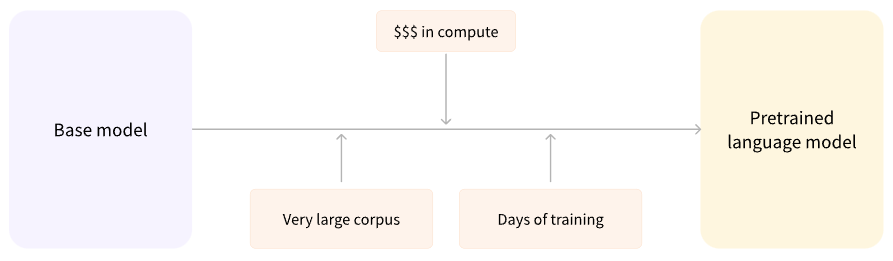
Lab2实验报告

高健翔521021910197

1. 实验设计：

采用transformer库中预训练的GPT2模型(属于解码模型)，在自行构造的数据集上进行迁移学习，转化为面向代码生成的模型，相当于将已经模型已经获得的知识转移到了新的领域, 即本实验中的代码生成。

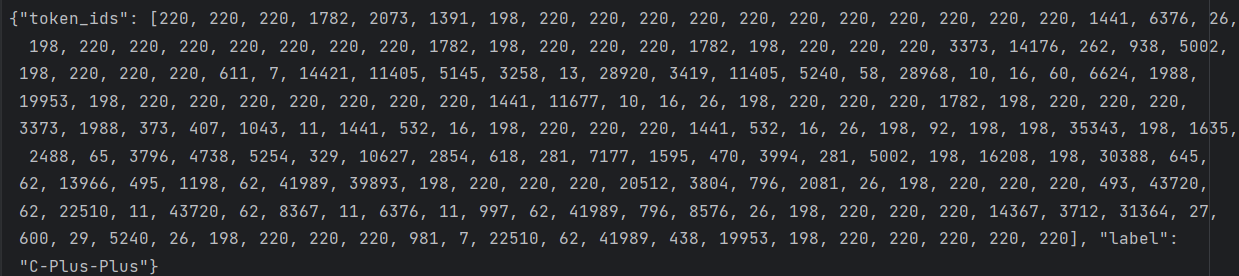


由于预训练的模型已经在大规模语料中进行训练，再使用代码语料进行微调，达到的效果可能比从头训练一个还好。

1. 训练方法：

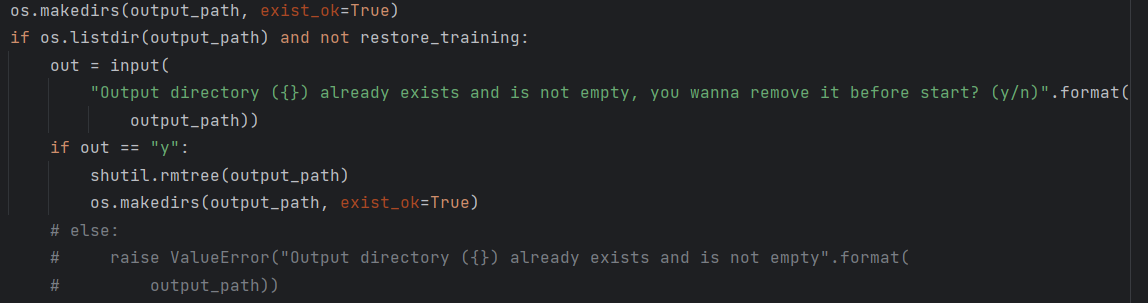
首先尝试将源代码放在自己电脑上运行，发现耗费时间较长且内存不够，于是将代码上传到kaggle上，采取训练一轮后停下，查看log并调整一些参数，然后在原来的基础上继续训练的方式。

由于原代码只提供了java和python两种语言的训练方式，我在将这两种语言进行训练之后又在数据集中添加了cpp语言进行训练，数据集通过python代码利用transformer库提供的分词器将java，python，cpp代码由token编码为input\_id，输入内容如下图所示：



然后由dataset类根据语言类型贴上对应的标签<java><cpp><python>对其进行编码之后与input\_id和eos进行拼接。

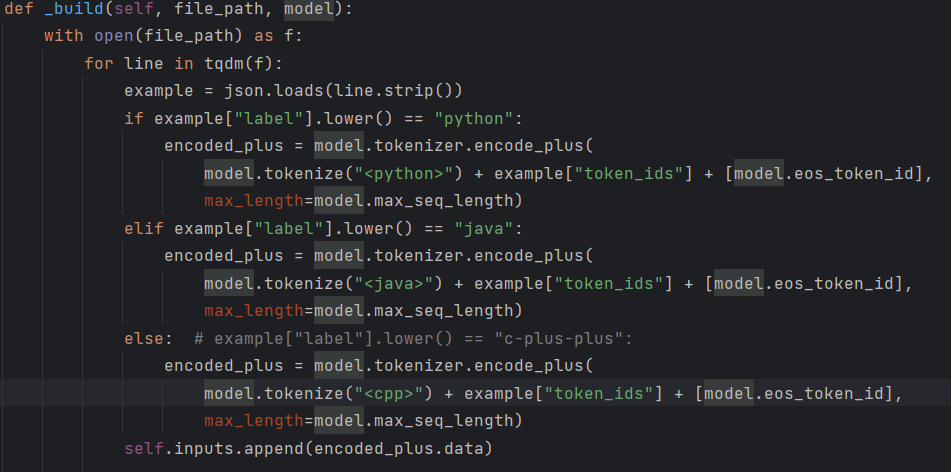
以下是对于代码的修改以适应上述的变化：



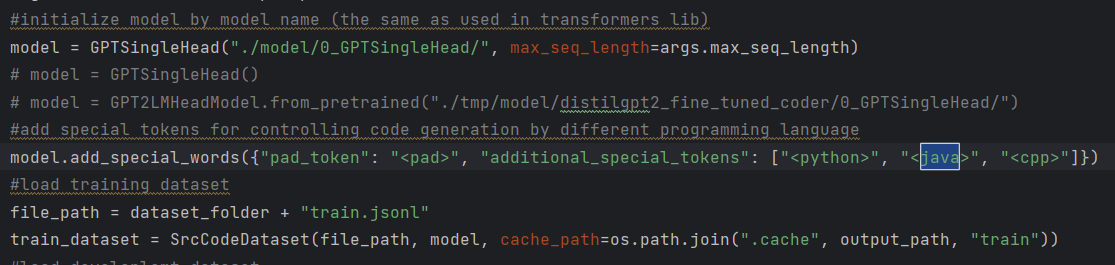
注释掉了是否删除./tmp文件夹的逻辑，kaggle好像无法应对这种运行时输入的情况.



在由源文件tokens转化为编码的时候(encode)添加c++语言的部分



在build dataset过程中添加C++数据集

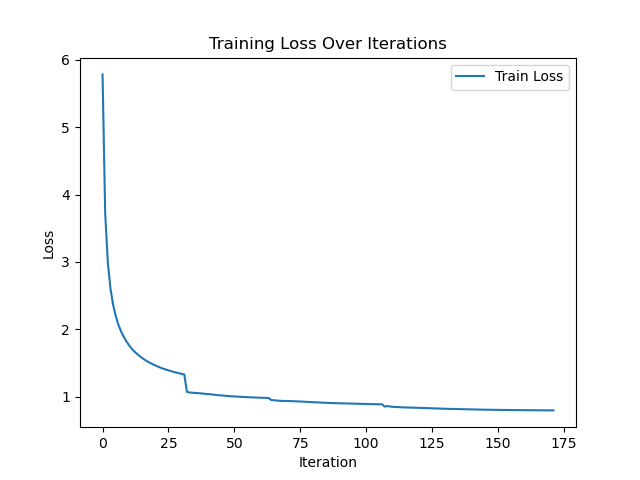


在初始化过程中添加<cpp>标签，并在训练一轮过后更改加载model的路径——从本地训练过的模型进行加载

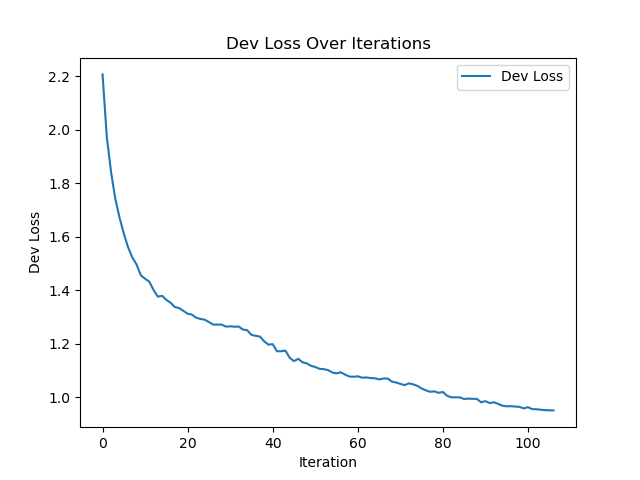
1. 训练过程及结果：
2. 在自己电脑上运行的过程中，观察log发现loss有一个大幅降低的过程，因此在kaggle第一轮训练过程中适当调大学习率，减小评估的频繁程度，期望能尽快到达局部最小值附近，之后观察第一遍训练的log，进行参数调整，尝试增大/减小batchsize，增加评估的频繁程度，使得loss尽快收敛。
3. 由于原本代码只将loss写入log文件，因此我额外写了一份代码将loss值提取出来并绘制图像，代码如下：



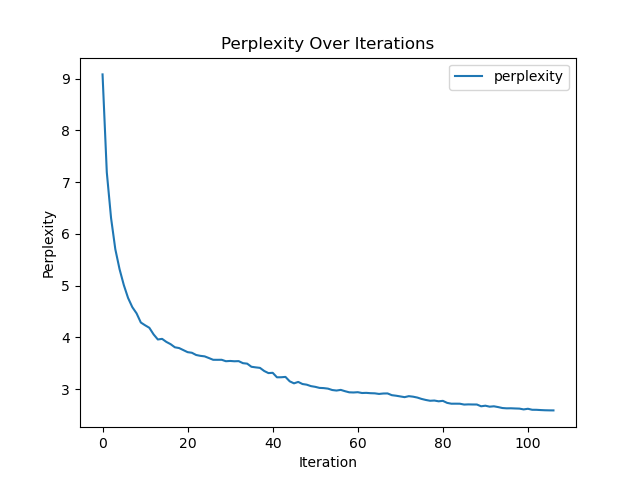
以下是对于只在<java><python>数据集上进行训练和在三种语言上进行训练的loss曲线：



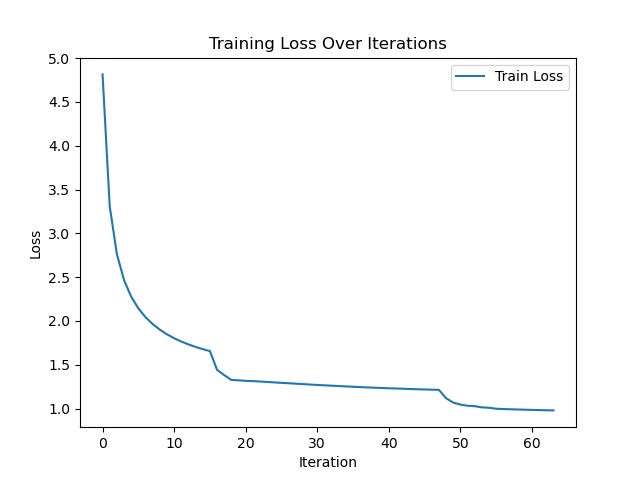
在<java><python>语料进行训练的trainloss曲线



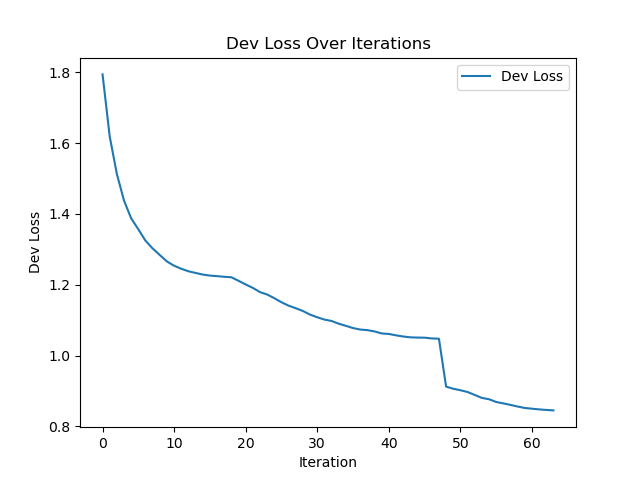
在<java><python>语料进行训练的devloss曲线



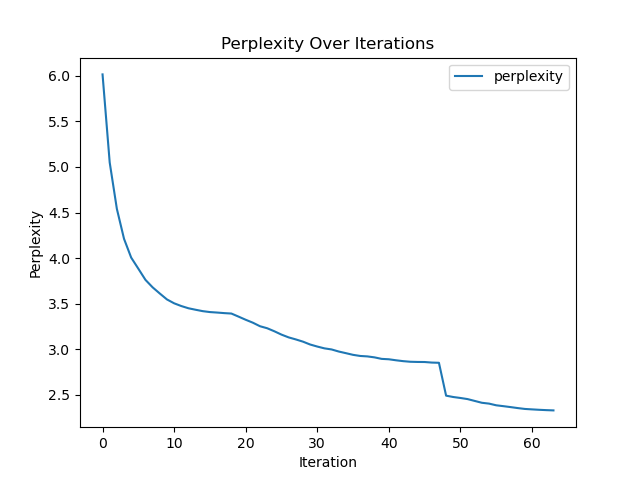
在<java><python>语料进行训练的perplexity曲线



在三种语言语料库进行训练的trainloss曲线



在三种语言语料库进行训练的devloss曲线



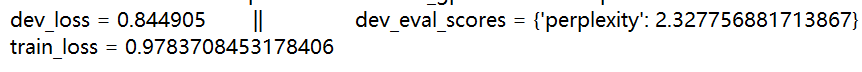
在三种语言语料库进行训练的perplexity曲线

对于loss曲线的分析：

1. 如图，可以看出loss曲线并不光滑，原因是我在kaggle上训练到第二轮的时候就会出现内存不足的情况，因此每训练一轮就会中止然后调整参数在原有的基础上继续训练，因此会出现突变的情况。
2. 在对<python><java>两种语料进行训练的过程中，出现了perplexity与devloss抖动的情况，集中在第二轮的训练过程，观察log发现，我在对于这批数据的训练过程中第一轮到第二轮并没有调整参数，可能原有的batch\_size与学习率在这一阶段已经不合适，造成了在训练集上的过度拟合。
3. 基本上loss与模糊程度都处于下降状态，事实上，在最后一轮的训练过程中发现loss的下降已经非常缓慢，由于kaggle限制gpu运行时间，我不能再次调整参数以判断loss的究竟能收敛到什么程度，以下是运行的最终结果

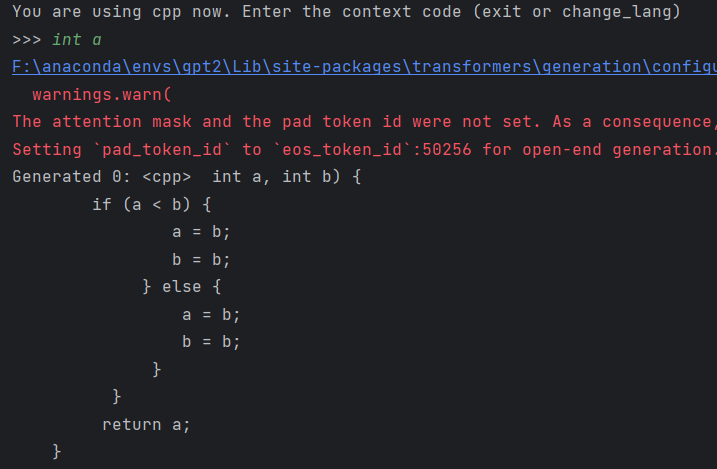


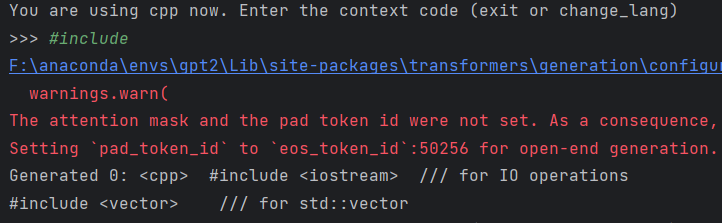
<python><java>

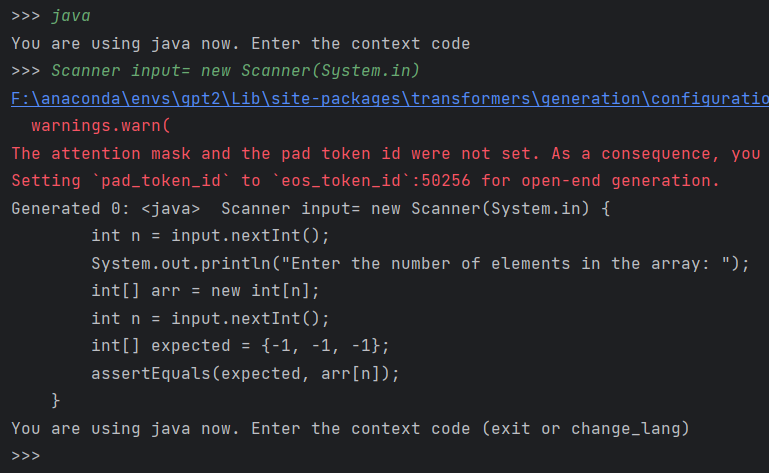


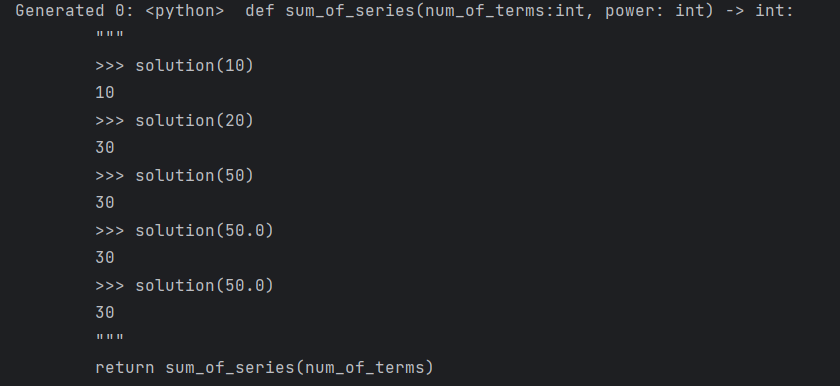
<python><java><cpp>

1. 运行效果展示：









对于python和cpp代码的生成结果并不理想，可能是因为我训练次数有限，并且为了提高速度，我减小了seq\_len，导致python函数之后基本上读入很多注释，以及cpp文件读入很多#DEFINE和#include的内容。