**04.28.2020**

**实现原理：**

数据加载，对联数据集的特点是输入与输出等长。

构造字典：为了将文字变成计算机能处理的id，我们需要构建一个字典。

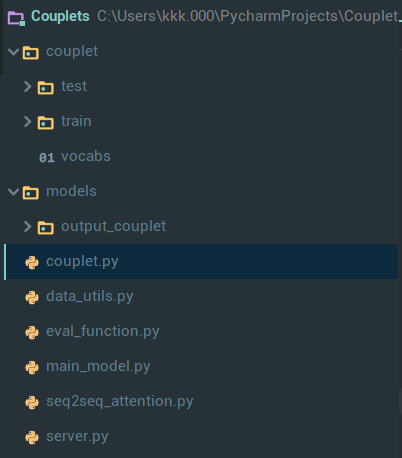
数据预处理：控制输出长度，使其与输入数据等长，方便交给模型处理。

模型构建：构建seq2seq模型，将encode的编码作为decode的输入。

模型训练：模型构建完成后，开始训练模型。模型预测：测试学习效果。

简单说就是一个翻译模型，通过使用深度神经网络将一个语言序列翻译成另一种语言序列。

**预设项目结构：**



couplet 文件夹用于存放一些数据（训练数据和测试数据，以及词表数据）

models 文件夹用于存放训练阶段保存的模型

couplet.py 训练模型保存到models

seq2seq\_attention 带有attantion机制的seq2seq模型

data\_utils 数据处理

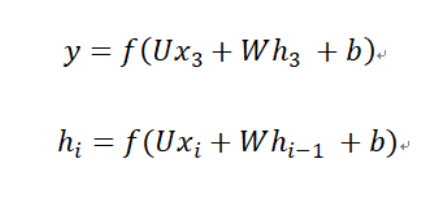
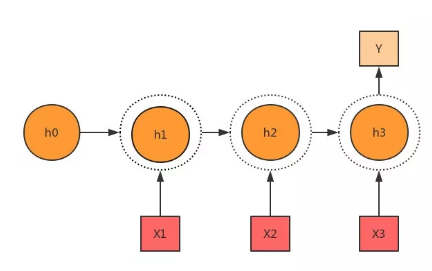
eval\_function bleu计算，用于评估训练阶段的模型效果

main 定义了model类型，里面包括了模型的输入，输出，重载保存，训练等对模型的操作，是整个项目的重点

Server 项目原型

**循环神经网络RNN基本概念理解**

RNN可以用先前的状态预测后来的状态，能够记忆先前的状态，适合用来解决具有上下文关系的算法问题。

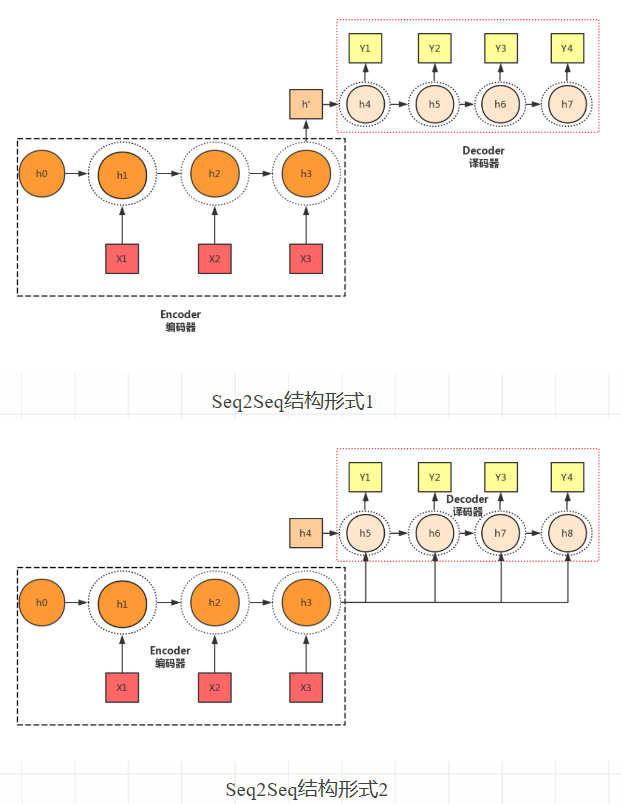


由结构和公式可知，整一个RNN结构共享1组（*U*, ***W***, ***b***），这是RNN结构**最重要的特性**，且每一个隐藏层神经元h的计算公式是由**当前输入X**与**上一个隐藏层神经元的输出**组成。

RNN的记忆是长记忆，一开始的记忆会一直保持到最后.利用BP算法训练网络时容易出现梯度消失的问题，当序列很长的时候问题尤其严重，因此RNN模型一般不能直接应用。

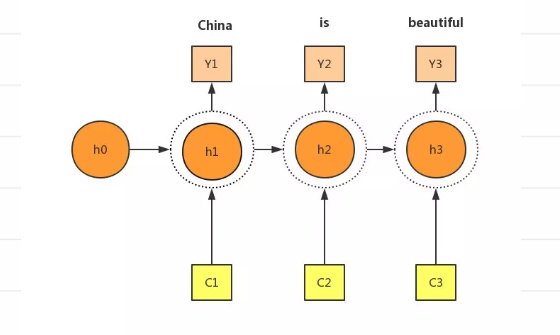
**05.03.2020**

**输入输出不等长的多输入多输出的RNN结构——**Seq2Seq**模型**

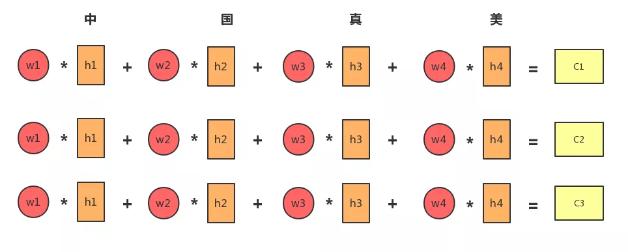


解码器decoder的输入都是译码器encoder的同一个输出，也就是说不过输入的语句是什么，编码器encoder都会将它转换成同一个中间语义h'。

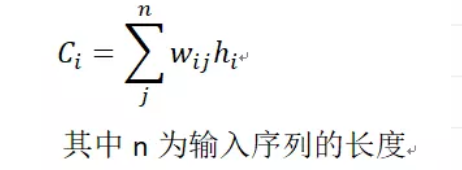
注意力机制下的Seq2Seq模型：



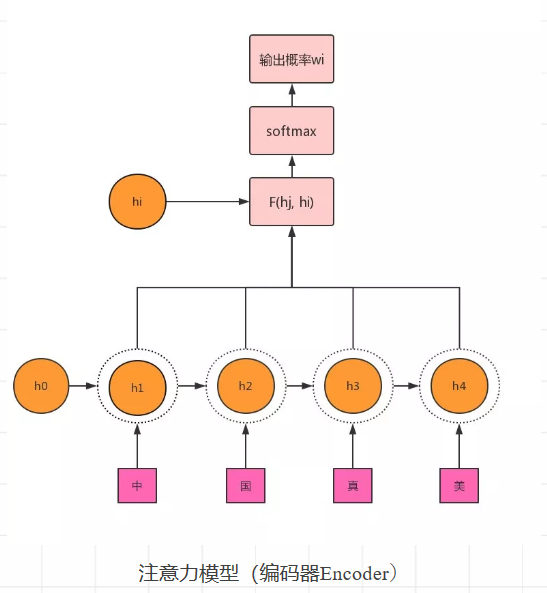
输入不是直接的序列输入，而是经过编码器encoder转换的中间语义C，而这些输入C也各不相同，每一个C都是由**权重w**和译码器的**隐藏层输出h**加权组成，如下图。



在解码器decoder部分，由于输入的顺序不同，中间语义C1，C2，C3之间的权值表征也不同，这也就是我们所说的注意力机制。换言之，随着不同时刻，重点一直在变化，而这些变化则由上图的权重w去表示，比如C1的重点在‘中’这个字，那么中间语义可以表示为C1=0.6 \* h1 + 0.2 \* h2 + 0.1 \* h3 + 0.1 \* h4 (权值可以看成概率，且概率加起来为1)。



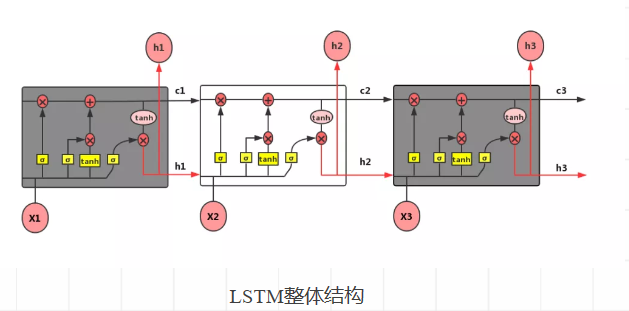
此时问题就变成了如何去求中间语义C的权值w表征。这就涉及到注意力模型的另一部分（译码器Encoder），如下图。F函数和softmax函数，大家可以理解为我们要计算当前的hi与全部h（包括hi）之间的差别，从而计算出在i时刻下，每一个h对应的权值（即概率）。换言之，大家可以将下图看成分类问题，与hi越相近的，输出的概率也就越大。



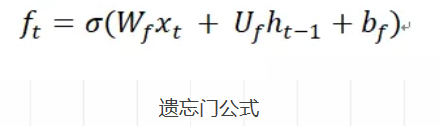
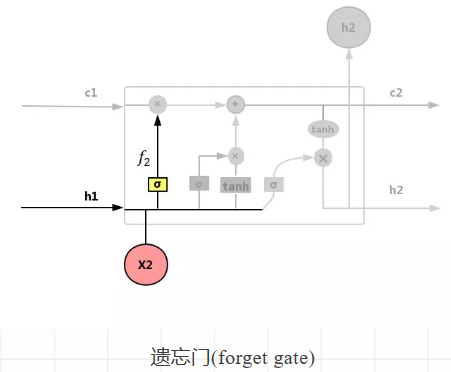
**5.06.2020**

**RNN特殊类型——LSTM**

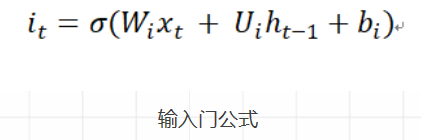
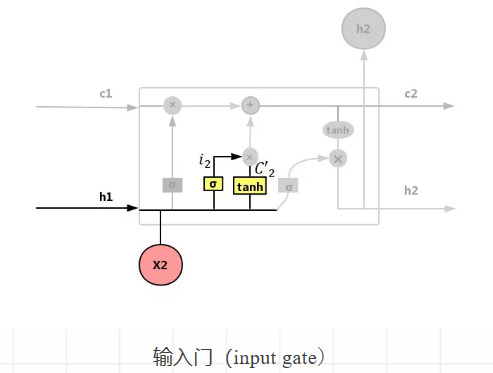
LSTM对有价值的信息进行记忆，放弃冗余记忆，从而减小学习难度。与RNN相比，LSTM的神经元还是基于**输入X**和**上一级的隐藏层输出h**来计算，只不过**内部结构**变了，也就是神经元的运算公式变了，而外部结构并没有任何变化，因此上面提及的RNN各种结构都能用LSTM来替换。



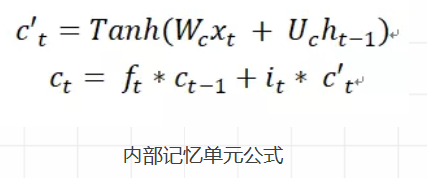
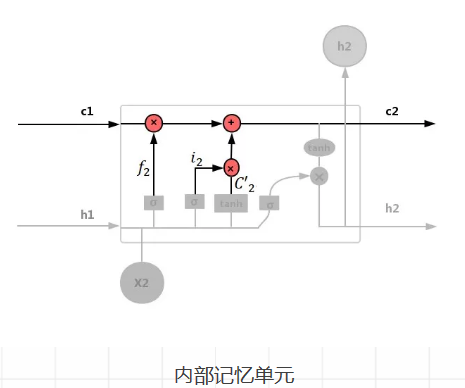
遗忘门f：控制输入X和上一层隐藏层输出h被遗忘的程度大小



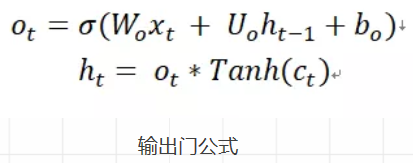
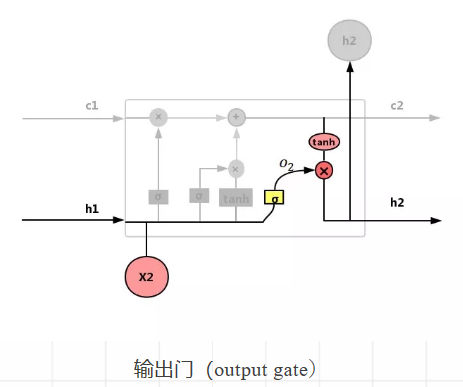
输入门i：控制输入X和当前计算的状态更新到记忆单元的程度大小。



内部记忆单元c：



输出门o：控制输入X和当前输出取决于当前记忆单元的程度大小



 对于一个训练好的LSTM模型，我们要知道它的每一个门（遗忘门、输出门和输入门）都有各自的(***U***, ***W***, ***b***)，上述公式也有所体现，这是在训练过程中得到的。而且当输入的序列不存在有用信息时，遗忘门f的值就会接近1，那么输入门i的值接近0，这样过去有用的信息就会被保存。当输入的序列存在重要信息时，遗忘门f的值就会接近0，那么输入门i的值接近1，此时LSTM模型遗忘过去的记忆，记录重要记忆。

**5.10.2020**

**BLEU得分**

一种**文本评估算法**，它是用来评估**机器翻译**跟**专业人工翻译**之间的对应关系，核心思想就是**机器翻译越接近专业人工翻译，质量就越好**，经过bleu算法得出的分数可以作为机器翻译质量的其中一个指标。使用bleu的目的是给出一个快且不差的自动评估解决方案，评估的是机器翻译与人翻译的接近程度。

核心在于：**N-gram（n元词组匹配度）**和**惩罚因子。**

****通过例子进行说明：****

**机器翻译：The cat sat on the mat.**

**人工翻译：The cat is on the mat.**

**1、计算各gram的精度（一般最多取到4-gram）**

**P1 = 5 / 6 = 0.833333333333333**

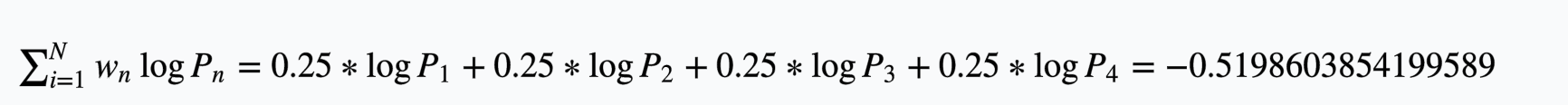
**P2 = 3 / 5 = 0.6**

**P3 = 1 / 4 = 0.25**

**P4 = 0 / 3 = 0**

**2、加权求和**

**取权重：Wn = 1 / 4 = 0.25**



**有0的项不做计算**

**3、求BP**

**这里c=r，则BP=1**

**4、求BLEU**

****优点****：方便、快速，结果比较接近人类评分。

**缺点：**不考虑语言表达（语法）上的准确性；

测评精度会受常用词的干扰；

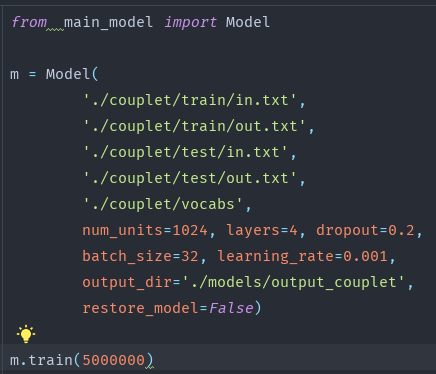
短译句的测评精度有时会较高；

没有考虑同义词或相似表达的情况，可能会导致合理翻译被否定；

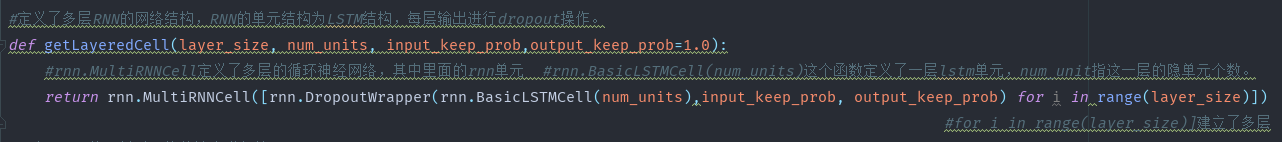
**关键代码理解**

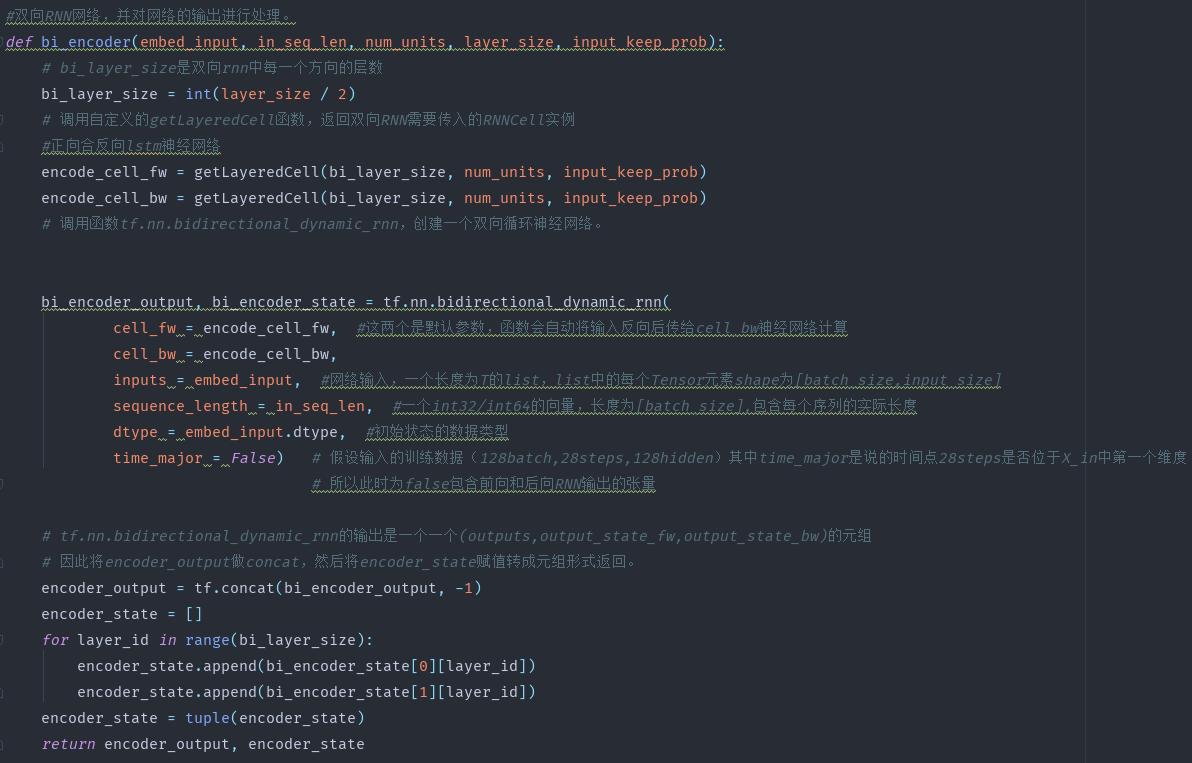
**5.15.2020**

**couplet训练模型,训练结果保存到models：**



**seq2seq模型的构造：**

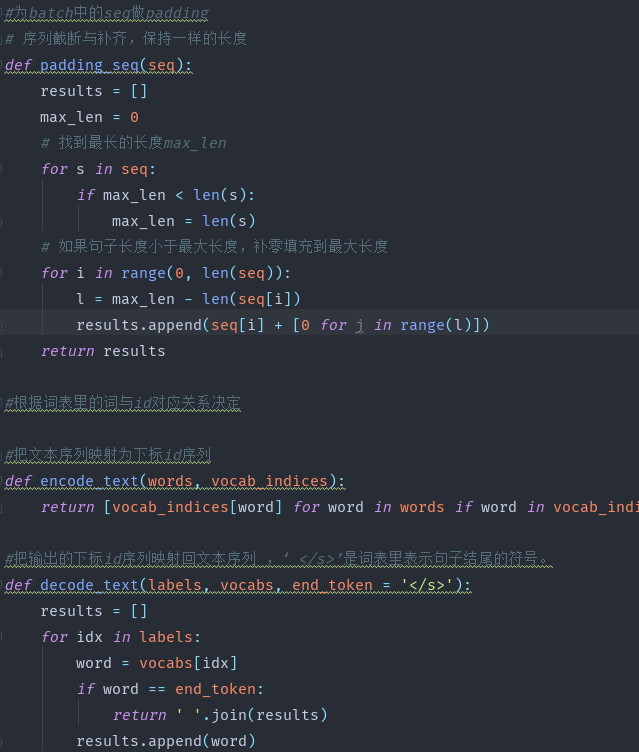


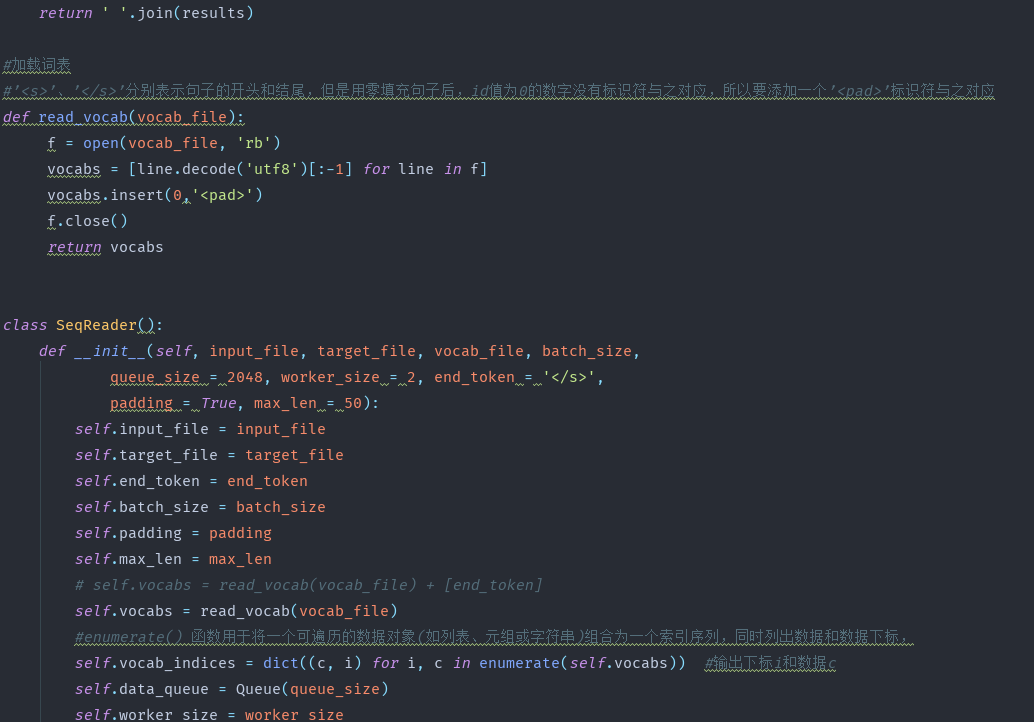




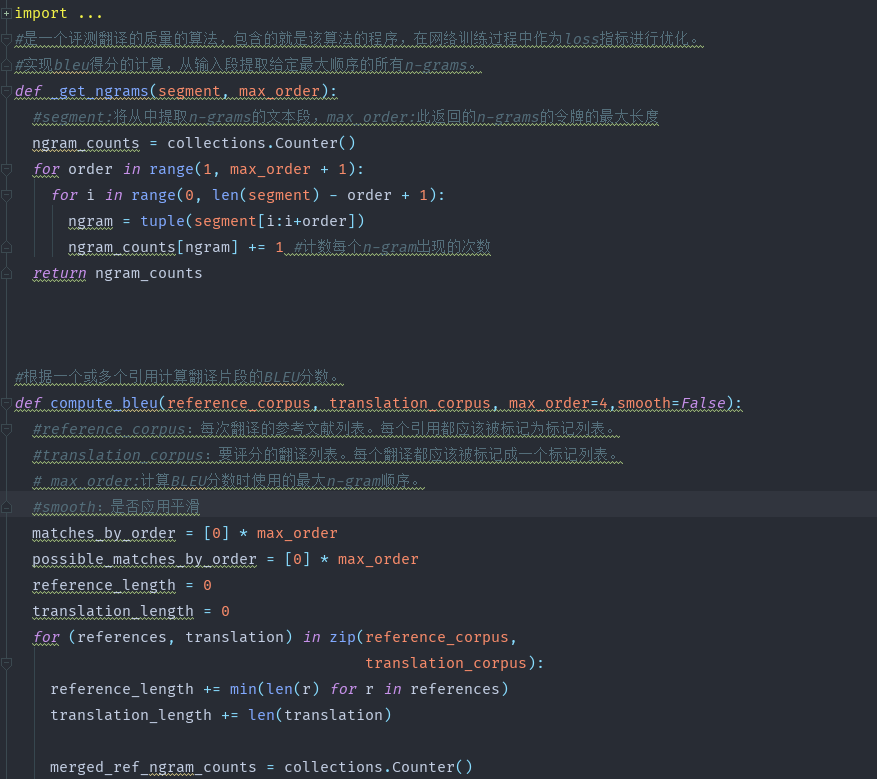
**5.19.2020**

**data\_utils数据处理：**





**bleu算法：**

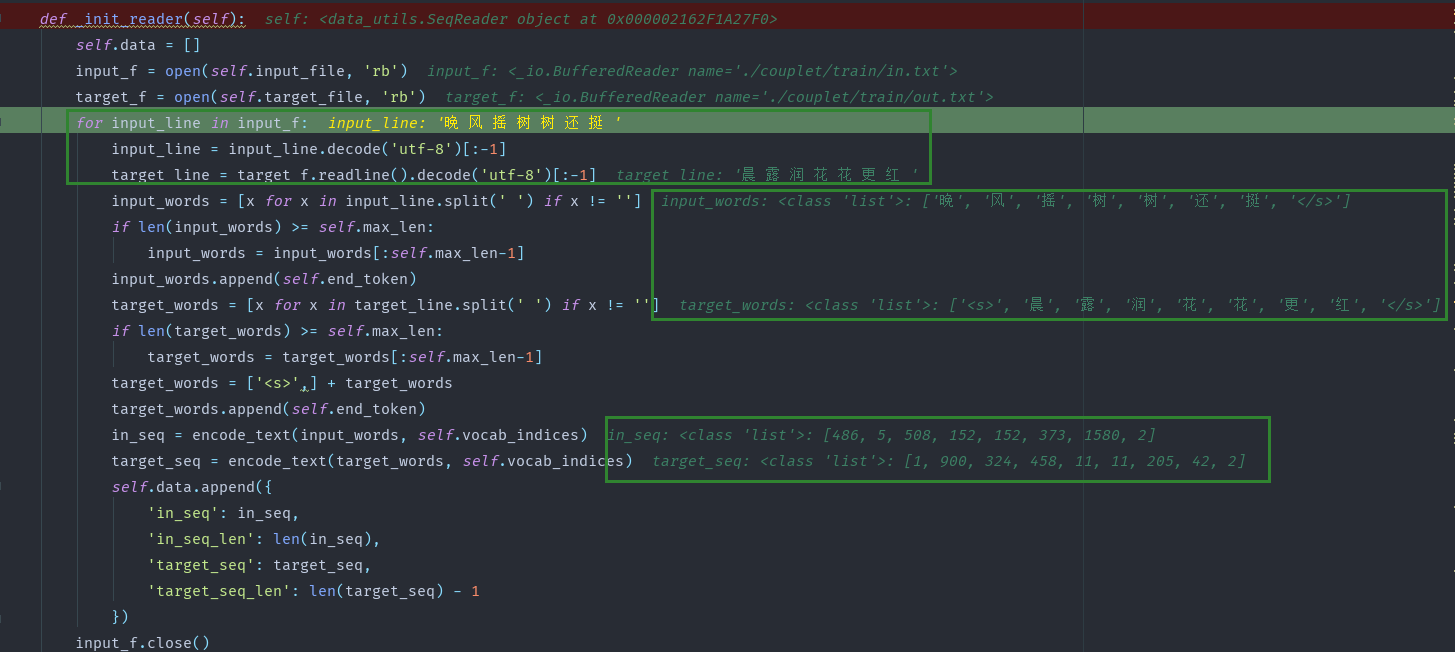
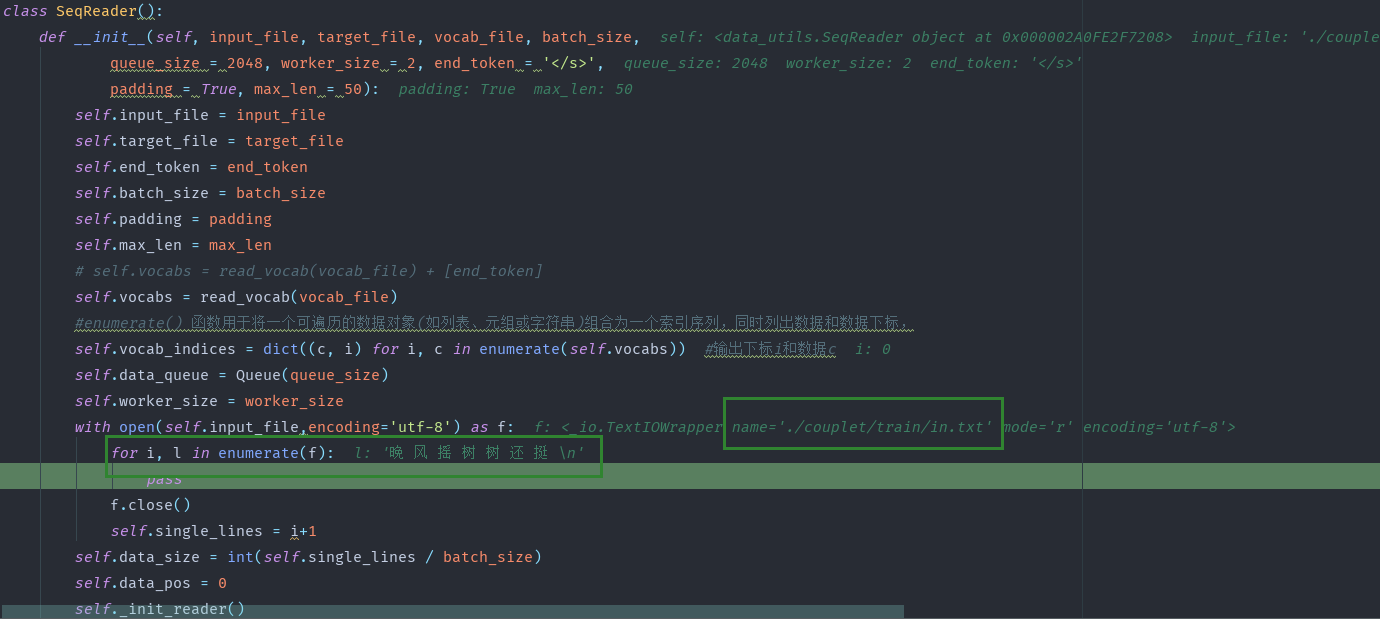


**5.22.2020**

**原型构造（tkinter）：**

****

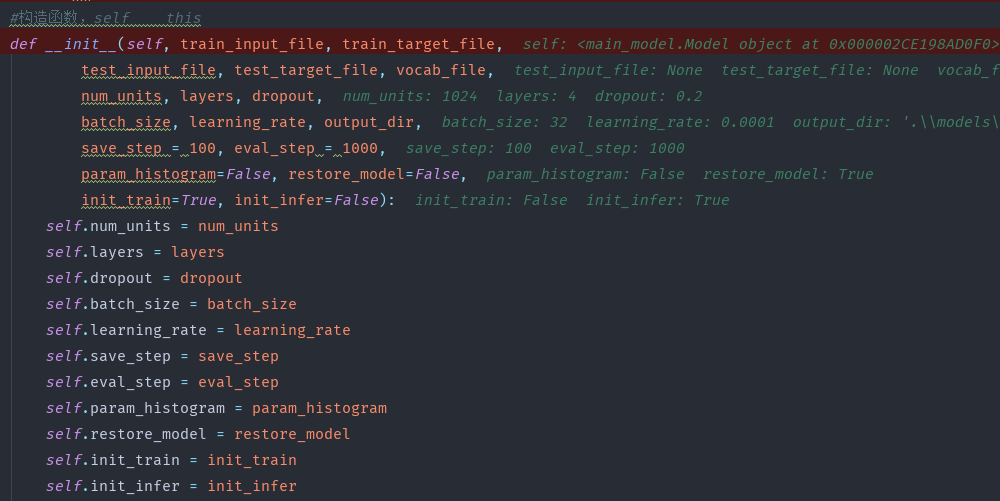
**训练模型：**



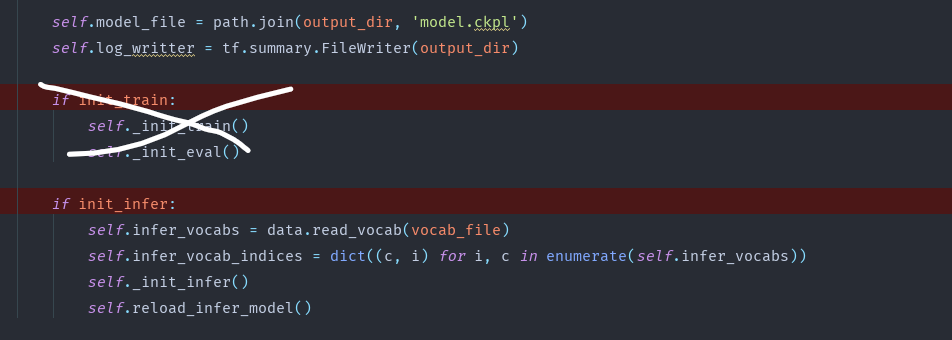
**5.27.2020**

**Debug调试**

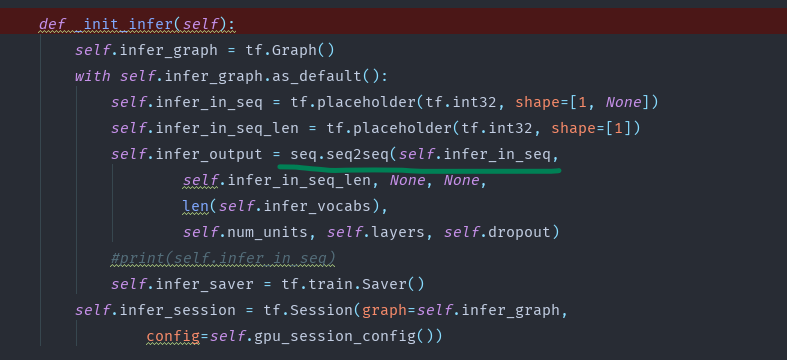
Server原型，初始化model（进入Main的Model函数）



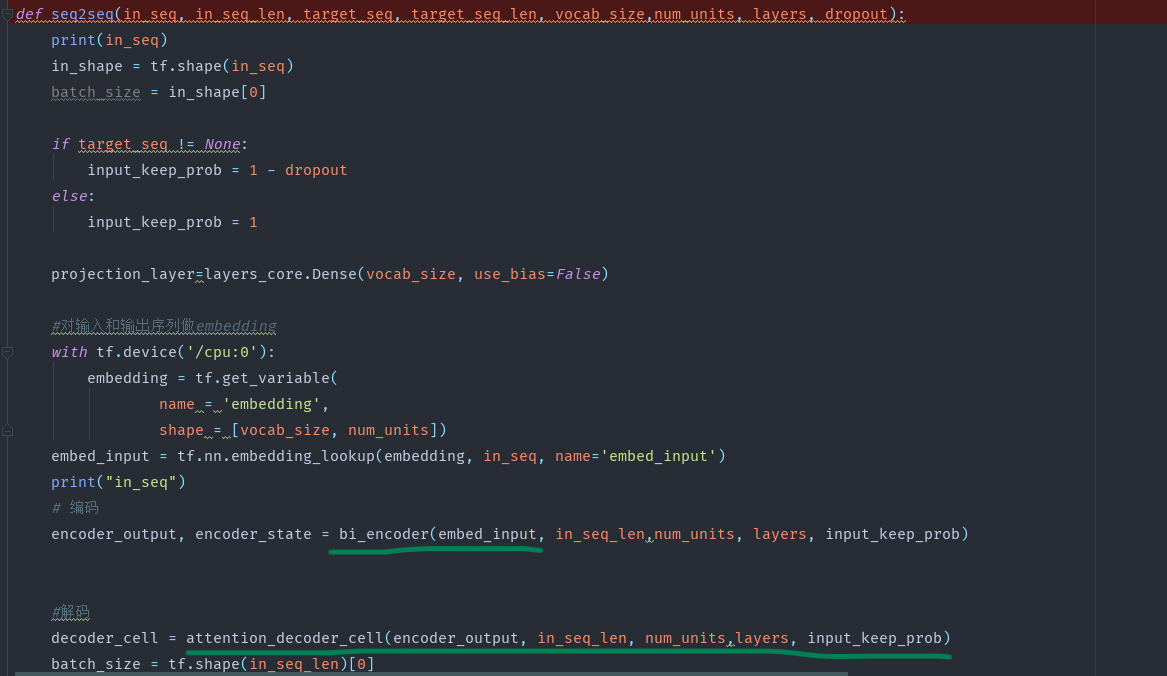
不是训练，所以跳过train，找到模型文件和词表：



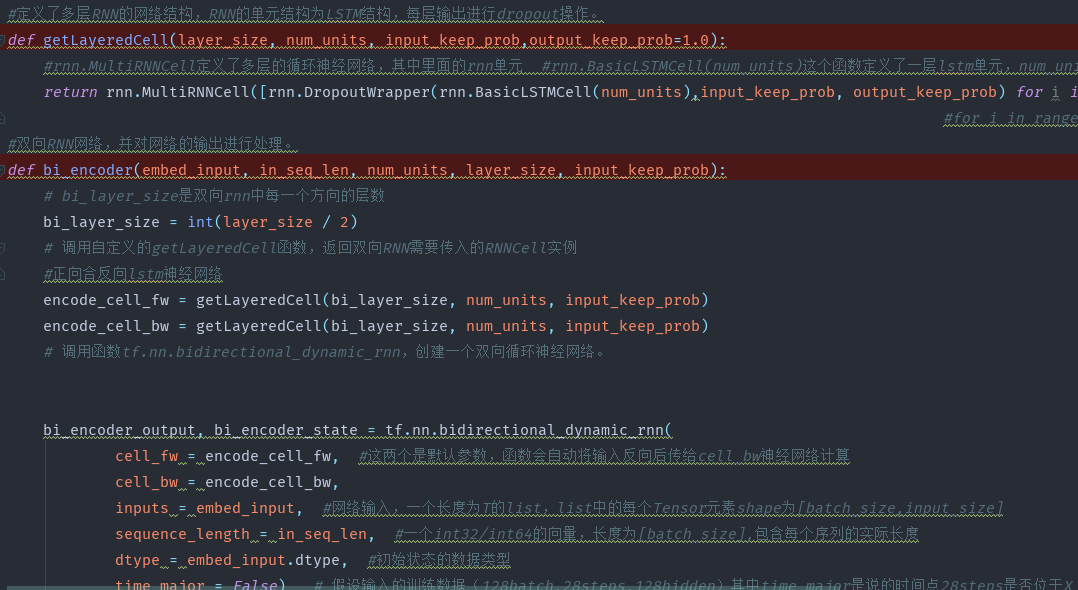
创建新的默认图：

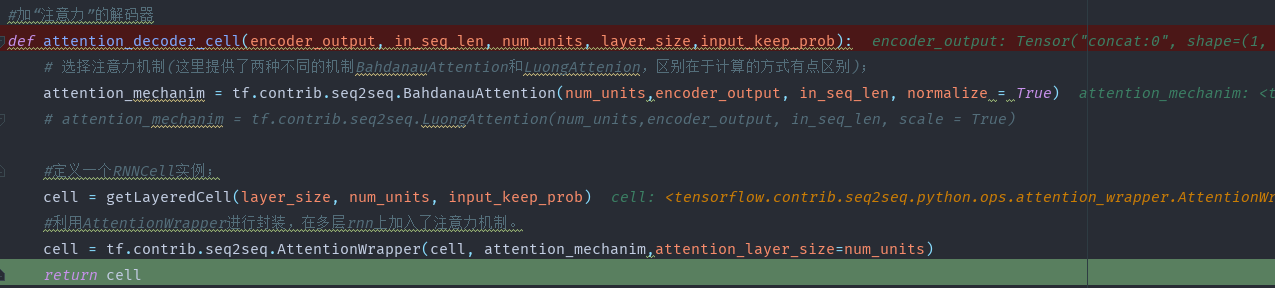


进入seq2seq模型：



调用到bi\_encoder和attention\_decoder\_cell (其中调用到getlayeredCell) :

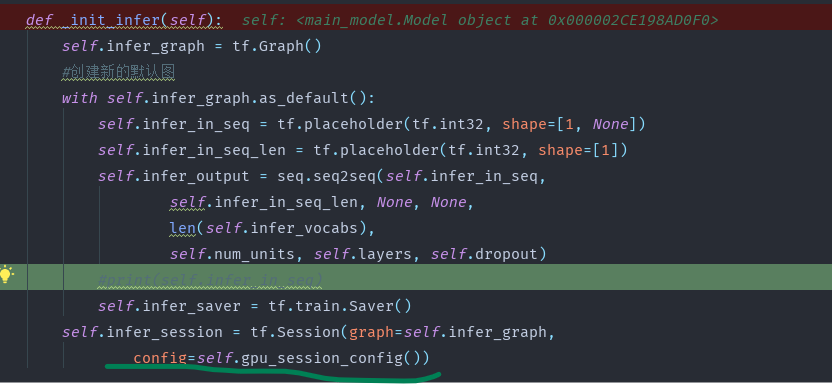


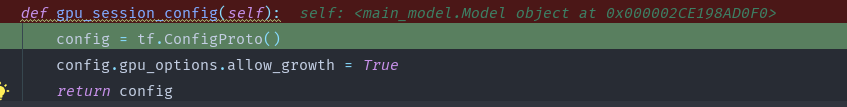


结束，然后回到seq2seq模型：

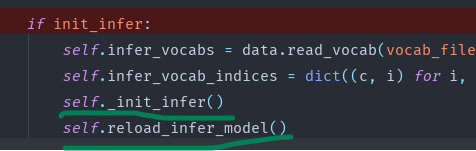


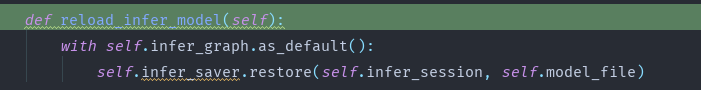
结束，回到infer，又调用了gpu\_session\_config：





Infer结束，回到第二步：





开始读取上联，编码，然后对下联：

