1. 概述

本次作业对1998年1月的人民日报做了命名实体识别，识别对象为机构名，使用预训练好的词向量模型，对词进行BIO三分类，B表示机构实体的首词，I表示机构实体的接续词，O表示非机构实体，神经网络使用了一个线性层，并使用softmax作为输出层激活函数。数据集来自北京大学计算语言学教育部重点实验室 (pku.edu.cn)，使用时先进行了预处理，梯度下降使用了小批量梯度下降，最终测试集的f1-measure 达到0.353，比上一次作业有较大下降。

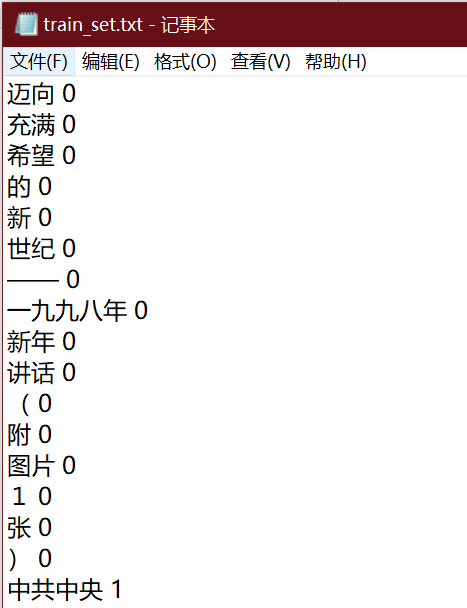
1. 模型原理

此次使用的词向量模型为50维，当前词与前后词拼接成150维的向量作为训练样本。对于BIO三种分类，我们使用一个的线性层将输入样本转化成长度为3的输出向量，再使用softmax函数将输出向量转化为和为1的概率分布，完成前向传播的过程，其中概率最大的类别便是此次预测的结果。然后再通过输出的概率向量与目标向量计算loss，通过反向传播进行梯度下降更新参数，便完成了一次训练。

对于loss的计算和梯度下降，使用了pytorch自带的cross\_entropy函数以及autograd机制。

1. 实验流程
2. 处理原始数据集

将原始数据集命名为people\_daily.txt，取19980101-19980120为训练集，19980121-19980125为验证集，19980126-19980131为测试集。调用source目录下collect\_set.py即可完成分割，得到train\_set.txt，validation\_set.txt，test\_set.txt。在分割时，对单个词作为的实体，若为机构名，标为1，否则标为0，对于多个词组成的实体，若为机构名，则将[]中的第一个词标为1，其余词标为2，否则都标为0。处理后的数据集如图所示



1. 获取词向量

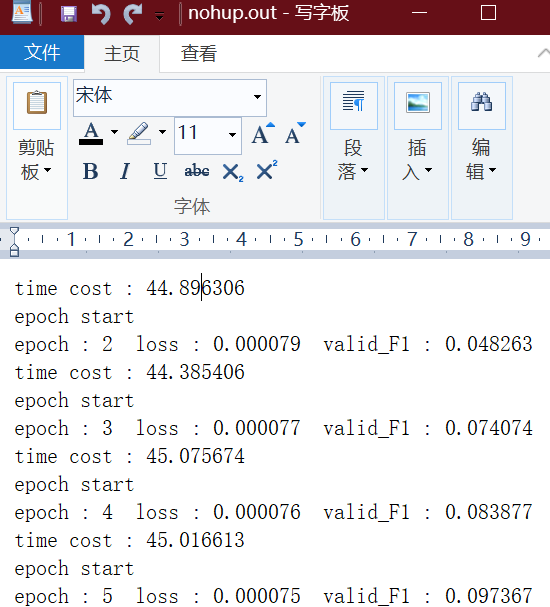
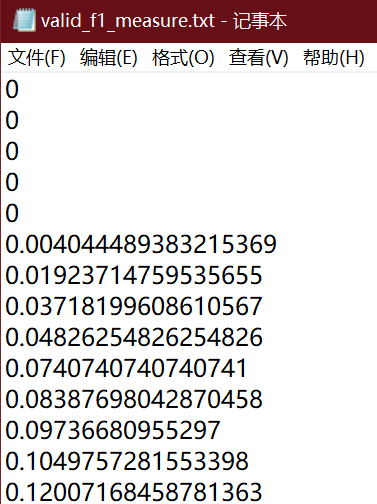
本次使用的词向量为网络资源，已预先训练好，资源链接

<https://github.com/jiesutd/LatticeLSTM>

在使用过程中，数据集中存在但词向量集中没有的词标为-unknown-。

1. 训练模型

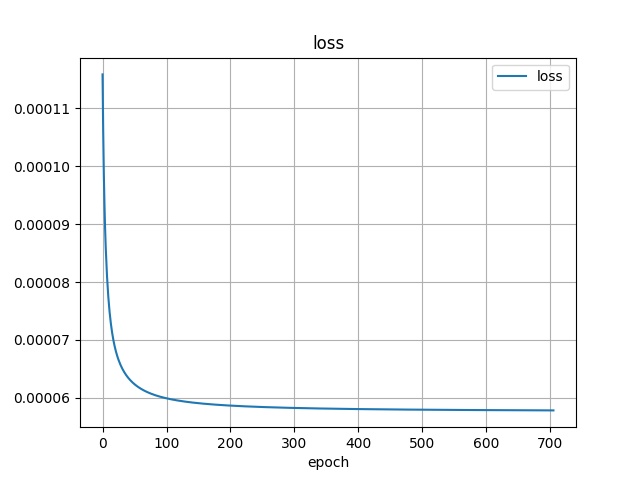
调用main.py，将词向量，训练集，验证集读入，存入自定义数据集MyDataSet中，这里使用了torch.nn中的TensorDataset类。若首次进行训练，则使用初始化的随机线性层，否则从protected data/softmax.pth中读取神经网络参数，并在训练后存入protected data/softmax.pth中。这里的线性层使用了torch.nn中的linear函数和module类。在训练时使用批量梯度下降，学习率取0.005，每次迭代使用，从MyDataset中每次获取的向量作为样本。在得到线性层的输出后，直接使用cross\_entropy函数计算logsoftmax对应的loss，然后使用backward函数进行反向传播更新参数。一次epoch训练完成后，计算训练集上的loss以及验证集上的f1-measure，并写入文件储存。loss.txt和valid\_f1\_measure.txt以及程序输出nohup.out如图所示

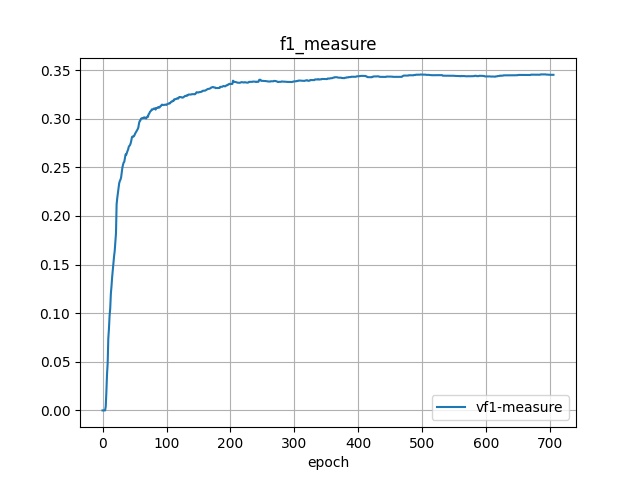


1. 结果分析

经过七百多个epoch训练后，模型的loss稳步下降至收敛，验证集的f1-measure也收敛在了0.34左右。

模型训练过后做出的预测效果较上次有较大下降，最终测试集的f1-measure仅有0.35。猜测主要原因在于模型本身，前后词与当前词拼接的方式包含的信息太少，难以做出准确预测。同时，上次作业中我手动将实体分离了出来，相当于只做了判断是否为机构名的工作，也使得上次的f1-measure偏高。





1. 总结与收获

这次作业过程中我依然花了大量的时间在网上查找相关资料，特别是pytorch的函数用法和使用技巧。最终的main.py中我用到了Module,TensorDataset,optimizer还有自带的softmax的损失函数cross\_entropy，让代码的实现更加清晰和简洁。同时我还在pytorch的官方教程中学习了代码结构，使整个代码的封装性更好，更加易读。最终实现代码只花了几个小时。

得益于上次作业中搭好了带有pytorch环境的ubuntu云服务器，这次写好代码可以直接上传到服务器上进行训练，不必担心自己的电脑过热等问题。但是目前的服务器不支持cuda，可能对后续作业的训练效率有影响，可以尝试找一下支持cuda的服务器。

在这次作业中找到一些有价值的资料

softmax loss的推导和cross entropy

https://blog.csdn.net/u014380165/article/details/79632950

https://blog.csdn.net/u014380165/article/details/77284921

softmax和log\_softmax的区别、CrossEntropyLoss() 与 NLLLoss() 的区别

https://blog.csdn.net/hao5335156/article/details/80607732

https://www.cnblogs.com/ranjiewen/p/10059490.html

https://blog.csdn.net/zcr1024/article/details/88410833

torch.nn

https://pytorch.org/tutorials/beginner/nn\_tutorial.html

https://blog.csdn.net/Spring\_24/article/details/100128412

指数分布族-GLM-softmax

https://blog.csdn.net/dataningwei/article/details/53728521