# 拼音输入法实验报告

2017011620 计 73 李家昊

2019 年 4 月 19 日

## 1 算法的思路及实现

### 1.1 二元语言模型

#### 1.1.1 提取模型

二元模型是一阶 Hidden Markov Model (HMM)，该模型的随机过程中，每个状态 $S_t$ 仅与它前一个状态 $S_{t-1}$ 相关。

拼音输入法的最终目标为，给定一串拼音 $O: p_1p_2\cdots p_n$ （观测序列）求出概率最大的对应中文字串 $S: w_1\cdots w_n$ （隐状态序列）。由贝叶斯公式，得：

$$S = \text{argmax } P(S|O) = \text{argmax } \frac{P(S)P(O|S)}{P(O)} \tag{1}$$

其中 $P(O)$ 为常量，$P(O|S)$ 用识别信度代替，即多音字读此音的概率，且由 Markov 假设，有

$$P(S) = \prod_{i=1}^{n} P(w_i|w_1\cdots w_{i-1}) = \prod_{i=1}^{n} P(w_i|w_{i-1}) = \prod_{i=1}^{n} \frac{P(w_{i-1}w_i)}{P(w_{i-1})} \tag{2}$$

由此，我们可以通过统计语料库中所有连续二字 $w_{i-1}w_i$ 出现的频率，以及单字 $w_{i-1}$ 出现的频率，从而近似计算出概率 $P(w_i|w_{i-1})$ ，存储成二元语言模型。

实际操作中，我们将语料的所有标点符号、数字、字母全部替换成空格，统计所有单字出现的频率，所有连续二字出现的频率，将字作为 key，频率作为 value，存为 dict，最后保存为 json 文件，得到二元语言模型。

#### 1.1.2 处理询问

对于 HMM，我们可以采用 Viterbi 算法。

设隐状态 $w_i$ 有 $m_i$ 种可能取值 $w_{i,1}, w_{i,2}, \cdots, w_{i,m_i}$ ，假设在 $w_1$ 和 $w_n$ 之间，我们找到了一条最优转移路径 $L = (w_{1,r_1}, w_{2,r_2}, \cdots, w_{n,r_n})$ ，则对 $\forall k: 1 \le k \le$

$n$，在状态 $w_1, w_k$ 之间，以及状态 $w_k, w_n$ 之间，$L$ 也必定为最优转移路径。由此得状态转移方程：

$$P(w_1 \cdots w_{k,j}) = \max_{1 \le i \le m_{k-1}} P(w_1 \cdots w_{k-1,i}) \cdot P(w_{k,j}|w_{k-1,i}) \tag{3}$$

对其动态规划，对 $k = 1, 2, 3, \cdots, n$，对于每个固定的 $k$，令 $j = 1, 2, 3, \cdots, m_k$，由公式 (3) 计算出 $P(w_1 \cdots w_{k,j})$ 的值。记隐状态平均取值可能数为 $m$ ，则时间复杂度为 $O(n \cdot m^2)$ 。

实际操作中，考虑到某些二元组合 $w_{i-1}w_i$ 在模型中可能从未出现过，为避免乘 0 的情况，考虑平滑操作：

$$P^*(w_i|w_{i-1}) = (1 - \lambda)P(w_i|w_{i-1}) + \lambda P(w_i) \tag{4}$$

其中 $\lambda$ 为超参数，具体调参实验见下文。需要注意的是，这里的$P(w_i)$ 表示在该拼音下，该字出现的概率，而非普遍意义上的出现概率。上式表明，当 $w_{i-1}w_i$ 从未在语料中出现过时，二元概率将退化为一元概率，此时将通过 $\lambda$ 降低其权重。

## 1.2 三元模型

三元模型中，每个字出现的概率与前两个字相关，因此，

$$P(S) = \prod_{i=1}^{n} P(w_i|w_1 \cdots w_{i-1}) = \prod_{i=1}^{n} P(w_i|w_{i-2}w_{i-1}) = \prod_{i=1}^{n} \frac{P(w_{i-2}w_{i-1}w_i)}{P(w_{i-2}w_{i-1})} \tag{5}$$

得动态转移方程

$$P(w_1 \cdots w_{k,j}) = \max_{\substack{1 \le r \le m_{k-2} \\ 1 \le s \le m_{k-1}}} P(w_1 \cdots w_{k-2,r}w_{k-1,s}) \cdot P(w_{k,j}|w_{k-2,r}w_{k-1,s}) \tag{6}$$

动态规划时间复杂度为 $O(n \cdot m^3)$ 。

同样采取平滑操作，令

$$P^*(w_i|w_{i-2}w_{i-1}) = (1 - \mu)P(w_i|w_{i-2}w_{i-1}) + \mu P^*(w_i|w_{i-1}) \tag{7}$$

其中 $\mu$ 为超参数。上式表明，当 $w_{i-2}w_{i-1}w_i$ 从未在语料中出现过时，三元概率将退化为二元概率，并根据公式 (4) 计算出概率值，此时将通过 $\mu$ 降低其权重。

考虑到模型规模，同时为了排除出现次数较低的无意义的三元组，实际操作中，三元模型只收录出现频率最高的前 5,000,000 个三元组，最终模型大小约为 200M。

### 1.3 多音字优化

处理多音字时，$P(O|S)$ 将不能被视为常量，应该用该字读该音的概率来代替。

实验过程中，我通过 pypinyin API 根据上下文对语料文字注音，我们统计出语料库中每个字读每个音的频率，将其视为概率，加权到上述语言模型计算的每个 $P(S)$ 上，达到识别多音字的效果。

## 2 实验效果展示

### 2.1 二元模型

#### 2.1.1 好的例子

- gao li lv yi jing zu ai le jing ji de fa zhan

  高利率已经阻碍了经济的发展

- zhong guo fei fan de ji shu shi li rang mei guo de zi you shi chang xin tu men luan le zhen jiao

  中国非凡的技术实力让美国的自由市场信徒们乱了阵脚

- bai du yi jing zai shen du xue xi shang tou ru ju zi

  百度已经在深度学习上投入巨资

- wan shan you zhi pu tong gao zhong zhao sheng zhi biao fen pei dao chu zhong xue xiao zheng ce

  完善优质普通高中招生指标分配到初中学校政策

- wo zui xi huan kan gong qi jun lao xian sheng de dong man

  我最喜欢看宫崎骏老先生的动漫

- wo ba duo yu de mao mai gei dang di de chong wu shang dian

  我把多余的猫卖给当地的宠物商店

- wo da suan jin tian xia wu gei ni xie yi feng xin

  我打算今天下午给你写一封信

### 2.1.2 不好的例子

- mei guo de jing ji xue jia men zai jin rong he qi ye jing ji xue ling yu zuo chu le zhong yao de gong xian

  美国的经济学家们在金融**合**企业经济学领域做出了重要的贡献

  错误原因：“融合”出现频率较高，因此“he”不会被视为单独的介词“和”，而是与前一个字合成一个词“融合”。

- wo men xi wang jian li yi ge mei you bo xue de she hui

  我们希望建立一个没有**博学**的社会

  错误原因：语料库中“博学”出现的频率比“剥削”更高。

- ke xue jia men xi wang zhao dao gai ji yin zai ran se ti zhong de wei zhi

  科学家们希望找到该基因在染色体中的**遗址**

  错误原因：“遗”为多音字，其中一个读音为“wei”，因此“wei zhi”被识别成“遗址”。

- qing hua da xue zi dong hua xi

  清华大学自动**画**系

  错误原因：“动画”出现频率高，因此“动画”被选中。二元模型未考虑前面两个字“自动”，这是它的一个缺陷。

- jin tian hui jia bi jiao wan

  今天回家比较**完**

  错误原因：这个例子在二元和三元模型都识别错误，反映了新闻语料中的口语词组较少的缺陷。

## 2.2 三元模型

### 2.2.1 好的例子

- mei guo de jing ji xue jia men zai jin rong he qi ye jing ji xue ling yu zuo chu le zhong yao de gong xian

  美国的经济学家们在金融和企业经济学领域做出了重要的贡献

- qing hua da xue zi dong hua xi

  清华大学自动化系

可见，三元模型解决了二元模型的部分缺陷，能识别“自动化系”，能识别介词“和”。

#### 2.2.2 不好的例子

- liang hui zai bei jing zhao kai

  量会在北京召开

  错误原因：在二元模型中正确的句子，放到三元模型就识别错误，说明两个模型都只能做到局部最优。

### 2.3 多音字优化

#### 2.3.1 好的例子

- ke xue jia men xi wang zhao dao gai ji yin zai ran se ti zhong de wei zhi

  科学家们希望找到该基因在染色体中的位置

  可以看出，经过多音字优化后，“wei zhi”被识别成“位置”，而不再被识别成“遗址”。

#### 2.3.2 不好的例子

- wo men ying dang xiao dui cuo zhe

  我们应当**校对**挫折

  错误原因：多音字模型中“校”字读“xiao”的频率过大，导致识别错误，若要进一步优化，则需建立多音字的二元模型或三元模型。

## 3 调整参数分析性能

在新浪网上选了几篇不同板块的新闻作为测试集，共 7102 字，网址如下：
https://news.sina.com.cn/c/2019-03-26/doc-ihsxncvh5560635.shtml
http://edu.sina.com.cn/l/2019-03-26/doc-ihtxyzsm0437835.shtml
https://cul.news.sina.com.cn/stickynews/2019-03-19/doc-ihrfqzkc5038736.shtml
https://finance.sina.com.cn/roll/2019-03-22/doc-ihsxncvh4527842.shtml

### 3.1 二元模型

调整参数 $\lambda$，在上述测试集上测试字准确率，结果如下。

| $\lambda$ | 1 | 1e-1 | 1e-2 | 1e-3 | 1e-4 | 1e-5 | 1e-6 | 1e-7 |
|---|---|---|---|---|---|---|---|---|
| Accuracy(%) | 39.06 | 69.88 | 85.12 | 87.81 | 88.21 | 88.30 | 88.30 | 88.30 |

Table 1: 二元模型参数与字准确率对照表

由表 1可见，字准确率随着 $\lambda$ 降低而增加，$\lambda$ 降低到 $10^{-5}$ 时，字准确率上升到最大，当 $\lambda \leq 10^{-5}$ 时，字准确率稳定在 88.30%。

### 3.2 三元模型

设置 $\lambda = 10^{-5}$ ，调节 $\mu$ 的取值，在上述测试集上测试字准确率。

| $\mu$ | 1 | 1e-1 | 1e-2 | 1e-3 | 1e-4 | 1e-5 | 1e-6 | 1e-7 |
|---|---|---|---|---|---|---|---|---|
| Accuracy(%) | 93.62 | 94.33 | 94.21 | 94.00 | 93.72 | 93.68 | 93.64 | 93.64 |

Table 2: 三元模型参数与字准确率对照表

由表 2 可见，字准确率随 $\mu$ 的变化几乎稳定在 94% 附近，波动不大。

### 3.3 多音字优化

取$\lambda = 10^{-5}$，取$\mu = 10^{-1}$，在上述测试集上使用带多音字优化的三元模型，得到的字准确率为 94.49%，比纯三元模型略有提升，但提升幅度较小。

### 3.4 补充数据集

为进一步提高准确率，我将大一小学期时爬取的 30,000 篇新闻（大小约 85M）作为补充数据集对其进行训练，在上述测试集上测试，得到的字准确率为 94.36%，比纯三元模型略有提升，但提升幅度较小。

## 4 改进方案

1. 对于多音字的处理仍存在优化空间，考虑到多音字读某一读音的概率与上下文相关，我们可以构建多音字的二元模型，计算出当前面一个字出现时，该字读该音的概率。

2. 本次实验开发了基于字的统计模型，在改进过程中，我们可以利用 jieba 分词等分词工具，开发基于词的统计模型，将能组词的序列的概率权重提高，将不能组词的序列的概率权重降低，则准确率将有一定程度的提升。

3. 也可以进一步利用深度神经网络改进，如使用 seq2seq 网络处理，将拼音序列端到端的转换为汉字序列。

# 5 总结收获

1. 通过本次实验，我第一次接触了 NLP，并掌握了 Markov 过程以及 Viterbi 动态规划算法，极大地锻炼了我的编程能力以及数据处理能力，使我对人工智能有了更深入的理解。

2. 写了一个通用的 Viterbi 算法模块，二元模型和三元模型都能直接调用此模块，虽然运行速度没有过程化编程快，但带来了更优的代码封装性。

3. 如何去掉语料中的标点符号这个问题困扰了我很久，一开始我将所有标点符号列出来，利用正则表达式将其替换为空格，但是无论我怎么列，语料里总有一些符号是我没有列出的，导致多音字的读音频率提取经常性失败。后来我采用逆向思维，通过查资料得到汉字的编码范围，然后将不在范围内的字符全部替换为空格，成功解决了这个问题。