
















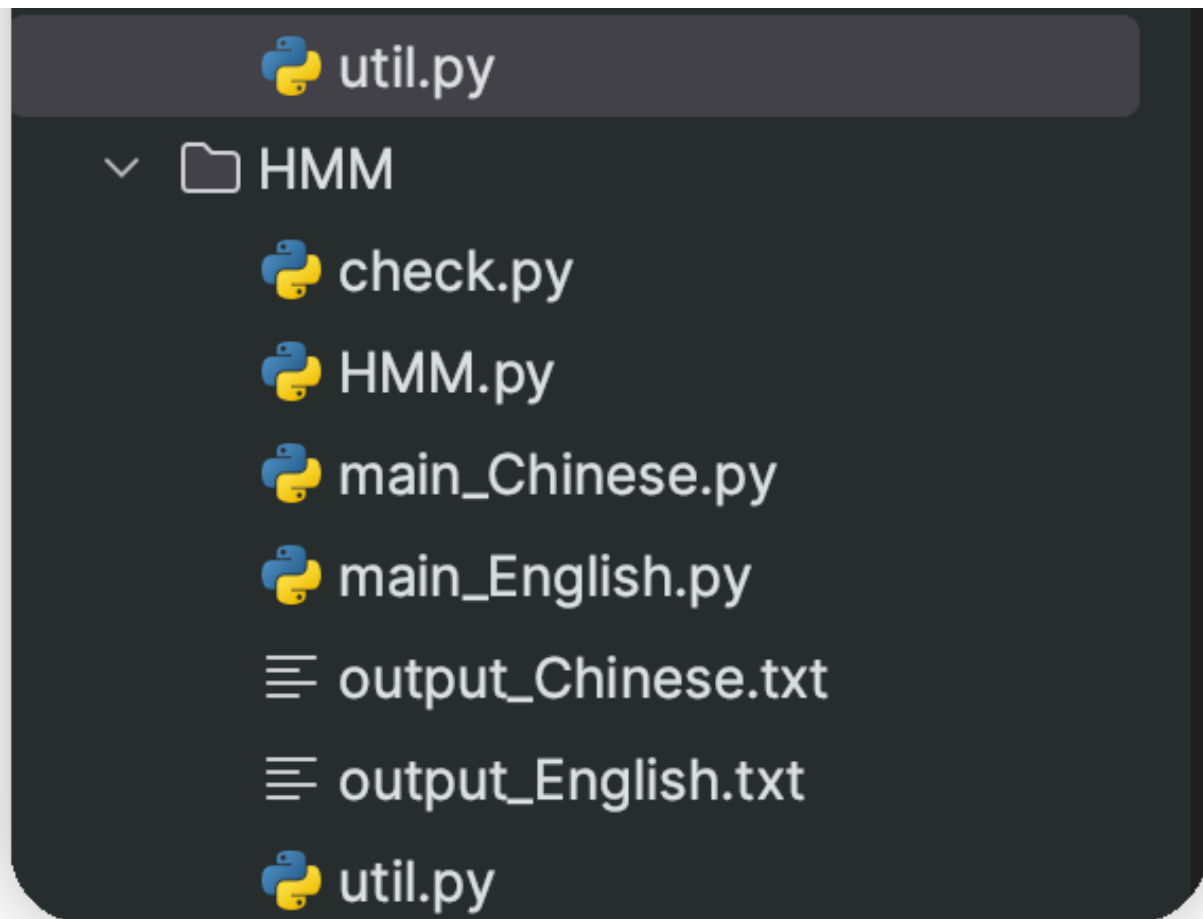


人工智能PJ2文档-20307130082-王骏飞

1.文件结构

- ▼  BiLSTM+CRF
 -  BiLSTM_CRF.py
 -  BiLSTM_CRF_Chinese.pkl
 -  BiLSTM_CRF_English.pkl
 -  check.py
 -  main_Chinese.py
 -  main_English.py
 -  output_Chinese.txt
 -  output_English.txt
 -  util.py
- ▼  CRF
 -  check.py
 -  CRF.py
 -  main_Chinese.py
 -  main_English.py
 -  output_Chinese.txt
 -  output_English.txt



本次提交的文件结构如图，分别按三个Part的任务分为了三个文件夹，即HMM、CRF、BiLSTM+CRF。

`util.py` 包含了对数据的处理函数。

`HMM.py`, `CRF.py`, `BiLSTM_CRF.py` 为三个模型的函数。

`main_Chinese.py` 与 `main_English.py` 分别代表利用模型使用不同数据集训练的主函数，在函数中会进行模型的训练与测试，并通过 `check.py` 进行了分数的输出。

`output_Chinese.txt` 与 `output_English.txt` 是生成的文件，用来运行check。

`BiLSTM_CRF_English.pkl` 与 `BiLSTM_CRF_Chinese.pkl` 是训练后保存的模型，分别是英文数据集和中文数据集。

2.HMM实现NER任务

运行结果

对于英文训练集与测试集，micro avg的f1-score为0.8089.

	precision	recall	f1-score	support
B-PER	0.9645	0.6629	0.7857	1842
I-PER	0.8844	0.7613	0.8183	1307
B-ORG	0.8369	0.7271	0.7781	1341
I-ORG	0.8442	0.6352	0.7249	751
B-LOC	0.9059	0.8334	0.8682	1837
I-LOC	0.8496	0.7471	0.7950	257
B-MISC	0.9164	0.8080	0.8588	922
I-MISC	0.8397	0.6358	0.7237	346
micro avg	0.8937	0.7388	0.8089	8603
macro avg	0.8802	0.7263	0.7941	8603
weighted avg	0.8958	0.7388	0.8074	8603

对于中文训练集与测试集，micro avg的f1-score为0.8757.

B-PRO	0.3714	0.7222	0.4906	18
M-PRO	0.2833	0.5152	0.3656	33
E-PRO	0.4857	0.9444	0.6415	18
S-PRO	0.0000	0.0000	0.0000	0
B-LOC	0.5000	1.0000	0.6667	2
M-LOC	0.8571	1.0000	0.9231	6
E-LOC	0.5000	1.0000	0.6667	2
S-LOC	0.0000	0.0000	0.0000	0
micro avg	0.8640	0.8878	0.8757	8437
macro avg	0.6221	0.7074	0.6533	8437
weighted avg	0.8678	0.8878	0.8771	8437

代码实现

```

1  def __init__(self, wordDictSize, tagDictSize):
2      self.wordDictSize = wordDictSize
3      self.tagDictSize = tagDictSize
4      self.transitionProb = np.random.rand(self.tagDictSize, self.tagDictSize)
5      for index in range(self.tagDictSize):
6          self.transitionProb[index] = self.transitionProb[index] /
np.sum(self.transitionProb[index])
7
8      self.initProb = numpy.random.rand(self.tagDictSize)
9      self.initProb = self.initProb / np.sum(self.initProb)
10
11     self.emitProb = numpy.random.rand(self.tagDictSize, self.wordDictSize)
12     for index in range(self.tagDictSize):
13         self.emitProb[index] = self.emitProb[index] / np.sum(self.emitProb[index])

```

在HMM算法中，最重要的为以下几个矩阵：

emitProb：观测概率矩阵

initProb：初始概率向量

transitionProb：转移概率矩阵

在初始化参数后，利用 `trainSup` 函数，使用已经经过处理的训练数据

- 计算并更新HMM模型的转移概率、发射概率和初始概率。
- 遍历训练数据中的词和标签，统计初始概率、发射概率和转移概率

然后利用 `forwardAlg` 函数进行前向算法

- 计算给定输入句子的前向概率矩阵。
- 使用初始概率和发射概率，递推计算每个词对应每个标签的前向概率。

利用 `backwardAlg` 函数进行后向算法

- 计算给定输入句子的后向概率矩阵。
- 使用发射概率和转移概率，递推计算每个词对应每个标签的后向概率。

利用 `viterbiAlg` 函数进行维特比算法

- 利用维特比算法找到最优路径，即最可能的状态序列。
- 根据初始概率、转移概率、发射概率计算每个词对应每个标签的分数，选择路径中分数最高的状态。

最后进行测试并写入文件。

- 在测试集上利用维特比算法预测标签序列。
- 将概率矩阵取对数，防止数值下溢。
- 计算预测准确率，并将预测结果写入文件（`output_English.txt` 或 `output_Chinese.txt`）。

任务原理

隐马尔可夫模型（HMM）在NER任务中的原理大概可以分为如下几步：

1. 定义状态：

- 每个状态对应一个标签，用来描述文本中的命名实体的边界和类型。比如B-PER，I-PER等。

2. 观察序列：

- 文本中的词序列被视为观察序列，每个词对应一个观察值。

3. 模型参数：

- 转移概率（Transition Probability）：描述从一个状态转移到另一个状态的概率。在NER中，这表示从一个标记转移到另一个标记的概率。
- 发射概率（Emission Probability）：描述从状态生成观察值的概率。在NER中，这表示在给定标记的情况下，生成特定词的的概率。
- 初始概率（Initial Probability）：描述模型在时间步0时处于每个状态的概率。

4. 训练模型：

- 利用带有标注的训练数据，通过监督学习来估计模型的参数。训练数据中包括了句子中每个词的标注信息，即哪些词属于命名实体，以及实体的类型。

5. 预测和解码：

- 利用Viterbi算法进行解码，找到最可能的状态序列，即对应于最可能的NER标注的词的序列。
- 给定观察序列（文本中的词），通过模型计算每个可能的状态序列的概率，并选择具有最高概率的状态序列。

6. 应用:

- 在应用阶段，给定一个未标注的文本，利用训练好的HMM模型进行NER预测，找到文本中的命名实体边界和类型。

3.CRF实现NER任务

运行结果

对于英文训练集和测试集，micro avg的f1-score为0.8620

I-ORG	0.7717	0.8282	0.7990	751
B-LOC	0.8891	0.8726	0.8808	1837
I-LOC	0.8707	0.7860	0.8262	257
B-MISC	0.8894	0.7852	0.8341	922
I-MISC	0.9088	0.7197	0.8032	346
micro avg	0.8765	0.8480	0.8620	8603
macro avg	0.8732	0.8241	0.8463	8603
weighted avg	0.8772	0.8480	0.8614	8603

对于中文训练集和测试集，micro avg的f1-score为0.9498.

S-RACE	0.0000	0.0000	0.0000	1
B-PRO	0.8095	0.9444	0.8718	18
M-PRO	0.7021	1.0000	0.8250	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	0.9490	0.9506	0.9498	8437
macro avg	0.7181	0.7334	0.7246	8437
weighted avg	0.9495	0.9506	0.9499	8437

代码实现

本部分的代码实现主要是利用了sklearn_crfsuite库来实现的。

```
1  def __init__(self,
2      type,
3      algorithm='lbfgs',
4      c1=0.1,
5      c2=0.1,
6      max_iterations=100,
7      all_possible_transitions=False
8  ):
9
10     self.model = CRF(algorithm=algorithm,
11                      c1=c1,
12                      c2=c2,
13                      max_iterations=max_iterations,
14                      all_possible_transitions=all_possible_transitions)
```

首先利用库初始化CRF模型对象，比如说优化算法采用了‘lbfgs’，并设置了正则项的权重c1和c2以及最大迭代次数，进行初始化。

然后利用 `train` 函数进行训练，输入句子列表和对应的标签列表，把每个句子都转换为CRF模型所需要的特征表示。再利用CRF模型的fit方法，传入特征和标签列表，以训练CRF模型。在这个过程中，模型会调整参数以最大化观测数据的似然性。

最后利用 `test` 函数把结果输出到文件中。

任务原理

CRF是一种概率图模型，用于建模标注问题，特别是序列标注问题。其原理可以简要概括如下：

CRF通常用于解决标注问题，其中需要为给定的输入序列分配一个相应的输出标签序列。在序列标注任务中，这些标签可以表示词性、命名实体类别等。

CRF属于判别式模型，它建模的是给定输入序列条件下输出序列的概率。在条件随机场中，给定一组输入变量（观察序列），CRF 模型要学习的是输出变量（标签序列）的条件概率分布。

对于每个输入序列（例如文本中的词序列），CRF通过提取一组相关特征来描述输入序列和输出序列之间的关系。这些特征可以包括当前词的属性（如词性）、前后相邻词的信息、词的前缀和后缀等。

CRF的模型参数包括转移概率和发射概率。其中，转移概率表示标签序列中相邻标签之间的概率分布，发射概率表示给定输入序列时，每个标签的发生概率。利用定义的特征和模型参数，CRF建立了输入序列和输出序列之间的条件概率分布。这个条件概率表示在给定观察序列的情况下，输出标签序列的可能性。然后利用已标注的训练数据，使用最大似然估计或其他优化算法，通过最大化对数似然函数的方式学习模型的参数，即调整模型的参数使得在训练数据上得到的条件概率分布最符合真实情况。

最后进行解码，在预测阶段，使用学习好的CRF模型，通过解码得到给定观察序列下最可能的输出序列。通常采用Viterbi算法等方法找到全局最优的标签序列。

4.BiLSTM+CRF实现NER任务

运行结果

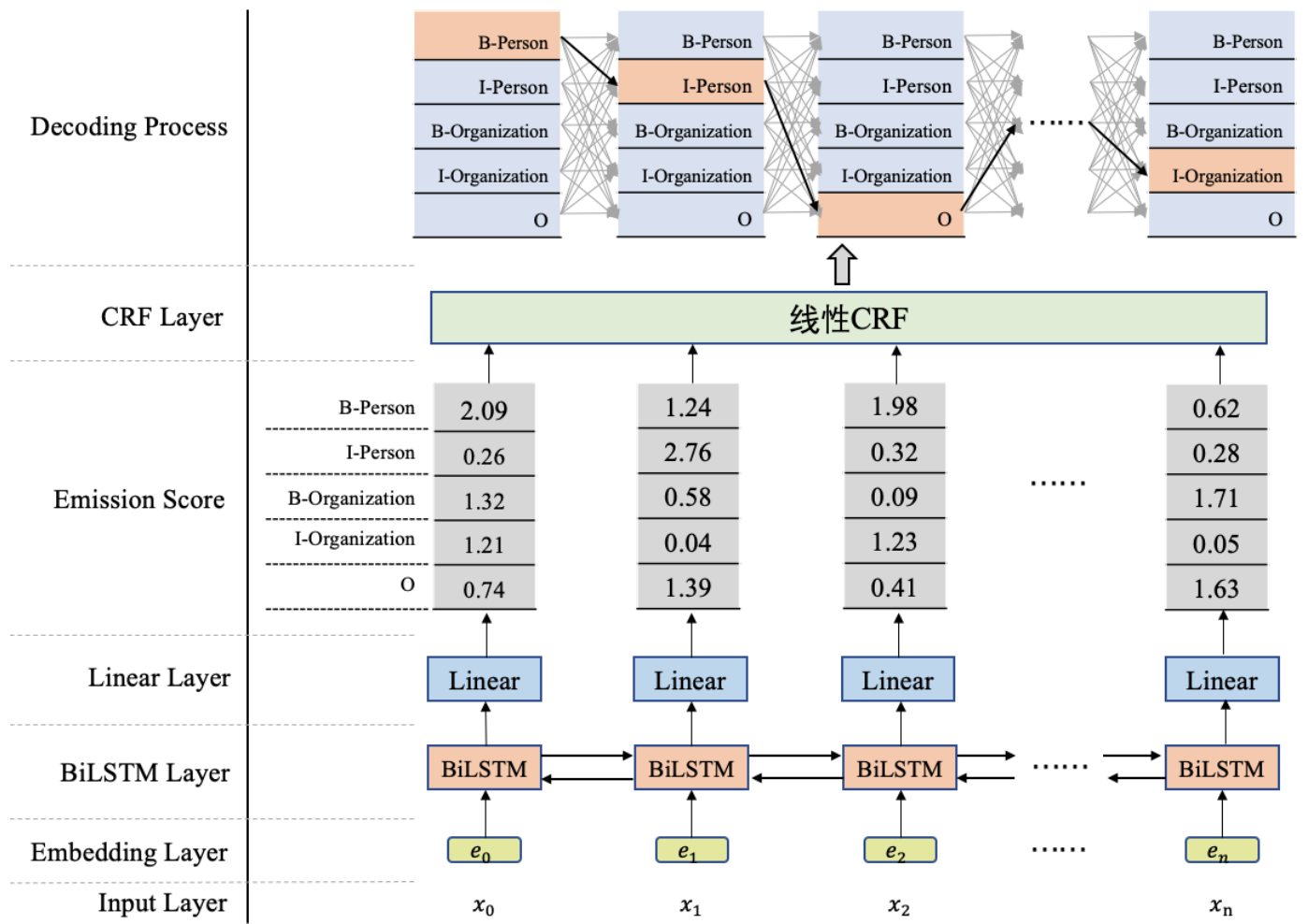
英文训练集分数为0.8841

B-ORG	0.9380	0.7785	0.8509	1341
I-ORG	0.9315	0.7790	0.8484	751
B-LOC	0.9439	0.8982	0.9205	1837
I-LOC	0.9024	0.8638	0.8827	257
B-MISC	0.9279	0.8232	0.8724	922
I-MISC	0.8657	0.7081	0.7790	346
micro avg	0.9365	0.8373	0.8841	8603
macro avg	0.9249	0.8219	0.8696	8603
weighted avg	0.9362	0.8373	0.8833	8603

中文训练集分数为0.9565

B-PRO	0.8182	1.0000	0.7000	18
M-PRO	0.7750	0.9394	0.8493	33
E-PRO	0.7727	0.9444	0.8500	18
S-PRO	0.0000	0.0000	0.0000	0
B-LOC	1.0000	1.0000	1.0000	2
M-LOC	0.6000	1.0000	0.7500	6
E-LOC	1.0000	1.0000	1.0000	2
S-LOC	0.0000	0.0000	0.0000	0
micro avg	0.9532	0.9599	0.9565	8437
macro avg	0.7027	0.7308	0.7143	8437
weighted avg	0.9536	0.9599	0.9566	8437

任务原理



利用BiLSTM+CRF实现NER大概的图类似这样。

在《统计学习方法》中，对线性链条件随机场的描述大概如下：

设 $X = [x_1, x_2, \dots, x_n]$, $Y = [y_1, y_2, \dots, y_n]$ 均为线性链表示的随机变量序列，若在给定随机变量序列的 X 的条件下，随机变量序列 Y 的条件概率分布 $P(Y|X)$ 构成条件随机场，即满足马尔可夫性：

$$P(y_i | X, y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_n) = P(y_i | X, y_{i-1}, y_{i+1})$$

$i = 1, 2, \dots, n$ (在 $i = 1$ 和 n 时只考虑单边)

则称 $P(Y|X)$ 为线性链条件随机场。

在经过BiLSTM后，会产生一个标签向量，这里我们会称之为发射分数。

然后是转移分数，这个转移分数表示一个标签向另一个标签转移的分数，分数越高，转移概率就越大，反之亦然。

然后 利用CRF以一种全局的方式建模，在所有可能的路径中选择效果最优，分数最高的那条路径。

最后使用Viterbi算法从全部的路径中解码出得分最高的路径。

5.一点发现与疑问

本次的分数统计大概如下

	英文训练集	中文训练集
HMM	8089	8757
CRF	8620	9498
BiLSTM+CRF	8841	9565

总的来说，三种模型效果越来越好，但是中文训练集的效果明显优于英文训练集的效果，推测可能是与训练集本身有关，但是不清楚具体是什么原因。