

自然语言处理作业

第一次大作业 2021.10.20

07111908 班 1120193583 张昊杰

一、实验内容：

- 实现基于词典的分词方法和统计分词方法：两类方法中各实现一种即可；
- 对分词结果进行词性标注，也可以在分词的同时进行词性标注；
- 对分词及词性标注结果进行评价，包括 4 个指标：正确率、召回率、F1 值和效率。

二、实现平台

MacBook Air M1，全部使用 Python 进行实验

三、实验过程

1. 基于词典的分词方法中，我们使用了四种分词方法，即完全切分式，正向最长匹配，逆向最长匹配，双向最长匹配。此处代码见附录 1。
这里的词典我选择使用了北京大学统计好的词典作为词典参考来进行实验。

```
pythonProject -> fenci.py
pythonProject ~/Downloads
  <fenci>
    <fenci>
      <fenci>
        <fenci>
          <fenci>
            <fenci>
              <fenci>
                <fenci>
                  <fenci>
                    <fenci>
                      <fenci>
                        <fenci>
                          <fenci>
                            <fenci>
                              <fenci>
                                <fenci>
                                  <fenci>
                                    <fenci>
                                      <fenci>
                                        <fenci>
                                          <fenci>
                                            <fenci>
                                              <fenci>
                                                <fenci>
                                                  <fenci>
                                                    <fenci>
                                                      <fenci>
                                                        <fenci>
                                                          <fenci>
                                                            <fenci>
                                                              <fenci>
                                                                <fenci>
                                                                  <fenci>
                                                                    <fenci>
                                                                      <fenci>
                                                                        <fenci>
                                                                          <fenci>
                                                                            <fenci>
                                                                              <fenci>
                                                                                <fenci>
                                                                                  <fenci>
                                                                                    <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
                                                                                      <fenci>
................................................................
```

Run: fenci
/usr/bin/python3 /Users/zhj/Downloads/pythonProject/fenci.py
85584
完全切分: ['就', '读', '于', '北', '大', '学', '']
前向切分: ['就读', '于', '北', '大', '学']
后向切分: ['研究', '物', '起源']
双向切分: ['研究', '生物', '起源']
Process finished with exit code 0

```
pythonProject / fenci.py
pythonProject ~/Downloads
  |- fenci.py
  |- main.py
  |- my_cws_corpus.csv
  |- renmin.csv
  |- renmin.txt
  |- renmincixing.csv
  |- renmincixing.txt
  |- temp.csv
  |- test.csv
  |- tongjfenci.py
  |- train.lm

pythonProject - fenci.py
28   # 向最长匹配
29   # 从当前扫描位置的单字所有可能的结果，我们找最长的
30   def forward_segment(text, dic):
31       word_list = []
32       i = 0
33       while i < len(text):
34           longest_word = text[i]
35           for j in range(i + 1, len(text) + 1):
36               word = text[i:j]
37               if word in dic:
38                   if len(word) > len(longest_word):
39                       longest_word = word
40               word_list.append(longest_word)
41           i += len(longest_word)
42       return word_list
43
44   # 逆向最长匹配
45   def back_segment(text, dic):
46       word_list = []
47       i = len(text) - 1
48       while i >= 0:
49           longest_word = text[i]
50           for j in range(0, i):
51               word = text[j:i + 1]
52               if word in dic:
53                   if len(word) > len(longest_word):
54                       longest_word = word
55               word_list.append(longest_word)
56           i -= len(longest_word)
57       word_list.reverse()
58       return word_list
59
60
61
for i in range(test_sents_num) : for tmp in ans_list
```

```
pythonProject / fenci.py
pythonProject ~/Downloads
  |- fenci.py
  |- main.py
  |- my_cws_corpus.csv
  |- renmin.csv
  |- renmin.txt
  |- renmincixing.csv
  |- renmincixing.txt
  |- temp.csv
  |- test.csv
  |- tongjfenci.py
  |- train.lm

pythonProject - fenci.py
51       longest_word = text[i]
52       for j in range(0, i):
53           word = text[j:i + 1]
54           if word in dic:
55               if len(word) > len(longest_word):
56                   longest_word = word
57               word_list.append(longest_word)
58       i -= len(longest_word)
59   word_list.reverse()
60   return word_list
61
62   # 统计单字成词的个数
63   def count_single_char(word_list: list):
64       return sum(1 for word in word_list if len(word) == 1)
65
66
67   # 双向最长匹配，实际上是做了个融合
68   def bidirectional_segment(text, dic):
69       f = forward_segment(text, dic)
70       b = back_segment(text, dic)
71       # 词数更少更优先
72       if len(f) < len(b):
73           return f
74       elif len(f) > len(b):
75           return b
76       else:
77           # 单字更少更优先
78           if count_single_char(f) < count_single_char(b):
79               return f
80           else:
81               return b
82
83
84
85   dic = load_dictionary()
86   for i in range(test_sents_num) : for tmp in ans_list
```

我们随意输入几个句子并输出结果，根据结果来看，各个方法分词的效果还算不错。接下来我们使用人民日报的分好的语料库进行一个全篇的预测。

```

pythonProject - fenci.py
pythonProject ~/Downloads
  |- pythonProject
    |- fenci.py
    |- chuli.py
    |- chulixing.py
    |- chixing_tongji.py
    |- CoreNatureDictionary.cs
    |- cws.csv
    |- danju.csv
    |- fenci.py
    |- main.py
    |- my_cws_corpus.csv
    |- remin.csv
    |- remin.txt
    |- reminicing.csv
    |- reminicing.txt
    |- temp.csv
    |- test.csv
    |- tongfenci.py
    |- train.lm
  |- External Libraries
  |- Scratches and Consoles

pythonProject - fenci.py
pythonProject ~/Downloads
  |- pythonProject
    |- fenci.py
    |- chuli.py
    |- chulixing.py
    |- chixing_tongji.py
    |- CoreNatureDictionary.cs
    |- cws.csv
    |- danju.csv
    |- fenci.py
    |- main.py
    |- my_cws_corpus.csv
    |- remin.csv
    |- remin.txt
    |- reminicing.csv
    |- reminicing.txt
    |- temp.csv
    |- test.csv
    |- tongfenci.py
    |- train.lm
  |- External Libraries
  |- Scratches and Consoles

for i in range(test_sents_num)
P,R,F1 0.8412775520097979 0.8864272778728313 0.863264710717806
26.91647696495056
Process finished with exit code 0

```

首先我们使用前向切分的方法，可以看到 P , R , $F1$ 三个值，以及用时 26.92s。

```

pythonProject - fenci.py
pythonProject ~/Downloads
  |- pythonProject
    |- fenci.py
    |- chuli.py
    |- chulixing.py
    |- chixing_tongji.py
    |- CoreNatureDictionary.cs
    |- cws.csv
    |- danju.csv
    |- fenci.py
    |- main.py
    |- my_cws_corpus.csv
    |- remin.csv
    |- remin.txt
    |- reminicing.csv
    |- reminicing.txt
    |- temp.csv
    |- test.csv
    |- tongfenci.py
    |- train.lm
  |- External Libraries
  |- Scratches and Consoles

pythonProject - fenci.py
pythonProject ~/Downloads
  |- pythonProject
    |- fenci.py
    |- chuli.py
    |- chulixing.py
    |- chixing_tongji.py
    |- CoreNatureDictionary.cs
    |- cws.csv
    |- danju.csv
    |- fenci.py
    |- main.py
    |- my_cws_corpus.csv
    |- remin.csv
    |- remin.txt
    |- reminicing.csv
    |- reminicing.txt
    |- temp.csv
    |- test.csv
    |- tongfenci.py
    |- train.lm
  |- External Libraries
  |- Scratches and Consoles

for i in range(test_sents_num):
    i += len(word)
    return ans

test_sents_num = len(test_sents)
print(test_sents_num)
P = 0
R = 0
for i in range(test_sents_num):
    xun_lian = back_segment(test_sents[i], dic)
    xun_lian_list = zhuan_huan(xun_lian)
    ans_list = zhuan_huan(ans_sents[i])

    xun_lian_set = set()
    for tmp in xun_lian_list:
        xun_lian_set.add(tuple(tmp))
    ans_list_set = set()
    for tmp in ans_list:
        ans_list_set.add(tuple(tmp))
    TP = ans_list_set & xun_lian_set
    p = len(TP) / len(xun_lian_list)
    r = len(TP) / len(ans_list)
    P += p
    R += r
    if i % 100 == 0:
        print(i / test_sents_num)

#求一个平均值
P,R,F1 0.8481196838880893 0.8934982673412537 0.8701751088698839
27.511391840649414
Process finished with exit code 0

```

接下来使用后向切分的分法，正确率微微提升，同时用时 27.51s。

The screenshot shows the PyCharm IDE interface. The left sidebar displays the project structure for 'pythonProject' with files like 'fenci.py', 'chulexing.py', 'chulei.py', etc. The main code editor window shows the 'fenci.py' file with Python code for processing Chinese text. The bottom right corner shows the run output terminal with command-line arguments and results.

```
pythonProject - fenci.py
pythonProject ~/Downloaded
fenci.py
chulei.py
chulexing.py
chulei_tongji.py
chulei_tongji.py
CoreNatureDictionary.csv
cws.csv
danju.csv
fenci.py
main.py
my_cws_corpus.csv
remmin.csv
remmin.txt
remnincinxing.csv
remnincinxing.txt
temp.csv
test.csv
tong/fenci.py
train.lm

> External Libraries
Scratches and Consoles

Run: fenci
P, R, F1 0.8488645010171012 0.8938939571664417 0.8707974927325003
52.729925870895386

Process finished with exit code 0
```

最后使用双向切分的方法，正确率几乎不变，用时 52s，几乎翻倍的时间。

	P	R	F1	Time
前向	0.8413	0.8864	0.8864	26.91
后向	0.8482	0.8934	0.8934	27.51
双向	0.8489	0.8939	0.8708	52.73

可见，基于词典的分词方式，不论哪种方法，正确率基本稳定在这个范围上。效果还算可以。

2. 基于语料的统计分词，使用二元语法模型来构建词库，然后将句子生成词网，在使用 viterbi 算法来计算最优解，在这当中使用+1 法来处理前后之间概率。代码放在附录 2 首先我们处理原始文件，将人民日报语料库的带空格的文件处理成 csv 的表格形式方便我们进行分词的整理。

pythonProject - chuli.py

Project

pythonProject - ~/Download

chuli.py
chulixing.py
cixing_tongji.py
CoreNatureDictionary.csv
cws.csv
danju.csv
fenci.py
main.py
my_cws_corpus.csv
remmin.csv
remmin.txt
remminicing.csv
remminicing.txt
temp.csv
test.csv
tongjfenci.py
train.lm

> External Libraries
Scratches and Consoles

1 import csv
2
3 mat = []
4
5 with open("renmin.txt", "r") as f: # 打开文件
6 for line in f:
7 mat.append([l for l in line.split()])
8
9 print(mat)
10
11 with open('renmin.csv', 'w', newline='') as csvfile:
12 writer = csv.writer(csvfile)
13 for row in mat:
14 if len(row) != 0:
15 writer.writerow(row)

Run TODO Problems Debug Python Packages Python Console Terminal

Python Debugger Extension Available: Cython extension speeds up Python debugging // Install How does it work (44 minutes ago)

8:1 CR UTF-8 4 spaces Python 3.10 Event Log

向 欢 希 望 的 新 世 纪 ——一九九八年 新 年 讲 话 (附 图 片 1 张)

中共中央总书记、国家主席江泽民

1998年1月1日，江泽民同志在中南海怀仁堂会见了全国各界人士、海外华侨、港澳同胞、归侨侨眷和归国留学人员，共迎新年。江泽民同他们亲切握手，表示诚挚的节日祝贺。江泽民说，1997年是新中国发展历史上非常重要的、很不平凡的一年。中国人民坚决继承邓小平同志的遗志，继续把建设有中国特色社会主义事业推向前进。中国成功地召开了十五次全国代表大会，确立了邓小平理论在全党的指导地位，总结百年历史，制定了党在世纪之交的行动纲领。江泽民指出，过去一年，中国的改革开放和现代化建设继续取得进展，国民经济保持了“高增长、低通胀”的良好发展态势，农业生产再获丰收，取得好的收成，企业改革继续深化，人民生活改善，社会安定。在这一年，中国同世界许多国家的友好合作关系进一步加强，同美国、俄罗斯、日本等国的双边关系有了新的发展，同周边国家的关系也有了新的发展，同亚非拉国家的友好合作有了新的进展。在这一年，中国的外交工作取得了新的成就，赢得了广泛的国际声誉。新的一年，我们将继续坚持独立自主的和平外交政策，同世界各国人民一道，为促进世界的和平与发展而努力。

江泽民强调，1998年是中国人民满怀信心地开创新的历史的一年。新的一年，我们在经济建设发展中还面临不少困难，但我们有邓小平理论的指引，有改革开放20年来取得的伟大成就和积累的丰富经验，还有其他的各种有利条件。我们一定能够克服这些困难，迎接新的挑战。让我们一起努力，进一步解放思想，实事求是，把握机遇，开拓进取，建好、建强社会主义的综合国力，实现祖国的完全统一，是海内外全体中国人的心声。通过中英双方的合作和努力，按照“一国两制”方针和澳门《基本法》，1999年12月澳门的回归一定会顺利实现。

江泽民指出，新的一年，是大有作为的一年，是大有希望的一年，是大有心所向的一年。任何企划，两个“中”，“台湾”大陆的“中”，“香港”澳门的“中”，都是实现和平、尽早、稳妥、更有效的，在一个中国的前提下，以和平、友大的方式，通过谈判解决的。两岸和平、稳定、繁荣、统一，是中华民族的根本利益所在。

江泽民强调，新的一年，科技正在为各国的经济发展、历史灿烂，但是，世界并不安宁。南北之间的贫富差距继续扩大；和平冲突时有发生，和平与发展的前景暗淡。新的一年，世界各民族人民要进一步团结起来，共同奋斗，建立公正合理的国际政治经济新秩序。

江泽民指出，新的一年，中国政府将继续奉行独立自主的和平外交政策，和平共处五项原则的原则，加强同联合国和其他国家组织的协调，促进在扩大开放、经济建设、环境保护、和平利用核能、打击国际犯罪等方面的合作。中国永远是维护世界和平与稳定的积极力量。中国人民愿与世界各国人民一道，为开创和平、稳定、繁荣的新局面而努力奋斗。

江泽民最后说，这到此为止，我将新年好，我拜大家新年快乐，家庭幸福！

谢谢！ (新华社北京1月1日电)

renmin	
125%	v
显示	缩放
+/-	工作表 1
	三 显
	添加类别 数据透视表
	插入 表格 图表 文本 形状 媒体 批注
	协作
	格式 管理
A	B C D E F G H I J K L M N O P Q R S T U V
1	迈向 充满 希望 的 新 世纪 —— 一九九八年 新年 讲话 (附 图片 1 张)
2	中共中央 总书记 、 国家 主席 江 泽民
3	(一九九七年 十二月 三十一日)
4	12月 31日 , 中共中央 总书记 、 国家 主席 江 泽民 发表 1998年 新年 讲话 (附 图片 1 张)
5	同胞们 朋友 们 女士 们 先生 们 :
6	在 1998年 来临 之际 , 我 十分 高兴 地 通过 中央 人民 广播 电台 、 中国 国际 广播 电台 和 中央 电视台
7	1997年 , 是 中国 发展 历史 上 非常 重要 的 很 不 平凡 的 一 年 。 中国 人民 决心 继承 邓
8	在 这 一 年 中 , 中国 的 改革 开放 和 现代化 建设 继续 向前 迈进 。 国民经济 保持 了 “ 高
9	在 这 一 年 中 , 中国 的 外交 工作 取得 了 重要 成果 。 通过 高层 互访 , 中国 与 美国
10	1998年 , 中国 人民 将 满怀信心 地 开创 新 的 业绩 。 尽管 我们 在 经济社会 发展 中 还 面临 不少 困难
11	实现 祖国 的 完全 统一 , 是 海内外 全体 中国人 的 共同 心愿 。 通过 中 菲 双方 的 合作 和
12	台湾 是 中国 领土 不可分割 的 一部分 。 完成 祖国 统一 , 是 大势所趋 , 民心所向 。 任何 企图 制造
13	环顾 全球 , 日益 密切 的 世界经济 联系 , 日新月异 的 科技 进步 , 正在 为 各国 经济 的 发展 提供
14	中国政府 将 继续 坚持 独立自主的 和平 外交 政策 , 在 和平共处 五项 原则 的 基础 上 努力 发展
15	在 这 辞旧迎新的 美好 时刻 , 我 祝 大家 新年 快乐 , 家庭 幸福 !
16	谢谢 ! (新华社 北京 12月 31日 电)
17	在 十五大 精神 指引 下 胜利 前进 —— 元旦 献辞
18	我们 即将 以 丰收 的 喜悦 送 走 牛年 , 以 昂扬 的 斗志 迎来 虎年 , 我们 伟大 祖国 在 新
19	刚刚 过去 的 一 年 , 大气磅礴 , 语惊四座 , 在 这 一 年 , 以 江 泽民 同志 为 核心 的 党 和 国家 实现 伟大 的 历史 转折 ,
20	1998年 , 是 全面 贯彻 落实 党 的 十五大 提出 的 第一年 , 各条战线 改革 和
21	今年 是 党 的 十一 届 三中全会 召开 20周年 , 是 我们 党 和 国家 实现 伟大 的 历史 转折 ,
22	我们 要 更 好 地 坚持 解放思想 、 实事求是 的 思想 路线 , 解放思想 、 实事求是 , 是 邓小平理论 的 精髓 , 是
23	我们 要 更 好 地 坚持 以 经济 建设 为 中心 。 各项 工作 必须 以 经济 建设 为 中心 , 以 经济 建设
24	我们 要 更 好 地 坚持 “ 两手抓 、 两手都 要 硬 ” 的 方针 。 在 坚持 以 经济 建设 为 中心 , 以 经济 建设
25	我们 要 更 好 地 发扬 求真务实 、 密切 联系 群众 的 作风 。 这 是 把 党 的 方针 、 政策
26	1998 瞩目 中华 。 新 的 机遇 和 挑战 , 催 人 进取 ; 新 的 目标 和 途经 ,

```

pythonProject - tongjifenci.py
pythonProject ~/Download
  |- Project
    |- pythonProject -> Download
      |- tongjifenci.py
      |- cixing_tongji.py
      |- chulicixing.py
      |- fenci.py
      |- main.py
      |- my_cws_corpus.csv
      |- remin.csv
      |- remin.txt
      |- remincixing.csv
      |- remincixing.txt
      |- temp.csv
      |- test.csv
      |- tongjifenci.py
      |- train.lm
    |- External Libraries
    |- Scratches and Consoles

```

```

# 二元语法模型
def bgram(sents: list):
    dic = {}
    dic["#开始#"] = dict()
    for sent in sents:
        for i in range(0, len(sent) - 1):
            if sent[i] not in dic:
                dic[sent[i]] = dict()
                dic[sent[i]][sent[i + 1]] = 1
            else:
                if sent[i + 1] in dic[sent[i]]:
                    dic[sent[i]][sent[i + 1]] += 1
                else:
                    dic[sent[i]][sent[i + 1]] = 1
        if sent[0] not in dic["#开始#"]:
            dic["#开始#"][sent[0]] = 1
        else:
            dic["#开始#"][sent[0]] += 1
    if sent[len(sent) - 1] not in dic:
        dic[sent[len(sent) - 1]] = dict()
        dic[sent[len(sent) - 1]]["#未表示#"] = 1
    elif "#未表示#" not in dic[sent[len(sent) - 1]]:
        dic[sent[len(sent) - 1]]["#未表示#"] = 1
    else:
        dic[sent[len(sent) - 1]]["#未表示#"] += 1
    return dic

bi_gram = bgram(fenci_sents)

fen_ci() -> calculate_weight()  if word_one in gram

```

这里是生成句子词网的核心代码：

```

pythonProject - tongjifenci.py
pythonProject ~/Download
  |- Project
    |- pythonProject -> Download
      |- tongjifenci.py
      |- cixing_tongji.py
      |- chulicixing.py
      |- fenci.py
      |- main.py
      |- my_cws_corpus.csv
      |- remin.csv
      |- remin.txt
      |- remincixing.csv
      |- remincixing.txt
      |- temp.csv
      |- test.csv
      |- tongjifenci.py
      |- train.lm
    |- External Libraries
    |- Scratches and Consoles

```

```

def fen_ci(text):
    # 生成一元语法词网
    def generate_wordnet(gram, text):
        net = [[ ] for _ in range(len(text) + 2)]
        for i in range(len(text)):
            for j in range(i + 1, len(text) + 1):
                word = text[i:j]
                if word in gram:
                    net[i + 1].append(word)
        i = 1
        while i < len(net) - 1:
            if len(net[i]) == 0: # 空白行
                i += 1
            for j in range(i + 1, len(net) - 1):
                # 寻找第一个非空行
                if len(net[j]):
                    break
            # 填补i, j之间的空白行
            net[i].append(text[i - 1:j - 1])
            i = j
        else:
            i += len(net[i] - 1)
        return net

    # 测试一个句子
    si_net = generate_wordnet(si_gram, text)

    # print(si_net)

    def calculate_gram_sum(gram: dict):
        num = 0
        for g in gram.keys():
            num += sum(gram[g].values())
        fen_ci() -> generate_wordnet() -> while i < len(net) - 1 -> if len(net[i]) == 0
    fen_ci() -> generate_wordnet() -> while i < len(net) - 1 -> if len(net[i]) == 0

```

这里是+1平滑处理的核心代码：

```

pythonProject - tongjifenci.py
pythonProject ~/Download 172
  chuli.py
  chulixing.py
  cixing_tongji.py
  CoreNatureDictionary.cs
  danju.csv
  cws.csv
  fenci.py
  main.py
  my_cws_corpus.csv
  remin.csv
  remin.txt
  reminicing.csv
  reminicing.txt
  temp.csv
  test.csv
  tongjifenci.py
  train.lm
  External Libraries
  Scratches and Consoles
  Favorites
  Structure
  Run TODO Problems Debug Python Packages Python Console Terminal
  Python Debugger Extension Available: Cython extension speeds up Python debugging // Install How does it work (50 minutes ago)
  191
  192
  193
  194
  195
  196
  197
  198
  199
  200
  201
  202
  203
  204
  205
  206
  207
  208
  209
  210
  211
  212
  213
  214
  215
  216
  217
  218
  219
  220
  221
  222
  223
  224
  225
  226
  227
  228
  229
  230
  231
  232
  233
  234
  235
  236
  237
  238
  239
  240
  241
  242
  243
  244
  245
  246
  247
  248
  249
  250
  251
  252
  253
  254
  255
  256
  257
  258
  259
  260
  261
  262
  263
  264
  265
  266
  267
  268
  269
  270
  271
  272
  273
  274
  275
  276
  277
  278
  279
  280
  281
  282
  283
  284
  285
  286
  287
  288
  289
  290
  291
  292
  293
  294
  295
  296
  297
  298
  299
  300
  301
  302
  303
  304
  305
  306
  307
  308
  309
  310
  311
  312
  313
  314
  315
  316
  317
  318
  319
  320
  321
  322
  323
  324
  325
  326
  327
  328
  329
  330
  331
  332
  333
  334
  335
  336
  337
  338
  339
  340
  341
  342
  343
  344
  345
  346
  347
  348
  349
  350
  351
  352
  353
  354
  355
  356
  357
  358
  359
  360
  361
  362
  363
  364
  365
  366
  367
  368
  369
  370
  371
  372
  373
  374
  375
  376
  377
  378
  379
  380
  381
  382
  383
  384
  385
  386
  387
  388
  389
  390
  391
  392
  393
  394
  395
  396
  397
  398
  399
  400
  401
  402
  403
  404
  405
  406
  407
  408
  409
  410
  411
  412
  413
  414
  415
  416
  417
  418
  419
  420
  421
  422
  423
  424
  425
  426
  427
  428
  429
  430
  431
  432
  433
  434
  435
  436
  437
  438
  439
  440
  441
  442
  443
  444
  445
  446
  447
  448
  449
  450
  451
  452
  453
  454
  455
  456
  457
  458
  459
  460
  461
  462
  463
  464
  465
  466
  467
  468
  469
  470
  471
  472
  473
  474
  475
  476
  477
  478
  479
  480
  481
  482
  483
  484
  485
  486
  487
  488
  489
  490
  491
  492
  493
  494
  495
  496
  497
  498
  499
  500
  501
  502
  503
  504
  505
  506
  507
  508
  509
  510
  511
  512
  513
  514
  515
  516
  517
  518
  519
  520
  521
  522
  523
  524
  525
  526
  527
  528
  529
  530
  531
  532
  533
  534
  535
  536
  537
  538
  539
  540
  541
  542
  543
  544
  545
  546
  547
  548
  549
  550
  551
  552
  553
  554
  555
  556
  557
  558
  559
  560
  561
  562
  563
  564
  565
  566
  567
  568
  569
  570
  571
  572
  573
  574
  575
  576
  577
  578
  579
  580
  581
  582
  583
  584
  585
  586
  587
  588
  589
  590
  591
  592
  593
  594
  595
  596
  597
  598
  599
  600
  601
  602
  603
  604
  605
  606
  607
  608
  609
  610
  611
  612
  613
  614
  615
  616
  617
  618
  619
  620
  621
  622
  623
  624
  625
  626
  627
  628
  629
  630
  631
  632
  633
  634
  635
  636
  637
  638
  639
  640
  641
  642
  643
  644
  645
  646
  647
  648
  649
  650
  651
  652
  653
  654
  655
  656
  657
  658
  659
  660
  661
  662
  663
  664
  665
  666
  667
  668
  669
  670
  671
  672
  673
  674
  675
  676
  677
  678
  679
  680
  681
  682
  683
  684
  685
  686
  687
  688
  689
  690
  691
  692
  693
  694
  695
  696
  697
  698
  699
  700
  701
  702
  703
  704
  705
  706
  707
  708
  709
  710
  711
  712
  713
  714
  715
  716
  717
  718
  719
  720
  721
  722
  723
  724
  725
  726
  727
  728
  729
  730
  731
  732
  733
  734
  735
  736
  737
  738
  739
  740
  741
  742
  743
  744
  745
  746
  747
  748
  749
  750
  751
  752
  753
  754
  755
  756
  757
  758
  759
  759
  760
  761
  762
  763
  764
  765
  766
  767
  768
  769
  770
  771
  772
  773
  774
  775
  776
  777
  778
  779
  779
  780
  781
  782
  783
  784
  785
  786
  787
  788
  789
  789
  790
  791
  792
  793
  794
  795
  796
  797
  798
  799
  799
  800
  801
  802
  803
  804
  805
  806
  807
  808
  809
  809
  810
  811
  812
  813
  814
  815
  816
  817
  818
  819
  819
  820
  821
  822
  823
  824
  825
  826
  827
  828
  829
  829
  830
  831
  832
  833
  834
  835
  836
  837
  838
  839
  839
  840
  841
  842
  843
  844
  845
  846
  847
  848
  849
  849
  850
  851
  852
  853
  854
  855
  856
  857
  858
  859
  859
  860
  861
  862
  863
  864
  865
  866
  867
  868
  869
  869
  870
  871
  872
  873
  874
  875
  876
  877
  878
  879
  879
  880
  881
  882
  883
  884
  885
  886
  887
  888
  889
  889
  890
  891
  892
  893
  894
  895
  896
  897
  898
  899
  899
  900
  901
  902
  903
  904
  905
  906
  907
  908
  909
  910
  911
  912
  913
  914
  915
  916
  917
  918
  919
  920
  921
  922
  923
  924
  925
  926
  927
  928
  929
  930
  931
  932
  933
  934
  935
  936
  937
  938
  939
  940
  941
  942
  943
  944
  945
  946
  947
  948
  949
  950
  951
  952
  953
  954
  955
  956
  957
  958
  959
  959
  960
  961
  962
  963
  964
  965
  966
  967
  968
  969
  969
  970
  971
  972
  973
  974
  975
  976
  977
  978
  979
  979
  980
  981
  982
  983
  984
  985
  986
  987
  988
  989
  989
  990
  991
  992
  993
  994
  995
  996
  997
  998
  999
  999
  1000
  1001
  1002
  1003
  1004
  1005
  1006
  1007
  1008
  1009
  1009
  1010
  1011
  1012
  1013
  1014
  1015
  1016
  1017
  1018
  1019
  1019
  1020
  1021
  1022
  1023
  1024
  1025
  1026
  1027
  1028
  1029
  1029
  1030
  1031
  1032
  1033
  1034
  1035
  1036
  1037
  1038
  1039
  1039
  1040
  1041
  1042
  1043
  1044
  1045
  1046
  1047
  1048
  1049
  1049
  1050
  1051
  1052
  1053
  1054
  1055
  1056
  1057
  1058
  1059
  1059
  1060
  1061
  1062
  1063
  1064
  1065
  1066
  1067
  1068
  1069
  1069
  1070
  1071
  1072
  1073
  1074
  1075
  1076
  1077
  1078
  1078
  1079
  1080
  1081
  1082
  1083
  1084
  1085
  1086
  1087
  1088
  1088
  1089
  1090
  1091
  1092
  1093
  1094
  1095
  1096
  1097
  1097
  1098
  1099
  1099
  1100
  1101
  1102
  1103
  1104
  1105
  1106
  1107
  1108
  1109
  1109
  1110
  1111
  1112
  1113
  1114
  1115
  1116
  1117
  1118
  1119
  1119
  1120
  1121
  1122
  1123
  1124
  1125
  1126
  1127
  1128
  1129
  1129
  1130
  1131
  1132
  1133
  1134
  1135
  1136
  1137
  1138
  1138
  1139
  1140
  1141
  1142
  1143
  1144
  1145
  1146
  1147
  1148
  1148
  1149
  1150
  1151
  1152
  1153
  1154
  1155
  1156
  1157
  1158
  1158
  1159
  1160
  1161
  1162
  1163
  1164
  1165
  1166
  1167
  1168
  1168
  1169
  1170
  1171
  1172
  1173
  1174
  1175
  1176
  1177
  1177
  1178
  1179
  1180
  1181
  1182
  1183
  1184
  1185
  1186
  1187
  1187
  1188
  1189
  1190
  1191
  1192
  1193
  1194
  1195
  1196
  1197
  1197
  1198
  1199
  1199
  1200
  1201
  1202
  1203
  1204
  1205
  1206
  1207
  1208
  1209
  1209
  1210
  1211
  1212
  1213
  1214
  1215
  1216
  1217
  1218
  1219
  1219
  1220
  1221
  1222
  1223
  1224
  1225
  1226
  1227
  1228
  1229
  1229
  1230
  1231
  1232
  1233
  1234
  1235
  1236
  1237
  1238
  1238
  1239
  1240
  1241
  1242
  1243
  1244
  1245
  1246
  1247
  1248
  1248
  1249
  1250
  1251
  1252
  1253
  1254
  1255
  1256
  1257
  1258
  1258
  1259
  1260
  1261
  1262
  1263
  1264
  1265
  1266
  1267
  1267
  1268
  1269
  1270
  1271
  1272
  1273
  1274
  1275
  1276
  1277
  1277
  1278
  1279
  1280
  1281
  1282
  1283
  1284
  1285
  1286
  1287
  1287
  1288
  1289
  1290
  1291
  1292
  1293
  1294
  1295
  1296
  1297
  1297
  1298
  1299
  1299
  1300
  1301
  1302
  1303
  1304
  1305
  1306
  1307
  1308
  1308
  1309
  1310
  1311
  1312
  1313
  1314
  1315
  1316
  1317
  1318
  1318
  1319
  1320
  1321
  1322
  1323
  1324
  1325
  1326
  1327
  1328
  1328
  1329
  1330
  1331
  1332
  1333
  1334
  1335
  1336
  1337
  1338
  1338
  1339
  1340
  1341
  1342
  1343
  1344
  1345
  1346
  1347
  1347
  1348
  1349
  1350
  1351
  1352
  1353
  1354
  1355
  1356
  1357
  1358
  1358
  1359
  1360
  1361
  1362
  1363
  1364
  1365
  1366
  1367
  1368
  1368
  1369
  1370
  1371
  1372
  1373
  1374
  1375
  1376
  1377
  1377
  1378
  1379
  1380
  1381
  1382
  1383
  1384
  1385
  1386
  1387
  1387
  1388
  1389
  1390
  1391
  1392
  1393
  1394
  1395
  1396
  1397
  1397
  1398
  1399
  1399
  1400
  1401
  1402
  1403
  1404
  1405
  1406
  1407
  1408
  1408
  1409
  1410
  1411
  1412
  1413
  1414
  1415
  1416
  1417
  1418
  1418
  1419
  1420
  1421
  1422
  1423
  1424
  1425
  1426
  1427
  1428
  1428
  1429
  1430
  1431
  1432
  1433
  1434
  1435
  1436
  1437
  1438
  1438
  1439
  1440
  1441
  1442
  1443
  1444
  1445
  1446
  1447
  1447
  1448
  1449
  1450
  1451
  1452
  1453
  1454
  1455
  1456
  1457
  1458
  1458
  1459
  1460
  1461
  1462
  1463
  1464
  1465
  1466
  1467
  1468
  1468
  1469
  1470
  1471
  1472
  1473
  1474
  1475
  1476
  1477
  1477
  1478
  1479
  1480
  1481
  1482
  1483
  1484
  1485
  1486
  1487
  1487
  1488
  1489
  1490
  1491
  1492
  1493
  1494
  1495
  1496
  1497
  1497
  1498
  1499
  1499
  1500
  1501
  1502
  1503
  1504
  1505
  1506
  1507
  1508
  1508
  1509
  1510
  1511
  1512
  1513
  1514
  1515
  1516
  1517
  1518
  1518
  1519
  1520
  1521
  1522
  1523
  1524
  1525
  1526
  1527
  1528
  1529
  1529
  1530
  1531
  1532
  1533
  1534
  1535
  1536
  1537
  1538
  1538
  1539
  1540
  1541
  1542
  1543
  1544
  1545
  1546
  1547
  1547
  1548
  1549
  1550
  1551
  1552
  1553
  1554
  1555
  1556
  1557
  1558
  1558
  1559
  1560
  1561
  1562
  1563
  1564
  1565
  1566
  1567
  1568
  1568
  1569
  1570
  1571
  1572
  1573
  1574
  1575
  1576
  1577
  1577
  1578
  1579
  1580
  1581
  1582
  1583
  1584
  1585
  1586
  1587
  1587
  1588
  1589
  1590
  1591
  1592
  1593
  1594
  1595
  1596
  1597
  1597
  1598
  1599
  1599
  1600
  1601
  1602
  1603
  1604
  1605
  1606
  1607
  1608
  1608
  1609
  1610
  1611
  1612
  1613
  1614
  1615
  1616
  1617
  1618
  1618
  1619
  1620
  1621
  1622
  1623
  1624
  1625
  1626
  1627
  1628
  1628
  1629
  1630
  1631
  1632
  1633
  1634
  1635
  1636
  1637
  1638
  1638
  1639
  1640
  1641
  1642
  1643
  1644
  1645
  1646
  1647
  1647
  1648
  1649
  1650
  1651
  1652
  1653
  1654
  1655
  1656
  1657
  1658
  1658
  1659
  1660
  1661
  1662
  1663
  1664
  1665
  1666
  1667
  1668
  1668
  1669
  1670
  1671
  1672
  1673
  1674
  1675
  1676
  1677
  1678
  1678
  1679
  1680
  1681
  1682
  1683
  1684
  1685
  1686
  1687
  1687
  1688
  1689
  1690
  1691
  1692
  1693
  1694
  1695
  1696
  1697
  1697
  1698
  1699
  1699
  1700
  1701
  1702
  1703
  1704
  1705
  1706
  1707
  1708
  1708
  1709
  1710
  1711
  1712
  1713
  1714
  1715
  1716
  1717
  1718
  1718
  1719
  1720
  1721
  1722
  1723
  1724
  1725
  1726
  1727
  1728
  1729
  1729
  1730
  1731
  1732
  1733
  1734
  1735
  1736
  1737
  1738
  1738
  1739
  1740
  1741
  1742
  1743
  1744
  1745
  1746
  1747
  1747
  1748
  1749
  1750
  1751
  1752
  1753
  1754
  1755
  1756
  1757
  1758
  1758
  1759
  1760
  1761
  1762
  1763
  1764
  1765
  1766
  1767
  1768
  1768
  176
```

3. 对分词的结果进行词性标注。使用了统计的方法。此处代码同附录 2。

我们先对人民日报带词性标注的 txt 文件进行一个转换，转换成便于我操作的 csv 文件。

pythonProject - chulicixing.py

Project

pythonProject ~/Download

- chull.py
- chulicixing.py
- cixing_tongji.py
- CoreNatureDictionary.csv
- cws.csv
- danju.csv
- fenci.py
- main.py
- my_cws_corpus.csv
- renmin.csv
- renmin.txt
- renmincixing.csv
- renmincixing.txt
- temp.csv
- test.csv
- tongji_fenci.py
- train.lm

> External Libraries

> Scratches and Consoles

```
1 import csv
2
3 mat = []
4
5 with open("renmincixing.txt", "r") as f: # 打开文件
6     for line in f:
7         line = line[22:]
8         print(line)
9         mat.append([l for l in line.split()])
10
11 print(mat)
12
13 with open('renmincixing.csv', 'w', newline='') as csvfile:
14     writer = csv.writer(csvfile)
15     for row in mat:
16         if len(row) != 0:
17             writer.writerow(row)
```

Run TODO Problems Debug Python Packages Python Console Terminal

Python Debugger Extension Available: Cython supports Python debugging // install How does it work (today 1:42 PM)

Event Log

renmingxing																					
显示		插入		协作		格式		管理													
125%	缩放	三	四	插入类别	数据透视表	插入	表格	图表	文本	形状	媒体	批注	协作	格式	管理						
+	工作表1	A	B	C	D	E	F	G	H	I	J	K	L	M	N						
1	迈向/vt	充满/vt	希望/n	的/ud	新/a	世纪/n	--/wp	一九九八年/t	新年/n	讲话/n	(/wkz	附/vt	图片/n	1/m	张/qe) /wky					
2	中共中央/t	总书记/n	/wu	国家/n	主席/n	江/nrf	津民/nrg														
3	(/wkz	—一九九七年/t	十二月/t	三十日/t) /wky																
4	12月/t	31日/t	/wu	中共中央	总书记/n	/wu	国家/n	主席/n	江/nrf	津民/nrg	发表/vt	1998年/t	新年/t	讲话/n	(/wkz	迈向/vt	充满/vt	希望/r			
5	同向/n	们/k	/wu	朋友/n	们/k	/wu	女士/n	们/k	/wu	先生/n	们/k	: /wm									
6	在/p	1998年/t	来临/vi	之际/f	/wd	中国中央	总书记/n	/wu	十分/dc	离高/vi	地/ul	通过/p	[中央]/n	人民/n	广播/vn	电台/n	/wu	【中国】/n	国际/n	广播/vn	
7																					
8	1997年/t	/wd	是/m	中国/ns	发展/vn	历史/n	上(shang5)/f	非/nc	重要/a	/wd	很/dc	不/dt	平凡/a	/wd	-/m	年/qt	/wj	【中国】/n	国际/n	广播/vn	
9	在/p	该(zhei4)/rz	/m	年/qt	中/f	/wd	中国/ns	的/ud	改革/vn	开放/vn	和/o	现代化/vn	建设/vn	继续/vt	向前/vd	迈进/vt	/wj	【中国】/n	国际/n	广播/vn	
10	在/p	该(zhei4)/rz	/m	年/qt	中/f	/wd	中国/ns	的/ud	外交/vn	工作/vn	取向/vt	了/ul	重要/a	成果/n	/wj	通过/vt	高层/n	互惠/vn	【中国】/n	国际/n	广播/vn
11	1998年/t	/wd	是中国/ns	人民/n	将/d	满怀信心/vi	地/ul	开创/vt	新/a	/wd	业绩/vn	/wj	尽管/c	我们/r	在/p	经济/n	社会/n	发展/vn	【中国】/n	国际/n	广播/vn
12	实现/vt	祖国/n	的/ud	完全/ad	统一/vt	/wd	是/vi	海内外/s	全体/n	中国/ns	人/n	的/ud	共同/b	心愿/n	/wj	通过/p	中/n	葡/v			
13	台湾/ns	是/vi	中国/ns	领土/n	不可分割/vi	的/ud	/m	部分/ln	/wj	完成/vt	祖国/n	统一/vt	/wd	是/vi	大势所趋/vn	/wd	正在/rd	为(wei4)/p	各路/n	条/ce	
14	环顾/vt	球体/n	/wd	日益/dc	密切/a	的/ud	世界/n	经济/n	联系/vn	/wd	日新月异/vn	的/ud	科技/n	进步/vn	/wd	正在/rd	为(wei4)/p	各路/n	条/ce		
15	【中国】/n	政策/jin	将/d	媒体/vt	坚持/vt	奉行/vt	独立自主/nv	的/ud	和平/a	/wd	政策/n	/wd	在/p	和平共处/vn	/m	项/qe	原则/n	的/ud			
16	在/p	该(zhei4)/rz	辞旧迎新/vt	/wd	美好/a	时则/h	/wd	我/tr	祝/vt	大家/n	新年/t	快乐/a	/wd	家庭/n	幸福/a	/i	!/wt				
17	感谢/vt	/i	/wkz	新华社	北京/ns	1/2月/t	31日/t	日电/n) /wky												
18	在/p	十五大	精耕/n	指引/vt	下/f	胜利/vd	前进/vi	--/wp	元旦/n	献辞/n											
19	We/wr	即刻/d	以/p	丰收/vi	的/ud	喜获/vi	逆/td>														

下面是我们词性标注的训练核心代码，统计所有词性和他们的个数，获得相应的词性转移矩阵，在统计每个词语的词性概率。

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** pythonProject
- File:** tongjifenci.py
- Code Preview:** The code implements a function to calculate the frequency of each character in a list of Chinese sentences. It uses a dictionary to store character counts and iterates through each sentence, splitting it into words and then into individual characters. It also handles punctuation and ensures each character is counted only once per word.

```
pythonProject - tongjifenci.py
[Run] [Edit] [TODO] [Problems] [Debug] [Python Packages] [Python Console] [Terminal]
pythonProject ~/Download/tongjifenci.py
100     part = dict()
101     # 统计所有词性个数
102     for sent in cixing_sents:
103         for word in sent:
104             word_part = word.split('/')[1].split('!')[0].split('。')[0]
105             if word_part in part:
106                 part[word_part] += 1
107             else:
108                 part[word_part] = 1
109
110             part_len = len(part)
111             print(part, "一共有多少类: ", part_len)
112             print("总频次: ", sum(part.values()))
113
114             # 或得转移矩阵
115             trans = dict()
116             for sent in cixing_sents:
117                 for i in range(len(sent) - 1):
118                     one = sent[i].split('/')[1].split('!')[0].split('。')[0]
119                     two = sent[i + 1].split('/')[1].split('!')[0].split('。')[0]
120                     if one in trans:
121                         if two in trans[one]:
122                             trans[one][two] += 1
123                         else:
124                             trans[one][two] = 1
125
126                     else:
127                         trans[one] = dict()
128                         trans[one][two] = 1
129
130             # 每个词的词频概率
131             percent = dict()
132             for sent in cixing_sents:
133                 for word in sent:
134                     word_word = word.split('/')[1].split('!')[0].split('。')[0]
135                     fen_ci_l = calculate_weight(word_word, one_in_gamn) else
```

这里是推测词性的核心代码：

```
pythonProject pythonProject - tongjifenci.py
pythonProject tongjifenci.py

Project Pro.~ D ~ PythonProject ~ /Download longfenci.py cixing_tongji.py chulixing.py fenci.py chull.py
  + pythonProject ~ /Download
    + chull.py
    + chulixing.py
    + cixing_tongji.py
    + CoreNatureDictionary.csv
    + cws.csv
    + danju.csv
    + fenci.py
    + main.py
    + my_cws_corpus.csv
    + remin.csv
    + remin.txt
    + reminicing.csv
    + reminicing.txt
    + temp.csv
    + test.csv
    + tongjifenci.py
    + train.lm
  > External Libraries
  > Scratches and Consoles

def ci_xing(text):
    text_percent = []
    # 这里我们假设所有单词都已进入人民日报语料库了
    for word in text:
        word_percent = percent[word]
        text_percent.append(word_percent)
    # 打印(text_percent)

    # 现在我们来使用Viterbi算法计算出最佳的组成
    dis = [dict() for _ in range(len(text))]
    node = [dict() for _ in range(len(text))]
    for first in text_percent[0].keys():
        dis[0][first] = 1
    for i in range(len(text) - 1):
        word_one = text[i]
        word_two = text[i + 1]
        word_one_percent_dict = text_percent[i]
        word_two_percent_dict = text_percent[i + 1]

        word_one_percent_key = list(word_one_percent_dict.keys())
        word_one_percent_value = list(word_one_percent_dict.values())
        word_two_percent_key = list(word_two_percent_dict.keys())
        word_two_percent_value = list(word_two_percent_dict.values())
        for word_two_per in word_two_percent_key:
            tmp_dis = []
            for word_one_per in word_one_percent_key:
                if word_two_per in trans[word_one_per]:
                    tmp_num = dis[i][word_one_per] * (
                        trans[word_one_per][word_two_per] + 1) / (part[word_one_per] + part_len) * (
                            text_percent[i + 1][word_two_per] / part[word_two_per])
                    tmp_dis.append(tmp_num)
            if word_two_per in trans[word_o...]
```

下面是对分词结果的正确性评估，这里我使用了书中使用的方法，仅计算一个 **Accuracy** 作为正确率评估标准，同时我只对之前分词正确的结果进行词性评估，这样可以避免其它的错误。

```

pythonProject - tongjfenci.py
pythonProject - tongjfenci.py
for tmp in ans_list:
    ans_list_set.add(tuple(tmp))
TP = ans_list_set & xun_lian_set
p = len(TP) / len(xun_lian_list)
r = len(TP) / len(ans_list)
# 我们只对分词正确的结果进行词性正确性评估
if ans_list_set == xun_lian_set:
    A_num += 1
    # 预测来的
    ci_xing_list = ci_xing(ans_sents[i])
    # 正确答案
    ci_xing_ans = fenci_sents[i]
    a = 0
    for j in range(len(ci_xing_list)):
        if ci_xing_list[j] == ci_xing_ans[j]:
            a += 1
    a = a / len(ci_xing_list)
    P += p
    R += r
    if i % 100 == 0:
        print(i / test_sents_num)
# 求一个平均值
P = P / test_sents_num
R = R / test_sents_num
F_1 = 2 * P * R / (P + R)
print("P,R,F1", P, R, F_1)
A = A / A_num
print("A", A)
end_time = time.time()
print(end_time - start_time)

for i in range(test_sents_num):
    if ans_list_set == xun_lian_set:
        for j in range(len(ci_xing_list)):
            if ci_xing_list[j] == ci_xing_ans[j]:
                a += 1
        a = a / len(ci_xing_list)
        P += p
        R += r
        if i % 100 == 0:
            print(i / test_sents_num)
# 求一个平均值
P = P / test_sents_num
R = R / test_sents_num
F_1 = 2 * P * R / (P + R)
print("P,R,F1", P, R, F_1)
A = A / A_num
print("A", A)
end_time = time.time()
print(end_time - start_time)

```

通过检测结果来看，词性标注的正确率大概在 95% 左右，同时共计耗时 315s。还算一个相对不错的效果。

```

pythonProject - tongjfenci.py
pythonProject - tongjfenci.py
a = 0
for j in range(len(ci_xing_list)):
    if ci_xing_list[j] == ci_xing_ans[j]:
        a += 1
a = a / len(ci_xing_list)
P += p
R += r
if i % 100 == 0:
    print(i / test_sents_num)
# 求一个平均值
P = P / test_sents_num
R = R / test_sents_num
F_1 = 2 * P * R / (P + R)
print("P,R,F1", P, R, F_1)
A = A / A_num
print("A", A)
end_time = time.time()
print(end_time - start_time)

for i in range(test_sents_num):
    if ans_list_set == xun_lian_set:
        for j in range(len(ci_xing_list)):
            if ci_xing_list[j] == ci_xing_ans[j]:
                a += 1
        a = a / len(ci_xing_list)
        P += p
        R += r
        if i % 100 == 0:
            print(i / test_sents_num)
# 求一个平均值
P = P / test_sents_num
R = R / test_sents_num
F_1 = 2 * P * R / (P + R)
print("P,R,F1", P, R, F_1)
A = A / A_num
print("A", A)
end_time = time.time()
print(end_time - start_time)

```

Run: tongjfenci
P, R, F1 0.982416582396446 0.974064073299225 0.9782224987136243
A 0.9587190558046981
315.101467159729
Process finished with exit code 0

四、实验遇到的问题与麻烦

- 首先就遇到的难题是处理人民日报那个 `txt`，一开始没有想到很方便的办法把它处理为 `list` 数据，最后通过转换为 `csv` 格式，在直接导入到 `list` 当中确实简化了不少步骤。
- 处理二元语法模型的时候，这个“#始始#”，“#末末#”的处理确实费了不少功夫，总是在这里缺少一些项目，`debug` 了很长时间。
- `Viterbi` 算法中存在的字典中缺少的值如何去补充，不存在的转化概率该如何去补充。

4. 这个也是尚未处理的问题，我做预测的时候仍然采用原人民日报的换行格式，即一行就为一个句子，但这个很明显并不是我们日常意义上所理解的句子，我们通常理解的句子中，是以“。”，“！”，“？”作为句子结尾的。我重新刷洗了数据，并让句子按照这三个标点符号作为结尾来预测数据。

```

pythonProject - pythonProject
pythonProject - Downloads/tongjifenci.py
Project: pythonProject -> Downloads/tongjifenci.py
  main.py
  chuli.py
  chulicing.py
  cixing_tongji.py
  CoreNatureDictionary.csv
  cws.csv
  danju.csv
  fenci.py
  main.py
  my_cws_corpus.csv
  renmin.csv
  renmin.txt
  reminicing.csv
  reminicing.txt
  temp.csv
  test.csv
  tongjifenci.py
  train.lm
  External Libraries
  Scratches and Consoles
  Structure
  Favorites
  Run: tongjifenci
    P, R, F1 0.9824165822396446 0.974064073299225 0.978222498713624
    A 0.9587190558046981
    311.5411648750305
    Process finished with exit code 0
  Run | TODO | Problems | Debug | Python Packages | Python Console | Terminal
  Python Debugger Extension Available: Cython extension speeds up Python debugging // Install How does it work (today:142 下午)
  Event Log
  3487:1 LF UTF-8 4 spaces Python 3.10

```

最终得出的训练结果中，正确率不变，说明我们能很好的将标点符号分开，但是时间却缩小了一些，说明缩短句子确实有助于减少大规模词网的运算量。

五、附录

附录一：基于词典的分词方法：

```

import csv
import time

start_time = time.time()

# 读入字典
def load_dictionary():
    word_list = set()
    csvFile = open("test.csv", "r")
    reader = csv.reader(csvFile)
    for item in reader:
        word_list.add(item[0])
    return word_list

# 完全切分式中文分词
# 如果在词典中则认为是一个词
def fully_segment(text, dic):

```

```

word_list = []
for i in range(len(text)):
    for j in range(i + 1, len(text) + 1):
        word = text[i:j]
        if word in dic:
            word_list.append(word)
return word_list

# 正向最长匹配
# 从当前扫描位置的单字所有可能的结尾，我们找最长的
def forward_segment(text, dic):
    word_list = []
    i = 0
    while i < len(text):
        longest_word = text[i]
        for j in range(i + 1, len(text) + 1):
            word = text[i:j]
            if word in dic:
                if len(word) > len(longest_word):
                    longest_word = word
        word_list.append(longest_word)
        i += len(longest_word)
    return word_list

# 逆向最长匹配
def back_segment(text, dic):
    word_list = []
    i = len(text) - 1
    while i >= 0:
        longest_word = text[i]
        for j in range(0, i):
            word = text[j:i + 1]
            if word in dic:
                if len(word) > len(longest_word):
                    longest_word = word
        word_list.append(longest_word)
        i -= len(longest_word)
    word_list.reverse()
    return word_list

# 统计单字成词的个数

```

```

def count_single_char(word_list: list):
    return sum(1 for word in word_list if len(word) == 1)

# 双向最长匹配，实际上是做了个融合
def bidirectional_segment(text, dic):
    f = forward_segment(text, dic)
    b = back_segment(text, dic)
    # 词数更少更优先
    if len(f) < len(b):
        return f
    elif len(f) > len(b):
        return b
    else:
        # 单字更少更优先
        if count_single_char(f) < count_single_char(b):
            return f
        else: # 都相等时我们更倾向于逆向匹配的
            return b

dic = load_dictionary()
print(len(dic))
print("完全切分：", fully_segment("就读于北京大学", dic))
print("前向切分：", forward_segment("就读于北京大学", dic))
print("前向切分：", forward_segment("研究生物起源", dic))
print("后向切分：", back_segment("研究生物起源", dic))
print("双向切分：", bidirectional_segment("研究生物起源", dic))

test_file = open("renmin.csv", "r")
reader = csv.reader(test_file)
test_sents = []
ans_sents = []
for item in reader:
    test_sent = ""
    ans_sent = []
    for word in item:
        test_sent += word.strip()
        ans_sent.append(word)
    test_sents.append(test_sent)
    ans_sents.append(ans_sent)

# 将分词出来的结果转换为集合中元素

```

```

def zhuan_huan(text: list):
    ans = []
    i = 1
    for word in text:
        ans.append([i, i + len(word) - 1])
        i += len(word)
    return ans

test_sents_num = len(test_sents)
print(test_sents_num)
P = 0
R = 0
for i in range(test_sents_num):
    xun_lian = bidirectional_segment(test_sents[i], dic)
    xun_lian_list = zhuan_huan(xun_lian)
    ans_list = zhuan_huan(ans_sents[i])

    xun_lian_set = set()
    for tmp in xun_lian_list:
        xun_lian_set.add(tuple(tmp))
    ans_list_set = set()
    for tmp in ans_list:
        ans_list_set.add(tuple(tmp))
    TP = ans_list_set & xun_lian_set
    p = len(TP) / len(xun_lian_list)
    r = len(TP) / len(ans_list)
    P += p
    R += r
    if i % 100 == 0:
        print(i / test_sents_num)

# 求一个平均值
P = P / test_sents_num
R = R / test_sents_num
F_1 = 2 * P * R / (P + R)
print("P,R,F1", P, R, F_1)
end_time = time.time()
print(end_time - start_time)

```

附录二：基于统计的分词和词性标注

```

import csv
import time

```

```

start_time = time.time()
fenci_file = open("renmin.csv", "r")
fenci_reader = csv.reader(fenci_file)
fenci_sents = []
for item in fenci_reader:
    sent = []
    for i in item:
        sent.append(i.strip())
    fenci_sents.append(sent)

sum_zero = 0
for sent in fenci_sents:
    if len(sent) == 0:
        sum_zero += 1
    print(sent)
print(sum_zero)

```

```

# 一元语法模型
def sgram(sents: list):
    dic = {}
    dic['#始始#'] = 0
    dic['#末末#'] = 0
    for sent in sents:
        for item in sent:
            if item in dic:
                dic[item] = dic[item] + 1
            else:
                dic[item] = 1
    dic['#始始#'] += 1
    dic['#末末#'] += 1
    return dic

```

```
si_gram = sgram(fenci_sents)
```

```
# print(si_gram)
```

```

# 二元语法模型
def bgram(sents: list):
    dic = {}
    dic["#始始#"] = dict()

```

```

for sent in sents:
    for i in range(0, len(sent) - 1):
        if sent[i] not in dic:
            dic[sent[i]] = dict()
            dic[sent[i]][sent[i + 1]] = 1
        else:
            if sent[i + 1] in dic[sent[i]]:
                dic[sent[i]][sent[i + 1]] += 1
            else:
                dic[sent[i]][sent[i + 1]] = 1

    if sent[0] not in dic["#始#"]:
        dic["#始#"][sent[0]] = 1
    else:
        dic["#始#"][sent[0]] += 1

    if sent[len(sent) - 1] not in dic:
        dic[sent[len(sent) - 1]] = dict()
        dic[sent[len(sent) - 1]]["#末#"] = 1
    elif "#末#" not in dic[sent[len(sent) - 1]]:
        dic[sent[len(sent) - 1]]["#末#"] = 1
    else:
        dic[sent[len(sent) - 1]]["#末#"] += 1
return dic

```

```
bi_gram = bgram(fenci_sents)
```

```
# print(bi_gram)
```

```
# 查看一元词频
```

```
def select_si_gram(gram: dict, word):
    return gram[word]
```

```
def selecr_bi_gram(gram: dict, word_one, word_two):
    return gram[word_one][word_two]
```

```
cixing_file = open("renmincixing.csv", "r")
cixing_reader = csv.reader(cixing_file)
cixing_sents = []
```

```

for item in cixing_reader:
    sent = []
    for i in item:
        sent.append(i.strip())
    cixing_sents.append(sent)

# print(sents)

part = dict()
# 统计所有词性个数
for sent in cixing_sents:
    for word in sent:
        word_part = word.split('/')[-1].split(']')[0].split('!')[0]
        if word_part in part:
            part[word_part] += 1
        else:
            part[word_part] = 1

part_len = len(part)
print(part, "一共多少类:", part_len)
print("总频次", sum(part.values()))

# 或得转移矩阵
trans = dict()
for sent in cixing_sents:
    for i in range(len(sent) - 1):
        one = sent[i].split('/')[-1].split(']')[0].split('!')[0]
        two = sent[i + 1].split('/')[-1].split(']')[0].split('!')[0]
        if one in trans:
            if two in trans[one]:
                trans[one][two] += 1
            else:
                trans[one][two] = 1
        else:
            trans[one] = dict()
            trans[one][two] = 1
print(trans)

# 每个词的词性概率
percent = dict()
for sent in cixing_sents:
    for word in sent:
        word_word = word.split('/')[0].split('{')[0].strip('[')
        word_part = word.split('/')[-1].split(']')[0].split('!')[0]

```

```

    if word_word in percent:
        if word_part in percent[word_word]:
            percent[word_word][word_part] += 1
        else:
            percent[word_word][word_part] = 1
    else:
        percent[word_word] = dict()
        percent[word_word][word_part] = 1

# print(percent)

def fen_ci(text):
    # 生成一元语法词网
    def generate_wordnet(gram, text):
        net = [[] for _ in range(len(text) + 2)]
        for i in range(len(text)):
            for j in range(i + 1, len(text) + 1):
                word = text[i:j]
                if word in gram:
                    net[i + 1].append(word)
        i = 1
        while i < len(net) - 1:
            if len(net[i]) == 0: # 空白行
                j = i + 1

                for j in range(i + 1, len(net) - 1):
                    # 寻找第一个非空行j
                    if len(net[j]):
                        break
                # 填补i, j之间的空白行
                net[i].append(text[i - 1:j - 1])
                i = j
            else:
                i += len(net[i][-1])
        return net

    # 测试一个句子
    si_net = generate_wordnet(si_gram, text)

    # print(si_net)

    def calculate_gram_sum(gram: dict):
        num = 0

```

```

for g in gram.keys():
    num += sum(gram[g].values())
return num

# 计算word_one后出现word_two的概率，带上+1平滑处理
def calculate_weight(gram: dict, word_one, word_two, gram_sum):
    if word_one in gram:
        word_one_all = gram[word_one].values()
        if word_two in gram[word_one]:
            return (gram[word_one][word_two] + 1) / (sum(word_one_all) +
gram_sum)
    else:
        return 1 / (sum(word_one_all) + gram_sum)
    else:
        return 1 / gram_sum

bi_gram_sum = calculate_gram_sum(bi_gram)

# print(bi_gram_sum)

def viterbi(wordnet):
    dis = [dict() for _ in range(len(wordnet))]
    node = [dict() for _ in range(len(wordnet))]
    word_line = [dict() for _ in range(len(wordnet))]
    wordnet[len(wordnet) - 1].append("#未未#")
    # 更新第一行
    for word in wordnet[1]:
        dis[1][word] = calculate_weight(bi_gram, "#始始#", word,
bi_gram_sum)
        node[1][word] = 0
        word_line[1][word] = "#始始#"
    # 遍历每一行wordnet
    for i in range(1, len(wordnet) - 1):
        # 遍历每一行中单词
        for word in wordnet[i]:
            # 更新加上这个单词的距离之后那个位置的所有单词的距离
            for to in wordnet[i + len(word)]:
                if word in dis[i]:
                    if to in dis[i + len(word)]:
                        # 要的是最大的概率
                        if dis[i + len(word)][to] < dis[i][word] * calculate_weight(bi_gram, word, to, bi_gram_sum):
                            dis[i + len(word)][to] = dis[i][word] * calculate_weight(bi_gram, word, to, bi_gram_sum)

```

```

        node[i + len(word)][to] = i
        word_line[i + len(word)][to] = word
    else:
        dis[i + len(word)][to] = dis[i][word] *
calculate_weight(bi_gram, word, to, bi_gram_sum)
        node[i + len(word)][to] = i
        word_line[i + len(word)][to] = word

# 回溯
path = []
f = node[len(node) - 1]["#末末#"]
fword = word_line[len(word_line) - 1]["#末末#"]
path.append(fword)
while f:
    tmpword = fword
    fword = word_line[f][tmpword]
    f = node[f][tmpword]
    path.append(fword)
path = path[:-1]
path.reverse()
return dis, node, word_line, path

(dis, _, _, path) = viterbi(si_net)
# print(dis)
# print(path)
return path

```

```

def ci_xing(text):
text_percent = []

# 这里我们假设是所有单词都已经人民日报语料库了
for word in text:
    word_percent = percent[word]
    text_percent.append(word_percent)

# print(text_percent)

# 下面我们来使用Viterbi算法计算出最佳的组成
dis = [dict() for _ in range(len(text))]
node = [dict() for _ in range(len(text))]
for first in text_percent[0].keys():
    dis[0][first] = 1
for i in range(len(text) - 1):

```

```

word_one = text[i]
word_two = text[i + 1]
word_one_percent_dict = text_percent[i]
word_two_percent_dict = text_percent[i + 1]

word_one_percent_key = list(word_one_percent_dict.keys())
word_one_percent_value = list(word_one_percent_dict.values())
word_two_percent_key = list(word_two_percent_dict.keys())
word_two_percent_value = list(word_two_percent_dict.values())
for word_two_per in word_two_percent_key:
    tmp_dis = []
    for word_one_per in word_one_percent_key:
        if word_two_per in trans[word_one_per]:
            tmp_num = dis[i][word_one_per] * (
                trans[word_one_per][word_two_per] + 1) /
            (part[word_one_per] + part_len)) * (
                text_percent[i + 1][word_two_per] /
            part[word_two_per])
            tmp_dis.append(tmp_num)
        else:
            tmp_num = dis[i][word_one_per] * (1 / (part[word_one_per] +
            part_len)) * (
                text_percent[i + 1][word_two_per] /
            part[word_two_per])
            tmp_dis.append(tmp_num)

    max_tmp_dis = max(tmp_dis)
    max_tmp_dis_loc = tmp_dis.index(max_tmp_dis)
    dis[i + 1][word_two_per] = max_tmp_dis
    node[i + 1][word_two_per] = word_one_percent_key[max_tmp_dis_loc]
# print(dis, node)

# 根据node来倒找答案
path = []
f_value = list(dis[len(dis) - 1].values())
f_key = list(dis[len(dis) - 1].keys())
f = f_key[f_value.index(max(f_value))]

path.append(f)
for i in range(len(dis) - 1, 0, -1):
    f = node[i][f]
    path.append(f)
path.reverse()
return path

```

```

# 对所有训练集进行测试
test_file = open("renmincixing.csv", "r")
reader = csv.reader(test_file)
test_sents = []
ans_sents = []
fenci_sents = []
for item in reader:
    test_sent = ""
    ans_sent = []
    fenci_sent = []
    for word in item:
        word_word = word.split('/')[0].split('{')[0].strip('[')
        word_part = word.split('/')[-1].split(']')[0].split('!')[0]
        if word_word == '。' and word_word == '！' and word_word == '？':
            test_sent += word_word.strip()
            ans_sent.append(word_word)
            fenci_sent.append(word_part)
            break
        else:
            test_sent += word_word.strip()
            ans_sent.append(word_word)
            fenci_sent.append(word_part)
    test_sents.append(test_sent)
    ans_sents.append(ans_sent)
    fenci_sents.append(fenci_sent)

```

```

# 将分词出来的结果转换为集合中元素
def zhuan_huan(text: list):
    ans = []
    i = 1
    for word in text:
        ans.append([i, i + len(word) - 1])
        i += len(word)
    return ans

```

```

test_sents_num = len(test_sents)
print(test_sents_num)
P = 0
R = 0
A = 0

```

```

A_num = 0
for i in range(test_sents_num):
    xun_lian = fen_ci(test_sents[i])
    xun_lian_list = zhuan_huan(xun_lian)
    ans_list = zhuan_huan(ans_sents[i])

    xun_lian_set = set()
    for tmp in xun_lian_list:
        xun_lian_set.add(tuple(tmp))

    ans_list_set = set()
    for tmp in ans_list:
        ans_list_set.add(tuple(tmp))

    TP = ans_list_set & xun_lian_set
    p = len(TP) / len(xun_lian_list)
    r = len(TP) / len(ans_list)
    # 我们只对分词正确的结果进行词性正确性评估
    if ans_list_set == xun_lian_set:
        A_num += 1
        # 预测来的
        ci_xing_list = ci_xing(ans_sents[i])
        # 正确答案
        ci_xing_ans = fenci_sents[i]
        a = 0
        for j in range(len(ci_xing_list)):
            if ci_xing_list[j] == ci_xing_ans[j]:
                a += 1

        a = a / len(ci_xing_list)
        A += a
    P += p
    R += r
    if i % 100 == 0:
        print(i / test_sents_num)

# 求一个平均值
P = P / test_sents_num
R = R / test_sents_num
F_1 = 2 * P * R / (P + R)
print("P,R,F1", P, R, F_1)
A = A / A_num
print("A", A)
end_time = time.time()

```

```
print(end_time - start_time)
```

附录三：处理人民日报分词数据：

```
import csv

mat = []

with open("renmin.txt", "r") as f: # 打开文件
    for line in f:
        mat.append([l for l in line.split()])

print(mat)

with open('renmin.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    for row in mat:
        if len(row) != 0:
            writer.writerow(row)
```

附录四：处理人民日报词性数据：

```
import csv

mat = []

with open("renmincixing.txt", "r") as f: # 打开文件
    for line in f:
        line = line[22:]
        print(line)
        mat.append([l for l in line.split()])

print(mat)

with open('renmincixing.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    for row in mat:
        if len(row) != 0:
            writer.writerow(row)
```