拼音输入法
实验报告

计 72 谢兴宇 2017011326

April 2019

目录

1 算法思路

我采用的是基于字的二元模型，算法的基本框架是用 Viterbi 算法求解 HMM 模型。

1.1 基本模型

1.1.1 公式推导

设 O 为输入拼音串，S 为输出汉字串，$P(S|O)$ 是这串汉字在这串拼音下的概率。

使用 Bayes 公式来计算概率：

$$P(S|O) = P(S)P(O|S)/P(O) \tag{1}$$

其中，$P(O) = 1$，$P(O|S)$ 我们也假设为 1（不考虑多音字的存在），而

$$P(S) = \prod_{i=1}^{n} P(w_i|\prod_{j=1}^{i-1} w_j)$$

引入 n-gram 基本假设：

- 齐次性假设：当前词的条件概率与当前词在词序列中的位置无关。
- 有限历史假设：当前词的条件概率只与前面的 n-1 个词相关。

当 n=2 时，便得到：

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

代入式 1：

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

问题转化为求 S，使得 $P(S|O)$ 最大。

1.1.2 统计建模

为了估计 $P(w_i)$ 和 $P(w_i|w_j)$ 的值，我们需要在训练语料库中做统计。

以 c_i 表示字 w_i 在语料库中出现的总次数，c_{ij} 表示字 w_i 和字 w_j 接连在语料库中出现的总次数，进而可以得到以下两个统计量：

$$unigram_i = \frac{c_i}{\sum c_j} \tag{3}$$

$$bigram_{ij} = \frac{c_{ij}}{c_i} \tag{4}$$

我们便可以用 $unigram_i$ 来估计 $P(w_i)$，以 $bigram_{ij}$ 来估计 $P(w_i|w_j)$.

1.2 Viterbi 算法

现在考虑如何对于一个输入的拼音串 O，求出 $P(S|O)$ 最大的 S。

借助动态规划的思想，考虑对于前 i 个拼音而言，最后一个汉字是 w_j 有最大概率 $P(S|O)$ 的汉字串 S，其最大的概率是 $f(i,j)$，S 中第 $j-1$ 个汉字是 $path(i,j)$，则：

$$f(i,j) = \max_k\{f(i-1,k)P(w_j|w_k)\} \tag{5}$$

$$path(i,j) = \arg\max_k\{f(i-1,k)P(w_j|w_k)\} \tag{6}$$

这样，通过 $path(i,j)$ 便可以得到 $P(S|O)$ 最大的 S。

1.3 Laplace 平滑

因为语料库过小，$P(w_i)$ 和 $P(w_i|w_j)$ 可能为 0，这不仅与实际情况不符，且会导致若 S 中出现了 w_i 或 w_jw_i，则必有 $P(S|O)=0$，所以我们需要引入 Laplace 平滑（Laplace Smoothing）。

若 $P(w_i)=0$，将其重设为 e^{-10}

若 $P(w_jw_i)=0$，将其重设为 $\alpha P(w_jw_i)+(1-\alpha)P(w_i)$

1.4 测试集构建

在新华网爬取了大约 4000 条新闻，抽取其中长度不小于 6 的连续一段汉字来做测试，使用 pypinyin 对其标注。

1.5 改进模型

1.5.1 多音字问题

为了处理多音字问题，我将基本字元换为 (pinyin, 汉字)。这样，多音字的不同发音便被视为不同的基本字元。

通过 pypinyin 对训练语料库进行拼音标注。

1.5.2 句首概率

对于同样的拼音，置于句中和句首对应的汉字概率分布是不同的，所以可以对句首进行特殊考虑。

我们在每一句话之前添加一个特殊字符 # 作为句首，这样式 2变为：

$$P(S|O) = \prod_{i=1}^{n} P(w_i|w_{i-1}), w_0 =' \#' \tag{7}$$

在训练语料库中取出长度不小于 6 的连续一段汉字来训练。

2 实验效果

2.1 正面例子

- gang ao tai di qu yuan zhu wen chuan di zhen zai qu hui gu

  港澳台地区援助汶川地震灾区回顾

- ju bei ji shu he chan ye lian you shi

  具备技术和产业链优势

- kua guo fan du tou mu bei zhua huo

  跨国贩毒头目被抓获

- bang zhu kun nan qun ti jiu ye

  帮助困难群体就业

- hua nan jiang zao yu kuang feng bao yu

  华南将遭遇狂风暴雨

- da qi wu ran zhi li cuo shi bu duan

  大气污染治理措施不断

2.2 反面例子

- chang yi wei fei zhou dai lai geng duo fa zhan ji yu

  倡议为非洲带来更多发展**基于**

- xi tu xian liang biao zhun qu xiao gei cha chan ye dai lai ji ji ying xiang

  西土限量标准取消给茶产业带来积极影响

- chao xian ban dao wu he hua ying xian yu zhong zhan xuan yan fa biao

  朝鲜半岛无核化应**限**于**中展**宣言发表

- kai hao che chuan ming pai

  开豪车**船**名牌

2.3 案例分析

- 二元字模型只考虑了相邻的两个字，目光较为短浅，无法准确识别“发展机遇”中的“机遇”、“终战宣言”中的“终战”。如果能换成四元字或者二元词模型，或者可以改进此点。
- 语料库中的词汇不够丰富，专有词汇匮乏，致使无法识别“稀土”中的“稀”。添加其他语料库，或者添加其他词库，或者可以改进此点。
- 对整句的涵义缺乏理解，无法识别“先于终战宣言发表”中的“先”。这一问题可能需要引入 NLP 领域中更加先进和复杂的算法方能解决。
- 缺乏对句子词性和成分的分析，导致会将动词“穿”误识为名词“船”。

3 性能分析

为分析程序的性能，我考虑了三个指标：

- 字准确率 (character accuracy): 在测试集中正确识别汉字的比例。
- 词准确率 (word accuracy): 在测试集中正确识别词的比例。（使用 pypinyin 分词）
- 句准确率 (sentence accuracy): 在测试集中正确识别整句的比例。

下面尝试对 Laplace 平滑中引入的参数 α 进行调整。

3.1 基本模型

α	character accuracy	sentence accuracy
0.7	0.765384	0.337552
0.85	0.768749	0.352086
0.9	0.768769	0.351148
0.95	0.767714	0.351383

表 1: 参数调整：基本模型

3.2 改进模型

α	character accuracy	word accuracy	sentence accuracy
0.85	0.785876	0.852945	0.391233
0.9	0.786533	0.853834	0.392639
0.93	0.786852	0.854352	0.394749
0.95	0.786513	0.854031	0.394749

表 2: 参数调整：改进模型

最终选择了 $\alpha = 0.93$

4 改进方案

通过对基本模型和改进模型的比较，以及对于错误案例的分析，我们可以得出以下几点改进方案：

- 尝试其他模型，比如三元字、四元字、二元词或三元词模型，LSTM 或 RNN 等深度学习模型。
- 考虑拼音旁边的标点、数字和句尾等特殊符号对数字的影响。
- 尝试其他的 Smoothing 方案，比如 Good-Turing Smoothing 等。
- 改变输入方式，使用以字母为单元的输入方式，而非以单字的拼音为单元。
- 改进前端界面，实现基于 terminal 或 GUI 的实时交互输入。
- 优化数据，扩大训练集，使用更加正确而合理的测试集。