Project2

一、HMM

（一）原理理解

在HMM中，我们可以把系统想象成一个有着隐藏状态和可观测状态的模型。隐藏状态表示我们不直接观察到的系统内部状态，而可观测状态则是我们能够观察到的数据。

HMM假设我们可以通过一个**发射概率矩阵**描述每个隐藏状态生成不同观测状态的概率，同时使用一个**转移概率矩阵**描述每个隐藏状态在下一时刻转移到其他隐藏状态的概率。此外，HMM还考虑了**初始标签概率**，即系统在起始时刻处于各个隐藏状态的概率分布。可以通过在训练集上利用**统计计算**的方法求解概率矩阵。

维特比解码的实质是在给定观测序列的情况下，寻找最有可能的对应隐状态序列。通过动态规划递归地计算每个时间步的最优路径，我们可以得到整个序列的最优路径和概率。

（二）代码基本结构与实验过程

代码包含初始化读取函数init()和类HMM。

- init(): 读取训练文件，得到wordlists，taglists，以及词汇表words；读取测试文件，得到wordlists
- 类HMM
  - train(): 根据从训练文件读取得到的wordlists和taglists，统计并计算初始标签概率、发射概率矩阵和转移概率矩阵。
  - viterbi(): 实现了HMM的维特比解码，通过动态规划寻找在给定观测序列（wordlist）下最有可能的隐状态序列。
  - test(): 通过调用之前定义的`viterbi`方法，对输入的多个词汇序列进行测试，并将结果写入一个输出文件中（后续借助`check.py`进行测试）。

以中文识别为例，最开始实现代码后，运行测试发现正确率很低，如下：

```
yuki@MacBook NER % python3 check.py
              precision    recall  f1-score   support

      B-NAME     0.0000    0.0000    0.0000       102
      M-NAME     0.0000    0.0000    0.0000        75
      E-NAME     0.0000    0.0000    0.0000       102
      S-NAME     0.0000    0.0000    0.0000         8
      B-CONT     0.0000    0.0000    0.0000        33
      M-CONT     0.0000    0.0000    0.0000        64
      E-CONT     0.0000    0.0000    0.0000        33
      S-CONT     0.0000    0.0000    0.0000         0
       B-EDU     0.0000    0.0000    0.0000       106
       M-EDU     0.0000    0.0000    0.0000       177
       E-EDU     0.0000    0.0000    0.0000       106
       S-EDU     0.0000    0.0000    0.0000         0
     B-TITLE     0.5476    0.0334    0.0629       689
     M-TITLE     0.4737    0.0609    0.1078      1479
     E-TITLE     0.0476    0.0029    0.0055       689
     S-TITLE     0.0000    0.0000    0.0000         0
       B-ORG     0.0734    0.0153    0.0254       522
       M-ORG     0.3990    0.2399    0.2997      3622
       E-ORG     0.5476    0.0441    0.0816       522
       S-ORG     0.0000    0.0000    0.0000         0
      B-RACE     0.0000    0.0000    0.0000        14
      M-RACE     0.0000    0.0000    0.0000         0
      E-RACE     0.0000    0.0000    0.0000        14
      S-RACE     0.0000    0.0000    0.0000         1
       B-PRO     0.0000    0.0000    0.0000        18
       M-PRO     0.0000    0.0000    0.0000        33
       E-PRO     0.0000    0.0000    0.0000        18
       S-PRO     0.0000    0.0000    0.0000         0
       B-LOC     0.0000    0.0000    0.0000         2
       M-LOC     0.0000    0.0000    0.0000         6
       E-LOC     0.0000    0.0000    0.0000         2
       S-LOC     0.0000    0.0000    0.0000         0

   micro avg     0.3884    0.1203    0.1837      8437
   macro avg     0.0653    0.0124    0.0182      8437
weighted avg     0.3414    0.1203    0.1597      8437
```

在代码中使用平滑技术处理未见过的单词或单词-标签组合，以防止零概率，提高模型的泛化能力，处理后正确率上升至50%以上，但依旧低；

后续又发现在维特比解码过程中可能会出现溢出，所以在解码前对于概率进行取对数操作，之后的预测结果正确率可以达到90%，如下图：

```
yuki@MacBook NER % python3 check.py
              precision    recall  f1-score   support

      B-NAME     0.9174    0.9804    0.9479       102
      M-NAME     0.9136    0.9867    0.9487        75
      E-NAME     0.9083    0.9706    0.9384       102
      S-NAME     1.0000    0.5000    0.6667         8
      B-CONT     0.8649    0.9697    0.9143        33
      M-CONT     0.8750    0.9844    0.9265        64
      E-CONT     0.8919    1.0000    0.9429        33
      S-CONT     0.0000    0.0000    0.0000         0
       B-EDU     0.8065    0.9434    0.8696       106
       M-EDU     0.7919    0.9887    0.8794       177
       E-EDU     0.8455    0.9811    0.9083       106
       S-EDU     0.0000    0.0000    0.0000         0
     B-TITLE     0.8194    0.8694    0.8437       689
     M-TITLE     0.8129    0.9020    0.8551      1479
     E-TITLE     0.9191    0.9724    0.9450       689
     S-TITLE     0.0000    0.0000    0.0000         0
       B-ORG     0.8869    0.9310    0.9084       522
       M-ORG     0.9185    0.9216    0.9201      3622
       E-ORG     0.7832    0.8238    0.8030       522
       S-ORG     0.0000    0.0000    0.0000         0
      B-RACE     0.9333    1.0000    0.9655        14
      M-RACE     0.0000    0.0000    0.0000         0
      E-RACE     0.9333    1.0000    0.9655        14
      S-RACE     0.0000    0.0000    0.0000         1
       B-PRO     0.4615    0.6667    0.5455        18
       M-PRO     0.5000    0.6364    0.5600        33
       E-PRO     0.5769    0.8333    0.6818        18
       S-PRO     0.0000    0.0000    0.0000         0
       B-LOC     0.0000    0.0000    0.0000         2
       M-LOC     0.0000    0.0000    0.0000         6
       E-LOC     0.0000    0.0000    0.0000         2
       S-LOC     0.0000    0.0000    0.0000         0

   micro avg     0.8697    0.9147    0.8916      8437
   macro avg     0.5425    0.5894    0.5605      8437
weighted avg     0.8714    0.9147    0.8918      8437
```

（三）实验结果

英文：

```
yuki@MacBook NER % python3 check.py
              precision    recall  f1-score   support

       B-PER     0.9435    0.5261    0.6755      1842
       I-PER     0.9211    0.6611    0.7697      1307
       B-ORG     0.7701    0.5846    0.6647      1341
       I-ORG     0.8053    0.5672    0.6656       751
       B-LOC     0.9129    0.7529    0.8252      1837
       I-LOC     0.8462    0.6848    0.7570       257
      B-MISC     0.9412    0.6594    0.7755       922
      I-MISC     0.8097    0.5289    0.6399       346

   micro avg     0.8831    0.6269    0.7332      8603
   macro avg     0.8688    0.6206    0.7216      8603
weighted avg     0.8859    0.6269    0.7309      8603
```

中文：

```
yuki@MacBook NER % python3 check.py
              precision    recall  f1-score   support

      B-NAME     0.9174    0.9804    0.9479       102
      M-NAME     0.9136    0.9867    0.9487        75
      E-NAME     0.9083    0.9706    0.9384       102
      S-NAME     1.0000    0.5000    0.6667         8
      B-CONT     0.8649    0.9697    0.9143        33
      M-CONT     0.8750    0.9844    0.9265        64
      E-CONT     0.8919    1.0000    0.9429        33
      S-CONT     0.0000    0.0000    0.0000         0
       B-EDU     0.8065    0.9434    0.8696       106
       M-EDU     0.7919    0.9887    0.8794       177
       E-EDU     0.8455    0.9811    0.9083       106
       S-EDU     0.0000    0.0000    0.0000         0
     B-TITLE     0.8194    0.8694    0.8437       689
     M-TITLE     0.8129    0.9020    0.8551      1479
     E-TITLE     0.9191    0.9724    0.9450       689
     S-TITLE     0.0000    0.0000    0.0000         0
       B-ORG     0.8869    0.9310    0.9084       522
       M-ORG     0.9185    0.9216    0.9201      3622
       E-ORG     0.7832    0.8238    0.8030       522
       S-ORG     0.0000    0.0000    0.0000         0
      B-RACE     0.9333    1.0000    0.9655        14
      M-RACE     0.0000    0.0000    0.0000         0
      E-RACE     0.9333    1.0000    0.9655        14
      S-RACE     0.0000    0.0000    0.0000         1
       B-PRO     0.4615    0.6667    0.5455        18
       M-PRO     0.5000    0.6364    0.5600        33
       E-PRO     0.5769    0.8333    0.6818        18
       S-PRO     0.0000    0.0000    0.0000         0
       B-LOC     0.0000    0.0000    0.0000         2
       M-LOC     0.0000    0.0000    0.0000         6
       E-LOC     0.0000    0.0000    0.0000         2
       S-LOC     0.0000    0.0000    0.0000         0

   micro avg     0.8697    0.9147    0.8916      8437
   macro avg     0.5425    0.5894    0.5605      8437
weighted avg     0.8714    0.9147    0.8918      8437
```

二、CRF

（一）原理理解

CRF的核心思想是考虑观测序列和标签序列之间的条件概率分布，通过最大化这个条件概率来预测最可能的标签序列。

在构建CRF模型时，我们通过对观测序列生成一系列**特征**，并为这些特征定义对应的特征函数。这些特征可以涵盖丰富的信息，包括自身词汇、上下文关系以及标签之间的关系，从而综合考虑多方面的因素。

在特征的基础上，CRF通过对这些特征进行加权评估，引入权重参数。这些权重参数是可以根据模型的训练数据进行动态调整的，通过训练过程中的优化算法，使得模型学到合适的权重参数。这样，CRF就能够在考虑多种特征的基础上，通过权衡不同特征对标签序列的影响，达到更准确的序列标注任务。

（二）代码基本结构与实验过程

本Part可以使用机器学习框架，代码上的工作主要是对特征的提取。

代码包含初始化读取函数init(), word2features(), sent2features()和类CRF。

- init(): 读取训练文件，得到wordlists，taglists；读取测试文件，得到wordlists。
- word2features(): 提取单词的特征（pre、nxt；对于英文也提取首字母是否大写）
- sent2features(): 对句子逐个单词的提取特征。
- 类CRF
  - train(): 利用提取的特征，调用sklearn框架进行训练。
  - test(): 调用sklearn框架对输入的多个词汇序列进行测试，并将结果写入一个输出文件中（后续借助`check.py`进行测试）。

（三）实验结果

英文：

```
yuki@MacBook NER % python3 check.py
              precision    recall  f1-score   support

       B-PER     0.8965    0.8746    0.8854      1842
       I-PER     0.9104    0.9564    0.9328      1307
       B-ORG     0.8559    0.7927    0.8231      1341
       I-ORG     0.7151    0.8322    0.7692       751
       B-LOC     0.8802    0.8639    0.8720      1837
       I-LOC     0.9248    0.8132    0.8654       257
      B-MISC     0.9217    0.8167    0.8660       922
      I-MISC     0.8975    0.7341    0.8076       346

   micro avg     0.8737    0.8546    0.8640      8603
   macro avg     0.8753    0.8355    0.8527      8603
weighted avg     0.8765    0.8546    0.8641      8603
```

中文：

```
yuki@MacBook NER % python3 check.py
              precision    recall  f1-score   support

      B-NAME     0.9901    0.9804    0.9852       102
      M-NAME     1.0000    0.9733    0.9865        75
      E-NAME     0.9901    0.9804    0.9852       102
      S-NAME     1.0000    1.0000    1.0000         8
      B-CONT     1.0000    1.0000    1.0000        33
      M-CONT     1.0000    1.0000    1.0000        64
      E-CONT     1.0000    1.0000    1.0000        33
      S-CONT     0.0000    0.0000    0.0000         0
       B-EDU     0.9906    0.9906    0.9906       106
       M-EDU     1.0000    1.0000    1.0000       177
       E-EDU     0.9906    0.9906    0.9906       106
       S-EDU     0.0000    0.0000    0.0000         0
     B-TITLE     0.9202    0.9202    0.9202       689
     M-TITLE     0.8921    0.9337    0.9125      1479
     E-TITLE     0.9826    0.9826    0.9826       689
     S-TITLE     0.0000    0.0000    0.0000         0
       B-ORG     0.9626    0.9368    0.9495       522
       M-ORG     0.9455    0.9671    0.9562      3622
       E-ORG     0.9272    0.9023    0.9146       522
       S-ORG     0.0000    0.0000    0.0000         0
      B-RACE     1.0000    1.0000    1.0000        14
      M-RACE     0.0000    0.0000    0.0000         0
      E-RACE     1.0000    1.0000    1.0000        14
      S-RACE     0.0000    0.0000    0.0000         1
       B-PRO     0.8571    1.0000    0.9231        18
       M-PRO     0.8049    1.0000    0.8919        33
       E-PRO     0.8571    1.0000    0.9231        18
       S-PRO     0.0000    0.0000    0.0000         0
       B-LOC     1.0000    1.0000    1.0000         2
       M-LOC     1.0000    1.0000    1.0000         6
       E-LOC     1.0000    1.0000    1.0000         2
       S-LOC     0.0000    0.0000    0.0000         0

   micro avg     0.9405    0.9553    0.9478      8437
   macro avg     0.7222    0.7362    0.7285      8437
weighted avg     0.9410    0.9553    0.9479      8437
```

三、BiLSTM+CRF

（一）原理理解

BiLSTM-CRF模型主体由Bi-LSTM和CRF组成，模型输入是字符特征，输出是每个字符对应的预测标签。

BiLSTM接收每个字符的embedding，并预测每个字符的对所有标签的概率。但直接选择该步骤最大概率的标签类别得到的结果并不理想。这样的模型无法学习到输出的标注之间的**转移依赖关系**（标签的概率转移矩阵）以及**序列标注的约束条件**，所以引入CRF层学习序列标注的约束条件，确保预测结果的有效性。

CRF层将BiLSTM的Emission_score作为输入，输出最大可能的预测标注序列。在训练过程中，对于某一序列，需要计算模型的真实路径得分和所有路径得分；训练目的即提高真实路径得分在所有路径得分中的占比，其最大似然函数可以表示为：

$$\log P(y \mid x) = \text{score}(x, y) - \log\left(\sum_{y} \exp(\text{score}(x, y))\right)$$

（二）代码基本结构与实验过程

代码包含类BiLSTM_CRF、类CRF、类NERdataset。

- 类CRF
  - _all_path(): 计算所有可能路径得分之和。
  - _real_path(): 计算真实路径得分。
  - calc_loss(): 计算损失函数。
  - viterbi(): 维特比解码，用于预测。
- 类BiLSTM_CRF
  - forward(): 前向传播，依次经过嵌入层、LSTM层、全连接层、CRF层。
- 类NERdataset: 为了构造数据加载器，继承自类Dataset。
- train(): 训练函数，并将模型保存至`ckpts/[language]_BiLSTM_CRF_Model.pkl`
- test(): 测试函数，调用已训练好的模型。

（三）实验结果

英文：

```
● yuki@MacBook NER % python3 check.py
               precision    recall  f1-score   support

        B-PER     0.9542    0.7693    0.8518      1842
        I-PER     0.9598    0.8409    0.8964      1307
        B-ORG     0.8558    0.8009    0.8274      1341
        I-ORG     0.8805    0.8242    0.8514       751
        B-LOC     0.9585    0.8677    0.9109      1837
        I-LOC     0.9624    0.7977    0.8723       257
       B-MISC     0.9513    0.8265    0.8845       922
       I-MISC     0.9222    0.7197    0.8084       346

    micro avg     0.9315    0.8159    0.8699      8603
    macro avg     0.9306    0.8058    0.8629      8603
 weighted avg     0.9329    0.8159    0.8697      8603
```

中文：

```
yuki@MacBook NER % python3 check.py
              precision    recall  f1-score   support

      B-NAME     0.9901    0.9804    0.9852       102
      M-NAME     0.9241    0.9733    0.9481        75
      E-NAME     0.9802    0.9706    0.9754       102
      S-NAME     1.0000    0.8750    0.9333         8
      B-CONT     1.0000    1.0000    1.0000        33
      M-CONT     1.0000    1.0000    1.0000        64
      E-CONT     1.0000    1.0000    1.0000        33
      S-CONT     0.0000    0.0000    0.0000         0
       B-EDU     0.9636    1.0000    0.9815       106
       M-EDU     0.9831    0.9887    0.9859       177
       E-EDU     0.9811    0.9811    0.9811       106
       S-EDU     0.0000    0.0000    0.0000         0
     B-TITLE     0.9419    0.9405    0.9412       689
     M-TITLE     0.9521    0.9412    0.9466      1479
     E-TITLE     0.9956    0.9956    0.9956       689
     S-TITLE     0.0000    0.0000    0.0000         0
       B-ORG     0.9769    0.9713    0.9741       522
       M-ORG     0.9718    0.9702    0.9710      3622
       E-ORG     0.9187    0.9310    0.9248       522
       S-ORG     0.0000    0.0000    0.0000         0
      B-RACE     1.0000    1.0000    1.0000        14
      M-RACE     0.0000    0.0000    0.0000         0
      E-RACE     1.0000    1.0000    1.0000        14
      S-RACE     0.0000    0.0000    0.0000         1
       B-PRO     0.8182    1.0000    0.9000        18
       M-PRO     0.7674    1.0000    0.8684        33
       E-PRO     0.8182    1.0000    0.9000        18
       S-PRO     0.0000    0.0000    0.0000         0
       B-LOC     1.0000    1.0000    1.0000         2
       M-LOC     1.0000    1.0000    1.0000         6
       E-LOC     1.0000    1.0000    1.0000         2
       S-LOC     0.0000    0.0000    0.0000         0

   micro avg     0.9637    0.9641    0.9639      8437
   macro avg     0.7182    0.7350    0.7254      8437
weighted avg     0.9640    0.9641    0.9639      8437
```