

# 词性标注

---

layout: post

title: HMM&Viterb POS Tagging

subtitle: Natural Language Processing homework

date: 2018-01-05

author: WeiXiao

header-img: img/tag-bg-o.jpg

catalog: true

tags:

- NLP

## - Python

---

### 任务定义

- This data set contains one month of Chinese daily which are segmented and POS tagged under Peking Univ. standard.
- Project ideas:
  - Design a sequence learning method to predicate a POS tags for each word in sentences.
  - Use 80% data for model training and other 20% for testing (or 5-fold cross validation to test learner's performance. So it could be interesting to separate dataset.

### 运行环境

- Mac OS 10.13
- Pycharm
- Python 3.6.1

### 输入输出

- 在这次实验中,我使用了nltk的brown数据集,这个数据集分为不同的种类,我选择新闻类作为我的输入集,并且在各个文件中剪切出一部分作为测试集,关于这个brown数据集的更多信息,请您访问[这个网站](#),我也是在这个博客学习使用数据集的

```

175
176
177
178
179
180
181
182
183
184
185
186
187
presb=0
for word in wor_tag:
    if (word[0]== "." or "," == word[0]):
        ans = self.hmm(sentence)
        #print(ans)
        for i in range(ans.__len__()):
            str=ans[i].split("+")
            estr=[]
            for tr in str:
                estr.extend(tr.split("-"))
            str=estr
            if ans[i] in str:
                pres=pres+1
    rate()

```

这里是每一个句子进行预测的准确度

这里是最终在整个测试集上的准确度

Precisious: 0.8142788892318801

IDE and Plugin Update  
PyCharm Community

- 如上图所示,这个程序的输出是一系列准确度,并且在最后输出整体平均水平的准确度

## 方法描述

- 首先从nltk的brown数据集中读取数据,并将结果保存在wordtag内

```

#载入词典
words_tag = brown.tagged_words(categories=['news'])

```

- 然后首先根据测试集,算出下面这些参数:
  - self.Cixin\_set#词性集合
  - self.Cixin\_map={}词性集合对应的位置
  - self.vocab\_map={}测试集中词性的
  - self.Cixin\_len=0词性的种类数量
  - self.Ci\_pro
  - self.Tran\_matrix=np.zeros()#转移概率矩阵
  - self.emitter\_pro\_matrix=np.zeros()#发射概率矩阵
- 然后读取要标注的测试集,使用词性选择的时候使用维特比算法算出最大可能的一个序列,然后把

这个结果存储在result\_cixin\_map中

- 再在test函数中对于每一个测试句子都分别对比result——cixin\_map和数据集的内容,并且计算出本句的准确度,最后再计算出平均值
- 最后一次输出每一句的平均值,再计算出总的平均值

## 结果分析

- 在抽取出来的测试集上运行,平均准确度**0.8224**

```
wor_tag = brown.tagged_words(fileids=['catest'])
```

```
1.0
1.0
0.90625
1.0
1.0
0.7631578947368421
0.5454545454545454
0.7391304347826086
0.75
0.9
1.0
0.15384615384615385
0.8421052631578947
Precisious: 0.8224880382775119
```

Process finished with exit code 0

- 由上图所示,在词性标注的过程中,对于一部分句子的标注准确度能达到1.0,对于某些句子的标注准确度只能达到0.545,这可能是由于句子长度大的时候,只要有一个词的词性标注错误,就会影响后面标注的所有词,在微观上来看,越往后面的词标注错误的概率呈现指数级增长.宏观上来看,句子长度较大的标注准确率相对较低

## 实验思考

- 正确处理unknown词

```
134
135
136
137
138
139
result_cixin_map=[]
sy = int(np.where(pro_table[-1,:]==np.max(pro_table[-1,:],0))[0][0])
t=-1
except KeyError:
    return "error!"
while sy:
```

HMM > rate() > for in wor... > if (word[0]=="...

tag

/Users/weixiao/anaconda/bin/python /Users/weixiao/PycharmProjects/nltk\_brown/tag.py

finish Initialization

0.0

0.0

0.7727272727272727

0.0

0.0

0.0

0.0

0.0

0.0

第一次处理unknown词的时候,直接出错返回error导致只要句子中含有未知词就会出错  
并且训练的准确率也不够

- 在上图中,由于未能正确处理未知词,所以只要句子中含有未知词都会出错,导致不能标注
- 词性一致性检查
  - 由于时间关系,我仔细看了一下课本,感觉计算马氏距离比较麻烦,所以在这次试验中处理兼类词的时候,只要预测的词性,是兼类词属性中的一个子集就判对,对应代码如下.也是以后改进的第一个点

```
for i in range(ans.__len__()):
    str=anse[i].split("+")
    estr=[]
    for tr in str:
        estr.extend(tr.split("-"))
    str=estr
    if ans[i] in str:
        pres=pres+1
        presb+=1
    else:
        pres+=1#最重要要修改的地方
```