the initial hidden state for each element in the batch. Defaults to zeros if not provided.

where:

$$
\begin{aligned}
N &= \text{batch size} \\
L &= \text{sequence length} \\
D &= 2 \text{ if bidirectional=True otherwise } 1 \\
H_{in} &= \text{input\_size} \\
H_{out} &= \text{hidden\_size}
\end{aligned}
$$

因此我们的input在输入RNN前需要transpose（这个问题很好解决，debug直接维数报错）

另外提一句，也可以不transpose，设置RNN参数 **batch_first=True** 即可

3. 如何实现forward

首先我们要知道encode和decode的思路是不一样的：encode阶段我们可以看到整个的句子信息，decode的阶段我们只能输入一个ground truth预测一个词。因此我们首先要确定代码框架

```
# encode
...
for i in range(len(sen)):
    # decode
    ...
```

为了保险起见，~~其实就是懒~~，我没有像transformer源码那样使用mask来遮盖词。显然非矩阵话的处理速度是比较慢的，对于数据集小的训练还是可以的。

另外，原论文中实现encode时使用了双向的RNN，而且特征是拼接在一起的，而不是相加。为了使encode得到的全局context（2*n_hidden）和针对每次输入的decode（n_hidden），我在原论文的基础上加了一层linear。

还是想吐槽：大牛写论文好随性：

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i)$$

告诉你输入输出，自己想对应维度。

4. 有关维度错误以及维度处理

其实这里是我报错最多的地方，但是整理总结是弄不完的，只能以后多写写代码把维数处理的那些函数用的熟练一些（cat、view、bmm...）

一些不足之处

1. 我在拆分数据集为batch的时候没有考虑到数据总数和batch_size的整除关系。具体地说：我在model.py中将所有的矩阵变换都用parser中设定的batch_size来做。这样虽然可读性好，但是当最后一个batch的数据集中的数据条数不足一个batch_size的时候，forward过程中维数就出错了。最后发现整个模型框架已经定下来了，不好改了。所以取下策，若最后一个batch不足batch_size，则删除该batch。个人感觉这个应该不会对训练结果产生太多的影响，但是日后写代码能避免还是避免。
2. 有关padding：为了使不同的句子能够放在一个batch中按矩阵处理，我将所有句子都用标志进行扩充。但是我们在计算loss的时候显然不想要计算这些的loss。但是感觉如果记录每一个句子的长度，然后根据不同的sentence_len用循环计算一个句子的loss有点浪费资源。在查找如何处理padding的问题时找到了解决方法：**CrossEntropyLoss(ignore_index)**

```
ref: https://blog.csdn.net/weixin_42287851/article/details/99419883
```

简单实验后发现：我们可以不计算ignore_index指定的ground truth相应的loss。具体地说

```
ignore_index = word2id['<pad>']
```

但是我想试验一下，计算和不计算的实验结果差距多大？很遗憾的是：没多大。可能是我 使用的数据集不够大，也可能是我模型有些地方出了问题。总之，之后有机会我还想针对该问题试验。

3. 最后就是没有GPU！！！根本不敢跑大数据集，运行代码整个图书馆都能听到我电脑响。开始想在云端服务器上做实验，但是之前在colab上买的会员到期了。~~老师如果能报销的话我就传到云端做（狗头保命~~