哈尔滨工业大学 2021 年秋季学期

计算学部本科生
“中文信息处理”课
大作业

作 业 题 目：中文名实体识别

姓　　名：李艺峰

学　　号：1190202002

学 生 专 业：计算机工程

任 课 教 师：刘秉权

2021 年 10 月 20 日

报 告 正 文

（根据具体情况，参照以下提纲撰写）

1. 实验内容

1. 使用任意方法实现任一类中文名实体识别；

2. 给定足够规模的测试文本，在其上标注至少 100 个实体识别结果（以附件形式提供）；

3. 计算出实体识别的准确率和召回率，并给出计算依据；

4. 针对识别结果中存在的问题给出具体分析；

5. 提交实验报告，给出详细实验过程、结果和结论；提交源代码、可执行程序和程序中使用的其他资源。

2.实验要求和目的

要求：

1. 自己构造必要的知识库；

2. 自己准备足够规模的语料；

3. 编程环境、汉字编码不限。

目的：

深入了解中文命名实体的处理过程，培养对于中文分词的兴趣和前沿领域的了解，锻炼自身中文信息处理的方法和能力。利用隐马尔可夫模型（HMM）进行命名实体（人名）的识别。

3.实验环境

Windows 10

Visual Studio Code 1.60.2

4. 程序主要算法

4.1 隐马尔可夫模型 HMM

隐马尔可夫模型 HMM 中，有 5 个基本元素：

param trainWordLists:观测序列，指每一个字本身。

param trainTagLists:隐状态，每一个字背后的标注信息。

param self.initProb:初始概率（隐状态），每一个标注的初始化概率。

self.initProb[tag] += 1

param self.transitionProb:转移概率（隐状态），某一个标注转移到下一个标注的概率。

self.emitProb[tag][word] += 1

param self.emitProb: 发射概率 （隐状态表现为显状态的概率），指在某个标注下，生成某个字的概率。

self.transitionProb[tag][nextTag] += 1

生成隐马尔可夫模型。

通过训练语料（data\train.txt）集中统计，将以上参数统计出来。最后，根据这些统计值，计算出产生此观测序列概率最大的隐状态序列，即选择 P（观测序列|隐状态序列 i）最大时（i=1,2,3.....n）的隐状态序列 i。

4.2 维特比（viterbi）算法

应用维特比（viterbi）算法，就可以算出测试集（test.txt）每个字序列背后的标注序列了。利用动态规划的思想，从头计算出每到下一个观测状态所对应的各个隐状态最大概率，这样逐渐向后计算，到达每一状态的时候都会删除不符合最大概率要求的路径，大大降低时间复杂度。

4.3 模式匹配

在生成的隐状态序列中找到相连的 B-NAME，M-NAME，E-NAME 或 B-NAME，E-NAME 进行匹配，匹配上则说明对应的为人名。

隐状态标志信息如下：

{'B-NAME': 0, 'M-NAME': 1, 'E-NAME': 2, 'O': 3, 'B-CONT': 4, 'M-CONT': 5, 'E-CONT': 6, 'B-EDU': 7, 'M-EDU': 8, 'E-EDU': 9, 'B-TITLE': 10, 'M-TITLE': 11, 'E-TITLE': 12, 'B-ORG': 13, 'M-ORG': 14, 'E-ORG': 15, 'B-RACE': 16, 'E-RACE': 17, 'B-PRO': 18, 'M-PRO': 19, 'E-PRO': 20, 'B-LOC': 21, 'M-LOC': 22, 'E-LOC': 23,'S-RACE': 24, 'S-NAME': 25, 'M-RACE': 26, 'S-ORG': 27, 'S-CONT':28, 'S-EDU':29,'S-TITLE':30, 'S-PRO':31, 'S-LOC':32}

5. 实验过程

训练模型

根据训练数据集（train.txt)用极大似然估计法计算隐马尔可夫模型中 5 个基本元素：

```
def trainSup(self, trainWordLists, trainTagLists):
```

```
        self.transitionProb = numpy.zeros((self.tagDictSize, self.tagDictSize))
        self.initProb = numpy.zeros(self.tagDictSize)
        self.emitProb = numpy.zeros((self.tagDictSize, self.wordDictSize))


        for i in range(len(trainWordLists)):
            for j in range(len(trainWordLists[i])):
                word, tag = trainWordLists[i][j], trainTagLists[i][j]
                self.initProb[tag] += 1
                self.emitProb[tag][word] += 1
                if j < len(trainWordLists[i])-1:
                    nextTag = trainTagLists[i][j+1]
                    self.transitionProb[tag][nextTag] += 1
        self.initProb = self.initProb / (self.initProb.sum())
        for index, value in enumerate(self.emitProb.sum(axis=1)):
            if value == 0: continue
            self.emitProb[index, :] = self.emitProb[index, :] / value
```

```
        for index, value in enumerate(self.transitionProb.sum(axis=1)):
            if value == 0: continue
            self.transitionProb[index, :] = self.transitionProb[index, :] / value
```

```
        self.initProb[self.initProb == 0] = 1e-10
        self.transitionProb[self.transitionProb == 0] = 1e-10
        self.emitProb[self.emitProb == 0] = 1e-10
```

生成隐马尔可夫模型。

预测状态序列

采用维特比算法对每个观测序列的隐状态序列进行估计，得出最大概率的隐状态序列

```
def viterbiAlg(self, sentence):

        sentenceSize = len(sentence)

        score = numpy.zeros((sentenceSize, self.tagDictSize))

        path = numpy.zeros((sentenceSize, self.tagDictSize))



        score[0] = self.initProb + self.emitProb[:,sentence[0]]
```

```
        state = numpy.zeros(sentenceSize)
```

```
        for index, word in enumerate(sentence):

            if index == 0: continue

            temp = score[index-1] + self.transitionProb.T

            path[index] = numpy.argmax(temp, axis=1)

            score[index] = [element[int(path[index,i])] for i, element in enumerate(temp)] +
self.emitProb[:,word]
```

```
        state[-1] = numpy.argmax(score[-1])



        for i in reversed(range(sentenceSize)):

            if i == sentenceSize -1: continue

            state[i] = path[i+1][int(state[i+1])]

        return state
```

识别人名

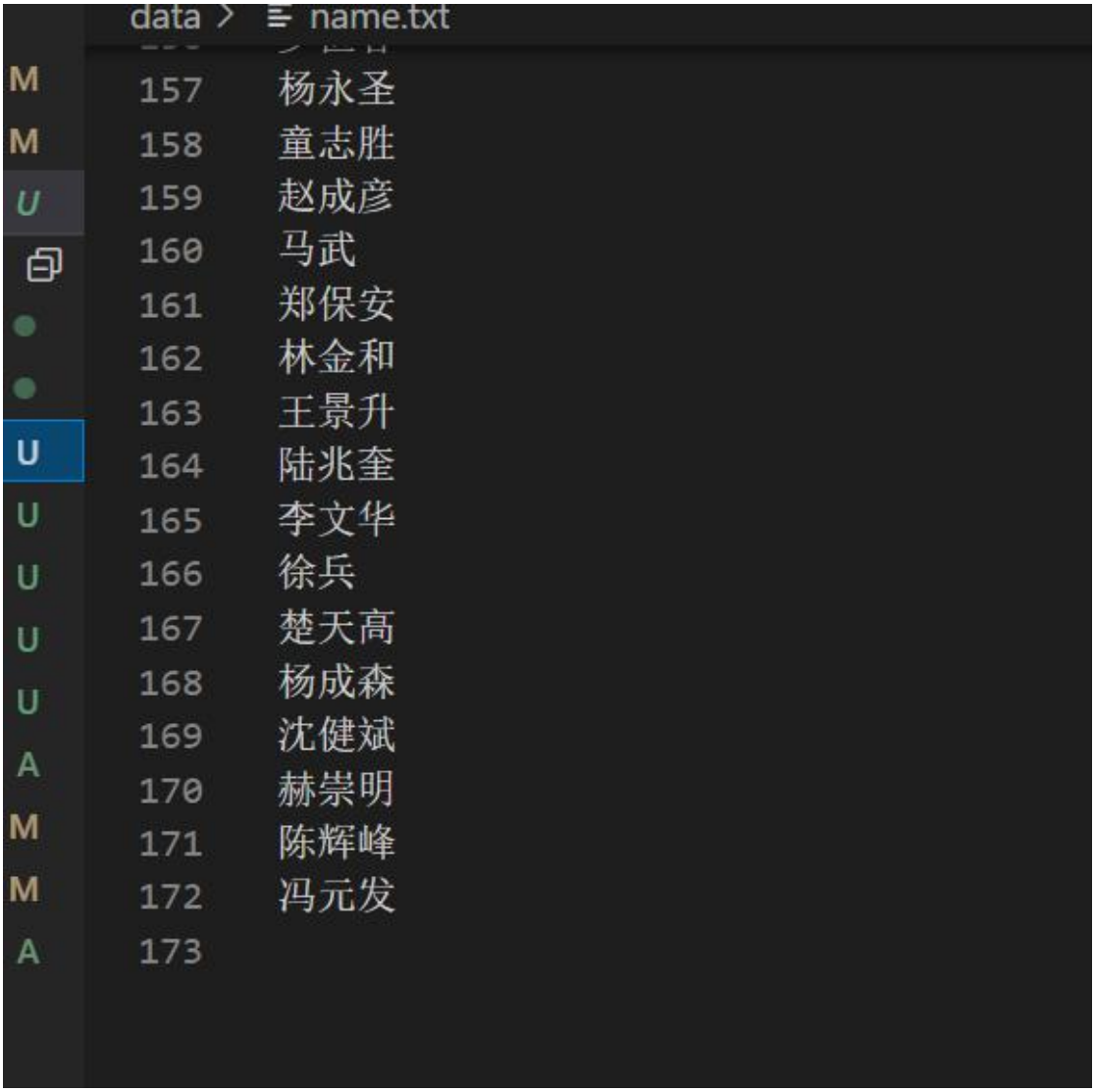
在隐状态序列中查找，如果有 3 个相连的状态为'B-NAME': 0, 'M-NAME': 1, 'E-NAME': 2，那么则判断为 3 字人名。若有 2 个相连状态为'B-NAME': 0,'E-NAME': 2,那么则判断为 2 字人名。其他不常见字数人名不计入考虑。

```
def accuracy(name):

   rtWordLists, rtTagLists = prepareData('data\\right_test.txt')

   rname=set()
```

```
for i in range(len(rtTagLists)):
   for j in range(len(rtTagLists[i])):
     if(rtTagLists[i][j-1]=="B-NAME" and rtTagLists[i][j]=="M-NAME" and rtTagLists[i][j+1]=="E-NAME"):
         rname.add(rtWordLists[i][j-1]+rtWordLists[i][j]+rtWordLists[i][j+1])
     elif(rtTagLists[i][j-1]=="B-NAME" and rtTagLists[i][j]=="E-NAME"):
         rname.add(rtWordLists[i][j-1]+rtWordLists[i][j])
with open("data/rightname.txt",'w',encoding='utf-8') as f:
 for n in rname:
     f.writelines(n+"\n")
pre=str(len(name&rname)/len(name))
print("准确率："+pre)
rcall=str(len(name&rname)/len(rname))
print("召回率："+rcall)
```

6. 实验结果和分析

用训练模型预测（test.txt）人名有 172 个，如下图所示：



实际上根据标准测试集标注结果（rightname.txt）进行计算，结果有 202 个，如下图所示：

计算准确率：n（正确识别的名字）/n（预测出的所有名字）

=n（预测结果&实际结果）/n(预测结果）

召回率:n（正确识别的名字）/n（实际的所有名字）

=n（预测结果&实际结果）/n(实际结果）

准确率和召回率计算结果如下：

准确率接近 85%，召回率也有 72%。

7. 实验结论和体会

实验结果证明，基于隐马尔可夫模型（hmm）的状态序列预测和人名识别准确率是可以接收的。但是因为名字可能有很强的随机性，对于较为特殊的名字识别还不够理想，如：超过 3 个字的名字，名字中有姓或不常见字或嵌套其他类型的字。

在正常的马尔可夫模型中，状态对于观察者来说是直接可见的。这样状态的转换概率便是全部的参数。而在隐马尔可夫模型中，状态并不是直接可见的，但受状态影响的某些变量则是可见的。每一个状态在可能输出的符号上都有一概率分布。因此我们可以根据输出符号的序列估计状态序列。

中文实体名大多都是未登录词，数量多、识别难度大、对分词效果影响大。从基于规则

和词典的方法，到基于统计的方法，再到二者混合再融入机器学习的方法，能够识别的中文实体名越来越精准广泛。中文实体名识别对分词很重要，它很大程度上决定了分词的好坏，而分词的好坏又决定了自然语言处理之后的一些列工作，所以这个工作是至关重要的。