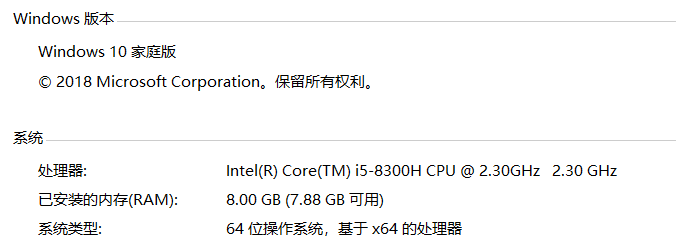
实验八 神经网络语言模型

**实验内容：**

使用Tensorflow在GPU上构建循环神经网络模型，在PTB（Penn Treebank）语料库上进行神经网络语言模型的训练和评估。

**实验环境：**

Lenovo Legion Y7000p，Intel 8代i5，8G内存，显卡：NVIDIA GeForce GTX 1050 Ti 4G



Python 3.6 + TensorFlow 1.11.0 + CUDA 9.0 + cuDNN 7.6.0 + VSCode

**实验原理**

1. 语言模型：

语言模型是一个为某一段词序列分配概率的模型。它对多种自然语言处理任务都有帮助。例如，在机器翻译任务中，需要由语言模型为系统输出打分，以提高输出目标语言输出的流畅性。在语音识别任务中，语言模型与声学模型一起预测下一个词。语言模型用来计算一个具有N个词的词序列概率，即：

******

上式可知，它也可以分解成对给定前缀（一般叫做上下文）的下一词出现概率的乘积。神经语言模型就是用于估计每一个词出现的条件概率的。

1. 循环神经网络：

循环神经网络对序列数据的处理有先天优势，它的结构使网络可以接受变长输入，当网络输入窗口被移位时，不需要重复计算。其网络结构如图1所示。

我们关注某一时刻的损失，它就等于此时刻之前所有时间步的损失之和。例如，对于时间时刻的损失，它的损失为

这种在时间步上展开进行反向传播的算法，称为基于时间的反向传播（Back-Propagation Through Time，BPTT）。循环神经网络的训练就是通过BPTT算法进行的。



图1 循环神经网络（RNN）结构

下面以图1为例，我们将通过BPTT算法来计算RNN的参数梯度。对于每一个节点N，我们需要基于N后面的节点的梯度，递归地计算梯度。我们从最后一个节点的损失开始递归：

对于时间步t输出的梯度的第i个元素为：

从序列的最后时间步开始反向计算梯度。对于最后时间步，只由得到，则梯度为：

根据这个梯度我们就可以依次计算时间步到时间步1的隐层节点的梯度。由于是同时有和两个后续节点。因此，对于每个时间步t的隐层节点的梯度为：

在得到了隐层节点的梯度后，我们就可以计算对于参数的梯度：

其中，**、、**表示时刻t时，**W**、**U**、**b**的副本，则表示时刻t的时的梯度贡献，对于**U**和**b**是类似的。

长短期记忆网络（LSTM）是一种具有门结构的特殊循环神经网络。它是为了应对长期依赖的挑战而提出的。LSTM网络结构如图2所示。



图2 LSTM网络结构

LSTM引入了“门”机制对细胞状态信息进行添加或删除，由此实现长程记忆。“门”机制由一个Sigmoid激活函数层和一个向量点乘操作组成，Sigmoid层的输出控制了信息传递的比例。每个LSTM基本单元包含遗忘门、输入门和输出门三个门结构。

1）遗忘门

LSTM通过遗忘门（forget gate）实现对细胞状态信息遗忘程度的控制，输出当前状态的遗忘权重，取决于和。

2）输入门

LSTM通过输入门（input gate）实现对细胞状态输入接收程度的控制，输出当前输入信息的接受权重，取决于和。

3）输出门

LSTM通过输出门（output gate）实现对细胞状态输出认可程度的控制，输出当前输出信息的认可权重，取决于和。

4）状态更新

**实验步骤：**

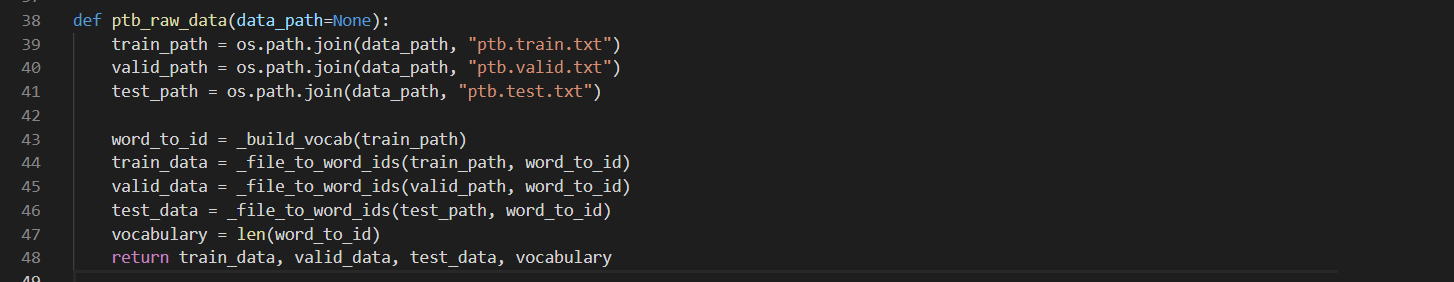
1. 下载语料库：

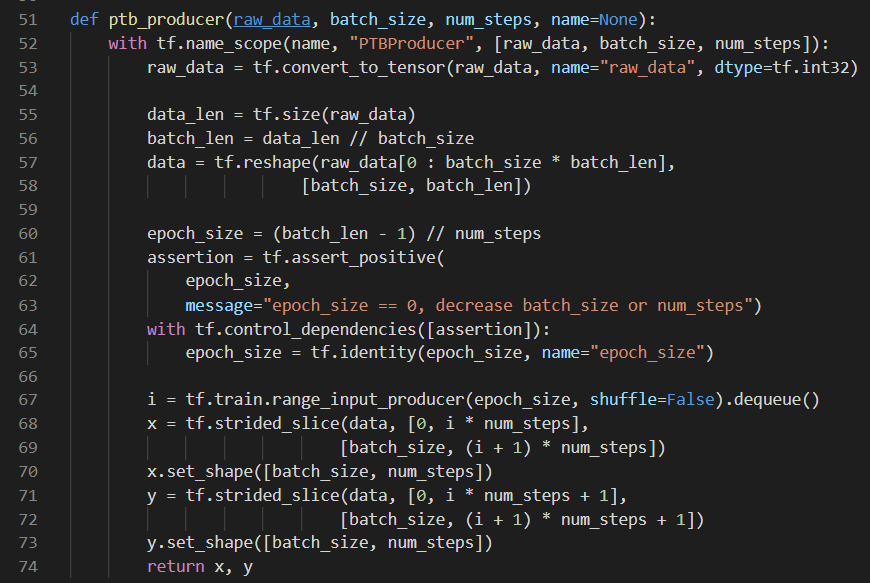
（1）老师给的文件里已经包含。

1. 加载及准备数据：

（1）从文件读入的语料库数据上迭代生成训练、验证、训练数据集，其中raw\_data是需要将词转换成词序号，方便后续处理。并将数据进行切分，去掉多余数据。从中获得批数据集。

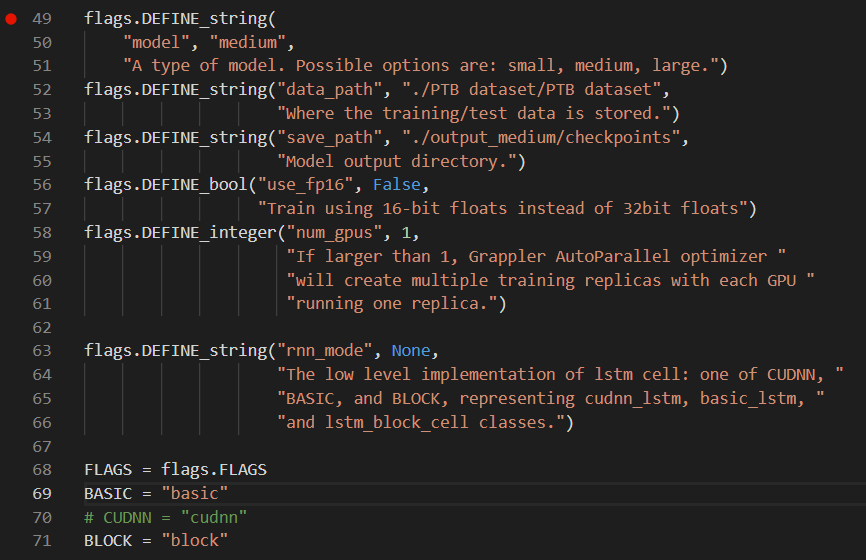
代码在**reader.py**中：





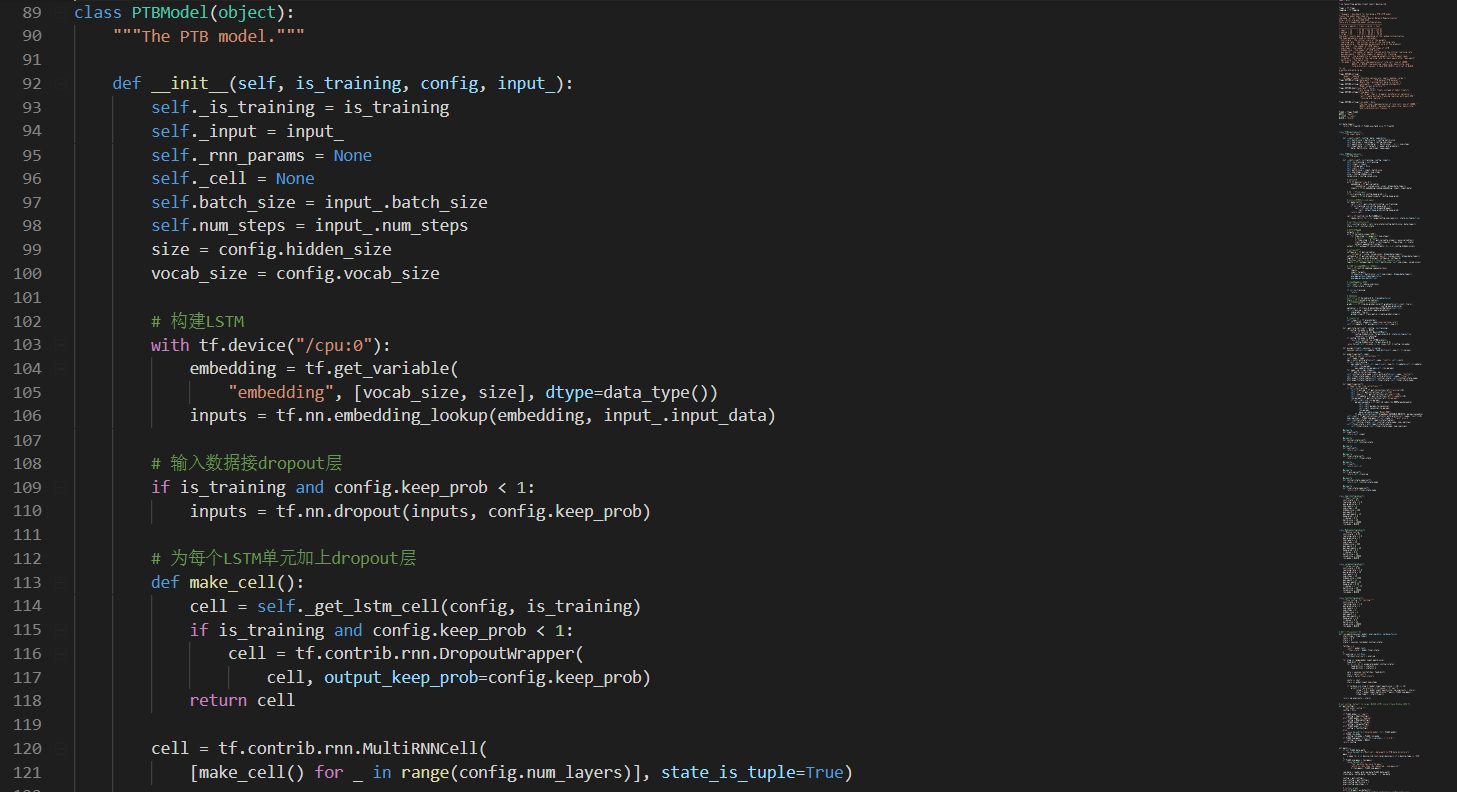
1. 构建计算图：

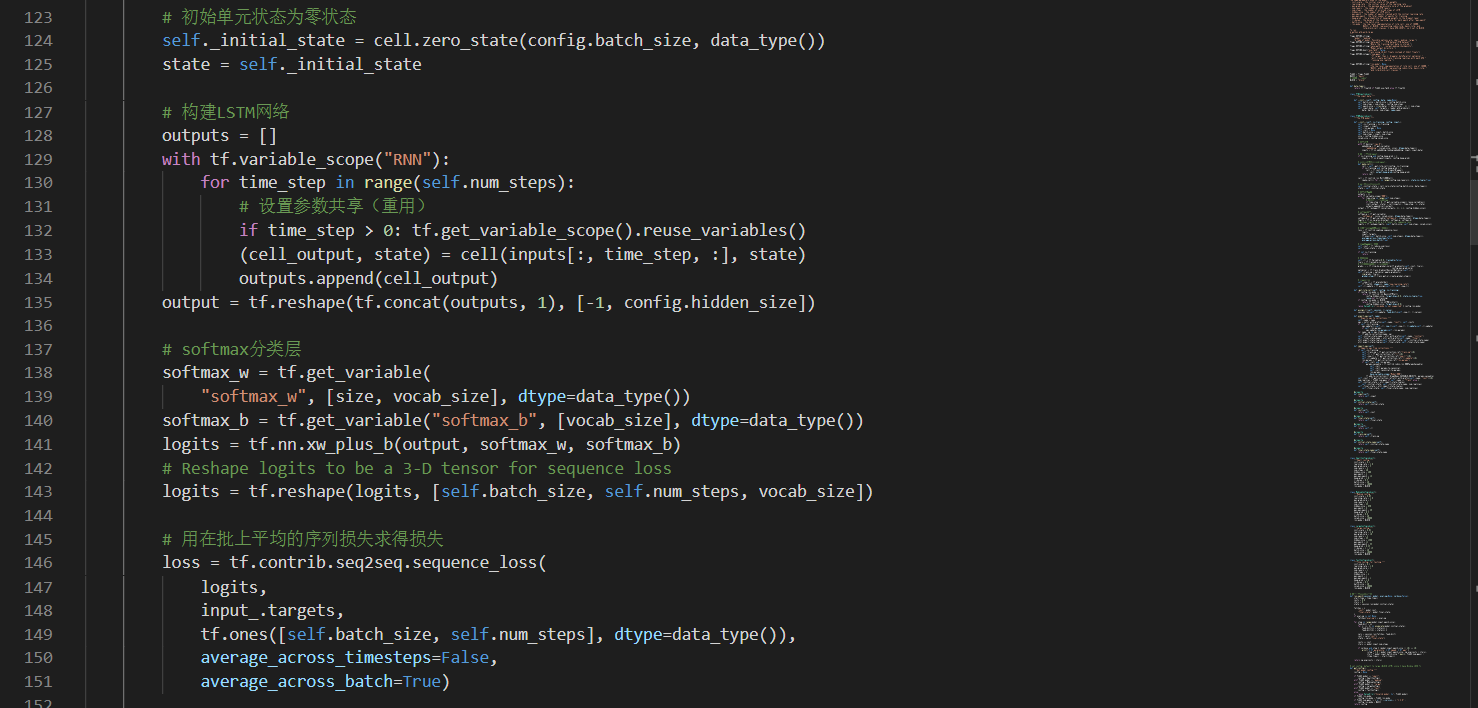
（1）构建模型初始参数(config)：



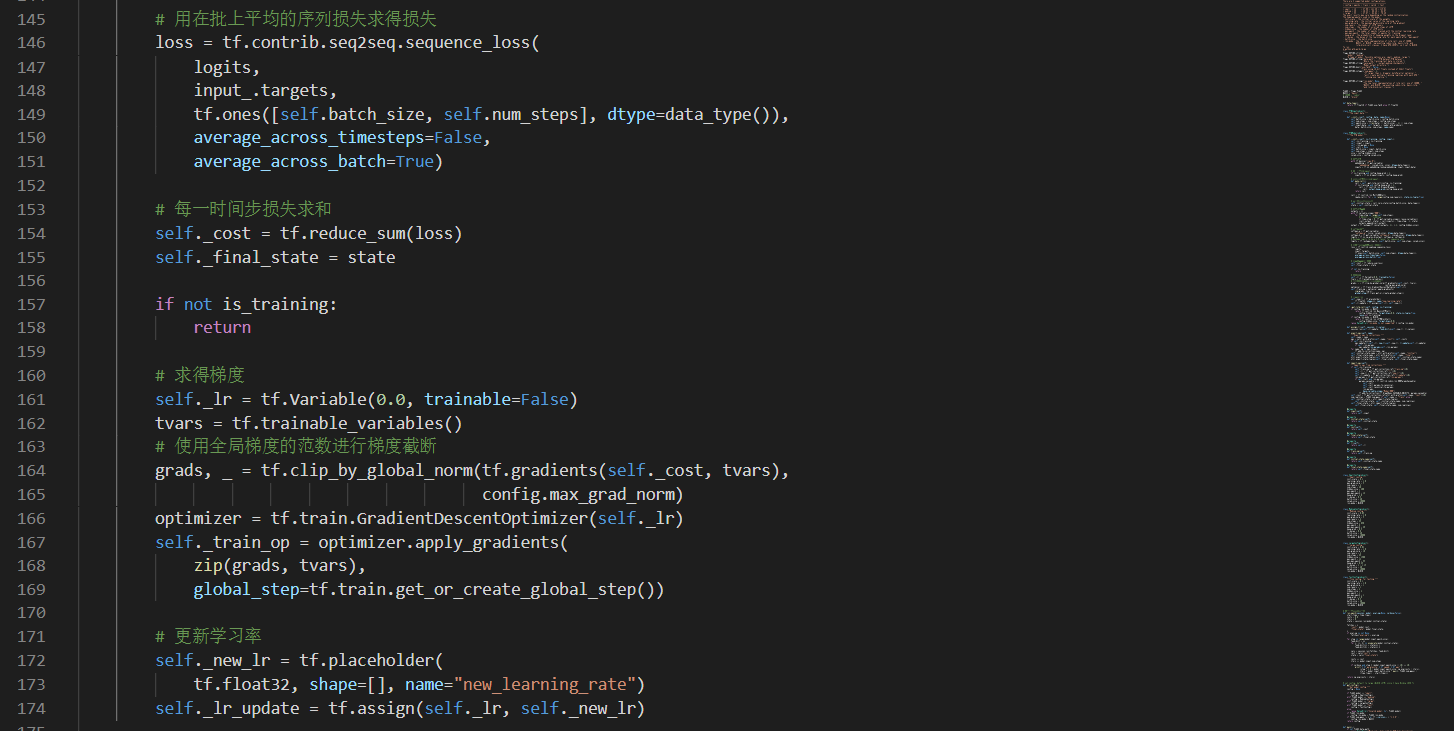
（2）构建LSTM：

代码在**main.py**：

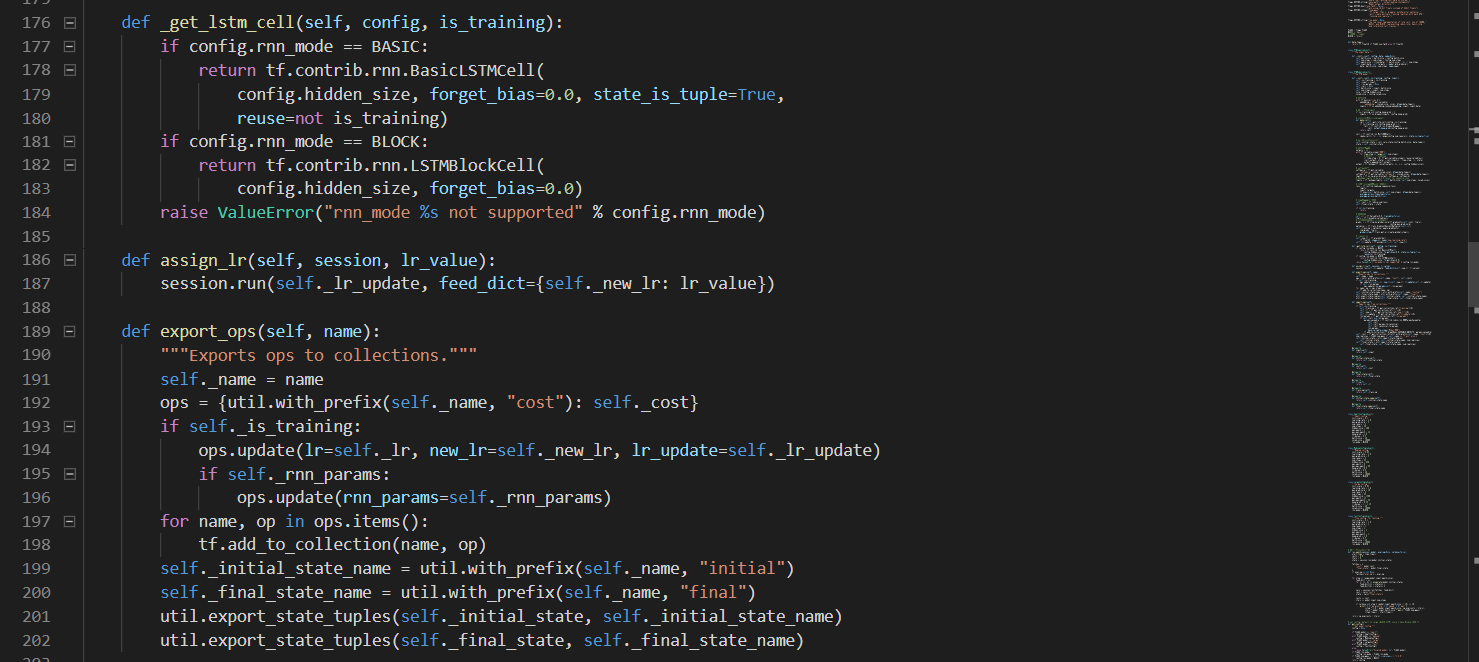


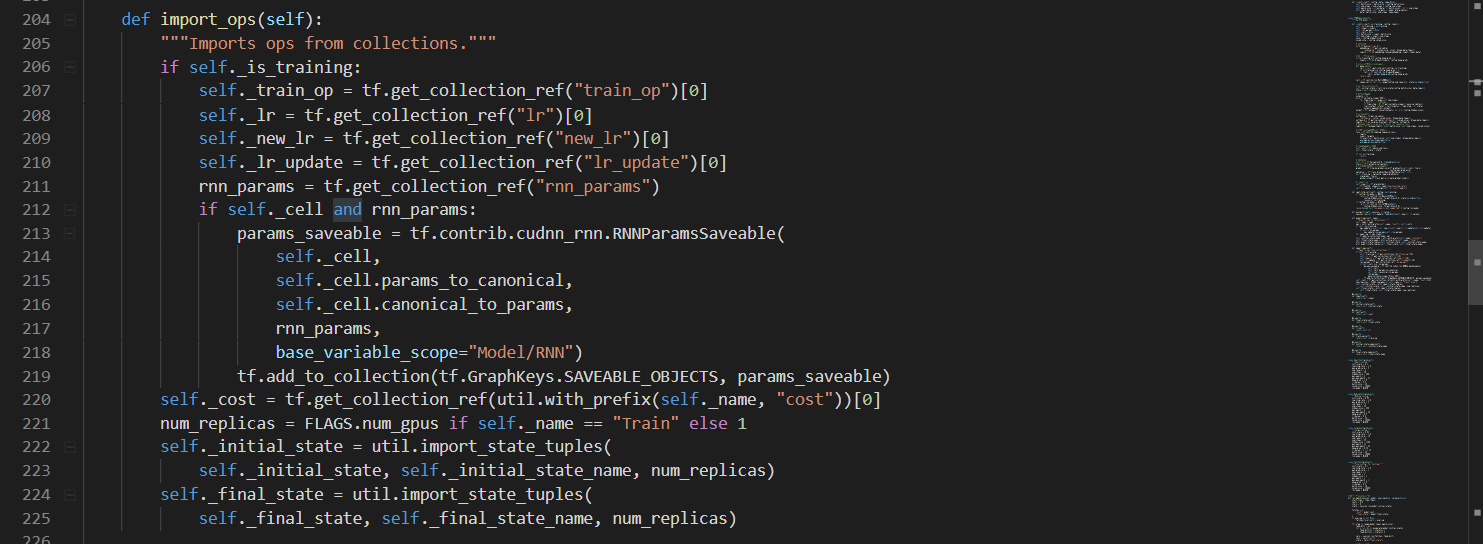


（3）训练和评估：

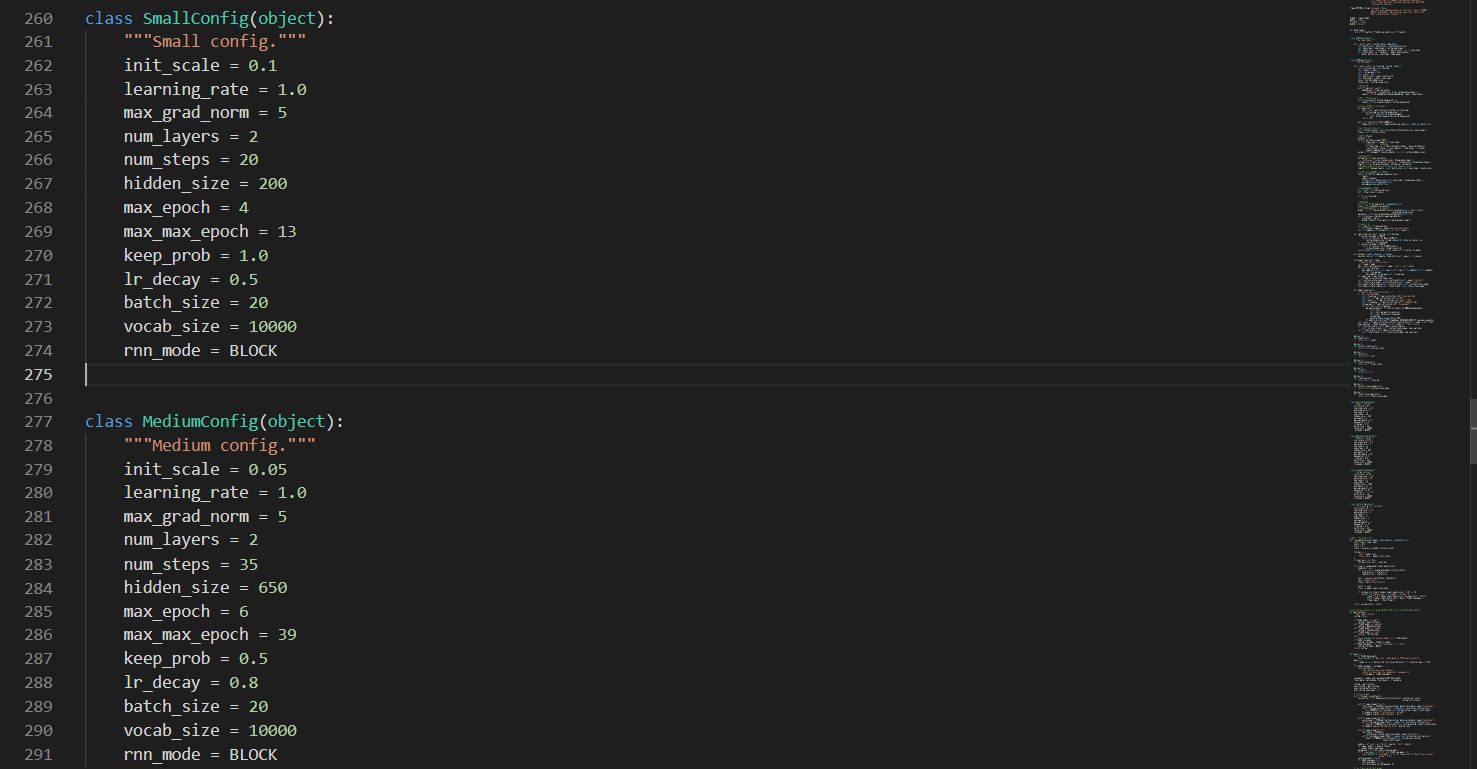


（4）LSTM神经元结构设置和图中Tensor的操作（op）:



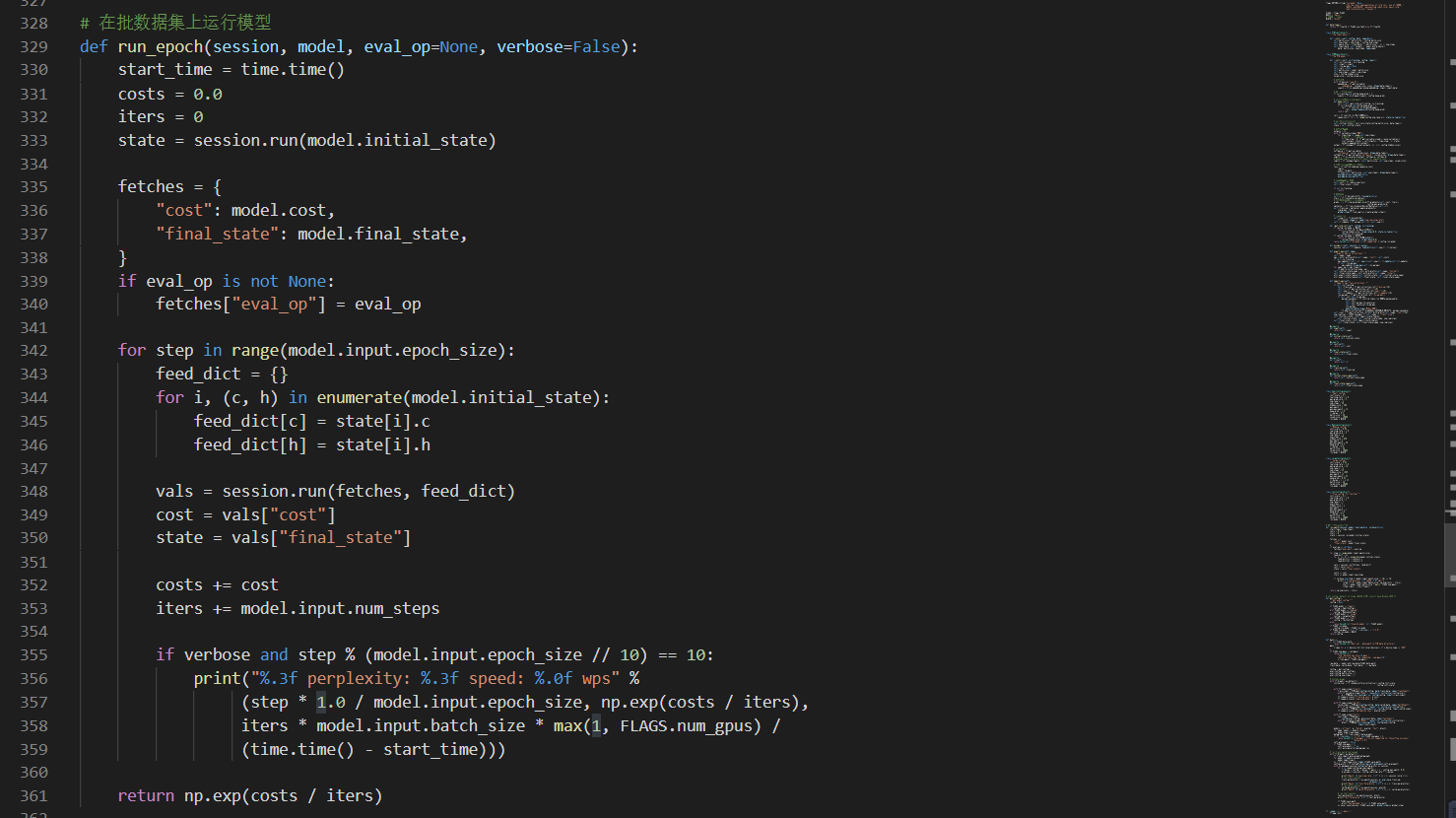


（5）超参设置:

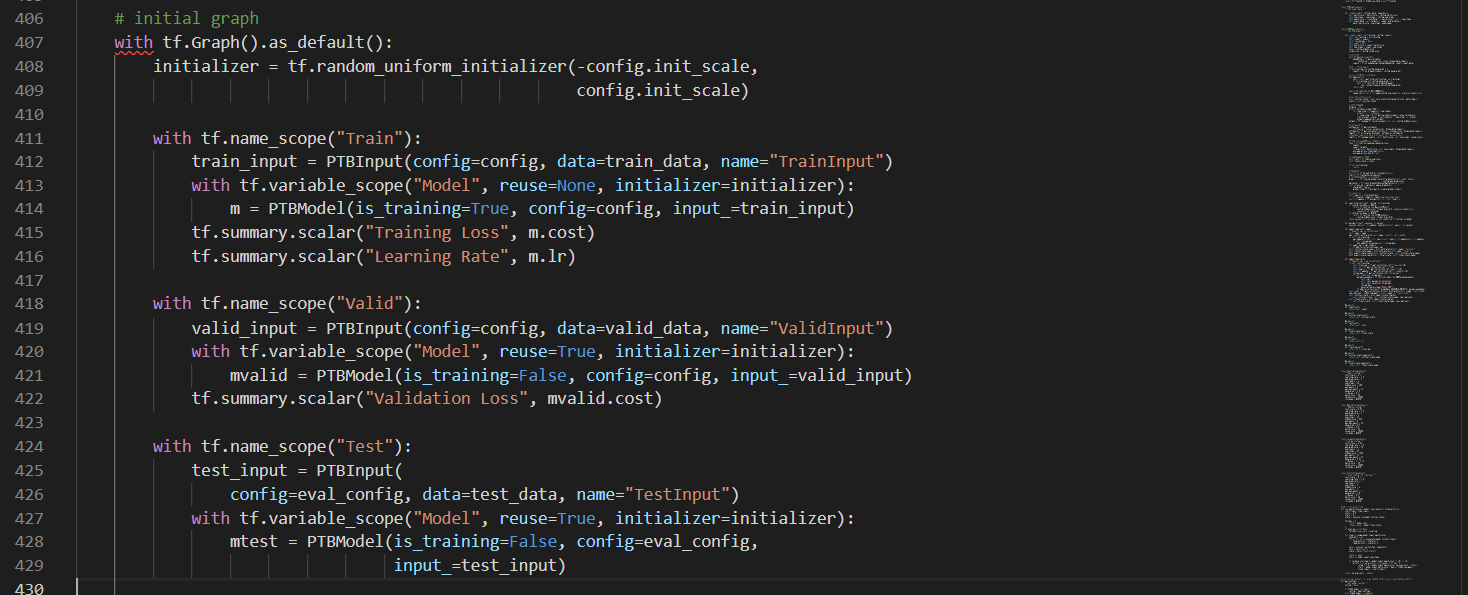


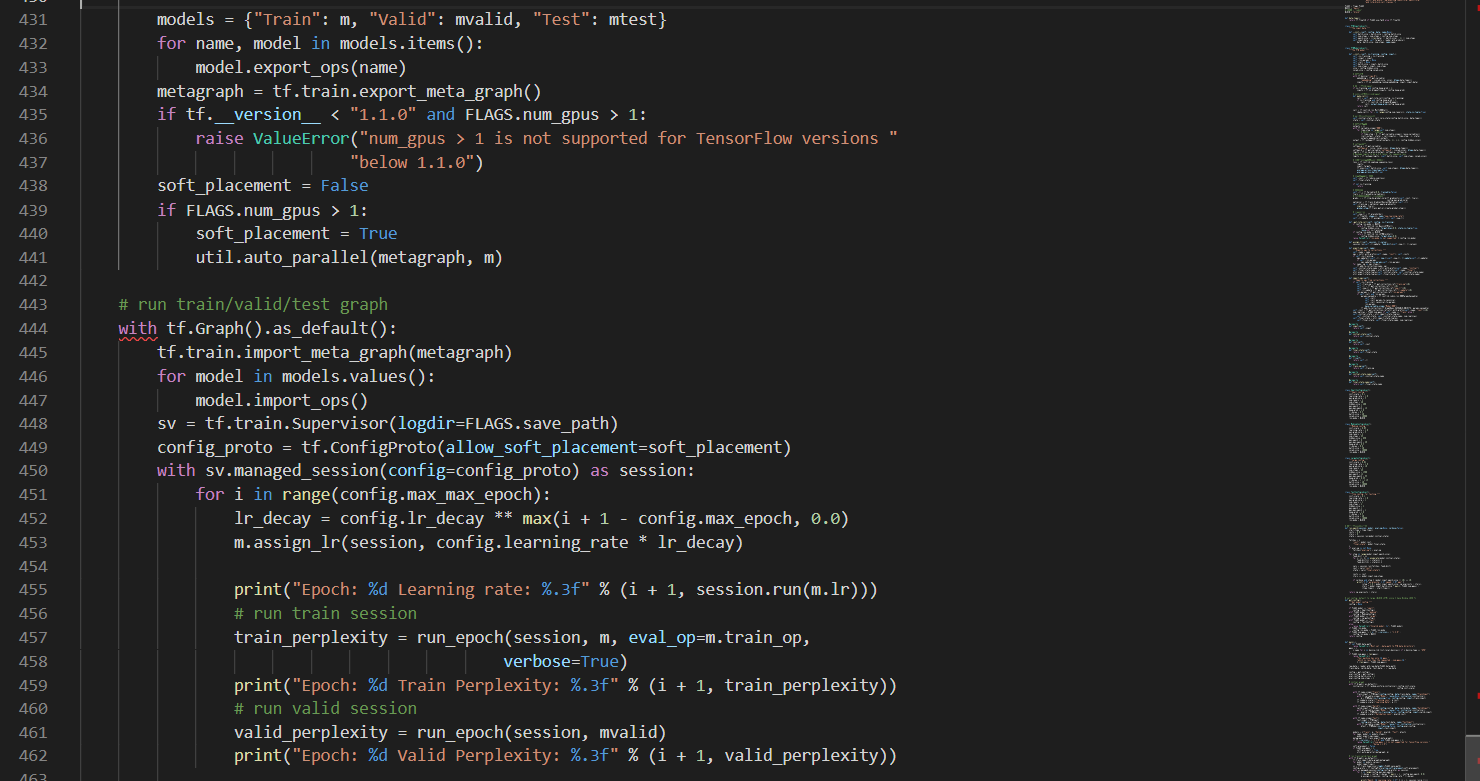


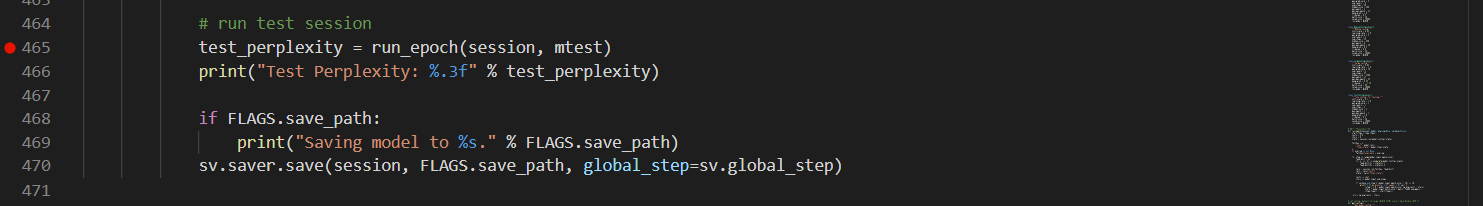
1. 创建会话，进行模型的训练与评估：



运行计算图：







**实验结果：**

直接运行脚本**main.py**

实验开始：

Small模式：

num\_steps = 20

hidden\_size = 200

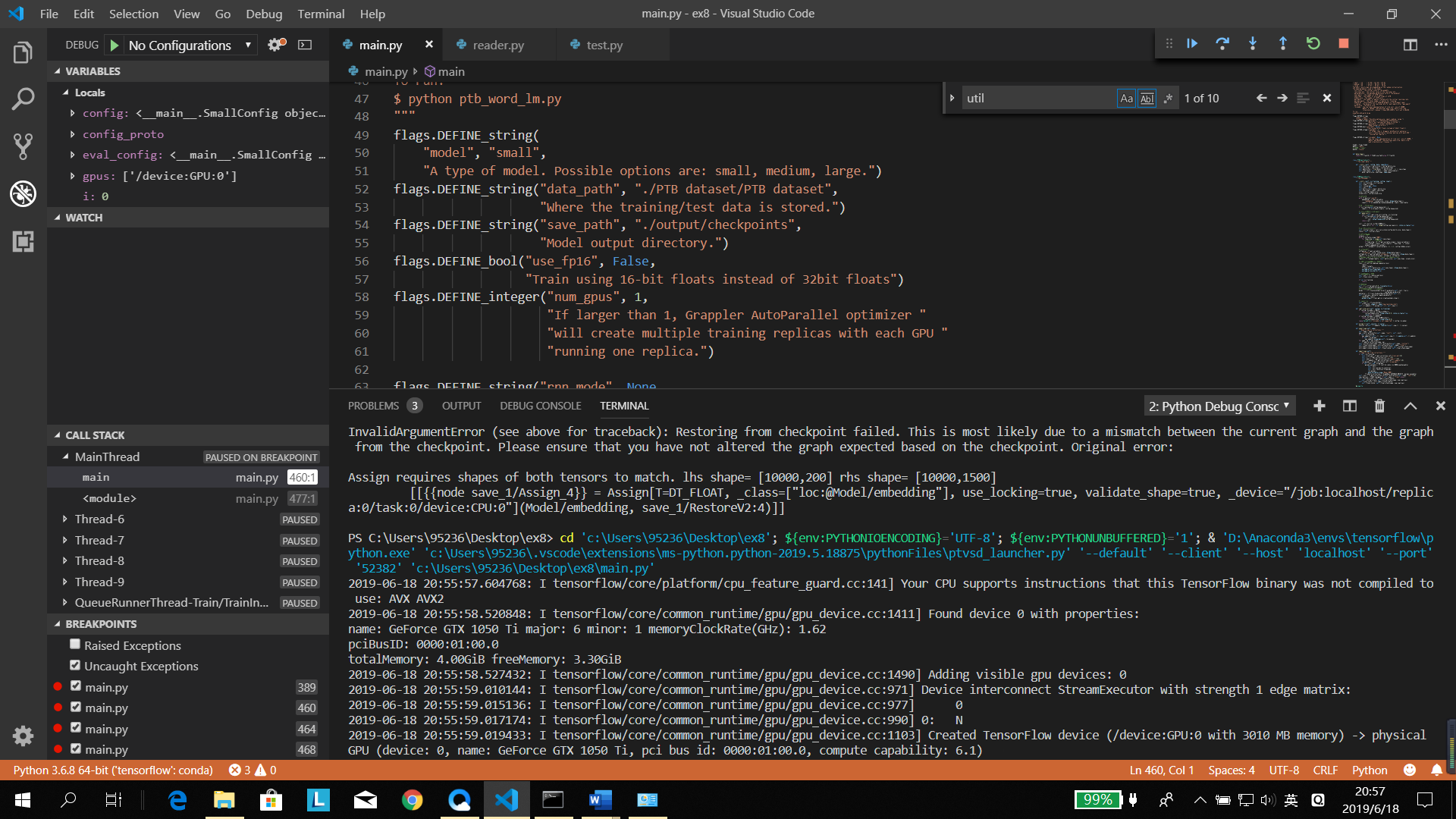
max\_epoch = 4

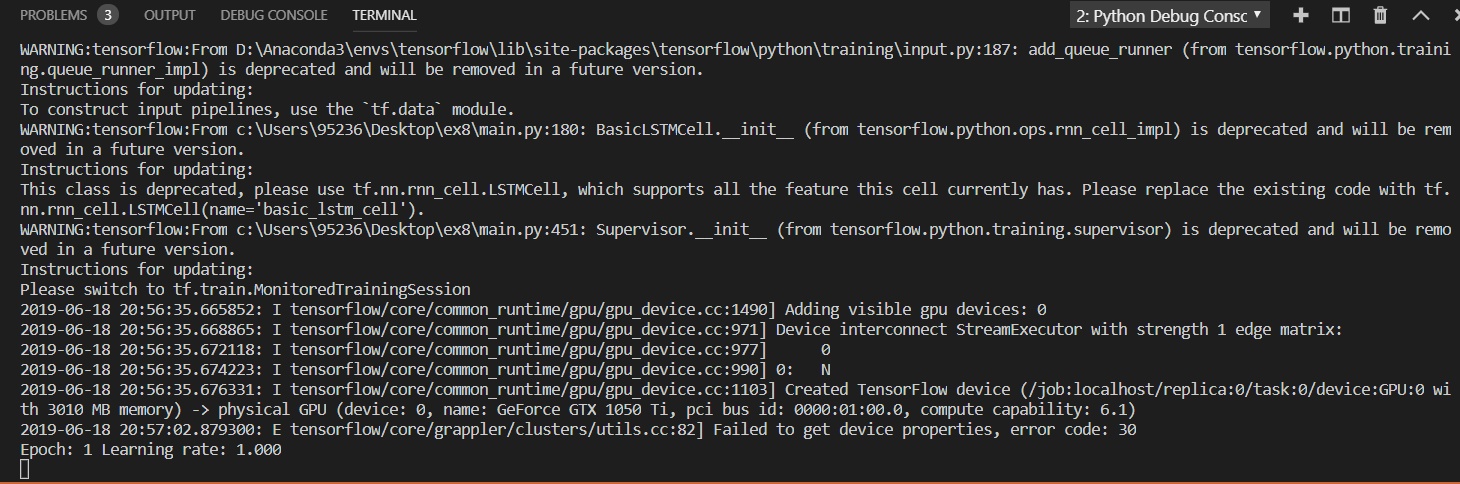
max\_max\_epoch = 13

keep\_prob = 1.0

lr\_decay = 0.5

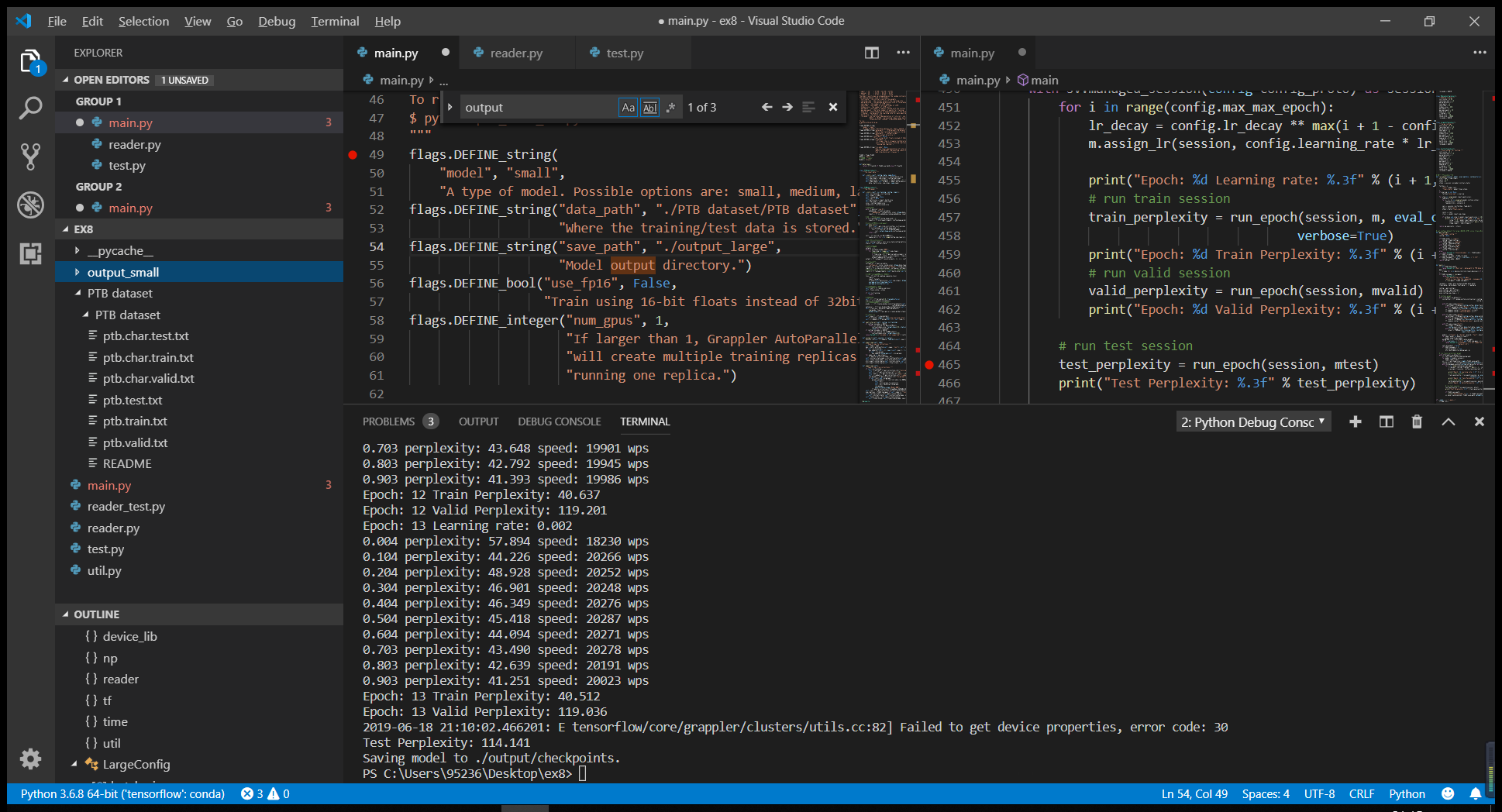
训练和验证：







测试：



Medium模式：

num\_steps = 35

hidden\_size = 650

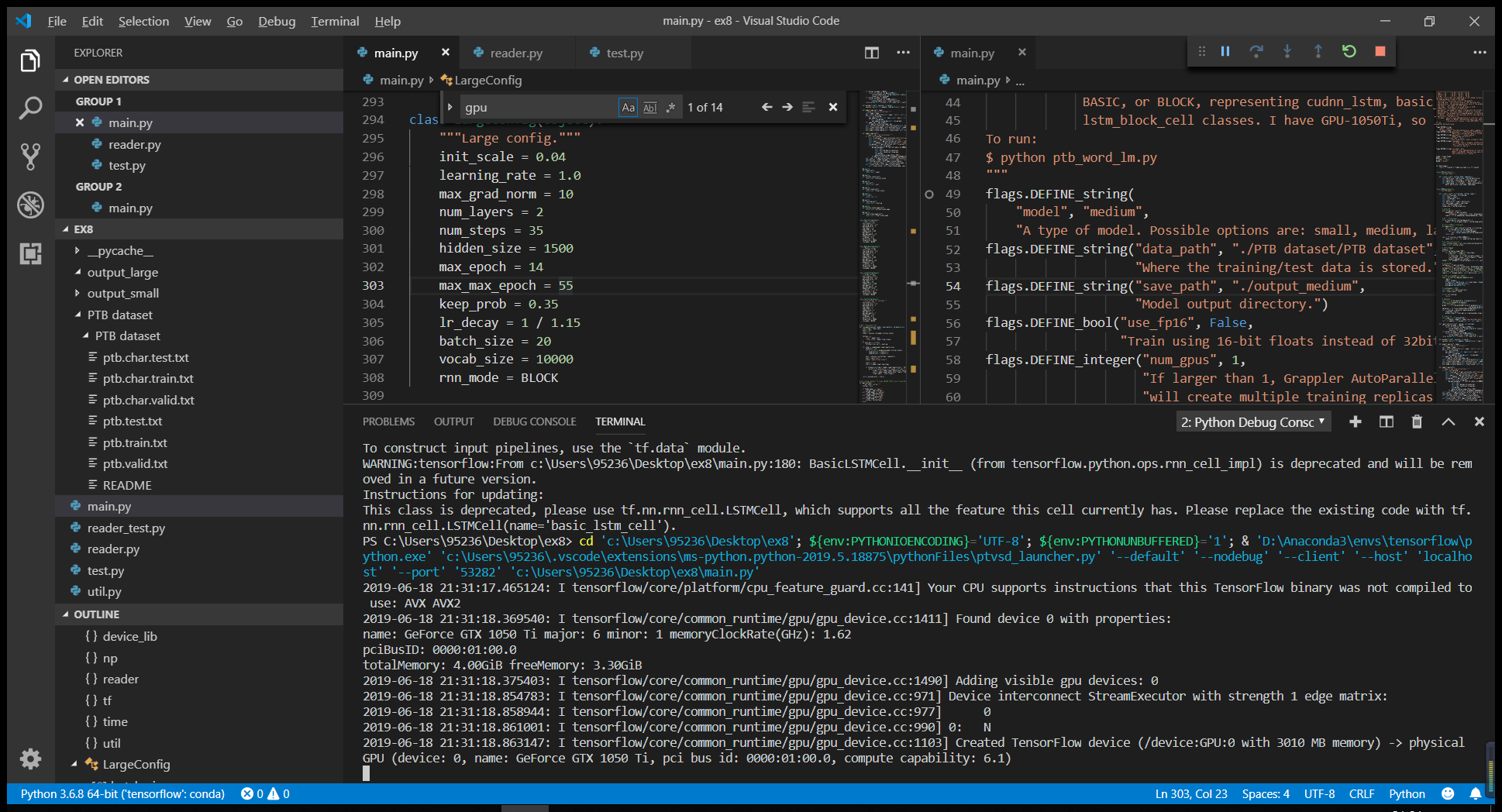
max\_epoch = 6

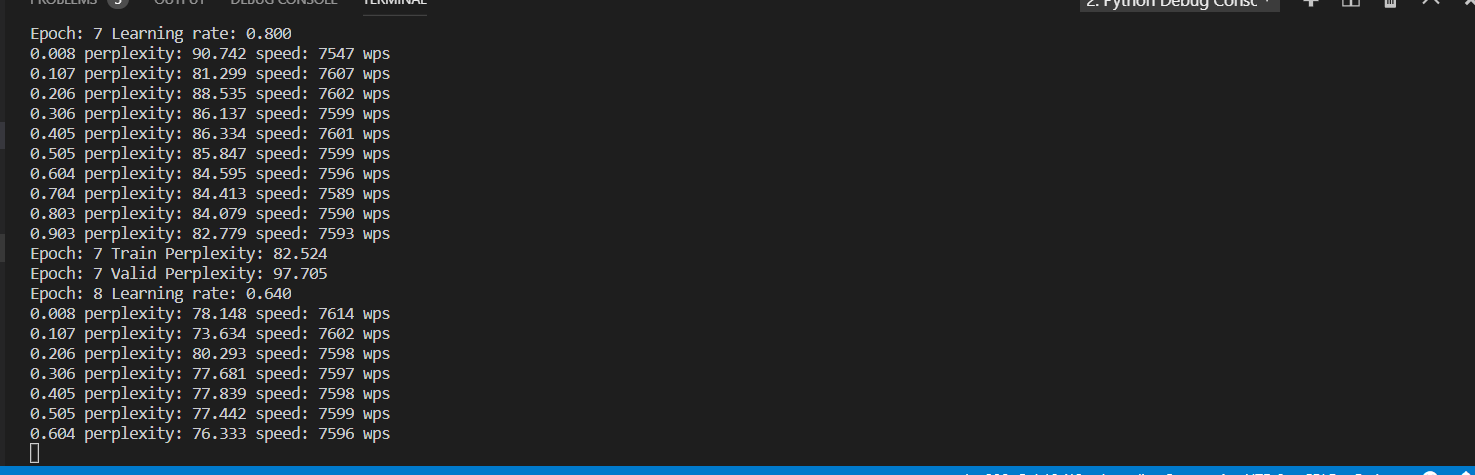
max\_max\_epoch = 39

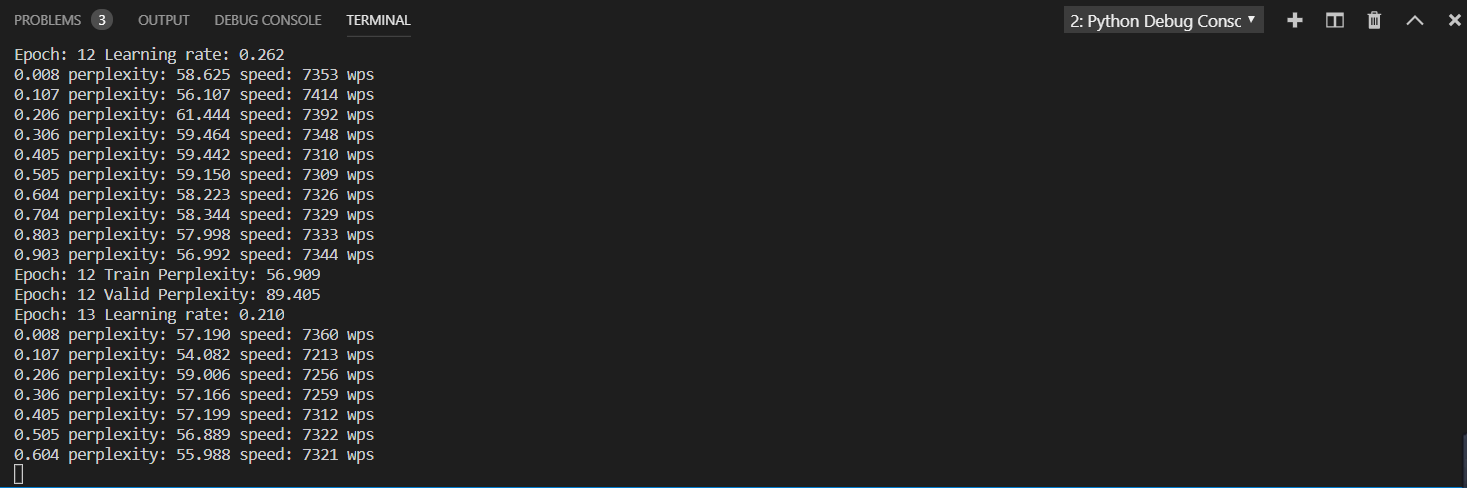
keep\_prob = 0.5

lr\_decay = 0.8

训练和验证：

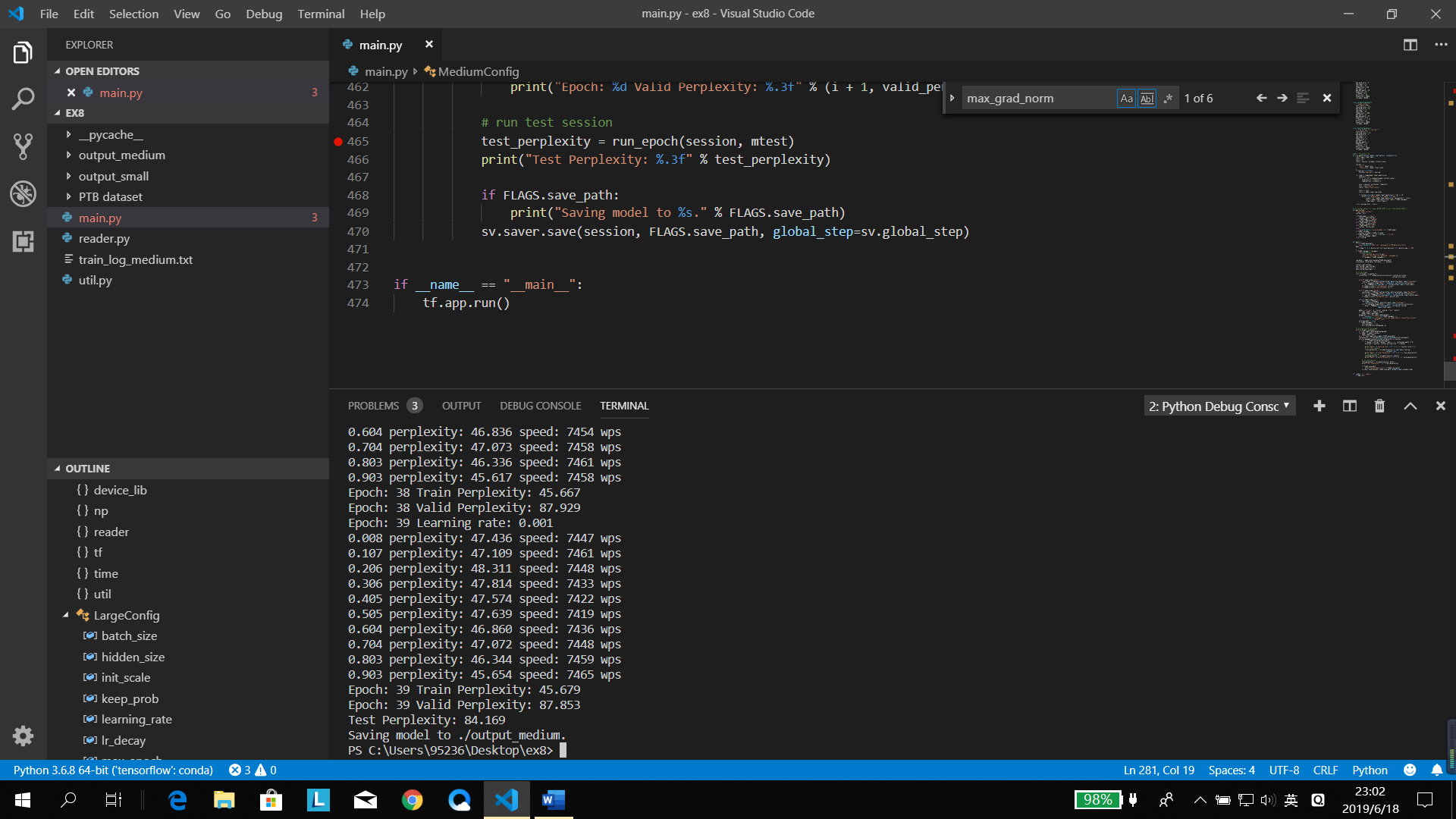








测试：



Large模型：主要是神经元个数翻倍所以导致计算图太大，显存溢出（OOM）,显存和内存不足，电脑跑不动。

num\_steps = 35

hidden\_size = 1500

max\_epoch = 14

max\_max\_epoch = 55

keep\_prob = 0.35

lr\_decay = 1 / 1.15

