NLP 第一次课程大作业报告

1120191731 王玉丹

目录

一、 实验问题

1. 中文分词
   i. MM
   ii. HMM+Viterbi
2. 中文词性标注

二、 实验原理

三、 实验实施与设计

1. 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：
   a 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

2. 分词

- N—gram

1) 构建 n-gram 语言模型
   a 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}

   i. 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：

```
《纽约时报》 {'nz': 236}
《华尔街日报》 {'nz': 46}
《今日俄罗斯》 {'nz': 5}
《俄罗斯之声》 {'nz': 1}
《国际收支统计申报办法》 {'nz': 6}
《铁路运输》 {'nz': 4}
```

```
BOS {'start': 286268}
EOS {'end': 286268}
人民网 {'nz': 3470}
1月1日 {'t': 906}
讯 {'ng': 1067}
据 {'p': 14727}
```

查询效率：

```
伟大 {'a': 651}
taking 1.1199999999988997e-05 time to find one
```

ii. 统计二元语法

内容：

2-gram 词频统计

格式：

{w1@w2:n1 , …}

结果展示：

```
人民网@1月1日 4
1月1日@讯 3
讯@据 189
据@《 372
《@纽约 262
纽约@时报 299
时报@》 705
```

查询效率：

```
最高@纪录 74
taking 8.79999999980896e-06 time to find one
```

iii. 统计三元语法

内容：

3-gram 词频统计

格式：

{w1@w2:n1 , …}

结果展示：

```
BOS@人民网@1月1日 4
人民网@1月1日@讯 3
1月1日@讯@据 3
讯@据@《 2
据@《@纽约 12
《@纽约@时报 236
纽约@时报@》 236
时报@》@报道 103
```

查询效率：

```
预期@目标@调低 1
taking 9.099999999762076e-06 time to find one
```

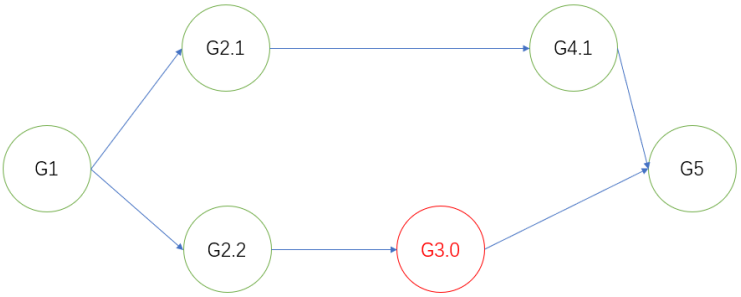
iv. n-gram LM：

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

2) 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
```

---

3) 维特比算法寻找最短路径

a) 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

$$\prod_{t=1}^{k+1} \hat{P}(w_t|w_{t\cdot 1}) \rightarrow -\sum_{t=1}^{k+1} \log \hat{P}(w_t|w_{t-1})$$

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：

$$\hat{P}(w_t|w_{t-1}) = \lambda\left(u\frac{c(w_{t-1}w)}{c(w_t)} + 1 - \mu\right) + (1-\lambda)\frac{c(w_t)+1}{N}$$

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

b) 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

3. 标注词性

1) 词性统计，构建词性转移频率矩阵
共 109 类
2) 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。

3) 维特比找最短路径

a) 计算距离：同分词相近，将联合概率转化为-log(p)的求和

$$\prod_{t=1}^{k+1}\hat{P}(w_t/\text{ti}\mid w_{t\cdot 1}/\text{tj}) \rightarrow -\sum_{t=1}^{k+1}\log\widehat{[P}(\text{ti-}>\text{tj})*\hat{P}(\text{wt-1/tj})$$

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：

$$\hat{P}(w_t|w_{t-1}) = \lambda\left(u\frac{c(w_{t-1}w)}{c(w_t)}+1-\mu\right)+(1-\lambda)\frac{c(w_t)+1}{N}$$

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：

$$\hat{P}(w_t|w_{t-1}) = \lambda\left(u\frac{c(w_{t-1}w)}{c(w_t)}+1-\mu\right)+(1-\lambda)\frac{c(w_t)+1}{N}$$

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

b) 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]

4. 实验实施中的问题以及解决方法汇总

①. 词频统计中的数据平滑（具体解决方法见上文公式）
②. 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
③. 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）

④. 计算出概率太小，导致结点之间的距离过大

四、 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

五、 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

```
--------------------------------------------------------------------------------------------------------------------
welcome to WYD_NLP 1.0 !
Please choose to deal with one sentence or one file  <1:sentence 2:file> : 1
choose your word split method <1:ngram  2:MM> :  1
now put your sentence  <请输入中文> : 这是展示系统的使用方法
The WS and POS_tagging result is as follow: (in a minute please wait a little while )
[['这', 'rzv'], ['是', 'vshi'], ['展示', 'v'], ['系统', 'n'], ['的', 'ude1'], ['使用方法', 'nz']]
--------------------------------------------------------------------------------------------------------------------
```

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

```
系统结构:

source:

        corpus_resource : 直接使用必须的数据

                n_gram : 已经训练好的 3-gram
                POS_source : 词性转移矩阵

        POS_tagging : HMM+Viterbi 分词

        WS_dict_MM : MM 分词  ( FMM BMM BIMM )

        WS_static_n_gram : n-gram + Viterbi 分词

        corpus_preprocess.py : 2014 人民日报预处理

        main.py : < 系统入口 >
```

参考文献：
[1]《自然语言处理入门》—何晗