实验六：PyTorch 下的神经网络训练用于 AGAC 的实体识别

2021 年 5 月 27 日

学　　校:	华中农业大学
学院班级:	信息学院生信 1801 班
姓　　名:	邓启东
学　　号:	2018317220103
指导教师:	夏静波

目录

1 实验目的

1.1 LSTM+CRF

本次实验采用 LSTM+CRF 这套代码，依旧完成 AGAC 实体识别 task1。通过 LSTM+CRF 的方式弥补单用 LSTM 本身无法对标签转移关系进行建模的问题。在 LSTM+CRF 模型中，前一类特征函数的输出由 LSTM 的输出替代，后一类特征函数就变成了标签转移矩阵。

2 数据处理

2.1 数据处理前后格式

首先我们从 Github 上将项目下载下来。

一开始语料库是 jason 格式,通过运行 json2bio.py 代码将 AGAC_train 和 AGAC_answer 这两个压缩包的据转换为 BIO 格式，分别得到 train.txt，valid.txt，test.txt。他们的内容都是单个的分词加上反斜杠再加上其标签，以空格分隔开每个词语（见图1），至此数据准备完毕。

```
Three/O subjects/O had/O compound/O heterozygotes/O genotypes/O containi
A/O variant/O allele/O of/O uncertain/O significance/O (/O p/O ./O R75Q/
ICP/O differs/O from/O other/O established/O CFTR/O -/O related/O condit
Having/O two/O CFTR/O mutations/O imparts/O a/O higher/O relative/O risk,
Mutation/O identification/O in/O a/O canine/O model/O of/O X/O -/O linke
X/B-Disease -/I-Disease linked/I-Disease hypohidrotic/I-Disease ectoderm
In/O a/O subset/O of/O affected/O individuals/O and/O animals/O ,/O muta
Ectodysplasin/O is/O a/O homotrimeric/O transmembrane/O protein/O with/O
```

图 1: 处理后数据格式

3 模型训练

3.1 训练前的处理

在整个过程开始之前还要首先运行 prepare.py 文件。

将 train.txt 中的每一个分词分成标签，字符，以及单词。并且生成存储他们的 Index 的文件，分别为 train.txt+ .char_to_idx/.word_to_idx/.tag_to_idx。

所有的参数设置可以在 parameters.py 找到。可以设置 EMBED 参数也就是 embedding 维数。

predict.py：（第 16 行）的 load_checkpoint('model.epoch20', model) 表示多少步长打印一下结果。

train.py: num_epochs = 20 也是步长检查一下训练效果。

运行 train.py 开始训练。这一步会得到训练好的模型并打印出模型结构。

3.2 出错与解决

出现报错（见下图2）：

```
    x = nn.utils.rnn.pack_padded_sequence(x, mask.sum(1).int(), batch_first = True)
  File "D:\Anaconda3\lib\site-packages\torch\nn\utils\rnn.py", line 244, in pack_padded_sequence
    _VF._pack_padded_sequence(input, lengths, batch_first)
RuntimeError: 'lengths' argument should be a 1D CPU int64 tensor, but got 1D cuda:0 Long tensor
```

图 2: 报错

解决问题：

可以发现以上报错内容提示，参数“lengths”应该使用 CPU int64，经过查阅出现以上报错的原因可能是由于 Pytorch 版本的问题，不能使用 GPU 版本进行代码的运行，此时，可以改用 CPU 进行代码尝试。

解决方法：

直接在报错指定的地方改成 CPU 进行运算。即调用.cpu() 直接添加在报错参数的后面。也就是在模型的 76 行后面加.cpu() （如下图3）。

```
76          x = nn.utils.rnn.pack_padded_sequence(x, mask.sum(1).int().cpu(), batch_first = True)
```

图 3: 改成 CPU 运算

可以看到随着模型的训练，loss 越来越小，也就是模型的拟合效果越来越高（图4）。

```
training model...
loss: 44.34273909593557
epoch = 1, loss = 44.342739, time = 21.323630
loss: 15.831912248165576
epoch = 2, loss = 15.831912, time = 20.281613
loss: 14.46740520774544
epoch = 3, loss = 14.467405, time = 20.549009
loss: 13.48328172928327
epoch = 4, loss = 13.483282, time = 21.020405
```

图 4: 训练过程

3.3 模型保存

最后到了 30 次 epoch 的时候，可以看到 (见图5)loss 已经下降到了 0.458291，已经非常小了，至此保存模型。

```
loss: 0.45829116833674444
epoch = 30, loss = 0.458291, time = 20.709930
saved model

macro precision = 0.224851
macro recall = 0.128854
macro f1 = 0.163826
micro f1 = 0.897525


Process finished with exit code 0
```

图 5: 模型训练完毕

4 标签预测

4.1 报错与解决

运行 predict.py 文件对标签进行预测，发现出现报错（见下图6）。

```
    result = predict('./prepare_data/test.txt', *load_model())
  File "E:/Desktop/NLP/作业6/LSTM_CRF_useAGAC-main (1)/LSTM_CRF_useAGAC-main/predict.py", line 37, in
predict
    text = fo.read().strip().split("\n" * (HRE + 1))
UnicodeDecodeError: 'gbk' codec can't decode byte 0xbc in position 227397: illegal multibyte sequence
```

图 6: 报错

这个报错比较简单, 就是打开文件的时候 txt 是 utf-8 编码，所以编码需要转为 utf-8，因此要在每一个文件打开的语句中间加上,encoding='utf-8'。分别在第 31 行和第 61 行添加上以后。报错解决

4.2 预测完成

看到下面的内容，说明预测已经完成了 (图9)。

```
loading model.epoch30
epoch = 30, loss = 0.458291

Process finished with exit code 0
```

图 7: 报错

5 结果评估

5.1 github 代码笔误

评估代码如下：

perl conlleval.pl -d $'\t' <test_out.tab | tee test_out_lstm.eval

却发生了报错，表明没有参数（见下图8）。

```
lianze@LAPTOP-5JCNK6CP:~/task6$ perl conlleval.pl  -d $'\t' <test_out.tab | tee test_out_lstm.eval
conlleval: unexpected command line argument
```

图 8: 报错

经仔细检查，github 上代码有误, 原因在于-d 的反斜杠是全角还是半角的问题，需要改成英文的斜杠 “-”。

5.2 评估结果

```
lianze@LAPTOP-5JCNK6CP:~/task5/2021Spring_CRF_AGACtask1-master$ perl conlleval.pl -d
C/train_out.eval
processed 62559 tokens with 2416 phrases; found: 1091 phrases; correct: 349.
accuracy:  91.64%; precision:  31.99%; recall:  14.45%; FB1:  19.90
              CPA: precision:  16.00%; recall:   7.14%; FB1:   9.88  25
          Disease: precision:  23.90%; recall:  15.55%; FB1:  18.84  272
          Enzyme: precision:   0.00%; recall:   0.00%; FB1:   0.00  1
             Gene: precision:  26.47%; recall:   3.86%; FB1:   6.74  68
      Interaction: precision:  10.00%; recall:  14.29%; FB1:  11.76  10
              MPA: precision:  15.94%; recall:   5.47%; FB1:   8.15  69
           NegReg: precision:  56.45%; recall:  27.34%; FB1:  36.84  62
          Pathway: precision:   0.00%; recall:   0.00%; FB1:   0.00  1
           PosReg: precision:  35.71%; recall:  24.39%; FB1:  28.99  56
          Protein: precision:   0.00%; recall:   0.00%; FB1:   0.00  3
              Reg: precision:  54.05%; recall:   5.03%; FB1:   9.20  37
              Var: precision:  35.93%; recall:  30.86%; FB1:  33.21  487
```

图 9: 最终评估结果

至此，我们本次 LSTM+CRF 训练模型、预测标签、评估结果一套流程全部完毕。可以看到预测的效果也比较一般，想要进一步提高效果，可以修改参数。例如循环次数。

6 参考链接

1. Pytorch 出现的小问题
2. Rulcy 的 CSDN 博客