NLP 第一次课程大作业报告

1120191731 王玉丹

目录

[一、 实验问题 3](#_Toc87558845)

[1. 中文分词 3](#_Toc87558846)

[2. 中文词性标注 3](#_Toc87558847)

[二、 实验原理 3](#_Toc87558848)

[三、 实验实施与设计 3](#_Toc87558849)

[1. 语料库整理 3](#_Toc87558850)

[2. 分词 3](#_Toc87558851)

[3. 标注词性 7](#_Toc87558852)

[4. 实验实施中的问题以及解决方法汇总 8](#_Toc87558853)

[四、 实验结果 9](#_Toc87558854)

[五、 使用系统说明 9](#_Toc87558855)

1. 实验问题
   1. 中文分词
      1. MM
      2. HMM+Viterbi
   2. 中文词性标注
2. 实验原理
3. 实验实施与设计
4. 语料库整理

本文用于统计的语料库：

2014 人民日报预料（部分），部分留给测试集

预处理前

|  |  |
| --- | --- |
| 词语个数 | 275439 |
| 句子个数 | 286268 |
| 前10长的词 | 英文句子 |
| 前70长的词 | 英文句子+一个书名 |

说明：

从前10长的词语统计，直到前70长的词中，有一个中文词语出现

1. 数据库预处理：
2. 英文处理：

为了不影响词语前后出现概率转移转移统计，同时方便中文统计，将所有的英文都改为“Eng” 代表该处是英文

1. 分词

* N—gram

1. 构建n-gram 语言模型
2. 训练：

采用json格式进行保存，文件的格式： 词{词性1：个数，词性2：个数}

1. 统计一元语法

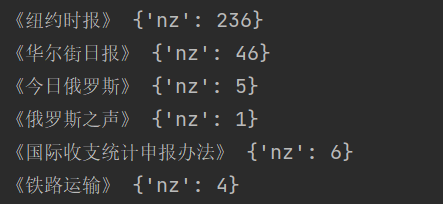
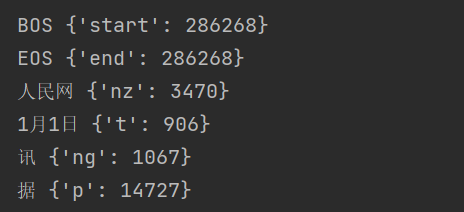
内容：

预料库原有的分好的最小单元的词、书名加书名号、语料库给出的由多个词语组成的新词、EOS、BOS

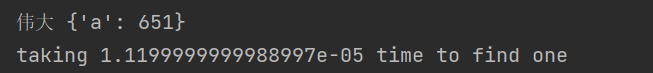
格式：

{w1:{t1:n1,t2:n2}, } (w:词语，t:词性，n:个数)

结果展示：

查询效率：



1. 统计二元语法

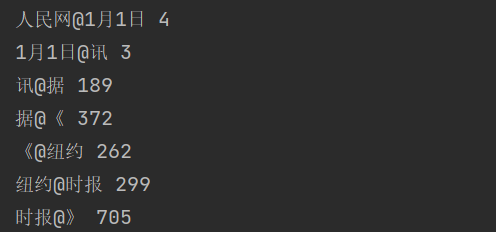
内容：

2-gram 词频统计

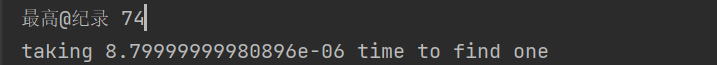
格式：

{w1@w2:n1 , …}

结果展示：



查询效率：



1. 统计三元语法

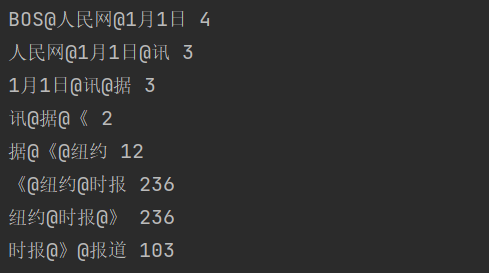
内容：

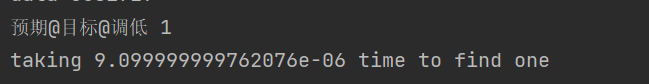
3-gram 词频统计

格式：

{w1@w2:n1 , …}

结果展示：



查询效率：

1. n-gram LM：

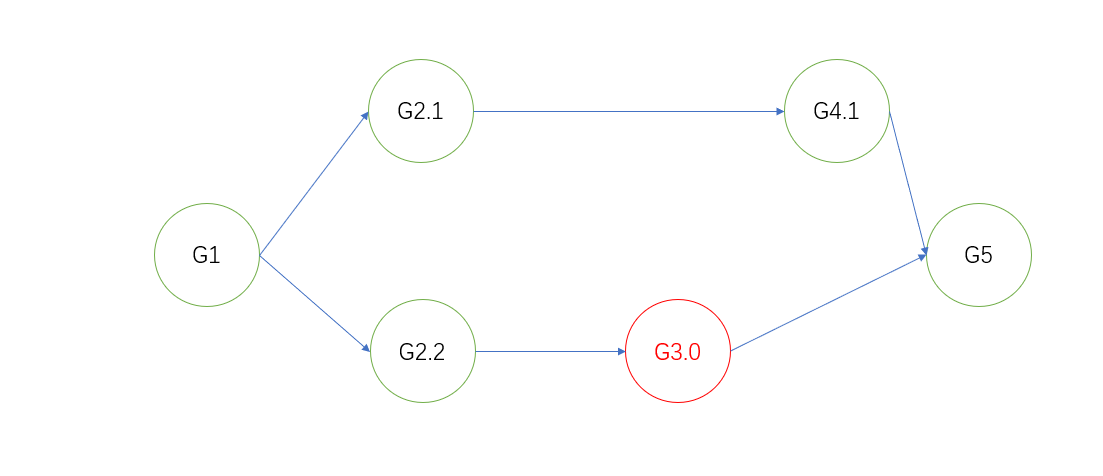
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n | 总数 | 最频繁有意义项 | 最大频数 | 频数为1的项数 | 频数为1的项占全部的比例 | 最长长度 |
| 1 | 15439391 | 的 | 558870 | 102402 | 37.12% | 23 |
| 2 | 2771625 | 李嘉瑞@王金跃 | 14442319 | 1471931 | 53.11% | 29 |
| 3 | 6581729 | 记者@李嘉瑞@王金跃 | 14156051 | 4377143 | 66.50% | 34 |

1. 生成词图

数据结构说明：

本实验采用python的list来存储由一个句子（句子的首位部分由人工加入B,E作为标识）表示生成的词图，graph

其中graph[i] 为在句子[i]位置可以产生的所有1-gram，（即字典中的1-gram）。graph[i]中的word与 graph[i+len(word)]中的词均有边相连

图示：（图中的红色结点代表语料库中没有的，用单字添加的）

算法说明：

伪代码如下，需要主要的是为了保证路径的连通性，如果当前结点的后继没有一元语法时，需要将单个的字加入词图中，直到有一元出现

Pseudocode

**Algorithm** Generate word-net

1. **Require**:
2. Open\_table: 1-gram need to be add in graph
3. Suu\_nodes : successive 1-grams for a particular 1-gram
4. Graph : word graph for current sentence
5. **Start**:
6. Add BOS into open\_table
7. While open\_table not empty :
8. Current\_1gram = Open\_table.pop()
9. Find suu\_nodes for Current\_1gram
10. If
11. suu\_nodes is empty
12. find latest 1-gram
13. add single word between lastest 1-garm and current\_word into graph
14. Else
15. Add suu\_nodes in grahp
16. Add suu\_nodes in open\_table
17. 维特比算法寻找最短路径
    1. 计算图中点与点的距离[1]

给词袋的联合分布律取对数，句子出现的概率转化为-log(p)的求和形式

记两个1-gram wi,wi+1之间的转移概率为p，图中结点之间距离即为-log(p)

数据平滑：[1]

采用拉普拉斯变换实现数据平滑，公式如下：

其中拉普拉斯变换的参数 λ μ 为服从U（1,0）的随机变量，由python的random库产生，N为1-gram 的总数

* 1. 数据结构说明

图的表示与上文相同，用python的list来维护最短路径，shortest\_path<list>

其中 shorest\_list[i]表示到graph第i层的节点，维护前是所有结点，维护后是到该层的最短结点，其中结点保存1.（layer\_index,in\_layer\_index）二维下标唯一确定2.到该结点的最短路径的距离

* MM

1. 正向最大匹配
2. 逆向最大匹配
3. 双向最大匹配：取FMM，BMM方法中词语数量较少的

最大匹配算法实现较易，此处不多说明

1. 标注词性
2. 词性统计，构建词性转移频率矩阵

共109类

1. 构建词性转移图

数据结构说明：

本实验采用python list数据类型 来存储词性转移图 class\_graph，共有n层，n为词的个数

其中class\_graph[i]代表第i个词语可能的词性

算法说明：

构建词性转移图较词网更为简单，只需添加对应的词语的词性即可，层与层之间的结点都是全连接的。

1. 维特比找最短路径
2. 计算距离：同分词相近，将联合概率转化为-log(p)的求和

数据平滑

词性标注中需要统计的概率：P1词性转移概率，P2某个词语的某个词性的概率

使用拉普拉斯数据平滑：

P1：

其中拉普拉斯变换的参数 λ μ 为服从U（1,0）的随机变量，由python的random库产生，N为词性转移概率矩阵的总数

P2：

其中拉普拉斯变换的参数 λ μ 为服从U（1,0）的随机变量，由python的random库产生，N为1-gram 的总数

1. 维特比算法寻找最短路径

数据结构说明：

用 python 的list数据类型存放最短路径信息shortest\_path，共n层，n为词语个数

其中shortest\_path [i]存储第i层每个结点的信息，[(该结点在本层的索引，其祖先结点在祖先层的索引)，到该结点的最短路径的距离]

1. 实验实施中的问题以及解决方法汇总
2. 词频统计中的数据平滑（具体解决方法见上文公式）
3. 最大匹配中最大检索范围的制定：最大检索范围和句长需取最小值
4. 构建词网时路径不连通问题（具体解决方法见上文伪代码块）
5. 计算出概率太小，导致结点之间的距离过大
6. 实验结果

本实验采用人民日报2014 作为测试集

测试集大小：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 正确率 | 召回率 | F1值 | 效率(每个句子的平均处理时间) |
| MM分词 | 22 | 30 | 25.38 | 0.6152188999999986 |
| n-gram分词 | 21 | 19 | 19.95 | 0.018221450000001305 |
| POS（HMM-viterbi） | 22 | 28 | 25 | 0.3553917499999999 |

实验结果分析：

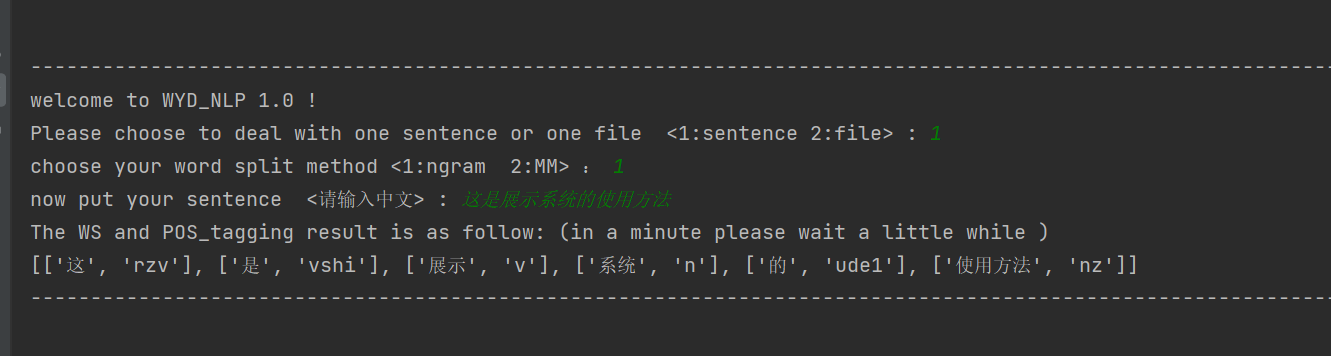
指标较低的原因：

1. 数字，系统将其全部分开，正确答案将数字作为一个
2. 概率统计数值不够科学，构建的词典不够完整
3. 使用系统说明

系统目前提供，n-gram，MM，两种分词方式和基于HMM的词性标注方式

可以选择处理单句，处理文件两种方式

处理单句效果如下

根据输出提示 <提示内容> 操作即可

关于系统的结构，详细信息可阅读source中的readme

以下为部分截图



参考文献：

[1]《自然语言处理入门》—何晗