

同济大学计算机系

中文信息处理实验报告



学 号 1553534

姓 名 李帅

专 业 计科

授课老师 卫志华

一、标注作业摘要

完成了基于 viterbi 算法的中文词性标注实验以及极大似然估计处理未登录词的新方法，最后实验验证集的准确率最高可达 95.51%。另外本实验是分词标注，所以对于验证集是分词完毕的直接标注即可，对于未被分词的测试文本，采用 jieba 分词先分词完毕再做 viterbi 算法标注。

二、实验环境

Python 3.5

Win 10 系统

第三方库：collections 导入 defaultdict，便于统计词性与单词次数

zhon 导入 punctuation（中文标点符号）

jieba 用来分词，同时与自设计 viterbi 算法结果对比

训练集の説明：

a; 形容词

Ag; 形容词语素

b; 区别词

Bg; 区别词语素

c; 连词

d; 副词

Dg; 副词语素

e; 叹词

Eg; 叹词语素

f; 方位词

g; 语素

h; 前接成分

i; 成语

j; 缩略语

k; 后接成分

l; 习用语

m; 数词

Mg; 数词语素

n; 名词

Ng; 名词语素

ns; 名词-表处所

nt; 名词-表时间

o; 像声词

p; 介词

q; 量词

Qg; 量词语素

r; 代词

Rg; 代词语素

s; 处所词
 t; 时间词
 Tg; 时间词语素
 u; 助词
 Ug; 助词语素
 v; 动词
 Vg; 动词语素
 w; 标点
 x; 字
 y; 语气词
 Yg; 语气词语素
 z; 状态词

三、算法框架

1. 基本模型

a. 问题

单词序列 $W=w_1w_2\dots w_n$ 词性序列 $T=t_1t_2\dots t_n$

b. HMM 模型 λ

状态集合: 词性标记集合

输出值集合: 待标记单词集合

转移概率: 由一种词性转移到另一词性的概率

初始分布: 各词性的概率

输出概率: 从某一词性观察到某一单词的概率

2. 理论分析

a. 已知词串 W (观察序列) 和模型 λ 的情况下, 求使得条件概率 $P(T|W, \lambda)$ 值最大的 T' , 一般记作:

$$T' = \arg \max_T P(T | W, \lambda) \quad (1)$$

b. 运用 Bayes 定理得:

$$P(T | W) = \frac{P(T)P(W | T)}{P(W)} \quad (2)$$

c. 由于 W 给定, $P(W)$ 不依赖于 T , (2) 式简化为:

$$P(T | W) \approx P(T)P(W | T) \quad (3)$$

d. $P(T)$ 是词性序列的概率:

$$\begin{aligned} P(T) &= \pi_{t_1} P(t_2 | t_1) P(t_3 | t_2, t_1) \cdots P(t_n | t_{n-1}, t_{n-2}, \dots, t_1) \\ &= P(t_1 | t_0) P(t_2 | t_1) P(t_3 | t_2) \cdots P(t_n | t_{n-1}) \end{aligned} \quad (4)$$

e. $P(W|T)$ 是已知词性标记串, 产生词串的条件概率:

$$P(W|T) = P(w_1 | t_{1,n}) P(w_2 | t_{1,n}) P(w_3 | t_{1,n}) * * * P(w_n | t_{1,n})$$

f. 由(3)(4)(5)式得:

$$P(T | W) \approx \prod_{i=1}^n [P(w_i | t_i) P(t_i | t_{i-1})] \quad (6)$$

g. 由(1)(6)式得确定一个句子的最优标注的等式:

$$T' = \arg \max_T \prod_{i=1}^n [P(w_i | t_i) P(t_i | t_{i-1})]$$

3. 实现过程

从本质上说, viterbi 算法是一个动态规划算法。

a. 训练数据

训练集: train 文件夹下的 TrainingText1-5 和 training.txt 文件

验证集: train 文件夹下的 TrainingText6 文件 (需要切割词组处理)

转移概率: $P(t_i | t_{i-1}) = \frac{\text{训练集中 } t_i \text{ 出现在 } t_{i-1} \text{ 之后的次数}}{\text{训练集中 } t_{i-1} \text{ 出现的次数}}$

输出概率: $P(w_i | t_i) = \frac{\text{训练集中 } w_i \text{ 的词性被标记为 } t_i \text{ 的次数}}{\text{训练集中 } t_i \text{ 出现的次数}}$

b. Viterbi 算法

符号定义:

①从第 m 个词的各个词性标记向第 m+1 个词的各个词性标记转移的概率, 记作

$$a_{ij} = P(t_j | t_i) \quad 1 \leq i, j \leq N$$

第 1 个词前面没有词, 其各个词性标记也满足一定的概率分布, 记作 π_i

②第 m 个词的各个词性标记取词语 w_m 的条件概率, 记作

$$b_i(w_m) = P(w_m | t_i) \quad 1 \leq i \leq N$$

③从起点词到第 m 个词的第 i 个词性标记的各种可能路径(即各种可能的词性标记串)中, 必有一条路径使得概率最大, 引入 Viterbi 变量来描述。

$$\delta_m(i) = \max_{t_{1,m-1}} P(t_{1,m-1}, t_m = i, w_{1,m} | \lambda) \quad 1 \leq i \leq N$$

④当扫描过第 m 个词, 状态转移到第 m+1 个词时, 需要记录已走过路径中的最佳路径, 即记住该路径上 w_m 的最佳词性标记, 引入如下变量:

$$\psi_m(i) = \arg \max_{1 \leq j \leq N} [\delta_{m-1}(j) a_{ji}] \times b_i(w_m) \quad 1 \leq i \leq N$$

c. 算法过程

① 初始化 $\delta_1(i) = \pi_i b_i(w_1), \psi_1(i) = 0 \quad 1 \leq i \leq N$

$$\delta_m(j) = \max_{1 \leq i \leq N} [\delta_{m-1}(i) a_{ij}] \times b_j(w_m)$$

$$\psi_m(j) = \arg \max_{1 \leq i \leq N} [\delta_{m-1}(i) a_{ij}] \times b_j(w_m)$$

② 归纳推导:

③ 终止和路径读出:

$$P^* = \max_{1 \leq i \leq N} [\delta_n(i)] \quad t_n^* = \arg \max_{1 \leq i \leq N} [\delta_n(i)]$$

$$t_k^* = \psi_{k+1}(t_{k+1}^*) \quad k = n-1, n-2, \dots, 1$$

d. 处理未登录词

本次实验采用三种方法分别登录词:

- (1) 将未登录词的输出概率统一为 1: 实现简单, 处理效率高, 但是没有利用任何统计先验知识, 准确率难以提高。
- (2) 扩充词典。将 jieba 第三方库调用起来, 补充本次实验训练集的不足, 一旦出现未登录词, 将其的概率设置为 1, 同时它的词性标注完全根据 jieba 的标注结果。
- (3) 采用极大似然估计的方法【1】:

根据参考文献【1】, 采用的极大似然估计方法:

假设有输入句子 $S = w_1, \dots, w_{j-1} x_j w_{j+1}, \dots, w_N$, 其中 S 表示整个句子, $w_i (1 \leq i \leq N)$ 表示单个的词, x_j 为生词。

现在面临的问题是如何来估算生词 x_j 的词汇发射概率? 我们可以设想这样的情景: 假设把 S 加入训练集中, 由于加入的只有一个句子, 因此对其他词的发射概率和整个模型的词性转移概率的影响可以忽略不计。遵循 HMM 模型的假设, x_j 的词性 c_j 由 w_{j-1} 的词性决定。即

$$P(c_j | x_j) \approx \sum_{m=1}^M P(c_m | w_{j-1}) P(c_j | c_m) \quad (5)$$

其中 M 表示词性种类的总数。而根据贝叶斯公式, 词汇发射概率

$$P(x_j | c_j) = \frac{P(x_j)}{P(c_j)} P(c_j | x_j) \quad (6)$$

将(5)式代入(6)得

$$P(x_j | c_j) = \frac{P(x_j)}{P(c_j)} \sum_{m=1}^M P(c_m | w_{j-1}) P(c_j | c_m) \quad (7)$$

对式(7)中的各概率值采用极大似然估计, 得

$$P(x_j | c_j) = \frac{C(x_j)}{C(c_j)} \sum_{m=1}^M P(c_m | w_{j-1}) P(c_j | c_m) = \frac{1}{C(c_j)} \sum_{m=1}^M \left(\frac{C(w_{j-1} c_m)}{C(w_{j-1})} \times \frac{C(c_m c_j)}{C(c_m)} \right) \quad (8)$$

其中 $C(c_j)$ 表示标注符号 c_j 在训练语料中的出现次数, $C(c_m c_j)$ 表示标注符号串 $c_m c_j$ 的共现次数。

(8)式即为我们将要测试的生词词汇发射概率估算模型。

四、函数说明

(1) 类成员变量介绍

本实验的主体模块是 `viterbi.py` 模块, 封装了一个 `viterbi` 类, 对外接口的测试文件路径, 测试集与验证集路径已经内置

```

ROOT = 'train'
class viterbi(object):
    def __init__(self, test_path):
        self.states=defaultdict(int)#词性统计
        self.words=defaultdict(int)#单词统计
        self.wordset=set()
        self.wordcount=defaultdict(int)
        self.transition_probability=defaultdict(float)#转移概率
        self.output_probability={}#输出概率
        self.pi={}
        self.data = open('training.txt', 'r').readlines()

        #1-5用做训练, 6用作验证
        for i in range(1,6):
            f=os.path.join(ROOT,'TrainingText%d.TXT'%(i))
            # print(f)
            p=open(f, 'r', encoding='utf-8').readlines()
            for line in p:
                self.data.append(line)

        self.test=open(test_path, 'r', encoding='UTF-8').readlines()

        self.output=None
        self.dictionary=[]

```

成员变量介绍:

Self.states: 词性统计词典, key 是词性字符串, value 是出现的次数。

Self.words 单词与词性统计词典词, key 是 (单词, 对应的词性) 的 tuple, value 是对应的次数。

Self.wordset 单词统计词典, 训练集出现的单词的集合。

Self.wordcount 单词与对应出现次数的字典

Self.transition_probability 转移概率, key 是二元 tuple, 对应 $p(x/y)$

Self.output_probability 输出概率, key 是二元 tuple, first 是单词 second 是词性

Self.pi 对应 π , 每个词性标记的概率分布

Self.data 训练集数据

Self.test 输入的测试集数据。

Self.output 输出

Self.dictionary 根据空格划分的词组 list.

(2) 函数说明

本程序分成一个 viterbi 类, 四个成员函数:

```
def __init__(self, test_path)
```

```
def process_trainset(self)
```

```
def print_result(self)
```

```
def validation(self)
```

```
from collections import defaultdict
import os
from zhon.hanzi import punctuation

ROOT = 'train'

class viterbi(object):
    def __init__(self, test_path):...
    def process_trainset(self):...

    def print_result(self):...
    def validation(self):...
```

a. __init__ 函数:

类的初始函数，初始成员变量。

b. process_trainset 处理训练集函数:

处理训练集，同时在扫描每一行，计算得出各个成员变量。

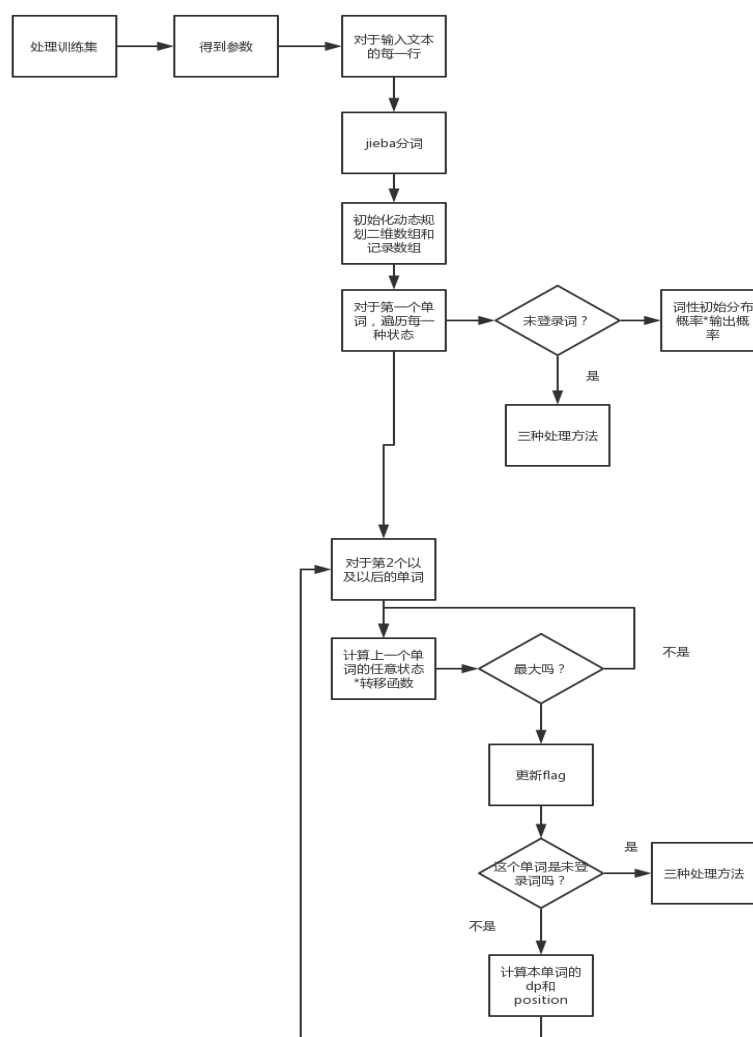
具体过程:

首先对于 每一行的字符串，去掉收尾多余的符号，根据空格分割成一个词组 list---dict ,对于每一 dict 中的元素，根据 '/' 划分出单词与词性标记，更新相关成员变量。在一遍扫描结束后，根据总体的统计结果，计算相关概率分布变量，对于 self.transition_probability 的 key(x,y),x 表示后一个词性标记，y 表示前一个出现的词性标记；对于 self.output_probability 输出概率，key(x,y) X 表示单词，y 表示词性，value 表示出现这一词性的单词的个数占整个这个词性的比重。

c. print_result 函数:

根据输入的 txt 文件，同时根据 viterbi 算法和第一种和第二种处理未登录词的方法处理未登录词，，再通过控制台输出结果

具体过程简略图



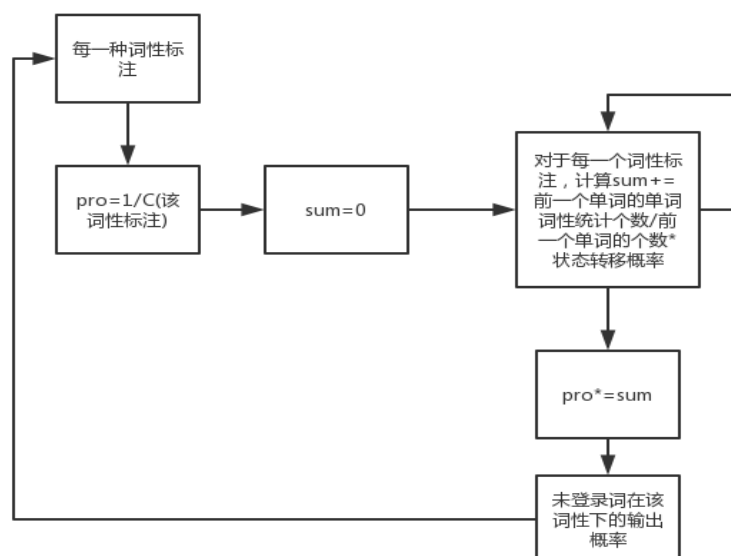
d. validation 函数:

计算验证集的正确率的函数，本函数的未登录词处理方法

验证集是 train 文件夹下的 TrainingText6.TXT

过程:

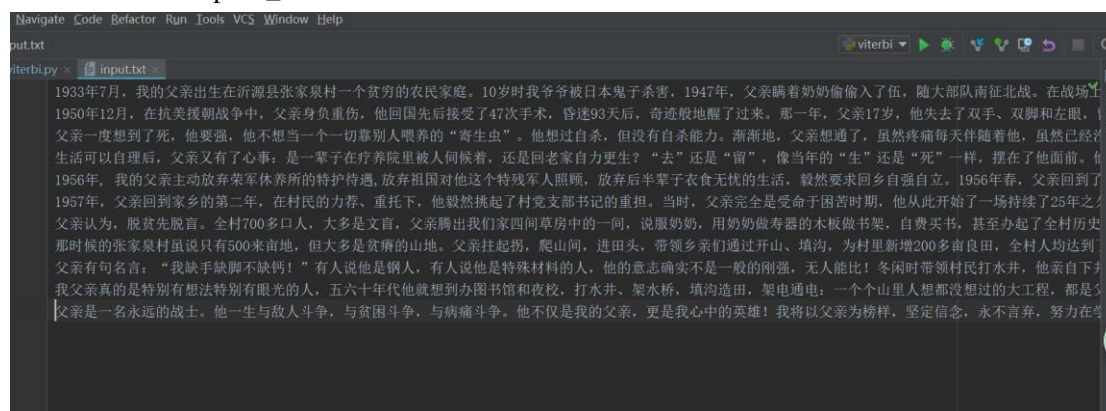
- (1) 首先将验证集每一行分割成数组，然后根据 '/' 将数组的元素分成单词与词性标注，将单词组成数组，将词性标注组成另一个数组，形成验证集。
- (2) 训练训练数据得到 dp 和 position 二维数组，然后按照上面流程图一样的过程，处理每一个单词
- (3) 其中，对于极大似然估计的计算流程如下:



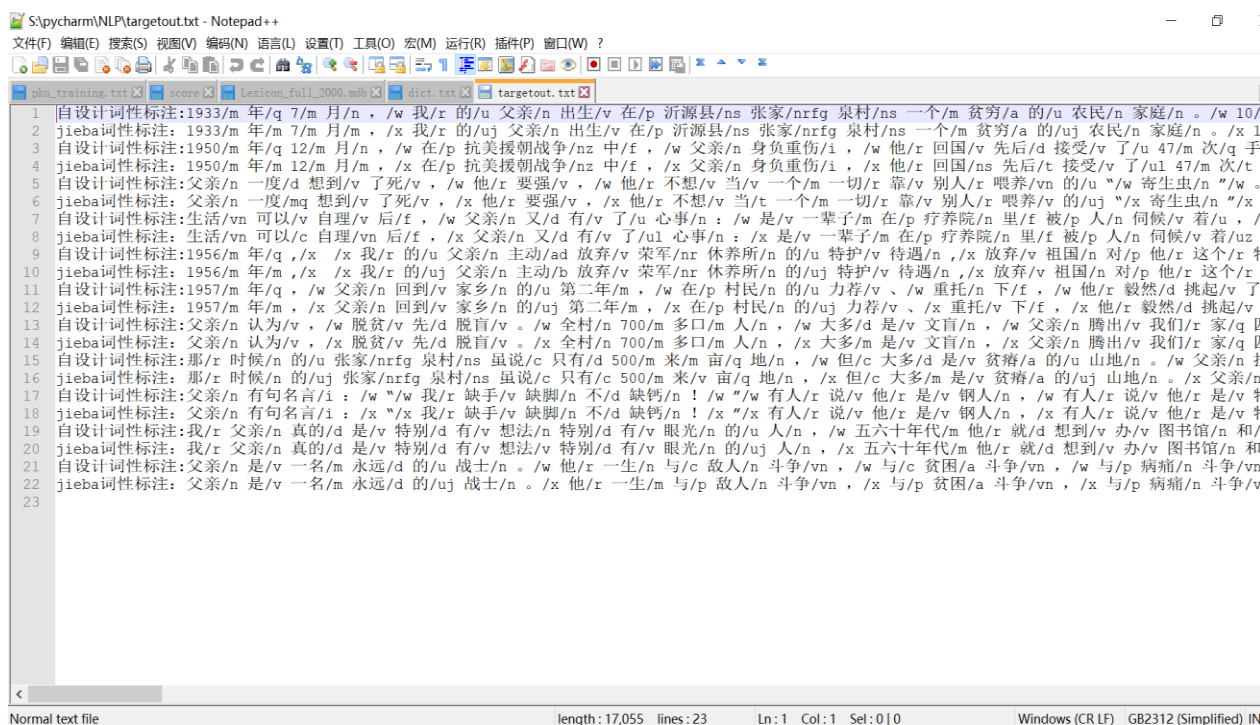
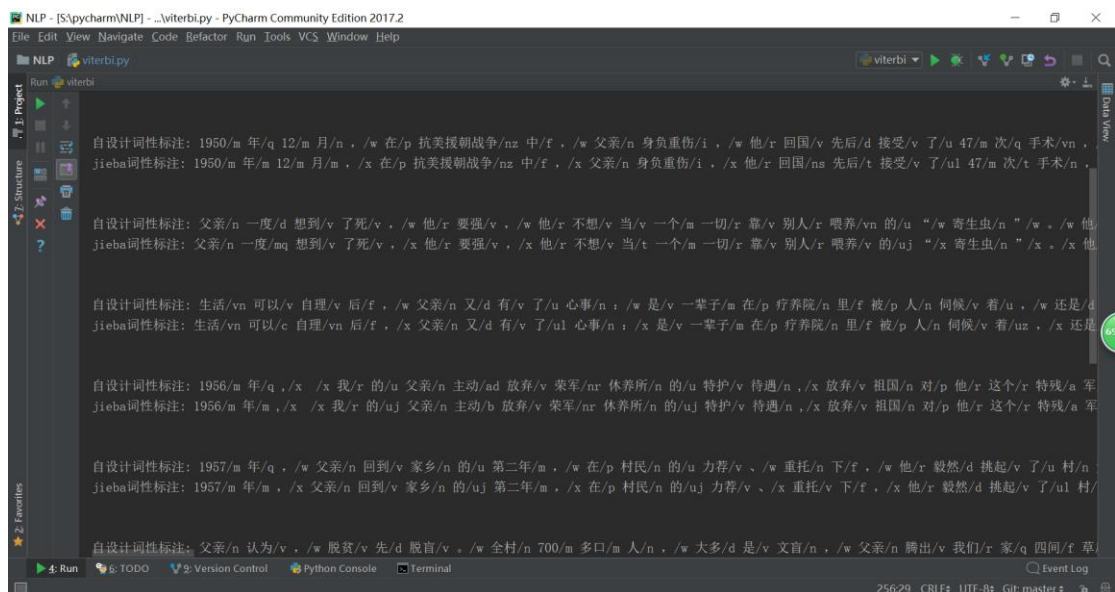
(4) 统计正确率。

五、实验结果截图

(1) 输入文本（对应 print_result 函数）

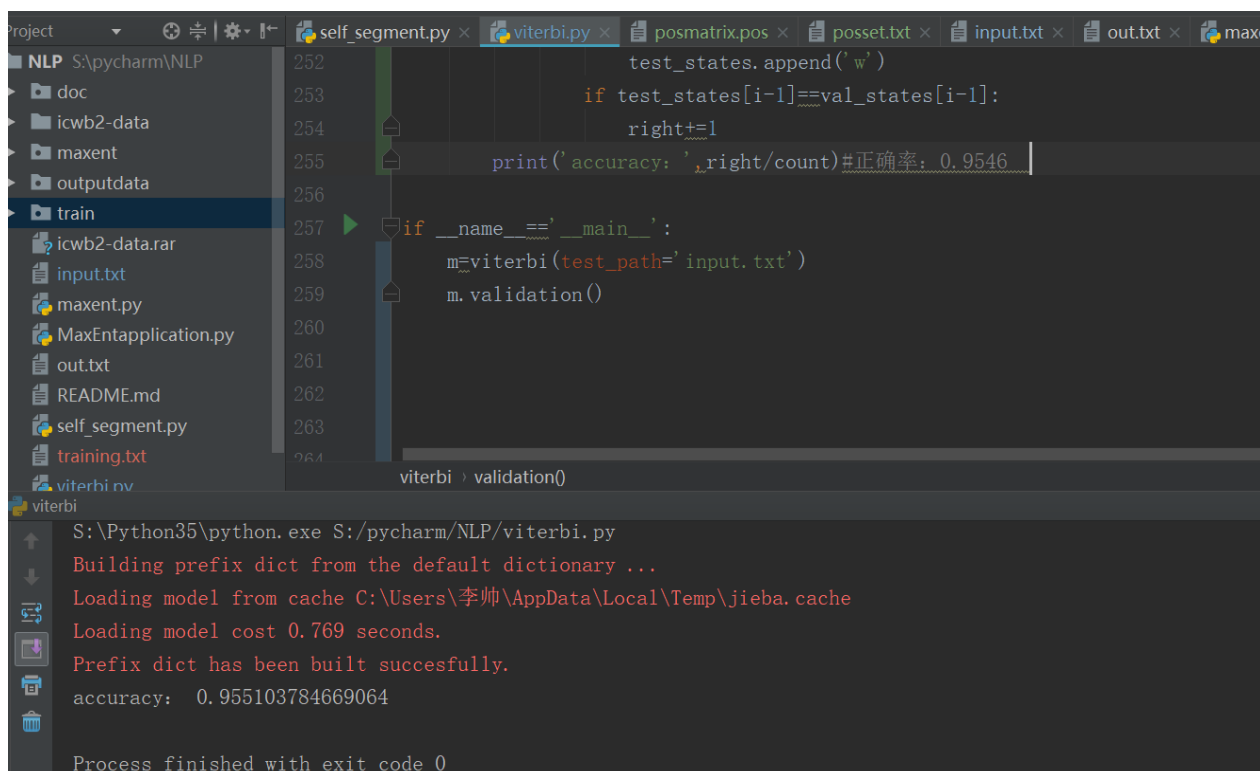


(2) 分词标注结果（我自己设计的维特比算法结果与 jieba 标注比对）



(3) 交叉验证

正确率：95.51%



```
project S:\pycharm\NLP
├── doc
├── icwb2-data
├── maxent
├── outputdata
├── train
│   ├── icwb2-data.rar
│   ├── input.txt
│   ├── maxent.py
│   ├── MaxEntapplication.py
│   ├── out.txt
│   ├── README.md
│   ├── self_segment.py
│   ├── training.txt
│   └── viterbi.py
└── viterbi.py

252
253
254
255
256
257
258
259
260
261
262
263
264

test_states.append('w')
if test_states[i-1]==val_states[i-1]:
    right+=1
print('accuracy: ',right/count)#正确率: 0.9546

if __name__=='__main__':
    m=viterbi(test_path='input.txt')
    m.validation()

viterbi validation()

S:\Python35\python.exe S:/pycharm/NLP/viterbi.py
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\李帅\AppData\Local\Temp\jieba.cache
Loading model cost 0.769 seconds.
Prefix dict has been built successfully.
accuracy: 0.955103784669064
Process finished with exit code 0
```

六、总结

基于统计的方法是目前词性标注的主流方法，其中 HMM 结合 Viterbi 算法的方法最常见的。利用几个标注好的语料库进行训练得到转移概率与输出概率，再用三种不同的处理未登录词的方法，使得开放测试的争取率基本在 95% 以上，但是第三种基于极大似然的估计算法只是比单纯设置输出概率为 1 的方法提高 0.1% 的准确率，可能是过于依赖未登录词的前一个单词，但是如果前一个单词也是未登录词，那么不准确的概率将会大大增加，效果甚至不如直接设置为 1，所以改善该算法的依赖词域将会是一个可能的方向。

七、源码与测试文件

Viterbi 算法实现以及测试验证源文件：viterbi.py

训练集：train 文件夹 trainningtext1-5.txt， training.txt 文件

验证集：trian 文件夹 trainnintext6.TXT 文件

测试集：input.txt

输出集：targetout.txt

八、参考文献

【1】张孝飞 等. 词性标注中生词处理算法研究. 中文信息学报.2003 05-0001-05