自动校正汉语拼音输入错误

张杰然-1900012159

项目概述

无论是手机还是电脑执行汉语拼音输入时，总是可能有错误，比如：前后鼻音、平翘舌音、触碰到临近键位等等。我们可以通过python代码完成此辅助功能，该项目具体分为词库建立、双字词语的自动纠错和长句的拼音自动纠错三个部分。

实现原理

拼音纠错的实现大体分为两个步骤，第一步要根据常出现的错误构造出候选句子集，第二步对所有的候选句子进行评分，将结果按照评分由高到低展示即可。

1. **构造候选句子：**
   通过构造混淆集的手段，将各个声母、韵母映射到其对应易错的集合中。拼音常见错误如下所示：

- 前后鼻音：n与ng；
- 平翘舌音：z、c、s与zh、ch、sh；
- 唇齿音：h与f；
- 齿龈边音：l与n；
- 临近键位误触：考虑错按的情况，一般来说分为以下几种情况：
  - 声母错按成另一个声母，这种情况较难分辨是误触还是打字者的本意，如jiao·huan和jiao·guan中hj是临近键位，很难判定打字者的本意具体是哪个词，且若将此类误触都引入，会对结果引入大量的噪声，因此暂时不做区分
  - 韵母错按成另一个韵母，uio三个字母靠的非常近，经常容易误触，且误触后形成的非法拼音较多见，如li和lo、wu和wi等等，因此可以将其纳入误触的范围；
  - 声母与韵母之间的错按，此类情况产生的大部分误触得到的拼音都是非法的，因此比较好判别不是打字者的原意，且也不会给最终结果引入大量噪声，可以全部纳入混淆集。

综上所述，我们可以构造如下的混淆集，根据混淆集，我们可以生成一次错按和两次错按后的待选集。

```
confuseSet_neighboringIncluded={
    #唇齿音
    "h":["f","u"],"f":["h","e"],
    #齿龈边音与前后鼻音
    "l":["n","o","i"],"n":["l","ng"],"ng":["n"],
    #平翘舌音
    "z":["zh","a"],"c":["ch"],"s":["sh","a"],
    "zh":["z"],"ch":["c"],"sh":["s"],
    #键盘误触
    "a":["q","w","s","x","z"],
    "e":["w","s","d","f","r"],
    "i":["u","j","k","l","o"],
    "u":["y","h","j","k","i"],
    "o":["i","k","l","p"],
    "y":["u"],"j":["u","i"],"k":["u","i","o"],"l":["i","o"],
    "p":["o"],"q":["a"],"w":["a","e"],"d":["e"],"r":["e"],
}
```

2. **候选句子评分：**
   可以综合候选词与目标词之间的最短编辑距离、候选句子对应词语总体出现的概率两种指标来对候选句子进行评分。

- 最短编辑距离（levenshtein算法）：
  将一个字符串编辑为另一个字符串，涉及的操作包括Insertion、Deletion、Subsititution和Matching。求两个字符串的最短编辑距离是一个二维的动态规划算法，状态转移方程如下：

$$D(i,j) = min \begin{cases} D(i-1,j)+1; //insertion \\ D(i,j-1)+1; //deletion \\ D(i-1,j-1) + \begin{cases} 1; if X(i) != Y(j) //subsititution \\ 0; if X(i) == Y(j) //matching \end{cases} \end{cases}$$

- 候选词出现概率：
  可以使用统计的方法来获取特定字词出现的概率，可以使用爬虫来获取大量的文字信息，利用jieba分词可以将句子划分为词语，基于此可以获取特定词语出现的频率。
  实际过程中，由于爬虫获取的数据有限且种类相对单一，因此借助搜狗实验室的语料库信息（https://www.sogou.com/labs/resource/w.php ）辅助完成，在本地建立数据库方便查询，利用SQL可以获取数据库所有词语的总频次为251909774405次，具有一定的统计意义，单个词出现的概率即为其频次除以总频次。
- 候选句子出现概率：
  可以依据隐马尔科夫模型与维特比算法来计算特定句子可能出现的概率，在github上可以找到现成的python扩展包（https://github.com/chuan717/Pinyin2Hanzi ）可以将拼音串转为对应中文，并给出其可能出现的概率。借助此扩展包可以完成候选句子概率的获取。

具体实现细节

- 共实现为三个文件，wordDataBase.py代码中完成对统计获取的双字词语及其频次的本地建库，具体数据存储在wordData.db文件中的wordCount表中；
- singleWord.py中实现了其中对target字符串的候选句子生成和打分工作。由于添加临近键位的纠错会对程序时间复杂度和结果引入一定的回声，因此设置了flag_neighboringIncluded的参数便于更换模式。程序对target基于混淆集可能产生的一次编辑错误、两次编辑错误情况进行枚举，并针对每一个情况均向本地数据库请求其出现概率。最后综合其编辑距离和出现概率对结果进行评分，并按照评分降序排列。
- sentence.py中也设置了target和flag_neighboringIncluded的参数，程序生成混淆情况的候选集后进入Pinyin2Hanzi的函数中计算出现概率，并综合编辑距离完成最终评分，展示得分最高的7个句子。

样例展示

输入为（target，flag_neighboringIncluded），输出为（result，评分）。

1. **双字词语：**

类型	输入	输出
正常	(zhongguo,0)	中国 zhongguo 388.90809 终归 zhonggui 10.40303 重轨 zhonggui 10.0186
前后鼻音	(yindu,0)	印度 yindu 108.28835 引渡 yindu 100.28788 阴毒 yindu 100.21363 银都 yindu 100.15303 硬度 yingdu 10.90392 影都 yingdu 10.12456
平翘舌音	(zisi,0)	自私 zisi 101.82673 子嗣 zisi 100.13859 恣肆 zisi 100.05359 只是 zhishi 66.23181 知识 zhishi 39.38177 姿势 zishi 14.34905 自是 zishi 11.32509 之死 zhisi 10.8224

类型	输入	输出
唇齿音与齿龈边音	(fulan,0)	腐烂 fulan 100.84049 弗兰 fulan 100.23172 富兰 fulan 100.03713 湖南 hunan 14.17345 护栏 hulan 10.51107 弗朗 fulang 10.23992 呋喃 funan 10.12518 护拦 hulan 10.0427 赴难 funan 10.01711 胡兰 hulan 10.01388 昏暗 hunan 2.18459 虎狼 hulang 1.17101
临近键位	(wrnnuan,1)	温暖 wennuan 18.72754

2. **短语及长句：**

类型	输入	输出
正常	(zhong hua ren min gong he guo,0)	中华人民共和国 -15.62662 中华人民共合国 -24.39093 综华人民共和国 -28.67557 总华人民共和国 -28.67974 中华迈民共和国 -36.35166 中华陨民共和国 -36.35166 中华迈諮共和国 -45.11671
前后鼻音	(zhong hua ren ming gong he guo,0)	中华人民共和国 -16.62662 中华人民共合国 -25.39093 中华人命共和国 -30.20306 中华人名共和国 -32.13012 中华迈民共和国 -37.35166 中华陨民共和国 -37.35166 综华人命共和国 -43.25201

类型	输入	输出
平翘舌音	(wo si sui,0)	我是谁 -13.73854 我死水 -18.67576 我死谁 -19.42131 我时随 -19.83367 我是随 -20.3825 我是说 -20.81866 我撕碎 -22.09992
键盘误触+平翘舌	(tian qo zen hao,1)	天气真好 -27.84141 天气真号 -28.59228 天气谮獐 -32.80067 天气譖獐 -32.80067 天奥真好 -33.66327 天气赠獐 -33.80067 天气嘱獐 -33.80067

反思与改进

- 拼音纠错混淆集构建时，忽略了声母之间的误触，然而此类情况有时可以辨别，比如duan、fuan、guan中理论上可以将fuan此类的非法输入的错误给纠正。因此可以进一步优化、细化混淆集来获得更优的纠错功能。
- 完成项目时，出于对本地数据库规模和时间的考虑，词语拼音纠错仅仅限于双字词语，可以进一步完善词库，进一步完成单、双、多字词语拼音的纠错。
- 句子拼音纠错时需要在各个字之间手动加入空格，与平常打字习惯不符合，因此可以仿照jieba进一步实现对拼音的分词。由于拼音音节的构成是有限的，可以不依靠大量数据就能构造有向无环图，完成分词路径的最大似然估计，对一串拼音进行自动分割。