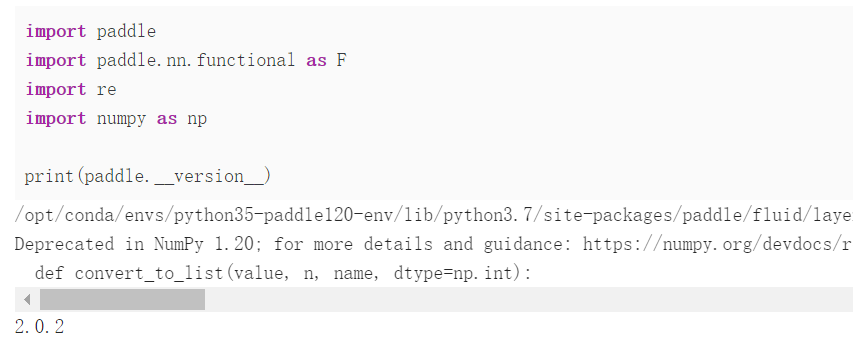
**2022自然语言处理大作业**

**李兆祺2019200166**

环境设置

使用百度飞桨paddlepaddle开源



**构建双语句对的数据结构**

接下来通过处理文本文件，将双语句对读入到python的数据结构中。这里做了如下的处理。

对于日文，按照分词切分。

对于中文，使用jieba做了切分。

为了后续的程序运行的更快，限制句子长度，得到了一个较小的数据集。

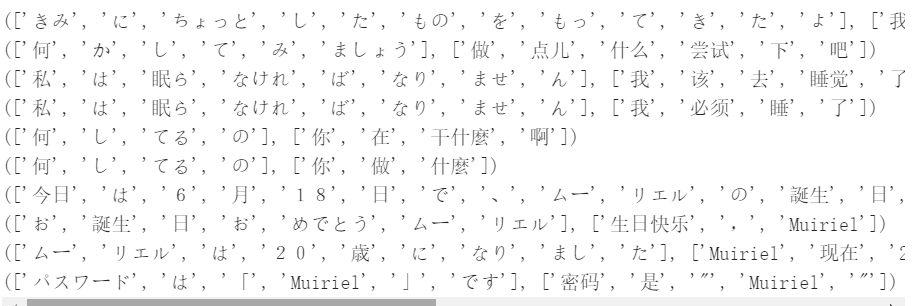
创建词表

接下来分别创建中日文的词表，这两份词表会用来将日文和中文的句子转换为词的ID构成的序列。词表中还加入了如下三个特殊的词：

<pad>: 用来对较短的句子进行填充。

<bos>: "begin of sentence"， 表示句子的开始的特殊词。

<eos>: "end of sentence"， 表示句子的结束的特殊词。



**创建padding过的数据集**

接下来根据词表，将会创建一份实际的用于训练的用numpy array组织起来的数据集。

所有的句子都通过<pad>补充成为了长度相同的句子。

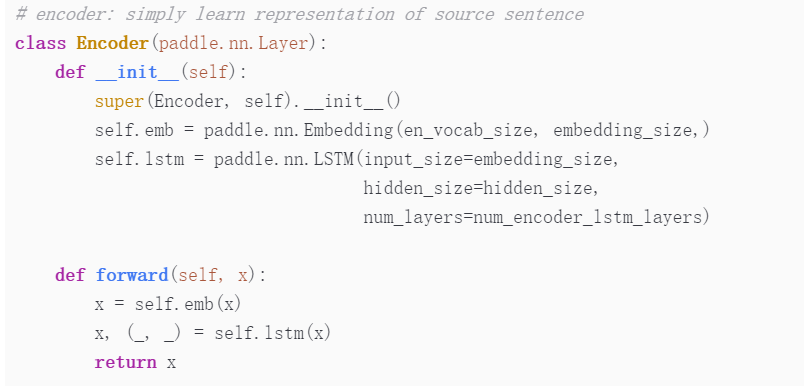
对于日文句子（源语言），将其反转了过来，这会带来更好的翻译的效果。

所创建的padded\_cn\_label\_sents是训练过程中的预测的目标，即，每个中文的当前词去预测下一个词是什么词。

**Encoder部分**

在编码器的部分，通过查找完Embedding之后接一个LSTM的方式构建一个对源语言编码的网络。飞桨的RNN系列的API，除了LSTM之外，还提供了SimleRNN, GRU供使用，同时，还可以使用反向RNN，双向RNN，多层RNN等形式。也可以通过dropout参数设置是否对多层RNN的中间层进行dropout处理，来防止过拟合。

除了使用序列到序列的RNN操作之外，也可以通过SimpleRNN, GRUCell, LSTMCell等API更灵活的创建单步的RNN计算，甚至通过继承RNNCellBase来实现自己的RNN计算单元。



**AttentionDecoder部分**

在解码器部分，通过一个带有注意力机制的LSTM来完成解码。

单步的LSTM：在解码器的实现的部分，同样使用LSTM，与Encoder部分不同的是，下面的代码，每次只让LSTM往前计算一次。整体的recurrent部分，是在训练循环内完成的。

注意力机制：这里使用了一个由两个Linear组成的网络来完成注意力机制的计算，它用来计算出目标语言在每次翻译一个词的时候，需要对源语言当中的每个词需要赋予多少的权重。

**训练模型**

接下来开始训练模型。

在每个epoch开始之前，对训练数据进行了随机打乱。

通过多次调用atten\_decoder，在这里实现了解码时的recurrent循环。

teacher forcing策略: 在每次解码下一个词时，给定了训练数据当中的真实词作为了预测下一个词时的输入。

a5b49618242ac2a4c881b05927a3432

