拼音输入法作业报告

作者

魏家栋，计72班，学号2017011445

实现内容

一个简单的汉语拼音输入法，即从全拼拼音到汉字内容的转换程序。基于提供的sina news汉语语料库，实现了字的二元模型、词的一元模型、词的二元模型，并对准确率进行测试和比较。

实现思路

处理语料库

首先从json格式的sina news语料库文件中抽取出标题和正文等文本信息，然后利用jieba库进行分词，同时利用正则表达式过滤掉非汉字字符的部分。对划分好的词语利用pypinyin库进行注音。然后统计字或词出现的频率（次数），字的二元模型统计字频和两字相邻的频率，词的一元模型统计词频，词的二元模型统计词频和两词相邻的频率。最后将字或词、拼音、频率这些信息写入相应模型的sqlite数据库。

调用jieba分词并过滤非汉字字符能够避免无注音的情况，使注音更为准确，也能够为词的一元、二元模型提供数据。这里先jieba分词再过滤非汉字字符，避免“我有5个苹果”先过滤掉非汉字字符导致“有”和“个”相连影响分词的准确性和两字相邻的频率。

调用pypinyin库是因为它对多音字的支持较好，能够根据词语等上下文环境给出相对正确的注音。当然它也会有出错或违反直觉的时候，比如它对“嗯”这个单字的注音为"n"（事实上“嗯”的读音众说纷纭，比如"en""n""ng"），对“略”和“虐”的注音分别为"lve"和"nve"（而我们一般会分别拼作"lue"和"nue"，提供的拼音汉字表也与这一习惯相符，然而事实上标准拼音确实为"lve"和"nve"）。对于出错的情况，由于数据量很大，出现一些小错误不会太影响结果的正确率；对于拼音不确定等特殊情况，比如"lve"和"nve"，在输入拼音时将"lue"和"nue"先自动转化为"lve"和"nve"再进行处理。

采用数据库是因为数据量较大，内存占用过多，需要边处理数据边将结果存储到外存中。采用多种措施加快写入数据的速度：关闭写同步、显式开启事务、设置合适的主键等。主键设置为字/词以及对应的拼音，或者是相邻的字/词以及对应的拼音。注意只将拼音设置为主键显然是不行的，因为一个拼音可能对应多个字/词，只将字/词设置为主键同样是不行的，因为有多音字和多音词的存在，如“朝阳”既可以拼作"chao yang"也可以拼作"zhao yang"。

建表

数据库有很多优点，比如占用内存较小等，但是读取的速度较慢，而且存储了一些模型可能用不到的冗余信息，占用外存空间较大。因此在数据库的基础上建立一个内存中的dict，加快查询速度。这个dict表以json格式存储在外存中，运行程序时首先将其加载入内存。字的二元模型包括两个dict，一个为单字拼音->list(汉字，log概率)，另一个为相邻字拼音和汉字->log概率。词的一元模型包括一个dict，词拼音->(出现概率最高的词，log概率)。词的二元模型包括两个dict，一个为词拼音->list(词，log概率)，另一个为相邻词->log概率。

这里采用概率而不是频率，方便后续运算，并且对概率取对数，避免概率过小导致浮点数下溢。值得注意的是，在这些dict表中，对单个字或单个词，我们用拼音作为key，用汉字和概率作为value，这是易于理解的，因为我们要做的是拼音到汉字的转换；而对相邻的字或相邻的词，我们用拼音和汉字整体作为key，用概率作为value，因为我们需要的是字/词相邻的概率，而不是拼音相连的概率。词的二元模型对相邻的词的dict做了进一步的简化，只用相邻词作为key，忽略了相同的相邻词而读音不同的情况，事实上在本数据库中也确实没有出现这种情况。但是字的二元模型相邻的字的dict就不能做类似的简化（只用相邻字而不带上拼音作为key），进行了这样的简化后出现了大量相同相邻字不同读音的情况，转换准确率大大降低。

转换拼音至汉字

字的二元模型采用基于动态规划思想的Viterbi算法，算法思路如下。

近似的认为整个句子出现的概率为`P(w_1)*P(w_2|w_1)*P(w_3|w_2)*...*P(w_n|w_n-1)`即`P(w_1)*P(w_1 w_2)/P(w_1)*P(w_2 w_3)/P(w_2)*...*P(w_n-1 w_n)/P(w_n-1)`，问题转化为求上式概率的最大值对应的w的序列。发现上式中`w_1`只与前两项有关，`w_2`只与第二项和第三项有关，因此可以采用动态规划减小搜索空间。具体来说，首先对第一个拼音对应的所有可能的`w_1`，查表获得概率`P(w_1)`；对第二个拼音对应的所有可能的`w_2`，与`w_1`交叉匹配，查表获得概率`P(w_1 w_2)`，计算`P(w_1)*P(w_1 w_2)/P(w_1)`，保留每个`w_2`该式概率最大值及其对应的`w_1 w_2`；对第三个拼音对应的所有可能的`w_3`，与`w_2`交叉匹配，查表获得概率`P(w_2 w_3)`，计算`P(w_1)*P(w_1 w_2)/P(w_1)*P(w_2 w_3)/P(w_2)`，其中`P(w_1)*P(w_1 w_2)/P(w_1)`已经计算过了，保留每个`w_3`该式概率最大值对及其对应的`w_1 w_2 w_3`...依次类推。最后再对所有可能的`w_n`对应的概率取最大值得到序列`w_1 w_2 ... w_n`。

由于`w_i-1 w_i`这样的序列可能从未在语料库中出现，即`P(w_i|w_i-1)=0`的情况，采用平滑策略，令`P(w_i|w_i-1)=P(w_i)/punishment`，其中的惩罚需要测试得到。惩罚的目的在于使得“相关”的字更容易被选中，即`P(w_i|w_i-1)!=0`的相邻字更容易被选中，从而提高句子的连贯性。

词的二元模型与上述字的二元模型类似，只不过需要在每一步多进行一层循环，用于将拼音分割为句子-尾词，而且动态规划也并不是逐层递推的了，而是每层可能与之前的所有层有关。需要注意的是，字的二元模型中可以采用概率也可以采用频率，因为输入拼音转换为句子后字的个数是确定的，而词的二元模型句子中词的个数不确定，如果采用频率可能会出现不同路径对应的常数不一样的情况，因此最好采用概率。

词的一元模型非常简单，依然是类似的动态规划算法，但是只需要将拼音分割为句子-尾词，则`P(w_1 w_2 ... w_n)=P(w_1 w_2 ... w_n-1)*P(w_n)`，不需要考虑尾词的所有可能，只需取概率最高的尾词即可。

参数选择

需要通过实验选择合适的punishment参数，以提高模型的准确率。由于程序中采用的是对数概率，因此平滑策略修正为`log(P(w_i|w_i-1))=log(P(w_i))-punishment`。下面列出的punishment指的就是这个被减去的惩罚。

在马老师提供的测试集上，采用字的二元模型，punishment参数对准确率的影响如下：

punishment	**单字准确率**	**句子准确率**
0	78.262%	26.717%

punishment	单字准确率	句子准确率
1	79.648%	28.589%
1.5	79.961%	28.839%
2	79.817%	28.714%
2.5	79.793%	28.714%
3	79.805%	28.714%
5	79.805%	28.714%
10	79.805%	28.714%

由此可见，加一个惩罚对提升准确率是有效的，但是惩罚过大并不会有进一步的提升。对比之后，我选定punishment为2。

在马老师提供的测试集上，采用词的二元模型，punishment参数对准确率的影响如下：

punishment	单字准确率	句子准确率
0	88.661%	54.057%
0.25	88.830%	55.181%
0.5	88.830%	56.180%
0.75	88.866%	55.930%
1	88.878%	55.680%
1.5	88.782%	55.431%
2	88.569%	53.932%
3	88.035%	52.185%

类似的，可发现加一个惩罚对提升准确率是有效的，但是惩罚过大反而会产生副作用。对比之后，我选定punishment为0.5。相比于字的二元模型，这个惩罚相对要小一些，说明词与词之间的联系相比于字与字之间的联系要弱一些。这是合理的，因为词本身已经包含了某种意义上的字与字之间的关系。

测试结果

在马老师提供的测试集上各个模型的准确率如下，共有801个句子，8299个单字：

模型	单字准确率	句子准确率
字的二元模型	79.817%	28.714%
词的一元模型	82.323%	34.332%
词的二元模型	88.830%	56.180%

从中可以看出，词的二元模型效果最好，词的一元模型次之，字的二元模型效果最差。而且词的二元模型显然比其他两个效果要好很多。

其中效果较好的例子比如：

输入：qing hua da xue ji suan ji xi
输出：清华大学计算机系
清华大学、计算机系两个词出现的频率可能较高，程序能够正确地将它们组词。

输入：jin tian ye shi hao tian qi
输出：今天也是好天气
这句话非常日常，出现频率可能很高，概率较大，程序能够正确转换。

输入：ren yu ren zhi jian wei shen me yao hu xiang shang hai ne
输出：人与人之间为什么要互相伤害呢
这句话也偏于日常，出现频率可能较高，因此程序能够正确转换。

效果较差的例子比如：

输入：mei jun fang cheng bu cheng ren zhong guo dong hai fang kong shi bie qu
应为：美军方称不承认中国东海防空识别区
实为：美军方承不承认中国东海防控识别区
这句话在字的二元模型、词的二元模型中能够正确转换，在词的一元模型中出现错误。这句话很像是新闻中的一句话，我们通过新闻训练出来的模型应该能够正确处理，可能词的一元模型过于强调单个词语的正确性，忽略了上下文的关联，导致出错。

输入：fu shou gan wei ru zi niu
应为：俯首甘为孺子牛
实为：扶手感为孺子牛
这句话在词的一元、二元模型中能够正确转换，在字的二元模型中出现错误。词模型在词与词连接时会乘上一个很低的概率，因此会倾向于词数更少的句子，可能这句话整个就被分割为了一个词语，因此词模型正确挑出了它。而字模型只局限于局部相邻字，没有宏观上的把握，导致生拉硬套组词，最终出错。

输入：zhe bu shi yi tiao sou gou shu ru fa dou da bu chu lai de ju zi
应为：这不是一条搜狗输入法都打不出来的句子
实为：这不是一条搜狗输入法都打不出来的巨资
实为：这不是一条搜狗输入法都大部出来的巨资
这句话在词的二元模型中能够正确转换，在字的二元模型、词的一元模型中出现错误。显然“打出来”和“句子”相隔太远，程序没有成功发现二者之间的联系，因此将"ju zi"错误转换为了“巨资”，而词的二元模型能够很好地处理词语较长导致的相隔较远的问题，因此能够正确转换。

输入：wo de peng you song le wo yi pen lv zhi zuo wei sheng ri li wu
应为：我的朋友送了我一盆绿植作为生日礼物
实为：我的朋友送了我一盆履职作为生日礼物
这句话在字的二元模型中能够正确转换，在词的一元、二元模型中出现错误。这很神奇，可能是因为“履职”出现的概率很高，而“盆”与“绿植”作为相邻词出现的概率又比较低，导致词模型错误地转换成了“履职”。

由于马老师提供的测试集中有些句子偏日常，甚至有些诡异，不能准确地反映出通过新闻训练出来的模型的准确程度，因此我从训练集中挑选了一些句子，制作了一个测试集，共有113个句子，1943个单字，各

个模型的准确率如下：

模型	单字准确率	句子准确率
字的二元模型	91.457%	38.053%
词的一元模型	92.331%	38.053%
词的二元模型	98.302%	83.186%

与上一个测试集类似的，词的二元模型效果最好，词的一元模型次之，字的二元模型效果最差，并且词的二元模型的准确率显著地比另外两个高。相较于上一个测试集，由于采用了新闻中的句子做测试，因此正确率比上一个测试集都高。

未来拓展

- 实现更高元的模型，比如字的三元模型、词的三元模型等。更高元的模型意味着对句子的宏观把握更强，准确率可能有所提升，但同时也意味着“词库”更大、更稀疏，占用内存、消耗时间也会更大。这是一个平衡取舍的问题。
- 词的二元模型载入很慢。这是因为词的二元模型的相邻词dict非常巨大也非常稀疏，达到了528.5MB，因此载入内存时会占用大量内存并且消耗较长时间。我尝试着在建表时去掉那些词频较低的词，但是发现最终转换的准确率会下降，这可能是因为数据量还不够特别大，某些词语确实出现的频率很小。
- 压缩存储或直接使用数据库查询。目前的模型dict表占用空间还是很大，如果能够通过某种方式进行数据压缩，有希望减少占用的外存。或者直接采用数据库进行查询，如果效率还能接受的话。