**课程编号：** **A3705060010**

**自然语言处理**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **姓名** | **王硕** | **学号** | | **20182338** |
| **班级** | **软信1802** | **学院** | | **软件学院** |
| **项目名称** | **搭建LSTM语言模型** | | | |
| **开设学期** | **2021-2022秋季学期** | | | |
| **开设时间** | **第1周——第8周** | | | |
| **报告日期** | **2021年11月13日** | | | |
| **仓库地址** | **https://github.com/Aa-bN/NLP\_work** | | | |
| **报告内容** | **1.设计思路2.代码说明3.模型测试4.问题总结** | | | |
| **评定成绩** |  | | **评定人** | **肖桐** |
| **评定日期** |  |

**1. 设计思路**

该部分为构建模型的具体思路，分为三部分：模型功能，数据形状，核心算法。

**1.1 模型功能**

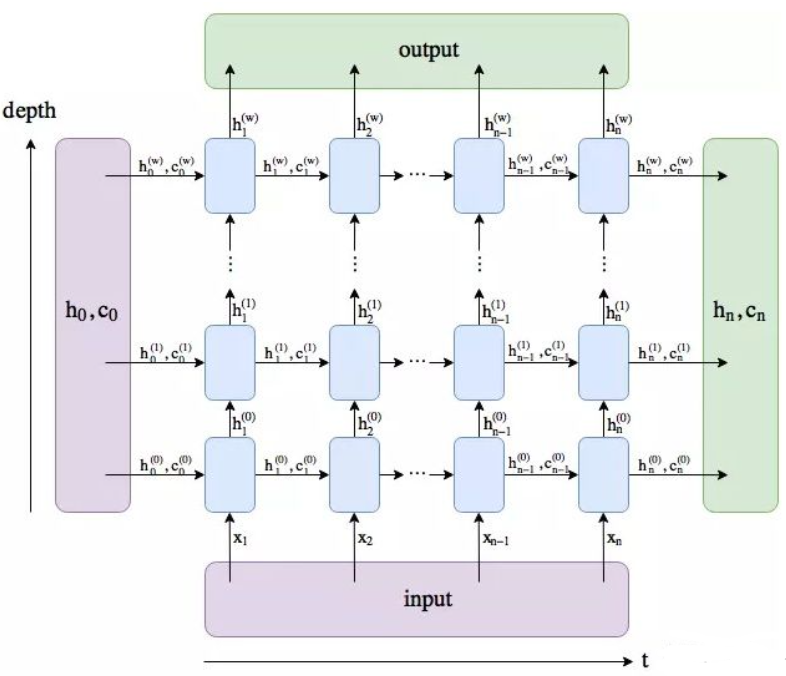
① 完成基本的LSTM过程；

② 可以选择LSTM的层数，通过传入参数，实现1层或多层LSTM的使用；

③ 可以选择性地传入hidden state、cell state、batch first等参数；

④ 输出每层最终的hidden state，cell state和outputs。

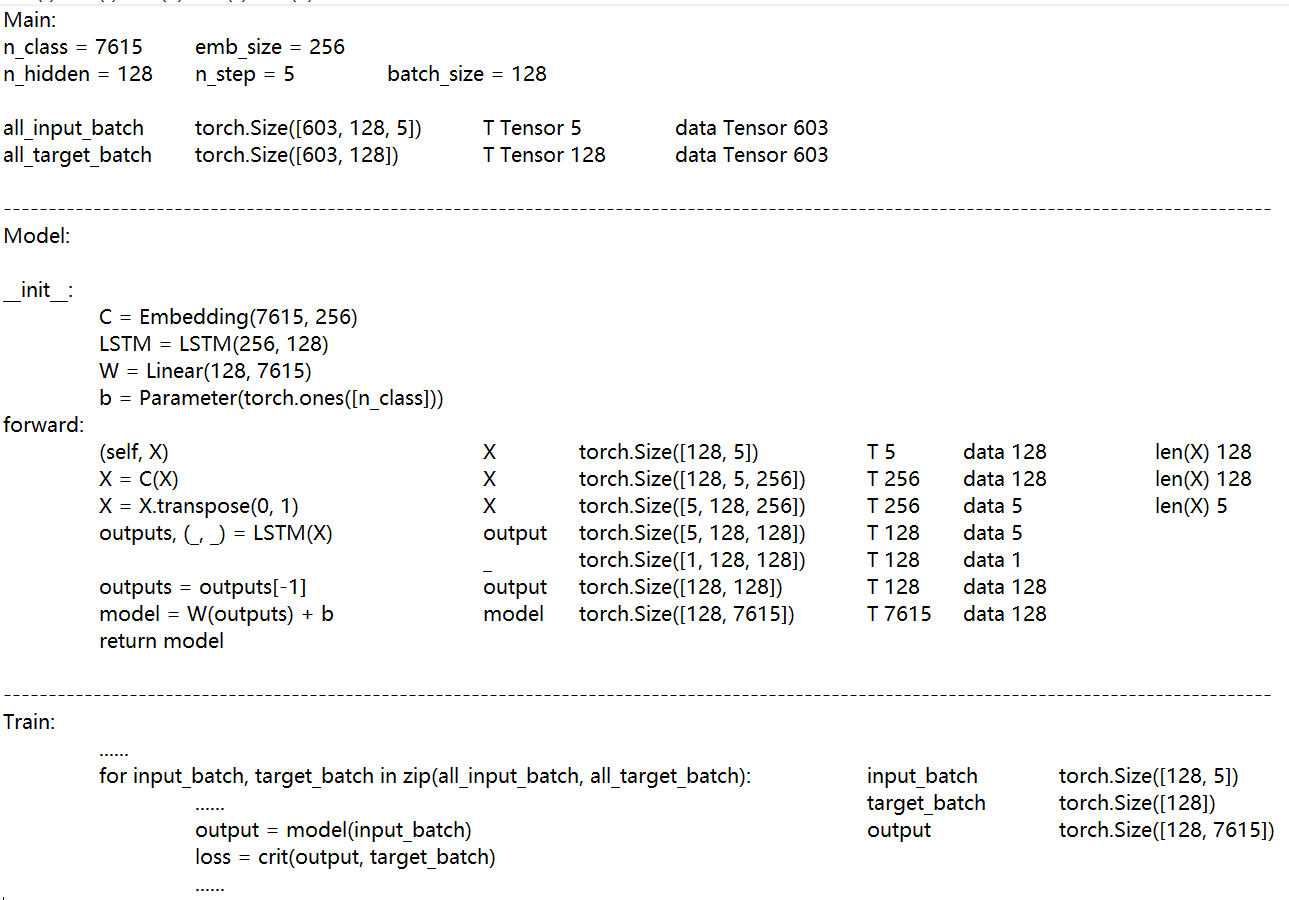
可用下图表示：



图源：<https://zhuanlan.zhihu.com/p/79064602>

**1.2数据形状**

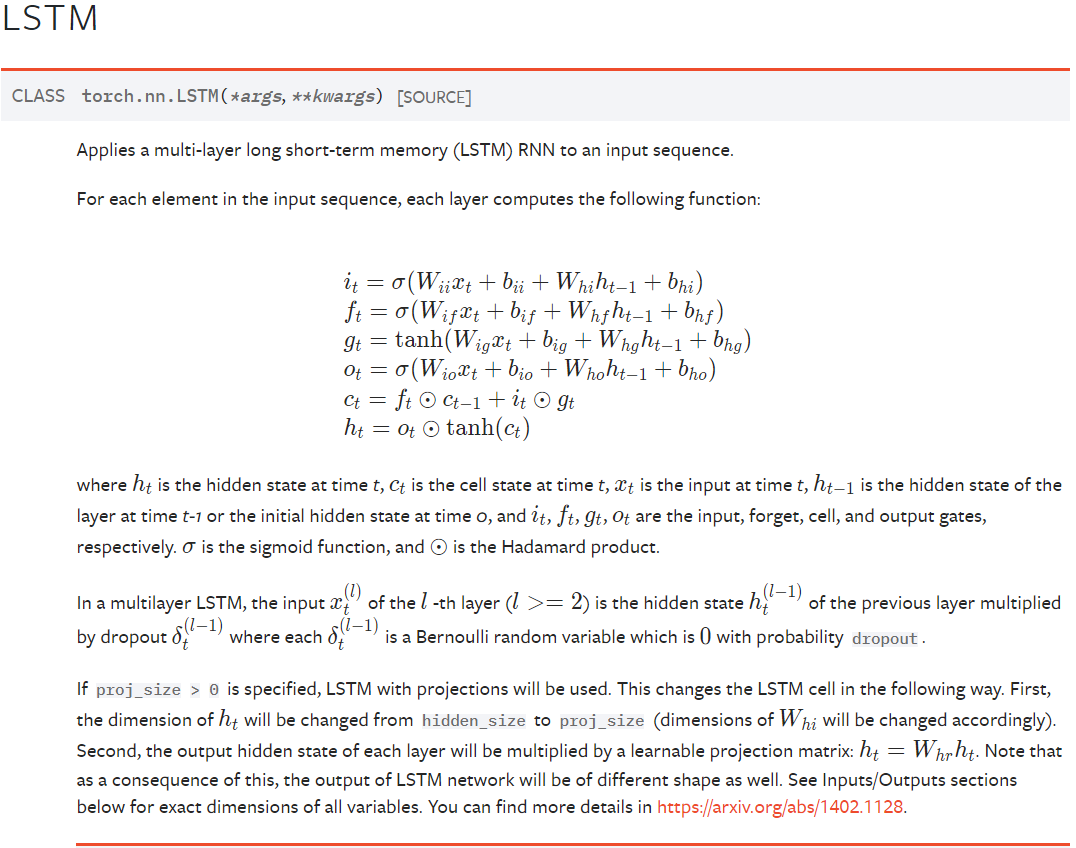
虽然项目只要求完成模型的搭建过程，但是对整个程序中数据类型或形状的分析，有助于更好地了解LSTM模型的输入和输出，也有利于为我们自己的函数提供良好的接口。通过对示例程序的debug，我提取并总结了程序中比较重要的变量，以及它们的形状。如下图：



图源：项目过程中的分析记录

**1.3 核心算法**

查阅官方文档，可以得到LSTM的核心算法：



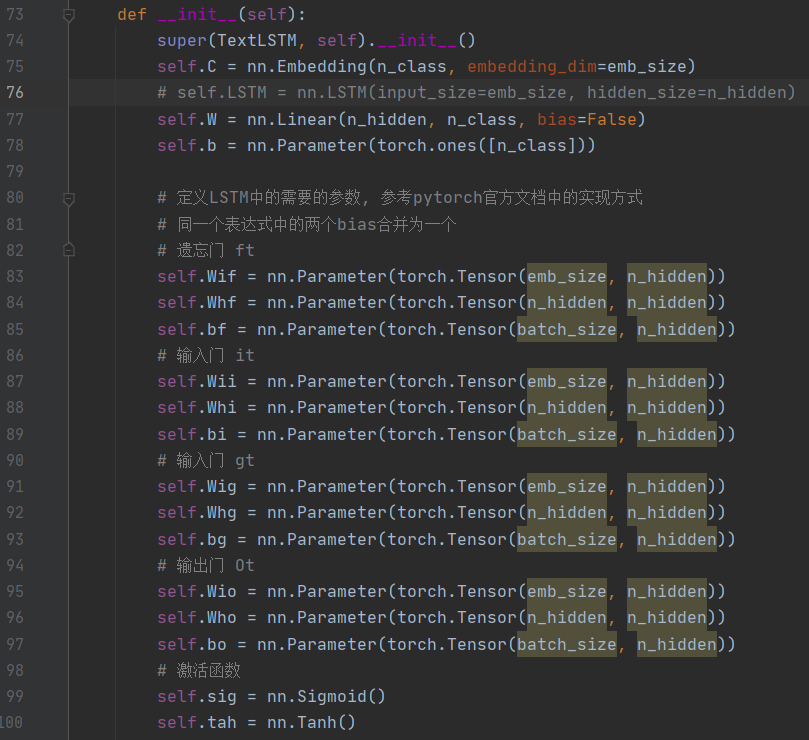
图源：pytorch官方文档

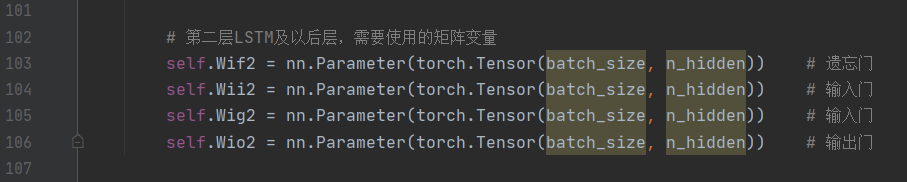
在实现时，可以把同一个表达式中的两个偏移量合并成一个。并且需要提前在init()函数中定义好权重和偏移量。在这些算法中，用到了两种不同的tensor数据类型的乘法，在实现过程中，需要加以区分。

**2. 代码说明**

模型在myLSTM.py的72-227行，下面**结合源码注释**进行代码说明。

**2.1 首先来看init()函数：（73-106行）**





在75-78行，定义了嵌入层、线性层和一个偏移量（不用于LSTM内部）。

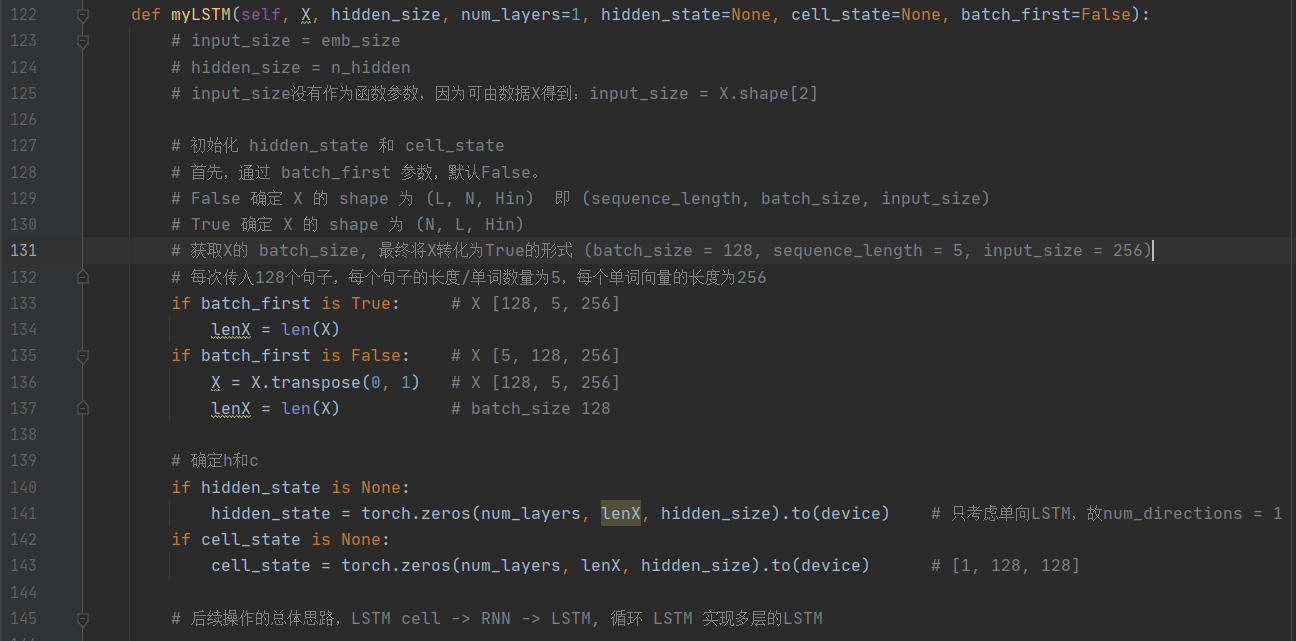
在80-100行，定义了LSTM内部的权重与偏移量，激活函数。

在102-106行，定义了第二层及以后层的LSTM，需要使用的权重和偏移量。

这些变量的size可以根据1.3核心算法推导出来。

**2.2 再来看我们建立的myLSTM函数：（122-227行）**

**第122-143行**



第122行，可以得到函数的六个参数。

在写myLSTM函数的过程中，参考了pytorch的官方文档，确定了函数的参数，输入数据的形状，以及函数的返回值。具体如下列表：

表2-1 参数列表

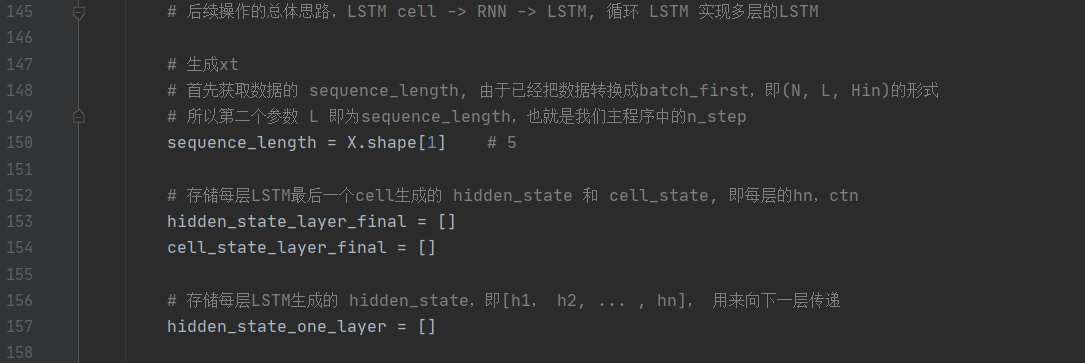
|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数说明 | 形状 |
| X | 数据 | [sequence length, batch size, input size] 或者  [batch size, sequence length, input size] |
| hidden\_size | 隐含层大小 | int（256） |
| num\_layers | LSTM的层数 | int（≥1，default=1） |
| hidden\_state | 隐层状态 | [num layers, batch size, hidden size] |
| cell\_state | 细胞状态 | [num layers, batch size, hidden size] |
| batch\_first | batch是否在首位 | bool，若为False，则X为第一种形状；若为True，则X为第二种形状 |

第123-132行，注释对数据X的shape和参数batch\_first进行了详细的说明。

第133-137行，通过对batch\_first参数的判断，调整X的shape，并获取batch size，即程序中的len(X)。

第139-143行，先判断有无初始的hidden\_state和cell\_state，若无，则创建一个默认值。

**第145-157行**



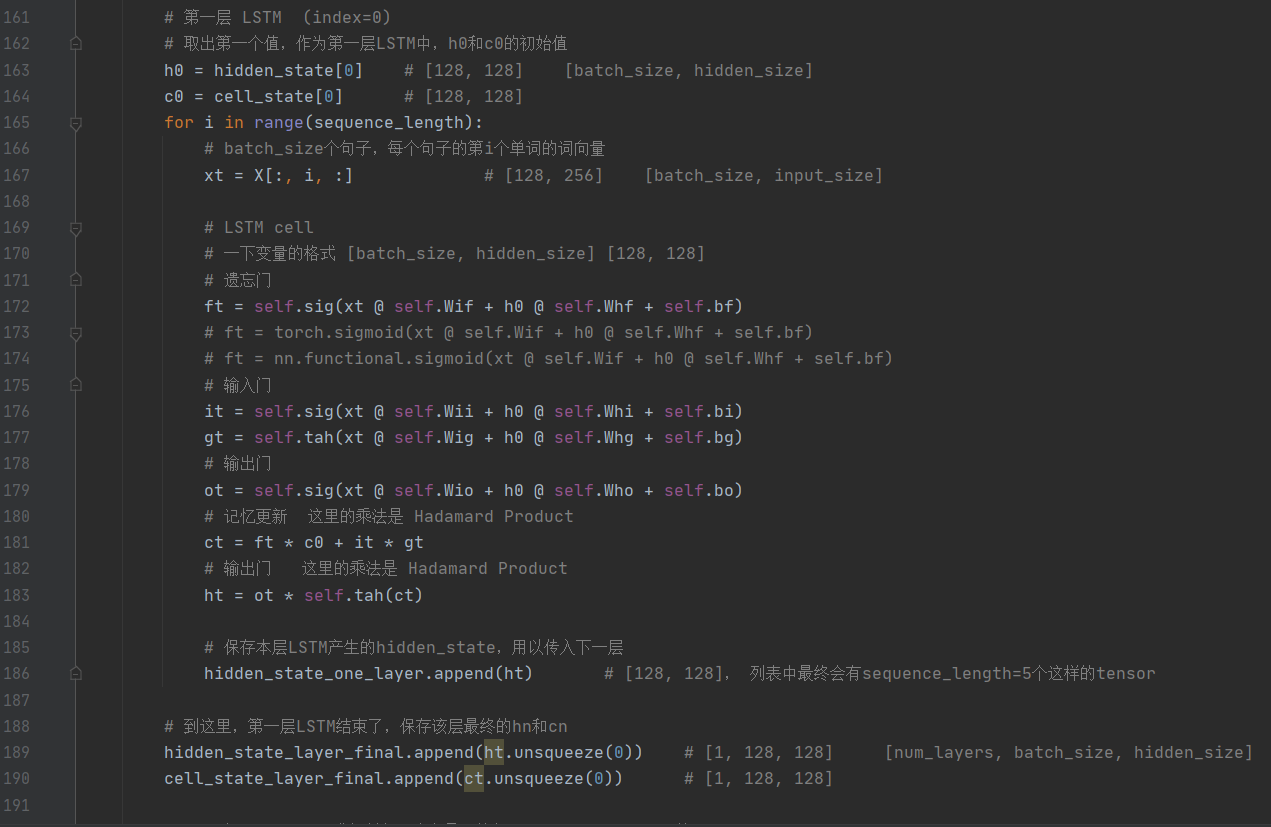
第150行，得到了序列长度，也可以直接使用主函数中的n\_step。

第153-154行，两列表分别存储每层LSTM最后一个cell生成的 hidden\_state 和 cell\_state。

第157行，存储每层LSTM生成的 hidden\_state，即[h1， h2, ... , hn]， 用来向下一层传递。

**第161-190行**

该部分代码，构建了第一层LSTM。同时，完成了核心算法的执行和相关数据的存储，为权重的传播做准备。在其中（第165-186行），以for循环的形式，完成了LSTM cell的实现。



第163-164行，获取了第一层LSTM，h0和c0的初始值。

第165行，以for循环的形式实现LSTM cell。

第167行，完成数据的变形，即batch\_size个句子，分别取每个句子的第i个单词的词向量，带入LSTM cell中。

第169-183行，完成了三个门的实现和记忆更新。

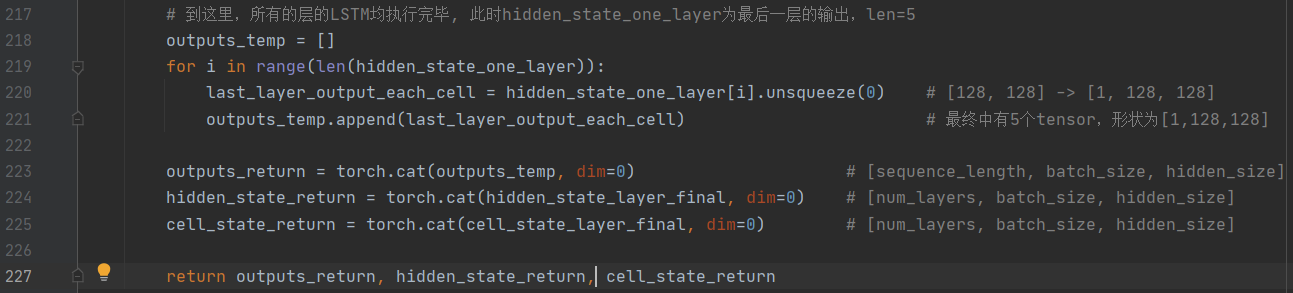
第186行，保存本层LSTM中，每一个LSTM cell产生的hidden\_state，用以传入下一层。

第189-190行，第一层的LSTM结束，将最终的hn和cn添加到列表，其shape由[128, 128]变为[1, 128, 128]。

**第192-215行**

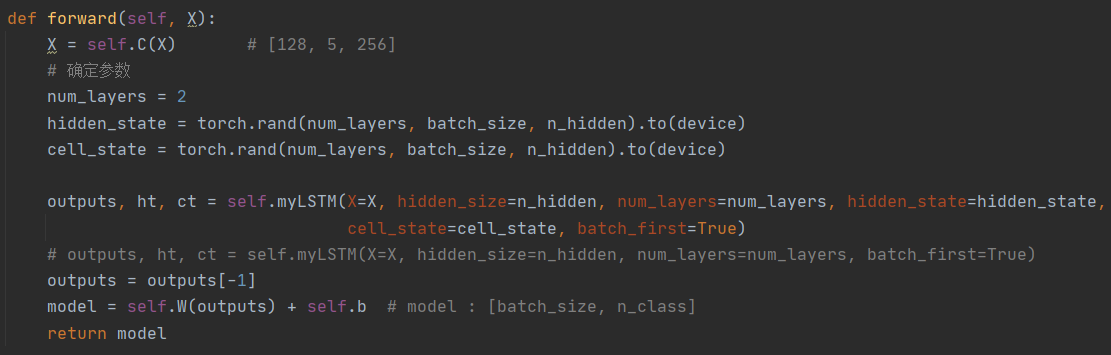
判断num\_layers是否大于等于2，若是，则意味着用户指定了多层LSTM，第二层及以后层的LSTM在这段代码中实现。代码思路与前面基本一致。

**第217-227行**



到这里，所有的LSTM层执行完毕，返回outputs，shape为[sequence\_length, batch\_size, hidden\_size]；返回hidden\_state和cell\_state，shape均为[num\_layers, batch\_size, hidden\_size]。

**2.3 最后看forward()函数：（108-120行）**



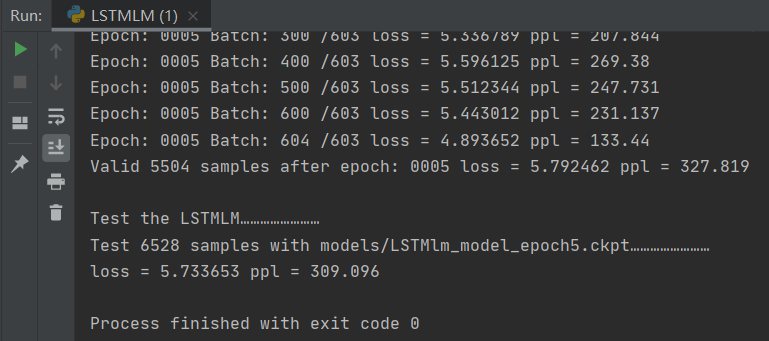
这部分主要完成了模型的前向传播过程，其中num\_layers, hidden\_state, cell\_state, batch\_first等参数，都是用户可选择的。

在最后，调整了数据的形状，从outputs中获取了我们需要的ht，经过后续线性层，完成了模型。

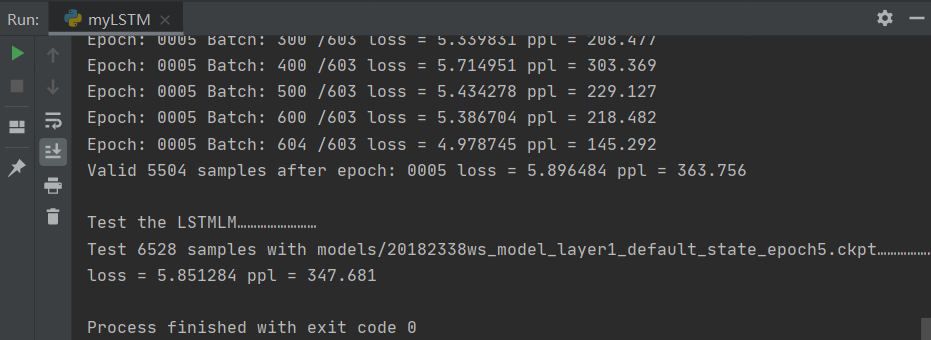
这里，我们的模型不仅完成了单层和双层的LSTM网络，理论上也可以实现任意n(n≥1)层的LSTM网络。

**3. 模型测试**

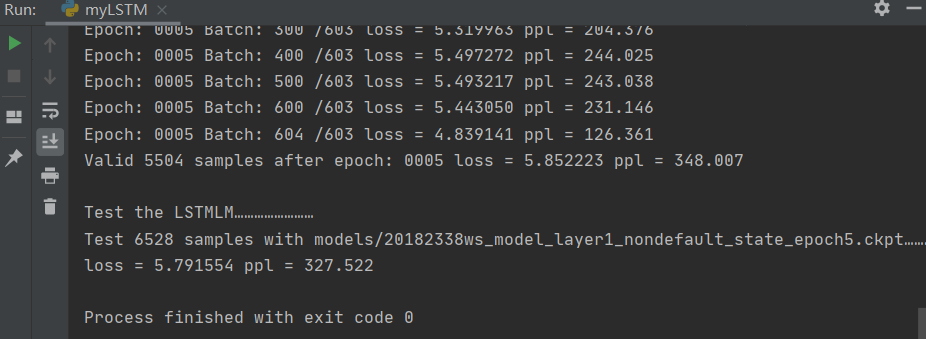
① 示例程序运行结果（层数=1， 默认hidden\_state, cell\_state）：



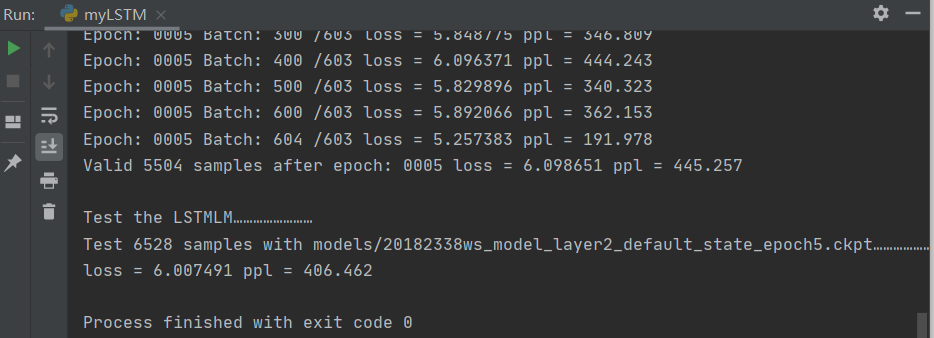
② 项目模型（层数=1，默认hidden\_state, cell\_state）:



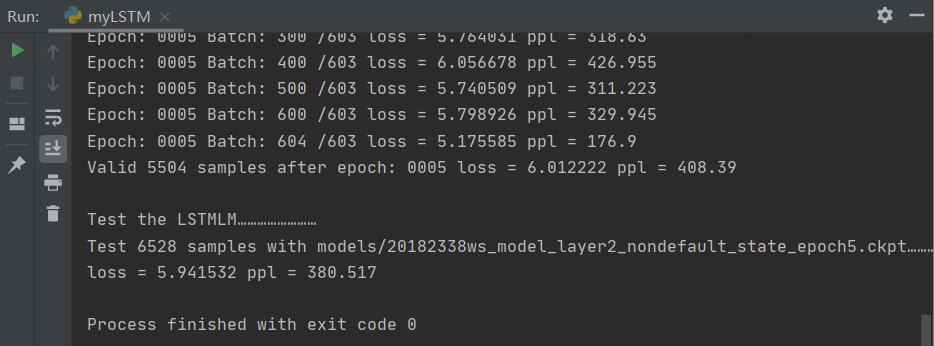
③ 项目模型（层数=1，随机hidden\_state, cell\_state）:



④ 项目模型（层数=2， 默认hidden\_state, cell\_state）:



⑤ 项目模型（层数=2， 随机hidden\_state, cell\_state）:



⑥ 项目模型（层数=3，默认hidden\_state, cell\_state）：略

⑦ 项目模型（层数=3，随机hidden\_state, cell\_state）：略

由于代码底层原因，使用项目模型时，当层数≥3时，计算速度过慢。我的GPU好像已经不允许我这么测试了。最终结果如下表：

表3-1 测试结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 模型 | LSTM层数 | hidden\cell state | loss | ppl |
| 1 | 示例模型 | 1 | default | 5.733653 | 309.096 |
| 2 | 项目模型 | 1 | default | 5.851284 | 347.681 |
| 3 | 项目模型 | 1 | random | 5.791554 | 327.522 |
| 4 | 项目模型 | 2 | default | 6.007491 | 406.462 |
| 5 | 项目模型 | 2 | random | 5.941532 | 380.517 |

**4. 问题总结**

Q1：注释中有的tensor数据的shape标注的不太准确，如示例程序LSTMLM.py的第106行。

Q2：如果使用GPU的话，构建或使用模型过程中，产生的新tensor要及时加载到GPU中，否则会报错。

Q3：自己构建的LSTM函数的接口不是很完善。

Q4：没有对传入的参数进行检查，存在安全性问题。

**Q5：在程序执行的过程中，有可能产生loss和ppl为nan的问题。原因或解决方案：① 减小学习率；② 数据归一化；③ 加入gradient clipping；④ 数据出了问题。（为了与示例程序做对比，没有加入上述操作。）**

Q6：由于底层原因，随着层数增多，运行速度变慢。