拼音输入法实验报告

计 62 杜政晓 2016011014

算法思路

首先我们需要建立一个汉语的语言模型，即给定一串汉字的序列，要求下一个字（词）的概率分布。利用马尔可夫模型，我们可以认为下一个字（词）的概率分布之和它之前的(k-1)个字（词）有关，也就是这个概率分布就是在之前的(k-1)个字的条件下的条件概率。以 k=2 的情况为例，根据贝叶斯公式我们知道

$$\mathrm{P}(w_i|w_{i-1}) = \frac{P(w_{i-1}w_i)}{P(w_{i-1})}$$

其中$P(w_{i-1}w_i)$和$P(w_{i-1})$可以认为与$w_{i-1}w_i$和w_{i-1}出现的频率成正比。所以在训练的时候的问题就是根据所给的语料统计单个字和两个字的不同组合出现的次数。

在进行拼音转化为汉字的时候，基本思想就是要使产生的汉字序列的概率最大化。也就是要最大化

$$\mathrm{P}(w_1w_2\cdots w_n) = \prod_{i=1}^{n} P(w_i|w_{i-1})$$

这个问题就可以转化成了搜索问题。这个问题很难提出很好的评价函数，所以我们直接用动态规划的方法来解决。

假设我们已经完成了对前 i 个汉字的搜索，在搜索第 i+1 个汉字时，第 i+1 个汉字是 t 的概率为

$$P\left(w_{(i+1)} = t\right) = \max_{r\in V} P(w_i = r) * P(t|r)$$

对 t 的所有可能都完成计算之后，就完成了第 i+1 个汉字的搜索。最后一个汉字

搜索完成后，按照最大可能的那一个选项向前回溯，就得到了概率最大的汉字序列。

实现过程

语料来源与处理

统计语言模型所基于的语料有两个来源。一个是作业中提供的新浪新闻。另一个是 https://github.com/nonamestreet/weixin_public_corpus 上提供的微信公众号文章的语料，我认为可以部分弥补新闻语料语体过于正式的问题。

原始的语料里混有大量的数字、标点和不在转化范围内的汉字，而且文本都是大段大段没有按句分开的。所以首先按照标号将文本按句分开，然后再用正则表达式将数字和符号去掉，最后只保留汉字表中的汉字。

Padding

对于每个拼音和汉字序列的开头增加了(k-1)个”b”，在结尾增加了一个”e”，这样可以不用特殊处理开头和结尾的概率，搜索仍然是从第一个汉字开始，搜索到 e 结束。

平滑化

在扩展到三元模型时有一个问题是因为语料的规模限制，不可避免的有一些三元组在语料中没有出现过或者出现次数很少，因为无法给出好的概率近似值。解决方法是进行线性插值：

$$\mathrm{P}(w_i|w_{i-2}w_{i-1}) = \lambda_1 p(w_i|w_{i-2}w_{i-1}) + \lambda_2 p(w_i|w_{i-1}) + (1-\lambda_1-\lambda_2)p(w_i)$$

但是这就引入了对这两个参数的选择，我们这里可以使用一种简单的方法：

$$\begin{aligned}\lambda_1 &= \frac{c(u,v)}{c(u,v)+\gamma} \\ \lambda_2 &= (1-\lambda_1)\times\frac{c(v)}{c(v)+\gamma} \\ \lambda_3 &= 1-\lambda_1-\lambda_2\end{aligned}$$

其中c 表示序列出现的次数。这样的话就只有一个参数γ需要选择。而且这种方法也是合理的，因为当$w_{i-2}w_{i-1}$出现的次数比较多时，λ_1就会接近于 1，λ_2、λ_3就会相应减小，表示$p(w_i|w_{i-2}w_{i-1})$占据了主导地位。

测试方法

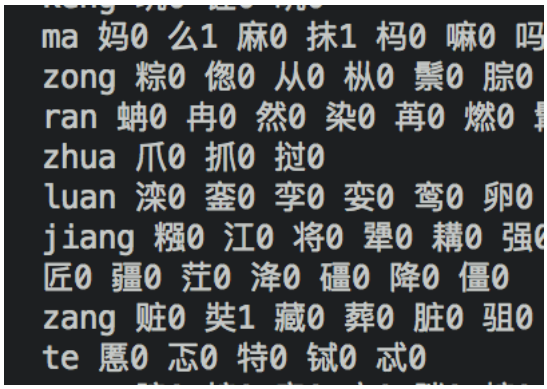
为了测试模型效果和选择最佳参数，从整体语料中随机抽取了 1/10 作为测试集。使用 pypinyin 转化成拼音之后作为 ground truth。

多音字的不同读音区分

在实际测试的时候，发现多音字的问题会给程序的准确性带来很大的问题。简比如行 (Xing) 和行 (Hang) 就是完全不同的意思，而现在的模型却会把 “Yin Xing” 转换成 “银行”，而且因为银行这个词出现很频繁，所以得到的概率很大，很可能被作为最终结果，但这样的转换显然是错误的。

为了解决这个问题，首先将原始语料用 pypinyin 进行了注音（pypinyin 处理多音字还是比较准确的），然后将一个字的不同读音用数字区分开，如银行的行是 “行 0”，很行的行是 “行 1”，然后再进行词频统计和拼音转换。当然拼音到字的字典也要对应做出修改。

这时的词频文件和拼音字典如下：



实验效果

我们使用了两种统计方法来衡量算法的效果，一个是完全正确的句子占总句子的总数的比例，即基于句的正确率。一个是转换正确的字占所有的字（即所有句子的长度之和）的比例，即基于字的正确率。

以下结果均未进行参数选择（见下一部分）

基本要求：基于字的二元模型

句正确率 44.0% 词正确率 85.1%

错误示例：

原句：性是人类本能 结果：省市人类本能

原句：在您花甲之年 结果：在您画家之年

原句：中国文字博大精深 结果：中国文字博大精神

分析：因为现代汉语中大量的词语都是二字词语，所以使用二元模型的结果就是

把句中的两个字单独拿出来看是成立的，比如“省市”“精神”，但是放到整个句子里看就会特别奇怪，比如“画家之年”，“博大精神”。

原句：自胜者强 结果：自胜者将

原句：将他和报警人一起带回了派出所 结果：将他和报警人一起大会了派出所

分析：这两个例子看了之后都让人有种苦笑不得的感觉。除了二元模型的缺陷意外，还因为“将”和“大”都有我们平常不太注意的读音。而且因为原始的语料是没有注音的，所以训练得到的语言模型也是没有考虑过多音字的，也就是在模型眼中“dai hui”是可以被解释成“大会”的，而显然“大会”在语言模型中是一个概率很高的二元词。

选做要求：基于字的三元模型

句正确率 77.5% 词正确率 97.0%

三元模型的正确度比起二元模型有了很大的提升，尤其是在整句的正确率上。但是代价是速度被大大地拖慢了，尤其是对于长句。“清华大学计算机系”在二元模型上只需要 48ms，在三元模型上却需要 720ms。

错误示例：

原句：人生最难处理的是 结果：人生最难处理的事

原句：于慕尼黑享用午餐后 结果：与慕尼黑享用午餐后

分析：这些句子其实两种方式都能解释，当然人也许能够给出正确的结果。比如第一个句子，我知道“最难处理的是”比“最难处理的事”更可能出现在句子的末尾，但是在基于字的三元模型下，“的是”和“的事”显然在句尾出现的概率不会有什么差别。第二个句子因为我知道“慕尼黑”是一个地名所以前面是“于”的可能比“与”更大，但是一方面是“慕尼黑”这个词本身在语料中出现的次数

就不多，另一方面三元模型也无法准确捕捉到这么长的范围里的搭配。这两个错误其实都是因为三元模型本身的缺陷。

原句：马大妈喜欢码代码 结果：马大妈喜欢么大妈

分析：这就是由于没有区分多音字而产生的错误，即使是三元模型也无法避免。“大妈”这个词的出现概率当然要比“代码”要更大。

扩展方法：基于字的三元模型+区别多音字不同读音

句正确率 86.3% 词正确率 97.8%

在区分了多音字不同读音之后，词的正确率提升不大，但是句的正确率有了很大的提升。这也是最终提交的版本里使用的方法。

正确实例：

原句：马大妈喜欢码代码 结果：马大妈喜欢码代码

可以看到程序没有再把大（da）妈（ma）和大（dai）妈（ma）认为是同一个词，所以能够正确地转换出“码代码”这个词。

参数选择

因为三元模型的完成一次完整的测试需要的时间太长了，所以这里以二元模型为例进行了参数选择。

对于前面说的第一种插值方法，所需要确定的参数是λ_1，这里我们用字正确率作为评估指标，结果如下：

λ_1	Acc(%)
0	47.87
0.5	80.51

0.8	84.07
0.9	84.88
0.95	85.11
0.98	85.17
0.99	85.19
0.992	85.18
0.995	85.18
0.999	85.15
1	84.79

当λ_1=0 时，会退化为一元模型，也就是每个字都选择出现的概率最大的单字，效果很差是可以预料的，比较小的λ_1也没有好的效果。当λ_1=1 时，就相当于没有插值，那么对于某些出现次数较少的二元组偏差就会比较大。根据结果选择λ_1=0.99 可以取得最佳的效果。

对于第二种插值方法，需要选择的参数是γ

γ	Acc(%)
10	85.14
100	85.14
1000	85.11
10000	84.92

可以看到γ的选择对于 performance 的影响是比较小的。这也是这种方法的一个优点：它是一种自适应的方法，对于出现次数不同的多元组可以自动调整出不同的插值系数。

收获与改进

在一开始看到要写一个输入法的时候感觉完全不知道从何入手，虽然实际只要求实现一个读写文本文件的拼音转换工具。但是反复看了课上用语音识别举例的那几张 ppt 之后感觉可以类比，然后通过自己在网上搜索找到了要用的 Vertebi 算法，得益于 Python 自带的各种数据结构和操纵字符串的方便性，倒是没有太麻烦。在这个过程中主要学到的是比如如何解析 json 文件，如何提取自己想要的部分，如何分句，如何去除不需要的标点，这些涉及到输入输出的问题才是最麻烦的。最后看到自己写的程序真的能够对拼音进行正确的转换的时候真的感到很高兴。

程序实现的是基于字的二元、三元模型，其实训练四元模型也是一样的，但是一个是语料不够多训练出的四元模型可能会不够好（据我了解一般商业的输入法也就到三元），一个是四元模型产生的数据文件可能大到没法读进内存。但是原理上和三元是没有任何差别的。另一个提高要求是基于词的，这个我还没有想出完成的办法。一个是分词本身就是汉语比较难的问题。另一个是词的长度是不同的，这样的话在某一步中存在两个长度不同的词，这里的选择就会影响到后面的所有序列，这样 Vertebi 算法基于的马尔科夫模型就不成立了。

除了这些之外，我觉得可能的一个改进方向就是对于从拼音到汉字的处理。在从单个拼音到单个字的时候我们实际上是假设了所有字的概率都相同的。但是实际上有的字的某些读音其实出现的很少，比如强（Jiang），我觉得除了在语文课上我基本没有见过这个读音，这个时候我们是不是应该假设从 Jiang 到“强”的先验概率是很低的？这样就能避免很多很奇葩的结果。

联系方式

微信号 duzx16

邮箱 duzx16@mails.tsinghua.edu.cn