拼音输入法实验报告

计 71 张程远 2017011429

2019.4.20

目录

1 问题描述

本次作业要求我们基于提供的新浪新闻语料库训练一个拼音输入法模型，使得可以把全拼串转化为正确的汉字字符串。输入为一系列小写拼音串，每个音之间用空格隔开，不含阿拉伯数字，标点符号，非英文输入等其他内容。输出为其相应的汉字转换，汉字之间没有空格。

输入样例（Input.txt）如下：

qing hua da xue ji suan ji xi

jin tian hui jia bi jiao wan

liang hui zai bei jing zhao kai

输出样例（Output.txt）如下：

清华大学计算机系

今天回家比较晚

两会在北京召开

2 实验原理

2.1 基于 viterbi 算法的二元字模型

Viterbi 算法是隐马尔可夫模型的核心算法，用于根据观测事件的序列寻找最有可能的隐含事件序列。其本质上是通过对已知情况的大量统计挖掘出概率信息，然后通过概率信息，基于贝叶斯公式，获得观测状态->隐状态的可能性。假设给定的隐马尔科夫模型的状态空间 S 中共有 K 个状态，初始状态 i 的概率为 π i，从状态 i 到状态 j 的转移概率为 a(i,j)。令观测到的时间序列为 y1,y2,…,yT。则最有可能的隐含状态序列 x1,…xT 由下面的公式给出：

$$V_{1,k} = \mathrm{P}(y_1 \mid k) \cdot \pi_k$$
$$V_{t,k} = \mathrm{P}(y_t \mid k) \cdot \max_{x \in S}(a_{x,k} \cdot V_{t-1,x})$$

此处 **Vt,k** 为前 t 个状态最终状态为 k 的观测结果最有可能对应的状态序列的概率。时间复杂度为 O(T|S|^2)。

在本次作业中，我们认为观测序列即为获得的拼音序列；而要求的隐含序列即为汉字序列。为了能够进行 viterbi 算法的计算，我们需要 3 个概率矩阵，这些概率是由语料库数据统计而来的，即初始概率矩阵——每个汉字的出现频率；转移概率矩阵——每个汉字之后的一个汉字是什么，各有怎样的可能；发射矩阵——即汉字对应于各种拼音的概率矩阵。获得前两个矩阵的统计方式为逐条读入新闻，一次检测两个字符，如果有任意一个字符不是中文，

则直接让指针前移；否则记录，从而最后算出频率。而最后一个矩阵，我采用了 pypinyin 的翻译结果作为汉字对应拼音的概率。

拥有 3 个概率矩阵以后，我们就可以按照下面的式子求出每步汉字的概率。令列表 V 存储状态和概率，V[t][‘汉字’]=(score，path)，其中 score 是第 t 个状态为‘汉字’的概率，path 为一直到 t 状态得到的汉字序列。假设我们现在拿到了 t+1 步拼音 pinyin 所对应的所有汉字集合 C，上一步可能的所有汉字集合为 P，那么 V[t+1]步的 score 就可以由下式计算：

$$\mathbf{Score = Max(V[t][pi].score * 转移概率(pi,cj) * 发射概率(cj,pinyin))}$$

将 cj 作为 pi 之后的新状态记录到 path 中，并不断如此做直到句子结尾，我们只需要在最后一步取概率最大者便可以获得整个拼音序列对应的最可能的汉字序列。而对于第一个字的选择，我们将这个字出现的整体字频作为它的 Score，并由此计算下去。这样便可以初步得到拼音序列的汉字翻译。

2.2 基于 viterbi 算法的三元字模型

三元字模型是对原有二元模型进行的改进。基本思想在于，之前我们推导 t+1 步的汉字时，完全基于上一步推导所提供的汉字进行，而实际上句子中的汉字绝不仅仅只受到前一个汉字的影响，它还受到前后语境的影响。为了实现方便起见，我们做更进一步的假设：一个汉字受到其前方两个汉字的影响。我们把一个句子中每两个汉字看成一个节点，这样一句含有 T 个汉字的句子就含有 T-1 个节点。我们要做的就是从第 t 个节点推导出第 t+1 个节点中含有的后一个汉字（t+1 节点的第一个汉字应该与 t 节点的最后一个汉字相同）。要做到这一点，首先需要改变转移概率矩阵——我们要一次性读入 3 个汉字，将前两个汉字作为 key，记录每个不同 key 都包含哪些值，及相应的概率为多少。V[t]中存储的信息也应该更多，之前只存一个拼音对应的汉字信息，现在要存两个。捋清楚这些，我们于是便有下面的式子：

假设现在要根据 t 节点求 t+1 节点。t 节点的前后两个拼音对应可能的汉字集合为 PP 和 P，所求的拼音对应的汉字集合为 C，那么 t+1 步的 Score 应由下式决定：

$$\mathbf{Score = max(V[t][ppi + pj].score * 转移概率(ppi + pj,ck) * 发射概率(ck,pinyin))}$$

可以看出前两个字的选择非常重要，所以在求解前两个汉字时我采用了二元与三元结合的方式——我统计了三元转移矩阵中每个 key 出现的频率（前面介绍过这里的 key 是两字组合），采用这个频率与二元转移矩阵中 key[0]到[key1]的概率相乘作为初始概率，确保选择到对

于输入拼音而言出现频率最高且关系最紧密的两个字。

3 问题与改进

在最开始的单独运行与测试中，我发现了一些问题。这些问题我尽力做出了改进，但限于水平原因，很多仍然存在无法改善的下界。以下就是问题和解决方案。

3.1 提升预处理速度

最开始的时候，无论是二元还是三元模型的预处理速度都很慢。原因是我采用了 list 作为存储工具，而我读入一组汉字时，首先要查找 key 是不是已经出现过，如果出现过则还要查找 value 在 key 对应的子 list 里面是不是也出现过。其结果是，由于 list 是线性查找，时间复杂度为 O(n)，所以在后期数据量较大的时候速度会非常慢——二元预处理要跑上两三个小时，三元预处理则需要跑整整一下午的时间。后来我全部改用 dict 进行存储，因为 dict 查找的时间复杂度是 O(logn)，在这样比较大的数据量下是不错的选择。经过测试，二元预处理的时间缩小到了 20 分钟出结果，而三元预处理所需时间缩小到了 50 分钟，极大提升了效率。据说使用其他读入文件的方式还可以进一步提升效率，不过这里我直接采用了 python 自带的 json 方式读入和输出，对于处理 json 格式的文件还比较方便。

3.2 三元字的平滑处理

这部分是三元模型呈现效果的关键。我们在处理时，获得 3 个输入拼音对应的汉字 XYZ，实际上在语料库中 XYZ 之间很可能并不存在 XY-Z 式的连接，这样我们一时无法去给 XYZ 的概率做准确的评价。XYZ 的连接情况一共可分 4 种，我将它们分为两类考虑，并相应地给它们各自一个权重（debuff），代表不存在 XY-Z 直接连接情况时对这一组合的惩罚。

（1）XY 不是三元转移矩阵的 key

这种情况下意味着 XY 本身的连接并不紧密。如果句子走到这样的情况，并不一定是前面的推导有错误，这有可能意味着 X 有可能是一个词汇的末尾，而 Y 是一个新词的开端。所以这里我采用二元模型的转移矩阵，用 Y 来推导 Z，即将三元模型公式中的转移概率(XY, Z)替换，公式变为：

$$\mathbf{Score(XYZ) = V[t][X+Y].score * 二元转移概率(Y,Z) * 1e-25 * 发射概率(Z,pinyin)}$$

其中 1e-25 是一个惩罚系数，表示只用 Y 导出 Z 获得的概率相比于三元直接连接评价要低一些。如果这里的 Z 甚至不在 Y 的二元转移词典里，这意味着 XY 根本无法推出 Z，这被认为是最糟糕的推导，故直接返回一个很小的参数 1e-40。加上之前的 1e-25，这会带来 1e-65 的惩罚系数，而一般 8 字的句子概率总计都在 1e-30 左右，因此这组参数代表了对 XYZ

的极低评价。

（2）XY 是三元转移矩阵的 key，但是 Z 不在 XY 的转移词典里

这意味着 XY 连接很紧密，但是与 Z 有些脱节。我还是考虑二元模型，即用 Y 推导 Z。如果 Y 可以推导出 Z，我认为这是一种比较好的情况，即 XY、YZ 分别连接地很紧密，这一定程度上代表了句子的连贯性尚可。所以这里的惩罚系数我只给了 0.1，即

$$\mathbf{Score(XYZ) = V[t][X+Y].score * 二元转移概率(Y,Z) * 0.1 * 发射概率(Z,pinyin)}$$

如果 Y 的转移词典里没有 Z，同样意味着 XY 无法推导出 Z。这意味着可能有这样的现象：XY 是一个词的末尾两字，而 Z 是下一个词的第一个字；或者这两者之间毫无联系。所以我做了权衡：由于二元转移如果不连接则返回 1e-40，这样得到的参数是 1e-41，这比 X-Y、Y-Z 都不存在连接要好，但比所有 Y 能推出 Z 的情况要差。

3.3 基于 jieba 分词结果的推荐机制

我们在最后实际上获得了很多的句子，并且常常出现这样的情况——即句子列表里概率最高的虽然并不是正确的答案，但正确答案排在概率第二或者第三。于是我考虑设计一个推荐算法，能够在最终概率排行前 5 的句子中推荐出最好的句子。

评价一个由拼音翻译出的中文句子效果是否足够好，我认为主要是刻画句子中汉字之间的连接紧密程度，即相邻几个汉字如果能够组成更多词语，那么这个句子的评价就会更高；同时词语的长度如果更长，那么句子的评价也会升高。所以我使用了 jieba 分词来处理概率排行前 5 的句子。在在“完全分词模式”下，句子分词将可能会重复覆盖一些汉字，这时我考虑词汇的平均长度，平均长度越长越好;。在“精确分词模式”下，汉语句子将被分割为几个词语，那么词语的数量将作为另外一个参数，词语数量越少越好。当然，原本的概率也是十分重要的参数。

需要注意的是，这个推荐机制只是一种辅助功能，句子本身的概率还是最重要的参数，因此两个 jieba 分词参数提供的对原概率值的扰动不能太大。所以我设计的最终分数公式为：

$$\mathbf{S = p + 1.7 * |List| * jiebascore1 - 0.2 * |List| * jiebascore2}$$

其中 S 为最终得分，|List|为可供推荐的列表长度（一般是 5），两个 jiebascore 分别对应上面的描述。由于各个句子的 jiebascore1 相差不会太大（一般平均词长都在 1-3 之间，相差可能不超过 1），jiebascore2 可能会相差较大（错误的单字往往导致它和周围的字独立成词），所以我给出了相应的参数以扩大或缩小参数绝对值的影响。

3.4 多句优化

如 3.2 的分析所见，汉字的组合实在太过玄学，任何一种算法对句子局面的评估都不是完全适用的。因此，我们考虑给 viterbi 算法犯错的机会，即对于到达 XY 节点这件事而言，之前我们只考虑到达 XY 节点的最优道路，这实际上是目光狭隘的，有贪心的意味在其中。我们考虑给每个节点存储 5 条通往它的最优道路，以及相应的概率值。

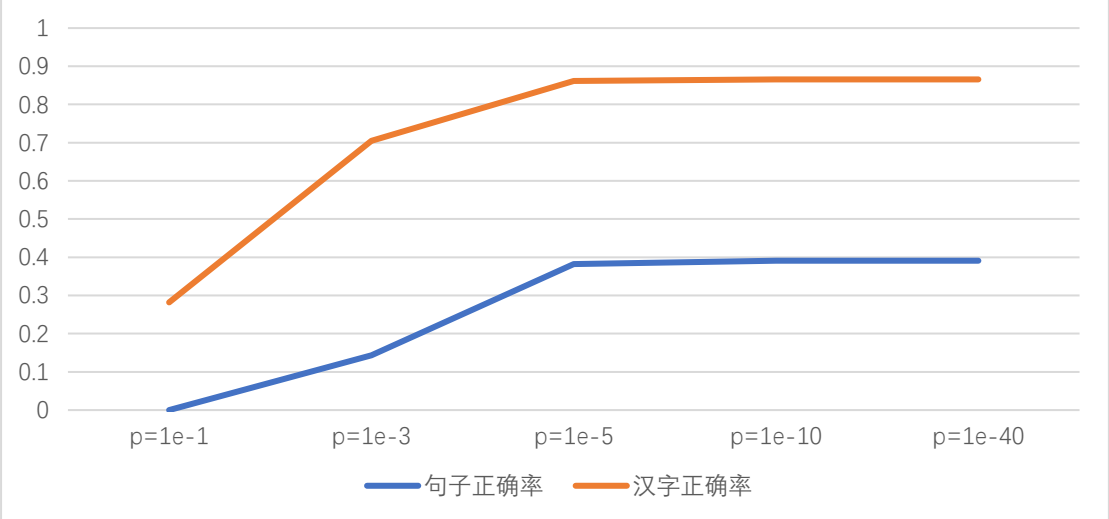
我们来看一下这样做会对推理过程产生什么影响。对于推导出 Y 字而言，这件事的发生可能来自于很多个节点，如 X_1Y, X_2Y, X_3Y 等等。接下来要推导 Z，也就是 YZ 节点，那么枚举所有可能的 XY，而对于每个 XY，我们内部存储了 5 个能够到达它的最好可能，于是就有 5*|S|条道路可供选择，其中 S 代表 X 的取值集合。接下来，我们把这 5*|S|个评分进行排序，找到最优的 5 个存进下一个节点 YZ 里。之前的做法里，我们对于每个 XY 只有一个最优值可供选择，这样在最后一步选择前 5 个句子进行优化时，往往只是最后一个字有所不同，因为之前的状态已经唯一确定了；而实际上，最后一步选出来的句子中，除了最优句子不会变动，其他次优句子并不一定来自于每个暂时最好的 XY——可能某个 XY 的前两条路径相比于其他 XY 都更有竞争力。换句话说，之前我们强制在每个 XY 里选一个最好的，直到组成最后 5 句；加入多句优化以后，可为推荐机制提供的句子质量更加高了。

4 实验结果

本实验所用的两个数据集分别为全部的学生数据集，以及 1000 行新闻数据集，数据均已放在 testdata 目录下。

4.1 二元字模型的准确率

在一千行新闻集上测试得到的平滑参数与正确率关系如图：



图中可以看到，p 的值小于一定程度后，句子正确率和汉字正确率都几乎不变。说明二元模型的平滑参数需要足够小使得让正确的字符能够与错误的字符严格区分开，这样才可以达到比较好的效果。

使用 p=1e-40 在学生数据集中测试，效果如图：

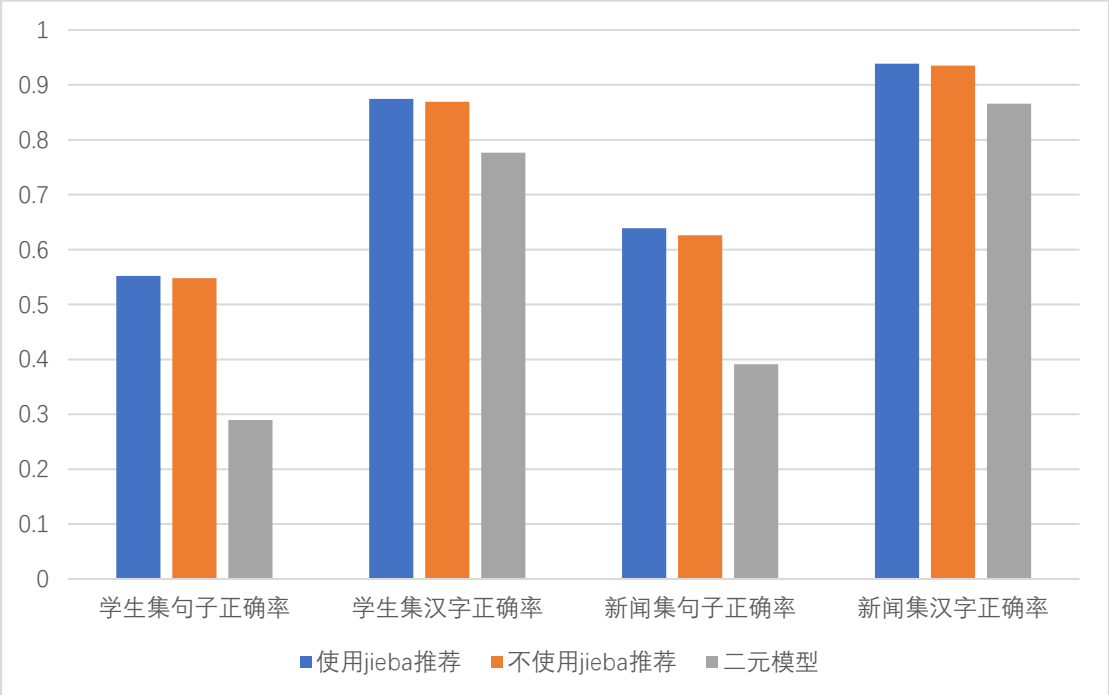
```
是不是以后可以用这个东西聊天  是不是以后可以用这个东西聊天

句子正确率为：  0.28967254408060455
单字正确率为：  0.7767068273092369
[Finished in 13.2s]
```

这说明，二元模型的准确率较低，还有很大的提升空间。

4.2 三元字模型的准确率

选取二元模型的平滑参数 1e-40，分别在学生集和新闻集上测试使用 jieba 推荐与不使用 jieba 推荐的效果，结果如下表：

测试数据集 / 使用方法	学生集		新闻集	
	句子准确率	汉字准确率	句子正确率	汉字准确率
jieba 推荐	0.552333	0.874515	0.639	0.938715
无 jieba 推荐	0.547859	0.869076	0.626	0.935011
二元模型	0.289673	0.776707	0.391	0.865645



从实验效果来看，三元模型的确要比二元模型效果优秀许多；同时使用了推荐算法的模

型在所有测试中都优于不使用推荐算法的模型，证明推荐算法的确有效，不过效果有限，一般能够将准确率提升 1-2 个百分点。

4.3 三元字模型中表现不好的例句及原因分析

以下例句中，前一句均为测试集的句子，后一句为输出的结果。

（1）生命宇宙及一切的终极答案是四十二 vs 声明与周济一切的终极答案是四十二

这一句的问题是后面的汉字被前面的汉字带偏。在新闻集中，宇和与的出现概率相差很大，虽然生命与声明都是常用词，但生命后面接宇的情况远比声明后面接与得分要低。这类问题的产生是因为前面的汉字由于前导较少，可推理的成分不多，只能基于统计矩阵处理的缘故，而前面的汉字一旦错了，后面的汉字很有可能会受影响。

（2）又劳心费神的是他们看似轻松 vs 又劳心费神地使它们看似轻松

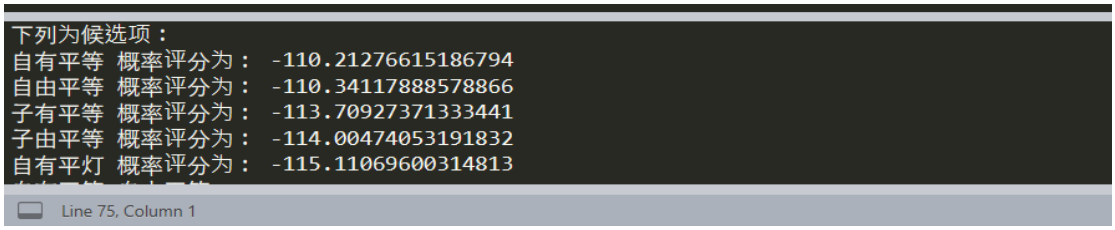
类似的错误还有 “今天晚上又好看的电影 vs 今天晚上有好看的电影”等。“有”和“使”在句子中充当修饰性词语与主体部分的连接，这样的词需要根据后半部分的词性和句子整体结构来看到底该如何使用，很显然这超出了三元模型基本假设的能力——每个汉字基于其前两个汉字进行推理。

（3）它游历西方各国 vs 他游历西方各国/ 他是我的母亲 vs 她是我的母亲

这样的错误是代词错误，同样超出了三元模型假设的能力，因为代词究竟使用什么词还应当看整个句子中是否存在能够给出答案的关键词，如“母亲”、“姐姐”等。但问题是，有些句子并没有关键词，如“游历西方各国”并不能体现出是“她”还是“他”，但是可以大概率否定“它”，这属于更加复杂的情况，目前可能没有太好的方法解决。

（4）自有平等公正 vs 自由平等公正

这句话的错误体现了三元模型的弱点，即我们其实无法用参数或公式很好评价一个句子是不是翻译地足够好。在输出时，显然自由比自有的概率要高出很多倍，但是自由后面可能接入的字太多了，所以‘平’的次数虽然很多，但是概率并不出众，不足以拉开差距；相反，推导‘等’时，等的上两个拼音是“you ping”，那么“有平等”的概率会非常高，而‘由平等’除了自由平等本身之外几乎不会再出现，这样就导致了自由平等稍逊于自有平等的现象——实际上在取对数之后相差只有 0.1，而‘自由’初始要比‘自有’高出很多。



这是一个很难权衡的问题——究竟是要更多的连接，即自有+有平等，还是牺牲一部分连接从而保护词语之间的独立性？我认为，汉语句子是很复杂的，我们无法从数学上直接证明哪种更好，不过我仍然倾向于保护更多的连接，尽管这有可能让本来独立的词语之间出现更多错误。

（5）善大夫王停止杀戮 vs 善待斧王停止杀戮

这句话中出现了多音字，虽然多音字不是这里的主要错误，但是体现了多音字产生的问题——大夫在这里读 dai，但是却使用了‘大’读‘da’和‘dai’转移概率，实际上‘dai’的转移概率应该排除读‘da’的情况，即把‘da’和‘dai’作为两个不同的字处理。

5 后续改进

除了上述多音字的改进，还有以下的几点后续改进措施。

5.1 输入鲁棒性

市面上的拼音输入法对于输入有着很高的鲁棒性，即便输入的是拼音首字母甚至错误的拼音都有可能获得正确的答案。由于拼音一共只有大约 340 种，故我们可以对输入错误的字音进行联想，自动找到最相似的字音并予以替换，从而提升输入的鲁棒性。

5.2 逆向处理

从前面的分析我们可以看出，整个句子的前两个字是最容易出错的，因为没有前导可做参考，只能直接基于词频和字频进行猜测。而且，由于后面的字出现概率只受前面的字影响，因此容易被前面的字“带偏”。我们可以考虑先进行一遍前向处理，得到效果最好的 5 个句子，之后以这 5 组句子的末尾为开头，再逆向操作一遍，这样一共能够得到 10 个句子，从这 10 个句子中再推荐出最好的结果，理论上可以解决一些词汇被识别错误的问题。

5.3 标注词性和关键词

标注词性和关键词也可以减少句子的错误。例如“她是我的母亲“，如果我们能够标注出”ta”为代词和主语，那么我们就会查看宾语的相关信息，如“mu qin”，程序识别出来为名词且为”女性“类别的关键词，这会对代词产生影响，故程序能够对”ta”进行修改，从而获得正确的结果。除此之外，标注词性还可以降低句子中的介词和助词被误判为名词的

概率，如“找到两个袋子，拿走红色的并藏起黄色的”，“的并”很容易被识别成“得病”，而标注词性会提示我们这里需要一个连词，从而得到“并”。这样的操作能够减少类似的错误情况发生。

5.4 基于 seq2seq 模型

在自然语言处理领域，seq2seq 模型常被用于中英互译，其功能是将一个 Input sequence 通过计算转换成需要的 Output sequence。理论上，拼音序列也可以通过设计好的 seq2seq 模型转换为汉字序列，但中英互译和拼音汉字互译实际上还存在着差距——中英互译中，每个单词的意思实际上基本固定，而模型需要学习如何将这些意思组合成一句通顺的话；拼音汉字互译的重点则是找到每个拼音对应的正确汉字。所以现有的 seq2seq 模型其实并不适用这个问题，还需要做一定的修改。

6 致谢与 Honor Code

预处理和算法部分的代码全部由本人完成；其中提升预处理速度采用了计 71 矫瑞同学的提议，关于三元模型的基本原理以及多句优化的具体实现与计 74 陈果同学进行了探讨，其他的算法、参数设计等均由本人独立完成。我对这两位同学表示诚挚的感谢。

另外还要感谢助教对于我关于本次作业相关问题的及时解答，以及老师精彩的上课讲授，让我对于本次作业的基本思路和原理有了初步的认识。期待下次大作业我能够做得更好！