《自然语言处理》实验报告

年级、专业、班级	2019 级计算机科学与技术（卓越）02 班	姓名	李燕琴
实验题目	基于 HMM 的拼音转汉字程序		
实验时间	2021/10/6	实验地点	虎溪校区 DS1421
实验成绩		实验性质	☐验证性 ☐设计性 ☐综合性

教师评价：

☐算法/实验过程正确； ☐源程序/实验内容提交 ☐程序结构/实验步骤合理；

☐实验结果正确； ☐语法、语义正确； ☐报告规范；

其他：

评价教师签名：

一、实验目的

理解、掌握隐马尔可夫模型，N 元语法等自然语言处理的基本思想、算法，并将其应用于从汉语拼音到汉字的自动转换过程。

假定：拼音串中已经用空格进行了分隔，如“wo ai wo jia”

二、实验项目内容

（1）对训练语料及相关资源进行预处理；

（2）通过学习算法，训练 HMM 模型；

（3）利用 HMM 模型和维比特算法，实现从任意拼音到汉字的自动转换。

（4）利用给定测试集，评价上述程序的转换准确率。

三、实验过程或算法（源程序）

1、 文本预处理

如图 1 所示，需要解决两个问题。

一是提取语句；本实验根据’_!_’分割，得到倒数第一、二段作为语料语句。

二是无拼音字符处理；给定训练语料中存在特殊字符、英文大小写字母、数字。这三类字符都没有对应的中文拼音，故在文本预处理时，直接去除英文大小写字母和数字，特殊字符替换为中文逗号，并在标点符号处进行断句。代码实现如图 2 所示。

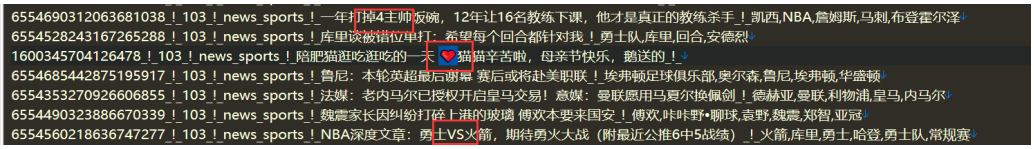

图 1 文本预览

```
# 非中文字符、数字、大小写字母等其他字符，替换为中文逗号
replacer = "([^\u4e00-\u9fa5\u0030-\u0039\u0041-\u005a\u0061-\u007a])"
# 数字、大小写字母直接去除（不替换其他元素）
remover = '[a-zA-Z0-9]'
# 分割中文标点符号
splitter = r',|\.|/|;|\'|`|\[|\]|<|>|\?|？
|:|"|\{|\}|\~|!|@|#|\$|%|\^|&|\(|\)|-|=|\_|\+|，|。|、|：|‘|’|【|】|·|！| |…|
（|）|：|《|》|—'

def text_preprocess(sentence):
    s1 = re.sub(replacer, ", ", sentence)
    s2 = re.sub(remover, "", s1)
    s_list = re.split(splitter, s2)
    s_list = [s for s in s_list if len(s) > 0]
    return s_list
```

2、 HMM 模型构建

经典 HMM 模型为 HMM(pi,A,B)。其中 pi 为初始概率，A 为状态转移概率，B 为发射概率。

①初始概率：

根据确定的初始状态字出现的频次，占所有初始状态字的总频次的概率。

②状态转移概率：

使用 curr 代表当前字，last 代表前一个字，其中 last 到 curr 的状态转移概率为 last_curr 出现的次数/last 中文字出现的次数。

③发射概率：

计算一个字到指定拼音的概率，即该中文字且对上该拼音出现的次数/中文字出现的总次数。

主要逻辑代码如下：

```
def get_init_log(self, pinyin):
    ''''' 根据 pinyin 获取初始状态词，并根据频率计算其出现的概率 log 值'''
    init_log = {}
    init_word_set = self.get_curr_search_set(pinyin)
    n_total = 0

    for word in init_word_set:
```

```
            n_total += self.word_counter[word]
        for word in init_word_set:
            init_log[word] = np.log10(self.word_counter[word] / n_total)
        return init_word_set, init_log

    def get_emission_log(self, word, pinyin):
        ''''' 计算 word_pinyin 的发射概率 '''
        n_total = sum(self.word_pinyin[word].values())
        emission_prob = self.word_pinyin[word][pinyin] / n_total
        return np.log10(max(emission_prob, self.min_prob))

    def get_transition_log(self, last, curr):
        ''''' 计算 last_curr 的状态转移概率 '''
        if self.word_word.get(last) == None:
            return np.log(self.min_prob)
        n_total = sum(self.word_word[last].values())
        transition_prob = self.word_word[last][curr] / n_total
        return np.log(max(transition_prob, self.min_prob))
```

3、 Viterbi 算法

根据 viterbi 算法原理，主要分为四个步骤。

①状态初始化，其中状态概率根据初始状态概率乘上发射到拼音的概率。

$$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N$$

②状态前推计算，需要记录最大可能状态转移字，及其发射到该拼音的联合概率。

$$\delta_t(j) = \max_{1 \le i \le N}[\delta_{t-1}(i) \cdot a_{ij}] \cdot b_j(O_t), \quad 2 \le t \le T, \quad 1 \le j \le N$$

$$\psi_t(j) = \arg\max_{1 \le i \le N}[\delta_{t-1}(i) \cdot a_{ij}] \cdot b_j(O_t), \ 2 \le t \le T, \ 1 \le i \le N$$

③记录最大概率的隐状态序列

$$\hat{Q}_T = \arg\max_{1 \le i \le N}[\delta_T(i)], \qquad \hat{p}(\hat{Q}_T) = \max_{1 \le i \le N} \delta_T(i)$$

④结果回溯

$$\hat{q}_t = \psi_{t+1}(\hat{q}_{t+1}), \quad t = T-1, \ T-2, \ \cdots, \ 1$$

Viterbi 算法原理实现代码如下：

```
def pinyin2hanzi(model, pinyin_list):
    '''''
    viterbi 算法，计算给定显状态 pinyin 序列，求解最优的隐状态汉字序列
    :param model: hmm 模型
    :param pinyin_list: 显状态 pinyin 序列
    :return:
```

```
        - pred_sent: 预测的语句
        - max_prob: 最大预测概率
    '''
    delta = [{} for i in range(len(pinyin_list))]

    # 状态初始化
    i = 0
    pinyin = pinyin_list[i]
    init_word_set, init_log = model.get_init_log(pinyin)
    for word in init_word_set:
        delta[0][word] = init_log[word] + model.get_emission_log(word,pinyin)

    # 状态前推
    max_last = [{} for i in range(len(pinyin_list))]
    for i in range(1,len(pinyin_list)):
        pinyin = pinyin_list[i]
        curr_candidate_set = model.get_curr_candidate_set(pinyin, delta[i - 1])
        for curr in curr_candidate_set:
            max_tran_log = -float('inf')
            max_tran_last = None
            for last in delta[i-1].keys():
                tran_value = delta[i-1][last]+model.get_transition_log(last,curr)
                if tran_value >= max_tran_log:
                    max_tran_log = tran_value
                    max_tran_last = last
            delta[i][curr] = max_tran_log + model.get_emission_log(curr,pinyin)
            max_last[i][curr] = max_tran_last

    # 获取最后的最大概率对应的状态字
    pred_sent = ['*' for i in range(len(pinyin_list))]
    last_i = len(pinyin_list)-1
    pred_sent[last_i],max_prob = max(zip(delta[last_i].keys(),delta[last_i].values()),key=lambda x:x[1])

    # 后向递归路径
    for i in range(last_i,0,-1):
        pred_sent[i-1] = max_last[i][pred_sent[i]]
    return pred_sent,max_prob
```

其中，根据语料库统计，大概有 6400 个汉字，如果全部作为状态字，算法复杂度将高达 1e6 以上，故在获取状态字候选集上，首选拼音对应的汉字。若为未注册拼音，则获取最大概率的上一个状态字的 curr 作为当前状态字候选集。若未给定上一个状态，则将全部汉字视为状态字候选集，代码实现如下：

```
1.  def get_curr_candidate_set(self, pinyin, last_delta=None):
2.      ''''' 确定搜索空间 '''
3.      curr_search_set = set()
4.
5.      # 获取 pinyin 对应的字列表
6.      if self.pinyin_word.get(pinyin) != None:
7.          curr_search_set.update(self.pinyin_word[pinyin])
8.      else:
9.          # 当该 pinyin 不存在于语料库时
10.         if last_delta == None:
11.             # 方法一：所有词语作为搜索空间
12.             curr_search_set.update(self.word_counter.keys())
13.         else:
14.             # 方法二：获取上一个最大出现概率的词，对应的 word_word 的所有词
15.             k = -1
16.             lastk_word = sorted(zip(last_delta.keys(), last_delta.values(
    )), key=lambda x: x[1])[k:]
17.             for last, _ in lastk_word:
18.                 if self.word_word.get(last) != None:
19.                     curr_search_set.update(self.word_word[last].keys())
20.
21.     return curr_search_set
```

四、实验结果及分析

根据给定的测试集，运行 test.py，得到的最终测试结果见报告结尾。

先分析一些准确率比较低的语句。

1、如下图，该句是因为语料库中，“他钢”出现的次数为 0

```
拼    音：  Ta gang qin tan de bu cuo
真实句子：  他钢琴弹得不错
预测结果：  塔港亲瘫的不错
max_prob=-41.098,acc=0.286
用时：0.015s
```

```
In  [60]:  1  model.word_word['他']['钢']
Out[60]:  0
```

2、这一句是因为“我带”的频次比“我戴”的更高，句首预测的值，也会影响到后续预测的值。

拼　音： Wo dai zhe yi fu hei kuang yan jing
真实句子： 我戴着一副黑框眼镜
预测结果： 我带着遗腹黑框演精
max_prob=-46.584,acc=0.444
用时：0.088s

```
In [61]:
1 print(model.word_word['我']['带'])
2 print(model.word_word['我']['戴'])

59
2
```

```
1. 加载参数，开始初始化 hmm...
2. 用时 0.06952238877614339 min
3.
4. 拼　音： jin tian wan shang you hao kan de dian ying
5. 真实句子： 今天晚上有好看的电影
6. 预测结果： 今天晚上有好看的电影
7. max_prob=-35.469,acc=1.000
8. 用时：0.024s
9.
10. 拼　音： bei jing ao yun hui kai mu shi fei chang jing cai
11. 真实句子： 北京奥运会开幕式非常精彩
12. 预测结果： 北京奥运会开幕是非常精彩
13. max_prob=-43.847,acc=0.917
14. 用时：0.063s
15.
16. 拼　音： quan guo ren min dai biao da hui zai bei jing ren min da hui tan
    g long zhong zhao kai
17. 真实句子： 全国人民代表大会在北京人民大会堂隆重召开
18. 预测结果： 全国人民代表大会在北京人民大烩汤隆重召开
19. max_prob=-80.803,acc=0.900
20. 用时：0.049s
21.
22. 拼　音： jin yong de wu xia xiao shuo fei chang jing cai
23. 真实句子： 金庸的武侠小说非常精彩
24. 预测结果： 金庸的武侠小说非常精彩
25. max_prob=-44.388,acc=1.000
26. 用时：0.035s
27.
28. 拼　音： ni de shi jie hui bian de geng jing cai
29. 真实句子： 你的世界会变得更精彩
30. 预测结果： 你的世界会变得更精彩
31. max_prob=-35.126,acc=1.000
```

```
用时：0.055s

拼　音：  shen du xue xi ji shu tui dong le ren gong zhi neng de fa zhan
真实句子：  深度学习技术推动了人工智能的发展
预测结果：  深度学习技术推动了人工智能的发展
max_prob=-56.259,acc=1.000
用时：0.109s

拼　音：  zai tian an men guang chang ju xing le long zhong de yue bing sh
i
真实句子：  在天安门广场举行了隆重的阅兵式
预测结果：  在天安门广场举行了隆重的阅兵式
max_prob=-65.977,acc=1.000
用时：0.037s

拼　音：  luo ji dai shu you xie chang yong de ji ben gong shi
真实句子：  逻辑代数有些常用的基本公式
预测结果：  逻辑大数有些常用的记本功是
max_prob=-59.386,acc=0.692
用时：0.116s

拼　音：  qing da jia xuan ze ni jue de ke yi de shi jian
真实句子：  请大家选择你觉得可以的时间
预测结果：  请大家选择你觉得可以的时间
max_prob=-46.991,acc=1.000
用时：0.094s

拼　音：  ju you liang hao de gou tong neng li he jiao liu neng li
真实句子：  具有良好的沟通能力和交流能力
预测结果：  具有量好的沟通能力和交流能力
max_prob=-67.454,acc=0.929
用时：0.063s

拼　音：  ren zai yue du shi shi cong zuo dao you zhu zi du ru de
真实句子：  人在阅读时是从左到右逐字读入的
预测结果：  人在阅读是失聪做到有竹子都入的
max_prob=-75.806,acc=0.467
用时：0.104s

拼　音：  jian shao ke cheng de ke shi shi shi fen ke xue de
真实句子：  减少课程的课时是十分科学的
预测结果：  减少课程的可是世十分科学的
max_prob=-56.403,acc=0.769
用时：0.141s
```

```
75.
76. 拼    音：  Wo dai zhe yi fu hei kuang yan jing
77. 真实句子：  我戴着一副黑框眼镜
78. 预测结果：  我带着遗腹黑框演精
79. max_prob=-46.584,acc=0.444
80. 用时：0.088s
81.
82. 拼    音：  Ta gang qin tan de bu cuo
83. 真实句子：  他钢琴弹得不错
84. 预测结果：  塔港亲瘫的不错
85. max_prob=-41.098,acc=0.286
86. 用时：0.015s
87.
88. 拼    音：  Xiao peng you men dou xi huan qu jiao you
89. 真实句子：  小朋友们都喜欢去郊游
90. 预测结果：  小朋友们都喜欢去郊游
91. max_prob=-34.240,acc=1.000
92. 用时：0.052s
93.
94. 拼    音：  Zhe ge wan ju hen you qu
95. 真实句子：  这个玩具很有趣
96. 预测结果：  这个弯矩很有趣
97. max_prob=-29.768,acc=0.714
98. 用时：0.032s
99.
100.    拼    音：  Ta hen sheng qi
101.    真实句子：  他很生气
102.    预测结果：  她很生气
103.    max_prob=-18.154,acc=0.750
104.    用时：0.014s
105.
106.    拼    音：  jin tian tian qi hen hao
107.    真实句子：  今天天气很好
108.    预测结果：  今天天气很好
109.    max_prob=-21.649,acc=1.000
110.    用时：0.014s
111.
112.    拼    音：  wo men qu san bu ba
113.    真实句子：  我们去散步吧
114.    预测结果：  我们去三部霸
115.    max_prob=-25.068,acc=0.500
116.    用时：0.009s
117.
118.    拼    音：  da jia ke neng bu zhi dao
```

```
真实句子： 大家可能不知道
预测结果： 大家可能不知道
max_prob=-19.476,acc=1.000
用时：0.036s

拼　音： wo men su she de deng huai le
真实句子： 我们宿舍的灯坏了
预测结果： 我们俗社的灯坏了
max_prob=-38.303,acc=0.750
用时：0.008s


Process finished with exit code 0
```