概率论在自然语言处理中的应用

twd2

2017 年 5 月 4 日

当下，人工智能是热议话题。人工智能中，有一个方向叫做**自然语言处理**。而在自然语言处理方面，有两个经典的问题：**光学字符识别（OCR）**和**拼音输入法**。它们都可以用概率的方法解决，本文就尝试讨论这个话题。

光学字符识别问题　所谓光学字符识别（OCR），就是给定一幅图片，让计算机程序识别出来图片中的文字。这涉及到图像匹配、模式识别等算法，但本文不关注于此，本文关注的是后处理的过程。

首先，对于一个字符的识别，识别结果就可能有多种，每一个结果都有一个置信度。所谓后处理过程，就是对于已经识别出来的字串（字串每个字都有多种可能选项），选择“最佳”的组合。这和下文讨论的拼音输入法十分类似，所以本文的重点放在对于拼音输入法的讨论。

拼音输入法问题　拼音输入法，指的是一个程序，它接受用户输入的拼音串（比如“qing hua da xue ji suan ji ke xue yu ji shu xi”），根据内部数据，将这个拼音串转换为文字串输出（对于上面的例子就是“清华大学计算机科学与技术系”）。

对于输入拼音串的每一个拼音（上面例子中的“qing”、“hua”、“da”、“xue”等），可以简单直接地查询字典来获得该拼音对应的所有可能的字，然后选择“最佳”的组合，认为是该拼音串对应的句子：

$$\begin{array}{cccc} 请 & \underline{清} & 氢 & \cdots \\ 画 & 话 & \underline{华} & \cdots \\ \underline{大} & 打 & 答 & \cdots \\ \underline{学} & 雪 & 血 & \cdots \\ & \cdots & & \end{array}$$

“最佳”的不同的定义方法，对应着寻找最佳组合的不同算法。这里，我讨论一个简单的**二元字模型**或**字的二元模型**。我理解中，字的二元模型就是将句子

中相邻的两个字作为一组，后一个字出现的概率只和前面一个字出现的情况有关。这能够极大地简化相关算法设计、提高算法速度，但是准确度也会因此受到不良影响。

对于任意一个句子S，如果记其长度$n \triangleq |S|$，并且记$S = w_1 w_2 \cdots w_n$，$w_i \in W\ (i = 1, 2, \ldots, n)$为字符集$W$中的一个字符，那么，其出现的概率$P(S)$可以表示成：

$$\begin{aligned} P(S) &= P(w_1) \cdot P(w_2|w_1) \cdot \cdots \cdot P(w_n|w_1 w_2 \cdots w_{n-1}) \\ &= \prod_{i=1}^{n} P(w_i|w_1 w_2 \cdots w_{i-1}) \end{aligned}$$

其中，$P(w_i)\ (i = 1, 2, \ldots, n)$为$w_i$出现的概率。
在字的二元模型下，这个表达式可以进一步简化成：

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

再由条件概率的定义及大数定律，

$$P(w_i|w_{i-1}) = \frac{P(w_{i-1}w_i)}{P(w_{i-1})} = \frac{P(w_{i-1}w_i)count(*)}{P(w_{i-1})count(*)} \approx \frac{count(w_{i-1}w_i)}{count(w_{i-1})}$$

其中，$count(w) = w$出现的次数，并且$count(*) \triangleq \sum_{w \in W} count(w)$，表示字和字的二元组出现次数总和。它们都可以由语料统计得到。
于是，

$$P(S) \approx \prod_{i=1}^{n} \frac{count(w_{i-1}w_i)}{count(w_{i-1})}$$

这就是一个句子出现概率的算法。

记$W(y) = \{\text{拼音}y\text{对应的所有字}\} \subset W$，不妨认为其中的元素按照某种顺序排好序。这样，给定一个包含m个拼音的拼音串$y_1 y_2 \cdots y_m$，如“qing hua da xue”，可以枚举每一个可能的句子$S \in \prod_{i=1}^{m} W(y_i) = \{\text{请}, \text{清}, \text{氢}, \ldots\} \times \{\text{画}, \text{话}, \text{华}, \ldots\} \times \{\text{大}, \text{打}, \text{答}, \ldots\} \times \{\text{学}, \text{雪}, \text{血}, \ldots\}$，计算$P(S)$，然后认为$P(S)$最大的$S$就是这个拼音串对应的句子，即对$y_1 y_2 \cdots y_m$求出

$$\underset{S \in \prod_{i=1}^{m} W(y_i)}{\operatorname{argmax}} \{P(S)\}$$

实现上有几个问题：精度问题、平滑问题以及速度问题。

精度问题　由于句子可能较长，最后计算出的概率可能很小，为了不致产生过于严重的浮点数精度的问题，可以将概率取对数使用：

$$\ln P(S) \approx \sum_{i=1}^{n} \ln\left(\frac{count(w_{i-1}w_i)}{count(w_{i-1})}\right)$$

平滑问题　$count(w_{i-1}w_i)$可能为0，这时，可以参考$P(w_i)$的值。使用新的“概率”值为

$$P'(w_i|w_{i-1}) = \lambda P(w_i|w_{i-1}) + (1-\lambda)P(w_i)$$

其中，λ为某一参数。

速度问题　如果实现中真的枚举每一个S，那么速度会很慢。这里，可以使用动态规划算法。

记w_{ij}为$W(y_i)$中的第j项。记f_{ij}为截止到y_i，如果w_i采用w_{ij}，$P(w_1w_2\cdots w_i)$的最大值。那么，有状态转移方程

$$\begin{aligned} f_{ij} &= \max_{1\le k\le |W(y_{i-1})|}\{f_{i-1,k}\cdot P(w_{ij}|w_{i-1,k})\}, \forall 1\le j\le |W(y_i)|, \forall 1\le i\le m \\ f_{1,j} &= P(w_j), \forall 1\le j\le |W(y_1)| \end{aligned}$$

如果取了对数，上面乘法变成加法。

最终，$\max_{1\le j\le |W(y_m)|}\{f_{mj}\}$为所求最大值，对应的$w_1w_2\ldots w_m$为所求句子。

至此，我对拼音输入法的字的二元模型就讨论完了。我曾编写过小程序测试这个模型及算法，单字的正确率为79.66%，句子的正确率为32.43%。值得一提的是测试集，这个测试集包括了日常用语、诗词歌赋以及最新网络语言，相当严格。

总之，本文尝试探讨了如何利用概率方法来解决自然语言处理中几个简单的问题。顺便一提，概率理论和方法在各个学科都有广泛的应用，在信息科学方面尤为突出。例如，**卡尔曼滤波（Kalman Filter）**用到了更多高级的概率论知识。简单来说，卡尔曼滤波就是根据对系统的预测和对系统的观测（通常是包含噪声的），做出对于系统当前状态的最佳的估计。卡尔曼滤波是一个非常有用的算法，在惯性导航、姿态解算、计算机视觉等方面都有应用，但是由于本人能力所限，这里就不做过多的探讨了。