作业四报告

宋尚儒 1120180717

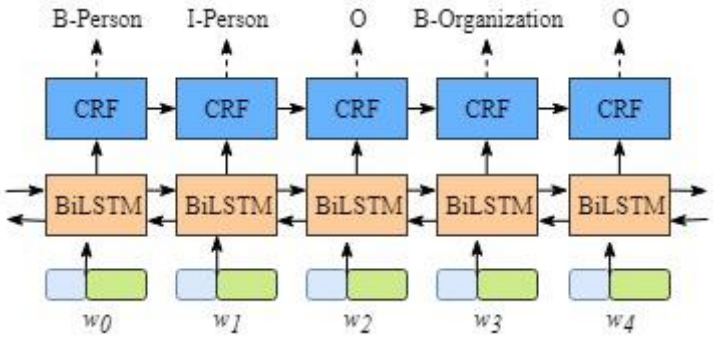
1 概述

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 BiLSTM-CRF 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，所有资源均为前三次作业提供，梯度下降采用随机梯度梯度下降，测试集 F1-measure 可达 0.7423。

2.模型原理简介

本次作业在作业 3 的基础上引入 CRF 层，其具体实现参考了 Pytorch 的官方文档，故对 CRF 层及部分模型函数做简单介绍。

条件随机场 CRF 为判别模型，常用于序列标注，可以通过定义约束保证预测的标签是合法的，在数据集训练过程中这些约束可以通过学习获得。可以将 CRF 与 Bi-LSTM 模型结合起来，在 BiLSTM-CRF 命名实体识别模型简图如下：



CRF 计算的是一种联合概率，其预测的优化目标是整个标注序列，而非将每个时刻最优值拼接，在 BiLSTM-CRF 模型中，CRF 层判别公式如下

$$\mathrm{P(y|x)} = \frac{\exp(Score(x,y))}{\sum_{y'}\exp(Score(x,y'))}$$

公式中 x 为词序列，y 为标注序列,也可以称为路径，Score(x,y)为单词序列 x 产生标记序列 y 的得分，得分越高，说明该标记序列产生概率越大。

用于 NER 的 Score(x,y)包含两个特征函数，一是转移特征函数，二是状态特征函数。

转移特征函数指标注序列中标注 i-1 到标注 i 的概率，或者称为转移分数，可以通过一个随机初始化的矩阵实现，该矩阵可以通过学习。

状态特征函数指的是词 xi 映射到各个标注的概率值向量，一个词序列 x 对应的即是一个矩阵，被称为发射矩阵，可以通过 Bi-LSTM 层输出获得。

至此，定义 Score(x,y)如下

$$\mathrm{Score(x,y)} = \sum_i A_{y_i,y_{i-1}} + \sum_i P_{i,y_i}$$

其中$A_{y_i,y_{i-1}}$表示从标签 yi-1 转移到标签 yi 的转移分数，P_{i,y_i}表示从词 xi 映射到标签 yi 的概率。

在模型训练中只需要计算最大化似然概率 P(y|x)，可以使用对数似然法：

$$\log\big(\mathrm{P(y|x)}\big) = \mathrm{Score(x,y)} - \log(\sum_{y'} e^{Score(x,y')})$$

定义损失函数 Loss 为：

$$\mathrm{Loss} = -\log\big(\mathrm{P(y|x)}\big) = \log\left(\sum_{y'} e^{Score(x,y')}\right) - \mathrm{Score(x,y)}$$

其中$\log\left(\sum_{y'} e^{Score(x,y')}\right)$计算相对复杂，因为需要计算所有可能路径的分数，可根据以下公式进行简化：

$$\log\left(\sum e^{\log(\sum e^{x})+y}\right) = \log(\sum\sum e^{x+y})$$

可以使用 dp 的思想，对于到 xi 的路径，可以先把到词 xi-1 的 log_sum_exp 值计算出来，进一步获得 xi 路径的 log_sum_exp 值，递推获得$\log\left(\sum_{y'} e^{Score(x,y')}\right)$。

在具体的代码实现中，$\log\left(\sum_{y'} \exp\big(Score(x,y')\big)\right)$通过模型函数_forward_alg()获取，$\mathrm{Score(x,y)}$通过模型函数_score_sentence()获取，在损失函数中正确调用即可获得 Loss 值。根据 Loss 值传递更新 BiLSTM 隐层和 CRF 层转移矩阵参数，即可达成训练效果。

在 BiLSTM-CRF 模型的实现代码中，会在关键部分以注释形式辅助说明，所有注释均为我手动添加，模型函数_forward_alg()和模型解码函数_viterbi_decode()较为复杂，故除了在代码处以注释形式说明外，在报告中简要说明其实现的思路细节。

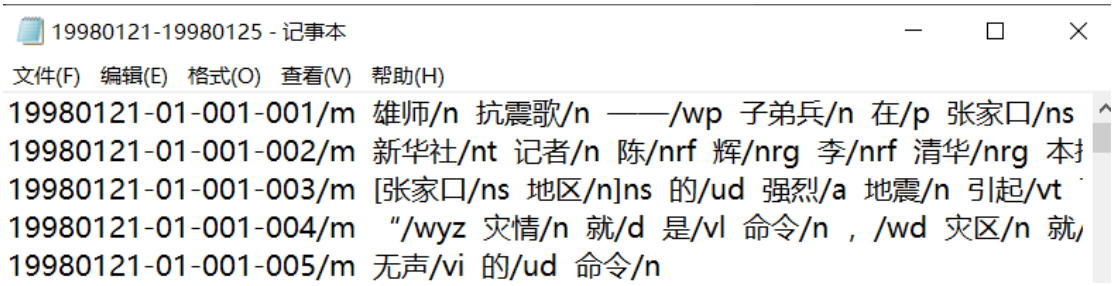
_forward_alg()用于获取损失函数中的$\log\left(\sum_{y'} e^{Score(x,y')}\right)$部分，输入为 LSTM 层获得的输出 feats，即发射矩阵，对应句子 x。设置向量 forward_var，用于存储每一步的接下来循环每一步的 log_sum_exp 值，forward_var[t]表示路径终点为标签 t 对应的 log_sum_exp 值。feats 行分解为 feat，第 i 个 feat 向量对应句子中的词 xi 映射到各个标注的概率分数值，对于每个 feat，需要遍历其下一个标注 next_tag 的所有可能（在本次实验中是 0-4），计算发射分数 emit_score（即所有值均为 feat[i]的向量），转移分数 trans_score（即转移矩阵对应值，trans_score[t]表示标签 t 转移到 next_tag 对应的分数），将两者与前向传播向量 forward_var 相加，并计算 log_sum_exp 值，所有 next_tag 对应的 log_sum_exp 值组成的向量将保存为新的 forward_var 向量。完成 feat 循环后，获得的 forward_var 向量再加上转移至 stop 标签的转移分数向量，计算 log_sum_exp 值，即可获得$\log\left(\sum_{y'} e^{Score(x,y')}\right)$

_viterbi_decode()使用了 viterbi 算法，用于模型预测，不在模型训练中使用，需要返回句子 x 对应的最优路径，即 Score 值最大的标注序列，输入为为 LSTM 层获得的输出 feats，即发射矩阵，对应句子 x。设置 backpointers 列表，用于存储后向指针列表。设置向量 forward_var，用于存储每一步的接下来循环每一步的 viterbi 向量值，forward_var[t]表示到达标签 t 的最大分数值。feats 行分解为 feat，第 i 个 feat 向量对应句子中的词 xi 映射到各个标注的概率分数值，对于每个 feat，需要遍历其下一个标注 next_tag 的所有可能（在本次实验中是 0-4），设置空列表 bptrs_t（后向指针）、viterbivar_t(viterbi 向量)，计算转移分数 trans_score（即转移矩阵对应值，trans_score[t]表示标签 t 转移到 next_tag 对应的分数）并与 forward_var 相加，获得临时向量 next_tag_var，分别将其最大值所在的索引和最大值添加记录到 bptrs_t 和 viterbivars_t 中，每一步最后需要讲 bptrs_t 添加至 backpointers 中，并用 viterbivars_t+feat 的向量值代替 forward_var。完成 feat 循环后，将获得的 forward_var 向量加上转移至 stop 标签的转移分数向量，即可获得最终的分数向量 terminal_var，terminal_var 的最大值即为可获得的最大分数，该最大值对应的索引即为最优路径的终点，然后即可将后向指针矩阵 backpointers 反向后从路径终点建立最优路径的反向路径，并将该反向路径再做一次反向，即为所要求的最优路径，最优标注序列。

3.实验流程

3.1 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：

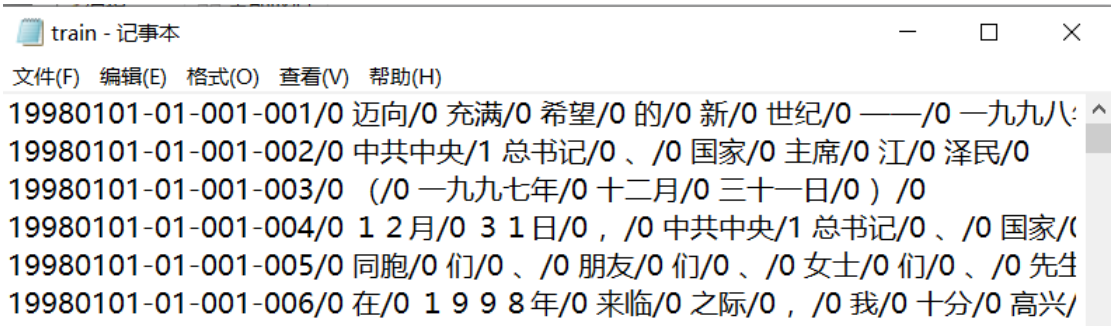


3.2 预处理训练集、验证集、测试集

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：



3.3 训练模型算 F1-measure

根据预处理过后的训练集，建立 BiLSTM-CRF 模型进行梯度下降，本次作业采取随机梯度下降的方式，每次迭代使用一文本段作为训练样本，每对训练集所有样本遍历一次计算一次验证集上的 F1-measure。

首先需要根据已有的词向量文件建立字典方便数据处理。读取预处理数据通过自定义类（需继承 torch 库中 Dataset）来完成，该类需包含三维列表，记录对应集合中每个样本对应的数据（每个样本数据包含两个 t 维张量，t 为样本句长，分别是该样本数据中每个词在字典中的序号和自身词性标签，未在字典中出现的词设置为“-unknown-”对应的序号），BiLSTM-CRF 模型如模型原理介绍进行编写，具体参数设置见 main.py，并实例化模型进行训练。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 BiLSTM-CRF 模型，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。梯度清零后，根据样本和模型中的损失函数计算 Loss，对 Loss 进行反向传播，并更新参数，即完成一次迭代。

每次 epoch 后计算验证集和测试集的 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，命名实体集的元素为二元组，分别表示一个命名实体起点索引和终点索引，查准率为 TP/(预测集大小)，查全率为 TP/(真实集大小)，进而计算 F1-measure。测试集同理。

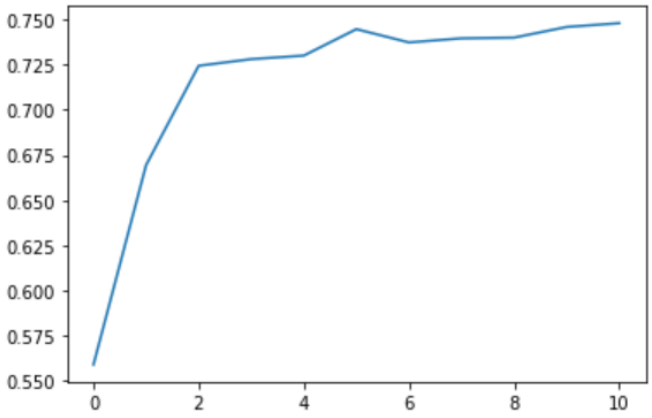
运行 main.py,修改参数即可进行训练模型，并生成验证集 F1-measure 上升图。

4.结果及分析

本次作业采取维数为 50 的词向量，使用 BiLSTM-CRF 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。在选择合适的模型参数后，验证集 F1 最高可达 0.74，对应计算出测试集 F1-measure 可达 0.7423。

以下为 main.py 文件运行的部分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 值和验证集和测试集上 F1-measure 值。

```
36.55704116821289
0.067779541015625
0.881317138671875
epoch:0 verify F1:0.5591277606933185
test F1:0.542833607907743
0.01671600341796875
0.0053863525390625
0.38608551025390625
epoch:1 verify F1:0.6693898391626244
test F1:0.6523378582202112
0.0007476806640625
0.0005950927734375
0.1004486083984375
epoch:2 verify F1:0.7243681449690034
test F1:0.6957439324657052
0.00193023681640625
7.62939453125e-05
0.07464599609375
epoch:3 verify F1:0.7280296022201664
test F1:0.7106875425459497
0.0001068115234375
1.52587890625e-05
0.09649658203125
epoch:4 verify F1:0.7299839412709337
test F1:0.7209145931405514
0.00028228759765625
3.0517578125e-05
0.00341796875
epoch:5 verify F1:0.7446140035906642
test F1:0.7423292642692181
0.0
0.0
0.00164794921875
epoch:6 verify F1:0.7372654155495978
test F1:0.7352746525479815
```



可以看到仍有训练空间，但由于时间限制，仅训练 11 个 epoch。

5.总结

在第四次作业中选择写 CRF 的起因是在阅读论文时看到 LSTM-CRF 模型的简单介绍，因为 CRF 层可以学习约束条件，其效果相较单 LSTM 模型应该有明显提升，所以准备做双向 LSTM-CRF 模型进行命名实体识别。本次作业 BiLSTM-CRF 模型代码的实现基本参考 Pytorch 官方文档，原本确实准备按照原理自己复现一个，但在实现损失函数的部分时，但并没有想出求所有路径分数 log_sum_exp 值的好方法，自己写的方法运行效率不佳，且结果相对差，对 pytorch 的掌握不熟练，最终还是参考了文档的前向算法和 viterbi 算法，其思路令人印象深刻，毕竟是第一次在知识工程课程中结合算法知识。

这次实验的结果相较于单 BiLSTM 模型结果相差不多，并没有预期中的明显提升。在选

择合理参数后才可以与单 BiLSTM 模型结果相当。推测原因有三，一是模型训练次数可能不足，CRF 层可能需要较多的训练次数，实验结果也表明仍有训练的空间，但受时间和设备限制难以进一步训练；二是原数据集质量不佳或其与该词向量数据集匹配不当，并且我在特征处理方面可能没有做到位，以后对中文 NER 任务或许可以逐字划分进行对比实验，也可以考虑自己训练词向量来使用；三是对模型的掌握仍有不足，对 BiLSTM 层的参数调优经验不足，对 CRF 层的细节实现虽然清楚原理，但尚无法在原基础上做出改进，以后可以考虑写一个单 CRF 的命名实体识别模型加深对细节的掌握，并考虑在此基础上改进模型。