

# 作业三报告

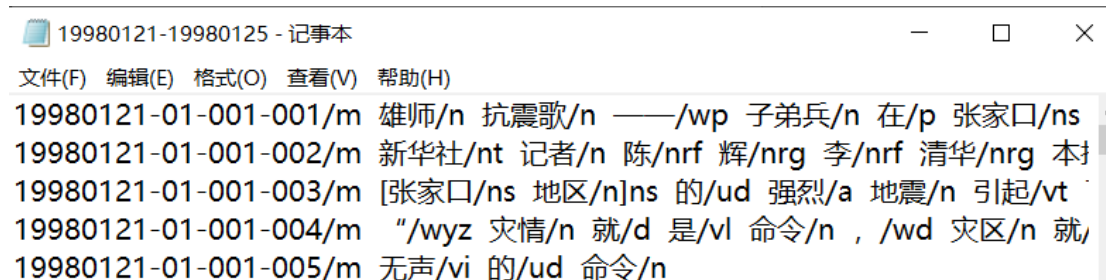
## 1 概述

本次作业对 1998 年 1 月人民日报做了命名实体识别，识别对象为机构名，使用了词嵌入表示并采用 lstm 模型实现多分类识别 BIO 标签。在实验中词嵌入为已训练完成的网络资源，其维度为 50，梯度下降采用随机梯度下降，相较于作业 2，F1-measure 有明显上升，测试集 F1-measure 可达 0.7754。

## 2.实验流程

### 2.1 分割原始数据

将原文件重命名为“renming.txt”,根据每行开头日期将其划分为三个 txt 文件，“19980101-19980120.txt”、“19980121-19980125.txt”、“19980126-19980131.txt”，分别作为训练集、验证集、测试集的初始文本文件。该步骤运行 data 文件夹下 split\_collection.py 即可完成，运行结果符合预期，以下是“19980121-19980125.txt”文件部分截图：



```
19980121-01-001-001/m 雄师/n 抗震歌/n ——/wp 子弟兵/n 在/p 张家口/ns
19980121-01-001-002/m 新华社/nt 记者/n 陈/nrf 辉/nrg 李/nrf 清华/nrg 本
19980121-01-001-003/m [张家口/ns 地区/n]ns 的/ud 强烈/a 地震/n 引起/vt
19980121-01-001-004/m “/wyz 灾情/n 就/d 是/vl 命令/n , /wd 灾区/n 就/
19980121-01-001-005/m 无声/vi 的/ud 命令/n
```

### 2.2 预处理训练集、验证集、测试集

根据获得的常用词典转化训练集、验证集、测试集，减少此后训练时的内存消耗。

分别转化三个集合，将每个词转化为 word1/word2 的形式。读取集合文件，对于每一个词判断其词性，若为机构名命名实体的首个词则标记 word2 为 1，并将此后该命名实体的其他词 word2 标记为 2，其余情况均标记为 0，具体操作方法是创建临时列表，非[]内词直接判断，[]内词需记录[]内所有词后一并修改，并根据[]词性判断词性。每段新闻占一行。

本步骤运行 predeal.py 即可完成，以下为“train.txt”文件部分截图：

并将其转化为词向量格式，输出预测值，配合 torch 库中 CrossEntropyLoss 损失函数，实现对模型的训练。F1-measure 的计算通过统计真实值和预测值中命名实体来进行，即将某一命名实体的起始序号和终止序号作为元组添加至对应的集合中，真实集和预测集的交集的大小即为 TP，即可获得查准率和查全率，进一步计算 F1-measure。

实例化 LSTM 模型，设定损失函数，定义优化器，并使用 torch 库中 Dataloader 设定迭代方式（batch 大小设置为 1），设定 epochs 参数，一个 epoch 表明训练集中每个样本都参与训练一次，每经过一次 epoch 计算一次验证集 F1-measure，每次 epoch 包含多次迭代操作，每次迭代取 1 个样本对模型进行更新，一个样本为一段文本。根据样本和模型预测结果，并用损失函数计算 loss，清零梯度后对 loss 进行反向传播，并更新参数，即完成一次迭代。

每次 epoch 后计算验证集 F1-measure 值。读入验证集文件，并根据模型进行预测，并形成验证集的命名实体预测集，和原有的命名实体真实集进行交操作即可获得 TP 值，查准率为  $TP/(\text{预测集大小})$ ，查全率为  $TP/(\text{真实集大小})$ ，进而计算 F1-measure。

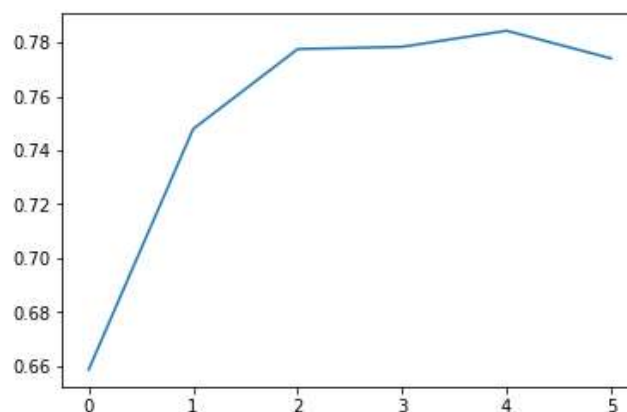
运行 main.py, 修改参数即可进行训练并得到，并生成验证集 F1-measure 上升图。

### 3. 结果及分析

本次作业采取维数为 50 的词向量，使用 LSTM 模型对机构名命名实体做 BIO 多分类处理以达到命名实体识别效果。运算时间相较前两次作业较长，但得到结果比起第二次作业 F1-measure 仅为 0.3404 的结果有了极为显著的提升。在选择合适的模型参数后，验证集 F1 在第一个 epoch 即可达到 0.65，在第五次 epoch 后可达最高值 0.78，第六次 epoch 便出现过拟合现象，停止迭代。对应计算出测试集 F1-measure 可达 0.7754。

以下为 main.py 文件运行输出以及生成的验证集 F1 上升图，其中每个 epoch 输出格式为某三个训练集中的样例的 loss 值和验证集上 F1-measure 值。

```
1.0918325185775757
0.0008369237184524536
0.023055804893374443
epoch:0 verify F1:0.6585193889541716
0.00021132301488975082
0.0002797544802532959
0.022485576570034027
epoch:1 verify F1:0.7478841870824052
8.333430741913617e-05
9.609758853912354e-05
0.017418993576048805
epoch:2 verify F1:0.7775847089487402
3.2368829124607146e-05
4.44948673248291e-05
0.0149372648447752
epoch:3 verify F1:0.7784175730432928
1.4333163562696427e-05
4.445016384124756e-05
0.008143100049955845
epoch:4 verify F1:0.7843968624125504
4.656174496631138e-06
2.142786979675293e-05
0.002776929410174489
epoch:5 verify F1:0.7740963855421686
test F1:0.775491881566379
```



### 4. 总结

根据第一次代码改写过程中发现输入条件差异较大，对数据接口做了较大的调整，并使用 torch 库中的 Embedding 类运算提高速度，但调整好输入输出条件后 F1-measure 停留在 0.3 左右，遂再次查找代码错误处，并发现是模型输入参数中步长和 batch 写反，导致每次步长相当为 1，效果与第二次作业相当，改正后便有了明显的提升。

同时因为运算较慢的问题，重新安装了 GPU 版本的 pytorch 并改写代码，每个 epoch 所需时间从 40 分钟降至 10 分钟以内，提升效果明显。

相较于第二次作业，本次作业效果提升极为显著，可见 LSTM 模型在处理词向量这种非线性特征相较线性拟合有着显著的优势。但不合理的参数选择也可能使得结果并不理想，譬如在此次实验过程中双层 LSTM 模型的结果相较单层 LSTM 表现略有不如，但尚未尝试过所

有可能的组合所以不可下定论，此后可以研究一下各种参数对实验结果的影响。