

《人工智能软件开发与实践》

(2023 学年 秋季 学期)

作 业 报 告

学 号： —

姓 名： —

班 级： —

任课教师： —

作业报告

实验名称： 名实体识别模型训练测试

成绩：

实验类别： 验证/综合型实验 实验要求： 1 人 1 组 时间： 2023 年 9 月 2 日

一、 实验目的

了解 CRF++的使用，熟悉将语料文件转换为模型输入的格式，然后训练测试，统计名实体识别精度，并想办法改进，提升模型识别性能的过程。

注意：只需要识别如下三个实体类型

符号	名称
nr	人名
ns	地名
nt	机构名

二、实验内容

- 1. 打开文件 199801.txt。
- 2. 删除每行开始的句子标识。将句子转化为如下的格式：

例子 1：

台 B_NS
湾 I
是 O
中 B_NS
国 I
领 O
土 O
不 O
可 O
分 O
割 O

例子 2：

以 O

江 B_NR

泽 I

民 I

同 O

志 O

为 O

核 O

心 O

的 O

党 B_NT

中 I

央 I

3. 切分过程中需要注意以下特殊情况：

- 语料中人的姓和名是分开标准的，统计时应该作为一个实体：
 - 江/nr 泽民/nr 算一个实体
 - 李/nr 鹏/nr 算一个实体
- 嵌套实体用[]括起来，即使里面还有粒度更小的实体，也按长度最大的实体计算，只算一次，不重复计算。

如：[中国/ns 香港/ns 特别/a 行政区/n]ns 整体算一个，里面的实体不算

4. 切分后，形成训练文件（来自 199801.txt），测试文件（test.txt），调用 CRF++ 进行训练测试，写程序统计名实体单元的识别精度（可以调用 conlleval.pl，但是需要安装 Perl 语言解释环境）

5. 加入新的特征或者改进 CRF 模型本身，提升名实体识别的性能。

三、使用的算法名称（若无，可以不填）

四、程序源码（拷贝至此处，同时作为附件和报告再一份单独的程序）

```
originalFile = open("199801.txt", "r", encoding='ANSI') # 打开文件
trainFile = open("train.data", "w", encoding='UTF-8')
trainFile.truncate(0)

nr_flag = 0
pa_flag = 0
entity = ""
```

```

for line in originalFile:
    if len(line) != 0: # 行非空
        words = line.split()[1:] # 以空格分词为列表 words 并去掉句子标识
    else: # 跳过空行
        continue

    for word in words:
        tag = word.split('/')[1] # 获得词的类型

        if nr_flag == 1: # 连续两个 nr，在处理前一个词时被合并为一个中文名，跳过此轮循环
            nr_flag = 0
            continue

        if word[0] == "[": # 括号开始
            pa_flag = 1
            entity = word.split('/')[0][1:] # 获取嵌套实体的第一个实体名并去掉括号
            continue

        if pa_flag == 1: # 词在括号中，将其拼接进实体名
            entity = entity + word.split('/')[0]
            if "]" in word: # 括号结束
                tag = word.split(']')[1]
            else:
                continue

        if tag == "nr": # 处理人名，若后一个也是人名则拼接
            if words.index(word) != len(words) - 1:
                nextWord = words[words.index(word) + 1]
                if nextWord.split('/')[1] == "nr":
                    nr_flag = 1
                    entity = word.split('/')[0] + nextWord.split('/')[0]

        # 非实体
        if tag != "nr" and tag != "ns" and tag != "nt":
            if entity == "":
                entity = word.split('/')[0]
            for i in range(len(entity)):
                trainFile.write(entity[i]+"\t0\n")
            entity = ""
            continue

        # 处理人名和括号的特殊情况：实体名已拼接完毕
        if pa_flag == 1:
            pa_flag = 0
        else: # 非特殊情况，直接获取实体名
            if not nr_flag == 1:

```

```

        entity = word.split('/')[0]
# 排除实体名为空
if len(entity) == 0:
    continue
else:
    for i in range(len(entity)):
        trainFile.write(entity[i]+"\\t")
        if i == 0:
            trainFile.write("B-" + tag.upper() + "\\n")
        else:
            trainFile.write("I\\n")
    entity = "" # 清空实体名

trainFile.write("\\n")

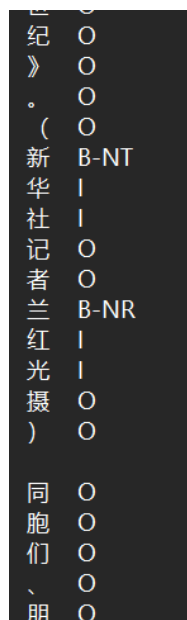
originalFile.close()
trainFile.close()

```

五、程序运行结果（将程序运行结果的截图，拷贝至此处）

1. 原程序结果

利用上述程序，将预料文件转为训练文件 train.data，将 test.txt 转为测试文件 test.data。生成的 train.data 内容如图 1。



```

世纪 O
》 O
。 O
( O
新华 B-NT
社 I
记者 O
兰 B-NR
红 I
光 I
摄 O
) O

同 O
胞 O
们 O
、 O
朋 O

```

图 1 train.data 内容

通过.\crf_learn -t -c 4.0 template train.data model 指令由训练文件生成 model 模型，所用模板 template 内容如图 2。

```
# Unigram
U00:%x[-2,0]
U01:%x[-1,0]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[2,0]
U05:%x[-1,0]/%x[0,0]
U06:%x[0,0]/%x[1,0]

# Bigram
B
```

图2 模板 template 内容

通过.\crf_test -m model test.data -o result 进行识别测试，result 内容如图3。

中	B-NT	B-NT
国	I	I
国	I	I
务	I	I
院	I	I
副	O	O
总	O	O
理	O	O
李	B-NR	B-NR
岚	I	I
清	I	I
3	O	O
1	O	O

图3 result 内容

通过perl conllevl.pl -d "\t" <result>result.info 利用 conllevl 脚本评估结果,result.info 内容如图4。

```
processed 71081 tokens with 4262 phrases; found: 3930 phrases; correct: 3419.
accuracy: 98.02%; precision: 87.00%; recall: 80.22%; FB1: 83.47
: precision: 88.89%; recall: 82.27%; FB1: 85.46 1963
: precision: 88.89%; recall: 82.27%; FB1: 85.46 1963
NR: precision: 91.01%; recall: 80.60%; FB1: 85.49 534
NS: precision: 86.12%; recall: 81.50%; FB1: 83.75 1059
NT: precision: 73.80%; recall: 65.87%; FB1: 69.61 374
```

图4 result.info 结果

2. 优化：增加特征

用 B、I、E 来标识名实体——名实体结尾的汉字标识为 E-N*。如图5。

坛	O	O	
中	B-NT		B-NT
国	I	I	
国	I	I	
务	I	I	
院	E-NT		E-NT
副	O	O	
总	O	O	
理	O	O	
李	B-NR		B-NR
岚	I	I	
清	E-NR		E-NR
3	O	O	
1	O	O	
日	O	O	
下	O	O	
午	O	O	
抵	O	O	
达	O	O	
瑞	B-NS		B-NS
士	E-NS		E-NS
东	O	O	

图 5 增加特征后的 result 内容

利用 conl1eval 脚本评估的结果见图 6，

```
processed 71081 tokens with 4739 phrases; found: 4431 phrases; correct: 3857.
accuracy: 97.87%; precision: 87.05%; recall: 81.39%; FB1: 84.12
: precision: 87.52%; recall: 81.85%; FB1: 84.59 1242
: precision: 87.52%; recall: 81.85%; FB1: 84.59 1242
NR: precision: 90.95%; recall: 82.75%; FB1: 86.66 939
NS: precision: 86.18%; recall: 82.05%; FB1: 84.06 1469
NT: precision: 83.23%; recall: 77.75%; FB1: 80.40 781
```

图 6 增加特征优化结果

与原结果相比，精确率有所下降，召回率有所上升，FB1 指标有所上升，而从结果可以看出这种优化方法对于 NT 类型的名实体各指标都有所提升且效果明显，因此，在下一步的优化中只对 NT 类实体采取 BIE 标识，其它两类实体采取 BI 标识。

3. 优化：更改 template 模板

在原 template 模板的基础上增加每个字生成特征函数的参考范围至-3，3，如图 7。

```
# Unigram
U00:%x[-3,0]
U01:%x[-2,0]
U02:%x[-1,0]
U03:%x[0,0]
U04:%x[1,0]
U05:%x[2,0]
U06:%x[3,0]
U07:%x[-1,0]/%x[0,0]
U08:%x[0,0]/%x[1,0]

# Bigram
B
```

图 7 新的 template 内容

优化后的评价结果见图 8，可知精确率较上一步优化有所提升。

```
processed 71081 tokens with 4262 phrases; found: 3903 phrases; correct: 3429.  
accuracy: 97.98%; precision: 87.86%; recall: 80.46%; FB1: 83.99  
: precision: 89.70%; recall: 82.51%; FB1: 85.95 1951  
: precision: 89.70%; recall: 82.51%; FB1: 85.95 1951  
NR: precision: 93.37%; recall: 81.76%; FB1: 87.18 528  
NS: precision: 87.35%; recall: 80.25%; FB1: 83.65 1028  
NT: precision: 72.73%; recall: 68.74%; FB1: 70.67 396
```

图 8 更改模板优化结果

六、心得体会和遇到的困难

优化思路较难以想到。要经常自己用程序实际运行来进行验证。