拼音输入法 实验报告

无07 陈宇阳 2020010816

拼音输入法马尔可夫模型

在拥有大量已知句例的情况下，如何从一系列可能的汉字中选取中最可能组成一句话的汉字？这便是拼音输入法所需要实现的基础功能。

定义一句话是合理通顺的话的概率为：$P(S)=\sum_i P(O_i)=f(O_1O_2O_3,\dots)$

其中$O_1,O_2\dots$为句子中包含的每一个字，注意字的排列顺序也是函数自变量的一部分。

考虑句子的构造以及条件概率，由马尔可夫模型容易得出：

$$P(S)=P(O_1O_2O_3\dots)=P(O_1|O_2O_3\dots)\cdot P(O_2|O_3O_4\dots)\cdot\dots\cdot P(O_i|O_{i-1}O_{i-2}\dots)=\prod_{i=1}^{n}P(O_i|O_1O_2\dots O_{i-1})$$

即这些字的组合成句的概率为每一个字在它前面的字的组合的条件下是这个字出现的概率的连乘。

我们取字的组合使得$P(S)$最大即可。

字的二元模型

直接求$\prod_{i=1}^{n}P(O_i|O_1O_2\dots O_{i-1})$显然是困难的，但我们可以对概率作简化，只考虑每个字的前一个字。

即求：$\prod_{i=1}^{n}P(O_i|O_{i-1})$，考虑到我们拥有的语料库句例，

定义$P(O_i|O_{i-1})=\frac{O_{i-1}O_i\text{出现次数}}{O_{i-1}\text{出现次数}}$，这便是字的二元模型。

平滑处理

为了避免$O_{i-1}O_i$出现次数为0的情况，可以加入平滑操作。

令：$P(O_i|O_{i-1})=\lambda\frac{O_{i-1}O_i\text{出现次数}}{O_{i-1}\text{出现次数}}+(1-\lambda)*P(O_i)$

$P(O_i)$要如何定义呢?

单字出现概率及其激活

相比于浩如烟海的语料库，单字出现的次数肯定是很小的。

因此令：$P(O_i)=tanh(\frac{8*O_i\text{出现次数}}{\text{语料库总字数}})$

上述公式是经调试确定的，保证了$P(O_i)$和$P(O_i|O_{i-1})$基本在同一数量级。

字的三元模型

稍作推广，求：$\prod_{i=1}^{n}P(O_i|O_{i-1}O_{i-2})$ 是更加准确的。

同时考虑到字的三字组合和二字组合，令：
$P(O_i|O_{i-1}O_{i-2})=q_1\frac{O_{i-2}O_{i-1}O_i\text{出现次数}}{O_{i-2}O_{i-1}\text{出现次数}}+q_2\frac{O_{i-1}O_i\text{出现次数}}{O_{i-1}\text{出现次数}}+(1-q_1-q_2)*P(O_i)$

便是我设定的字的三元模型。

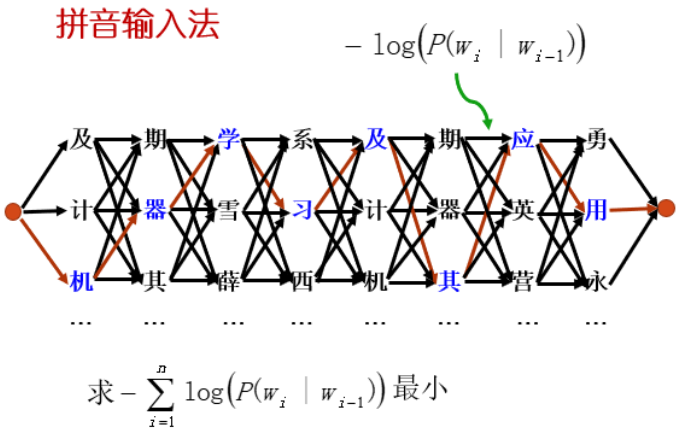
Viterbi求最大值

现在要求$\prod_{i=1}^{n} P(i)$的最大值，以及相应取最大值时的i取值。

只需定义$dis(i) = -log(P(i))$，就把问题转化为了求$\sum_{i-1}^{n} dis(i)$的最小值。

想象$dis(i)$为节点i到某个点的距离，最终问题转化为一个在节点之间寻找最短路径的问题。

图为二元模型的情况。



实验环境

本实验全程在Windows系统下用Anaconda 4.10.3虚拟环境运行，程序语言全部为Python，版本为3.10.0 。

实验语料库以及预处理

语料库

语料库为新浪新闻2016年的新闻语料库，语料库中包括了对应新闻的标题、正文、URL和发布时间。

需要对语料库进行预处理，把有用的信息先储存起来，有需要时直接使用，从而提高程序运行效率。

预处理方法

语料库预处理程序为`/src/preprocess.py`。

- 拼音字典

  把拼音表逐行读取，按空格分割，得到每个拼音对应的字的字典，以`json`格式储存下来。

- 语料库处理

  用正则表达式删去语料库中的URL和发布时间。按照标点符号把新闻切割成一个个分句，这一步是必要的，如果只删去标点而不作切割会产生很多并不正确的连接。

- 储存单字字典

  遍历整个处理后的语料库，把每一个在一二级汉字表中的汉字储存下来，同时记录出现次数，以`json`格式储存。

- 储存双字字典

  遍历整个处理后的语料库，把每一个相连接的双字词组储存下来，同时记录出现次数，以`json`格式储存。

- 储存三字字典

  遍历整个处理后的语料库，把每一个相连接的三字词组储存下来，同时记录出现次数，以`json`格式储存。这一步花费了超过3小时的时间，三字词组比二字词组要多得多，说明分布也稀疏得多，不禁让我怀疑我所设计的三元字模型能否发挥作用。

- 单字概率激活

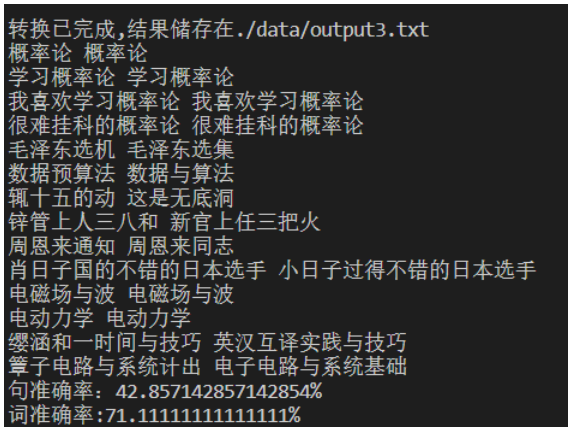
  这一预处理程序为`/src/tanh_process.py`

读入单字字典，每一个除以语料库总字数后通过$tanh$函数激活，再以`json`格式储存。此激活函数在趋近于0处斜率较大，在趋近于1处斜率趋于平缓，能使得单字的出现频率的数量级与前面的$P(O_i|O_{i-1})$在同一数量级。

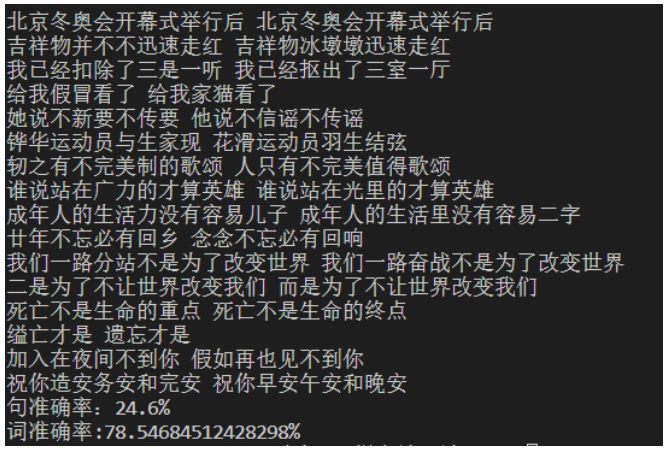
实验效果

二元模型

自己的测例：

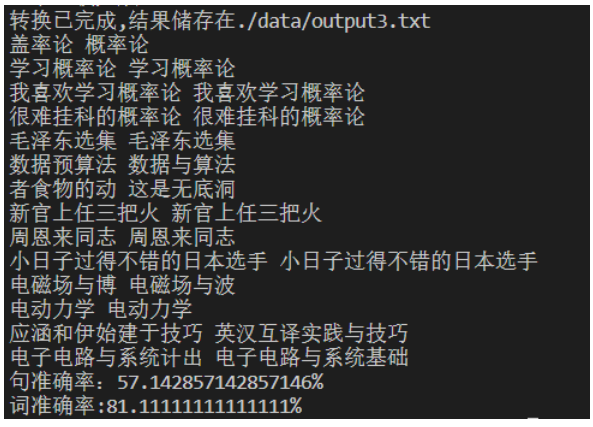


标准测例：



三元模型

自己的测例：



标准测例：

```
这就是我们试图作出无法预测得行动的原因 这就是我们试图做出无法预测的行动的原因
北京冬奥会开幕式矩形后 北京冬奥会开幕式举行后
吉祥物并不不迅速走红 吉祥物冰墩墩迅速走红
我已经口出了三十一斤 我已经抠出了三室一厅
给我假冒看了 给我家猫看了
他说不信谣不传谣 他说不信谣不传谣
画画运动员与省界线 花滑运动员羽生结弦
人只有不完美质的歌颂 人只有不完美值得歌颂
说说站在广里的才算英雄 谁说站在光里的才算英雄
成年人的生活力没有容易儿子 成年人的生活里没有容易二字
年念不忘必有回乡 念念不忘必有回响
我们一路分站不是为了改变世界 我们一路奋战不是为了改变世界
二十为了不让世界改变我们 而是为了不让世界改变我们
死亡不是生命的重点 死亡不是生命的终点
以往才是 遗忘才是
加入在夜间不到你 假如再也见不到你
助你早安无安和完安 祝你早安午安和晚安
句准确率：37.8%
词准确率:84.60803059273422%
```

讨论与分析

可见三元模型在句准确率和词准确率上都要比二元模型高出约10%，但翻译的效果并不是总是三元优于二元。下面挑选一些有意义的句子进行分析：

- 概率论&盖率论

  二元翻译正确，而三元翻译为了“盖率论”。三元模型前两个字的选择其实是用二元模型的，那么为什么会出现与二元模型不相同的结果？只能解释为在选取第三个字的时候三元模型发现“盖率论”的组合要优于“概率论”。但其实“盖率论”在语料库中是没有出现的。同样的问题还出现在“电磁场与博”的错误翻译中。经过检查后发现，其实是三元模型的前两个字”盖率“就已经选择错了。其实这是由于我对二元距离作了一些小小的修改，把前面的字和后面的字换了一下位置，这样得出的正确率更高。但也导致了某些词语更加难以辨认。

- 很难挂科的概率论&我喜欢学习概率论

  有了前面的铺垫以后，概率论二者都翻译对了。可见“概率论”其实才是普遍的结果。

- 周恩来通知&周恩来同志

  三元模型相对能更加充分地掌握上下文的信息，而二元模型更可能把前后割裂开来，选择最常见的二字词组合。

- 古诗词、俗语的翻译

标准测例中的“莫听穿林打叶声”和我的测例中“新官上任三把火”三元模型都基本转换出来了，但二元模型则翻译得一塌糊涂。首先这与语料库的选择有关，新闻语料库中相对缺少古诗词和俗语，且此类词语要求对整体的把握很高，因此相对割裂的二元模型很能把握得住。同样，专有名词的翻译如“羽生结弦“、”冰墩墩“等，都是很难译出的。

- 新闻词句的翻译&口语化句子翻译

  由于这是新闻语料库，句子的用词都相对正式，因此对于新闻类语句翻译的准确率要更高。而对于”给我家猫看了“、”抠出了三室一厅“这一类流行语，语料库中基本缺乏储备，因此转换效果会比较差。

- 多音字

  多音字会导致很少用的拼音被识别为一个很常用的字。比如说：”谁说“可能被识别为”说说“。

- 平滑操作的另一种作用

在$P(O_i|O_{i-1}) = \lambda \frac{O_{i-1}O_i\text{出现次数}}{O_{i-1}\text{出现次数}} + (1-\lambda) * P(O_i)$，当中，分母小同样可以导致很大的概率，这样会导致那些如”贪婪“的”婪“字一样出现次数少、可组词语少的字成为被青睐的对象。因此需要加入后面的单字概率，这样可以在一定程度上减少生僻字带来的困扰。