**3.3版本改进的问题**

1. 改进了dataloader的机制：把它从model\_handler移到了input\_handler里面
2. 改进了训练过程中accuracy的计算机制：之前是冗杂在model\_handler里，现在移到了output\_handler里
3. 添加了early stopping机制
4. 改进了保存模型的机制：保存训练中表现最好的模型
5. 实现了新模型：BoW-BiLSTM联合的网络
6. 把代码全部重构了一遍，简化了主运行文件的调用复杂度
7. 输出模块加入了更多的评估指标：confusion matrix, f1 score of each class；也都把这些测试结果写入了结果文件（加入confusion\_matrix图像的保存机制）
8. 补充了每个函数的代码注释
9. 补充了代码说明文档：Code description.docx

**还存在的问题**

1. BoW和BiLSTM有没有更好的融合方法
   1. 现有的就行
2. 句向量网络：加入ensemble机制（Bonus）
   1. 不搞了
3. 多类分类F1要什么机制（weighted, micro, macro）
   1. 全用，写入结果全部写，报告全部要写，但调参只用weighted
4. 需要能从命令行调用代码 - Parsing command line arguments（最终提交的版本）
5. 还差一个READ ME文件需要写
6. 模型调参要以accuracy为基准吗
   1. 用weighted f1

**4.0版本改进的问题**

1. 模型选择变成了9种，配置文件也更新了
2. BoW和BiLSTM的合成定下来了；ensemble不搞了
3. 修复了src路径引入的问题
4. 添加入F1三种不同指标的测试：micro, macro, weighted
5. 加入了GPU的适配（适合BiLSTM的调参）
6. 修复了“如果先没有train模型，跑test会保存，因为没有模型可以加载”的问题
7. 模型训练中的测试结果输出，全部换成了f1 score（包括早停机制）
8. 更新了所有的注释和code description

还存在的问题

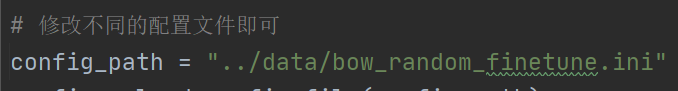
1. 模型的分配
   1. 我觉得BOW完全可以一个人来弄
   2. LSTM可以分到剩下5个人 - LSTM是要好好调整的
2. 调参的机制
   1. 目前是epoch 100; early stopping 50
   2. 可变的还有:
      1. max\_len(我觉得这个就20不变了)
      2. Lstm\_layers 和Classification layer的层数可以不用变
      3. batch\_size: 就定在64就行了
      4. learning\_rate: [0.001, 0.005, 0.01] - 当然这个是要看具体表现再调整范围的
      5. Embedding\_dimension：
         1. 使用glove时是固定300
         2. 使用random时可以调整: [100, 200, 300]
      6. Lstm\_hidden\_dim: [100, 200, 300]
   3. 调参过程中，不只是要看最后的F1是多少，过程中的training的accuracy的变化是什么，loss的趋势是什么都要看，才有助于更好的缩小参数的范围
   4. 而且调参基于的是development set，不要用到test set上去 - 看看要不要重新写一个代码
3. 模型读取方面要handle读不到的情况
4. 调参要综合GPU（那就要开一个新的分支）

**4.1版本 - 调参版本**

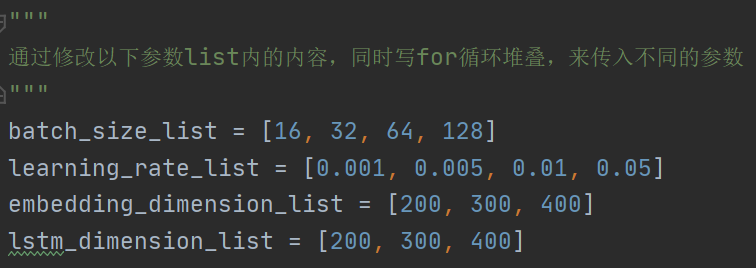
1. 更改了代码结构，使得调参更加容易

**Parameter\_tuning.py调参文件使用教程**

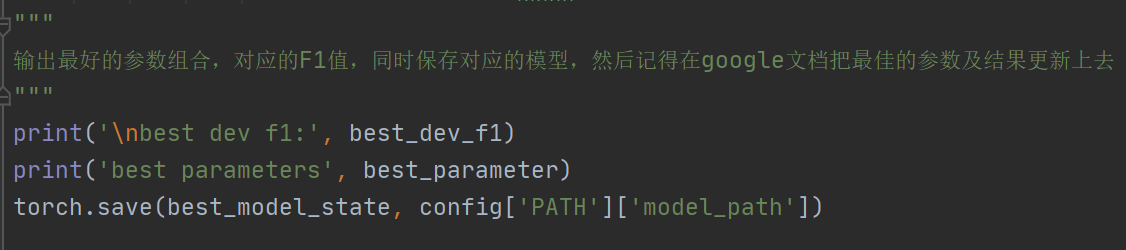
1. 修改第7行的config\_path找到自己需要调参的模型的配置文件



1. 修改26-29行的需要进行调参的参数，以及参数的范围



1. 修改后续程序的for循环以匹配当前的调参逻辑
   1. 比如如果有涉及glove的，embedding\_dimension就必须指定300不能动了，在参数范围选择和for循环中就需要剔除embedding\_dimension
   2. 比如有涉及LSTM的，在for循环时就无需加入对lstm\_dimension的调节
2. 后续代码会自动调参、记录最好的参数组合，对应的f1 score，以及保存最好的模型



1. 最后跑完了代码大家记得去共享文档里更新数据

**4.3.1版本 - 调参最新版本**

1、减少了调参的范围，不然要很久才能出结果

**每个人分配的模型：**

阳昀周：BoW-Glove-Finetune, BoW-Glove-Freeze, BoW-Random-Finetune

戚书豪: BiLSTM-Glove-Finetune, BiLSTM-Glove-Freeze

孙津梁: BoWBiLSTM-Glove-Finetune, BoWBiLSTM-Glove-Freeze

郭嘉昕: BiLSTM-Random-Finetune

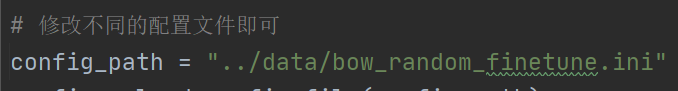
王子瑞: BOWBiLSTM-Random-Finetune

**需要调参的范围：**

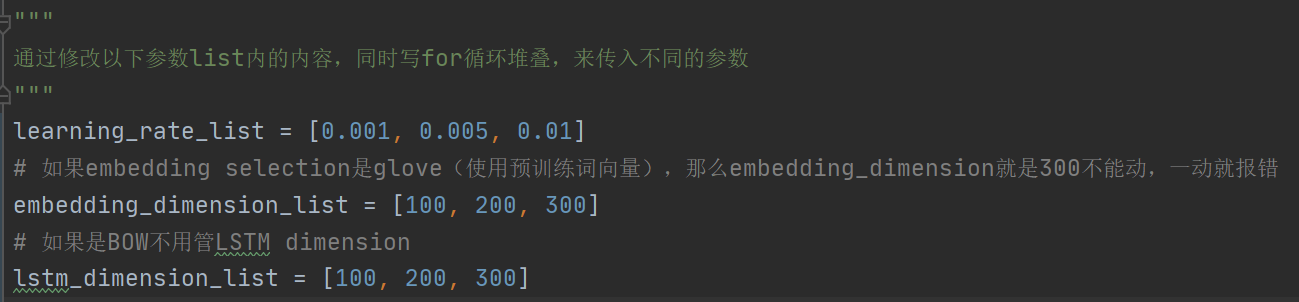
1. learning\_rate: [0.001, 0.005, 0.01]
2. Embedding\_dimension：
   1. 使用glove时是固定300
   2. 使用random时是: [100, 200, 300]
3. Lstm\_hidden\_dim: [100, 200, 300]
   1. 使用BOW，则无需考虑这一项

**Parameter\_tuning.py调参文件使用教程**

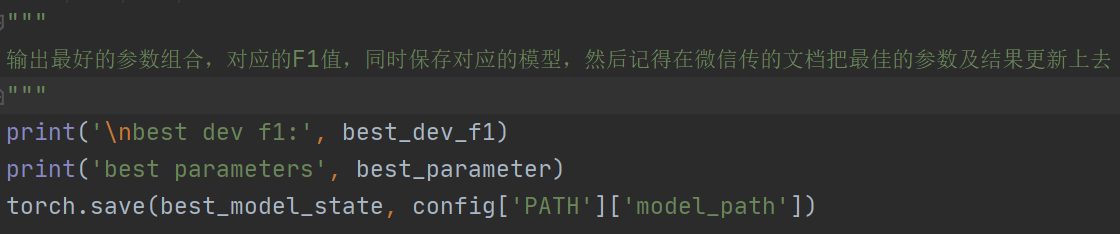
1、修改第7行的config\_path找到自己需要调参的模型的配置文件



2、修改26-30行的需要进行调参的参数，以及参数的范围



1. 修改后续程序的for循环以匹配当前的调参逻辑
   1. 有涉及glove的，embedding\_dimension就必须指定300不能动了，在参数范围选择和for循环中就需要剔除embedding\_dimension
   2. 有涉及LSTM的，在for循环时就无需加入对lstm\_dimension
   3. 反正想好模型需要的参数组合，修改参数范围和for循环的逻辑就OK
2. 后续代码会自动调参、记录最好的参数组合，对应的f1 score，以及保存最好的模型



1. 最后跑完了代码大家记得去共享文档里更新数据

**5.0版本 - 调参结束回归正式代码**

1. 数据方面
   1. 代码文件的data文件下：所有的模型都在test上做了测试，详细的results结果都写进入对应的txt文件中
   2. 微信的共享文档中：汇总了所有模型的最佳参数组合，dev上的f1，test上的accuracy和f1，这些粗略的信息（有些列我做了隐藏和折叠）
2. 代码方面
   1. 回滚到了了之前可用的代码版本
   2. 所有的配置文件，都更新为了调参中找到的最好参数组合

**目前还存在的问题**

1. 提交之前，要把所有的结果，预训练好的词向量和模型删除，只留下数据、代码、文档和配置文件；
2. 提交之前，代码需要调整为全部通过命令行去调用训练和测试；
3. 提交之前，需要再在VM上测试能不能跑；

**6.0版本 - 更改为以accuracy为基准重新跑结果**

1. 代码方面：
   1. 更新了评测机制和早停机制，一切都以accuracy为准
2. 数据方面：
   1. 更新了所有模型配置文件的参数
      1. 所有模型的embedding\_dimension和lstm\_dimension都是300+300
      2. learning\_rate：bow是0.005，bilstm和bowbilstm是0.001
      3. Epoch增加为300，early\_stopping增加为40
   2. data文件下
      1. 所有的模型都在test上做了测试，详细的results都写进入对应的txt文件中
      2. 多打印了每个模型的training\_log，记录了其在训练中输出的所有信息
         1. E.g. training\_log\_bow\_glove\_finetune.txt
   3. 微信的共享文档中：更新了所有模型在dev上的accuracy，test上的accuracy和f1

**目前还存在的问题**

1. 提交之前，要把所有的结果，预训练好的词向量和模型删除，只留下数据、代码、文档和配置文件；
2. 提交之前，代码需要调整为全部通过命令行去调用训练和测试；
3. 提交之前，需要再在VM上测试能不能跑；
4. Doc的read me和code description还没写