拼音输入法 - Scik

王科 201300008 203536673@qq.com

1 基本功能

基于`HMM`的拼音输入法,数字`0-9`选择候选序列，空格键选择`0`号候选，减号键`-`向前翻页，等号键`=`向后翻页。

2 系统特色

1. 实时、高效，能够即时生成候选汉字序列。
2. 准确的排序，基本能保证目标语句出现在第一页候选汉字序列中。
3. 支持长序列。
4. 支持自动纠错。
5. 根据分词结果在原拼音序列中插入`'`，更加用户友好。

3 使用介绍

安装依赖后直接运行`main.py`即可。

```
pip install -r requirements.txt
```

配置文件`src\conf\config.py`：

```
TRAIN_CONFIG = {
    'intact_weight': 21,  # 训练时全拼的权重
    'zcs_weight': 6,  # 训练时['zh','ch','sh']的权重
    'first_weight': 4,  # 训练时首字母的权重
}  # 训练时不同拼音所占的权重

CUT_CONFIG = {
    'has_correct': True,# 选择是否需要自动纠错
    'has_first': True,  # 选择是否支持首字母拼写
    'has_zcs': True,  # 选择是否支持['zh','ch','sh']拼写
}

IME_CONFIG = {
    'max_pinyin_len': 52,  # 选择输入法支持的最长的拼音序列
    'candidate_per_page': 10  # 每页最多显示的候选汉字序列条数
}
```

4 代码模块的功能划分与描述

```
MYPINYIN
|
|  all_requirements.txt  // 由pip freeze生成的依赖
|  requirements.txt  // 由pipreqs生成的依赖
├─db
|  |  correct.txt  // 用于更正错误拼音的规则
|  |  pinyin.txt  // 拼音表
|  |  preprocess.py  // 预处理数据文件的代码
|  |  wordfreq.txt  // 根据原始语料数据计算出的词频表
|  └─corpus  // 原始语料数据
├─model  // 训练得到的模型
|  |  emission_log_probability.json  // 从汉字到拼音的发射概率矩阵
|  |  start_log_probability.json  // 初始概率矩阵
|  |  transition_log_probability.json  // 从前一个字推测后一个字的状态转移矩阵
|  └─reverse
|          compute_nxt.json  // 根据上一个字和拼音推测最可能的下一个字
|          reversed_emission.json  // 从拼音到汉字的发射概率矩阵
|          reversed_transition.json  // 从后一个字推测前一个字的状态转移矩阵
├─path
|      mypinyin.pth  // 用于解决python解释器找不到自建模块的问题
└─src  // 源代码
   |  main.py  // 程序入口
   ├─conf  // 配置模块
   |      config.py  // 输入法的可选配置
   ├─hmm  // 隐马尔可夫模型模块
   |  |  hmm.py  // 用viterbi算法转换拼音
   |  └─train  // 模型训练模块
   |         freqdata.py  // 用于获取词频数据，对数据库进行抽象封装
   |         train.py  // 用于计算HMM模型所需要的参数，结构保存在model文件夹下
   ├─interface  // 输入法界面的模块
   |      ime.py  // 输入法主程序
   |      imeui.py  // 输入法UI界面，由imeui.ui自动生成
   |      imeui.ui  // 输入法UI界面
   ├─split  // 分割拼音的模块
   |      pycut.py  // 用于分割拼音的模块
   └─util  // 工具
          tools.py  // 加载文件、保存文件等通用函数
```

5 方案描述

5.1 数据

5.1.1 语料获取

原始数据来源：https://github.com/brightmart/nlp_chinese_corpus

使用了其中的`wiki2019zh`和`webtext2019zh`。

语料保存在`db\corpus`。

5.1.2 数据清洗与整合

原始数据是`json`格式，直接处理开销较大，我们利用正则表达式提取其中的中文字符。

```
chinese = re.compile(
    r'[\d\
        |\u4e00-\u9fa5\
|\u3002\uff1b\uff0c\uff1a\u201c\u201d\uff08\uff09\u3001\uff1f\u300a\u300b\
        |\n\
      ]'
)
```

原始的语料由多个小文件组成，我们将其合并，然后利用`jieba`分词将原文分割成一个个词语，并且在此基础上计算每个词出现的次数，最终得到词频文件`db\wordfreq.txt`。

源代码在`db\preprocess.py`，生成的文件形如：

```
的 49026416
是 13385754
我 12141126
了 11858449
在 9083079
有 6375791
…… ……
```

5.1.3 数据封装

为模型训练提供统一的数据接口，构建一个迭代器，每次调用返回一个`(word,word_frequency)`二元组，源代码在`src\hmm\train\freqdata.py`。

```
words_path = 'db\\wordfreq.txt'
def wordfreq() -> tuple:
    """ word frequency data set, iteratively return (word, freq). """
    with open(words_path, 'r', encoding='utf-8') as fr:
        for line in fr:
            word, frequency = line.split()
            yield word, int(frequency)

```

5.2 拼音划分

根据输入的拼音序列生成候选的汉字序列，首先要将输入的长拼音序列分割成一个个单独的拼音。

5.2.1 划分方法

参考 [再谈Python之拼音拆分 - 简书 (jianshu.com)] 。

使用最简单的递归方式，提供一个`pinyin_table`，从头开始遍历拼音序列，每当遇到一个在`pinyin_table`出现的拼音，就递归地继续划分剩下的序列，直到划分完。代码位于`src\split\pycut.py`。

5.2.2 `pinyin_table`的选择

基础的拼音表保存在`db\pinyin.txt`中，除此之外，为了支持首字母拼写，还增加了如下两个拼音表：

```
first_pytable = ('b', 'c', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'm',
                 'n', 'p', 'q', 'r', 's', 't', 'w', 'x', 'y', 'z')
zcs_pytable = ('zh', 'sh', 'ch')
```

5.2.3 自动纠错

根据日常经常出现的拼音错误情况，总结出替换表`db\correct.txt`，如果在配置中选择了自动纠错，会在划分前安装替换表中的规则进行纠错。

注意：自动纠错只是增加了纠错后的转换结果，用`-`、`=`键前后翻页可以找到纠错前的拼音得到的转换结果，这样即使本不应该纠错也无伤大雅。

5.3 HMM模型

5.3.1 训练

训练代码位于`src\hmm\train\train.py`，计算出的矩阵保存在`model`目录下。此处用数据集中计算出的频率估计真实的概率，为了便于计算，保存的都是原概率的自然对数。为了保证输入法的实时高效，提前将所有由前一个字和后一个字的拼音推测后一个汉字的结果保存在`model\reverse\compute_nxt.json`中，后续转换拼音序列时直接查表即可。

1. start_log_probability.json

   首字出现的概率：

   ```
   {
       "一": -3.8405453029044274,
       "丁": -9.536892549273839,
       ……
       "顱": -19.790367735448843
   }
   ```

2. transition_log_probability.json

   由前一个字推测后一个字的概率：

   ```
   {
       "一": {
           "一": -6.41359328062042,
           "丁": -8.180800311192229,
           ……
           "龟": -14.199798958493531
       },
       ……
   }
   ```

3. emission_log_probability.json

   由汉字推测其拼音的概率：

```
{
    "一": {
        "y": -1.8325814637483102,
        "yi": -0.1743533871447778
    },
    ……
}
```

4. reversed_emission.json

   由拼音推测汉字的概率：

```
{
    "a": {
        "佼": -1.8325814637483102,
        ……
        "鼇": -1.8325814637483102
    },
    ……
}
```

5. reversed_transition.json

   由后一个字推测前一个字的概率：

```
{
    "一": {
        "一": -4.453024061737384,
        ……
        "龚": -10.681085121563946
    },
    ……
}
```

6. compute_nxt.json

   由前一个字和后一个字的拼音推测后一个汉字的概率：

```
{
    "一": {
        "a": [
            "案",
            -10.721640535695247
        ],
        ……
        "zuo": [
            "座",
            -5.6612005031209405
        ]
    },
    ……
}
```

5.3.2 转换

使用维特比算法，具体代码位于`src\hmm\hmm.py`。

5.4 交互界面

5.4.1 主程序

实现`keyReleaseEvent`函数，每次有按键释放就调用`HMM`将当前的拼音序列转化为汉字序列。

5.4.2 图形界面

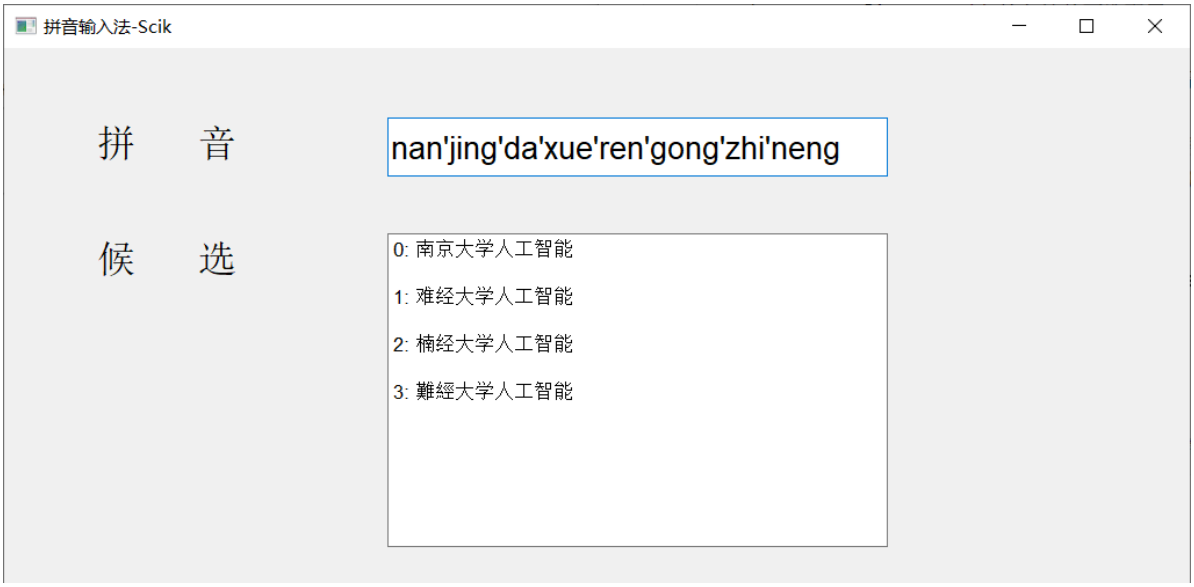
采用`PyQt5`进行编写，利用`designer.exe`进行界面设计，保存为`src\interface\imeui.ui`，然后用命令

```
pyuic5.exe .\src\interface\imeui.ui -o .\src\interface\imeui.py
```

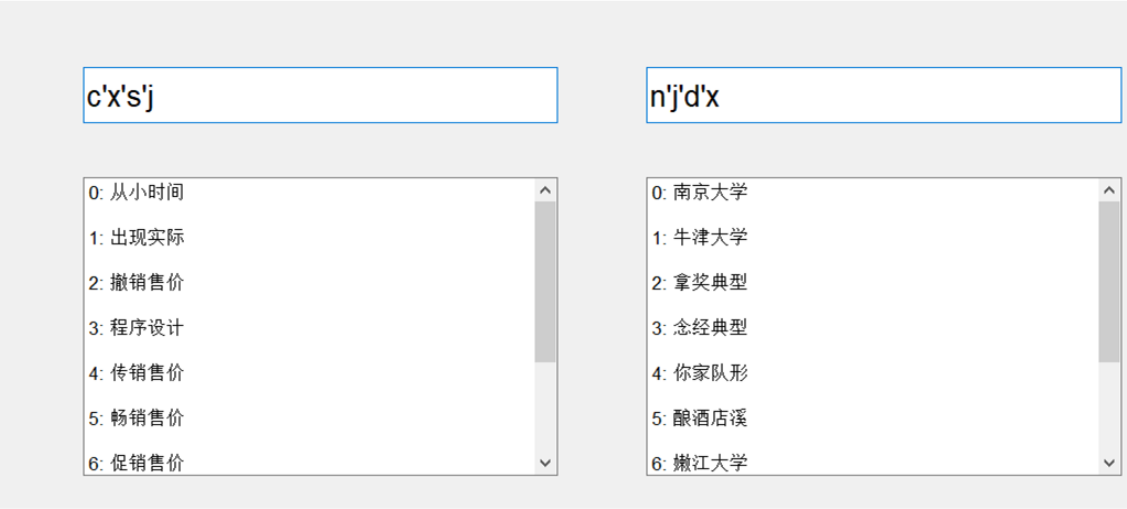
生成python文件`src\interface\imeui.py`。

6 实现效果

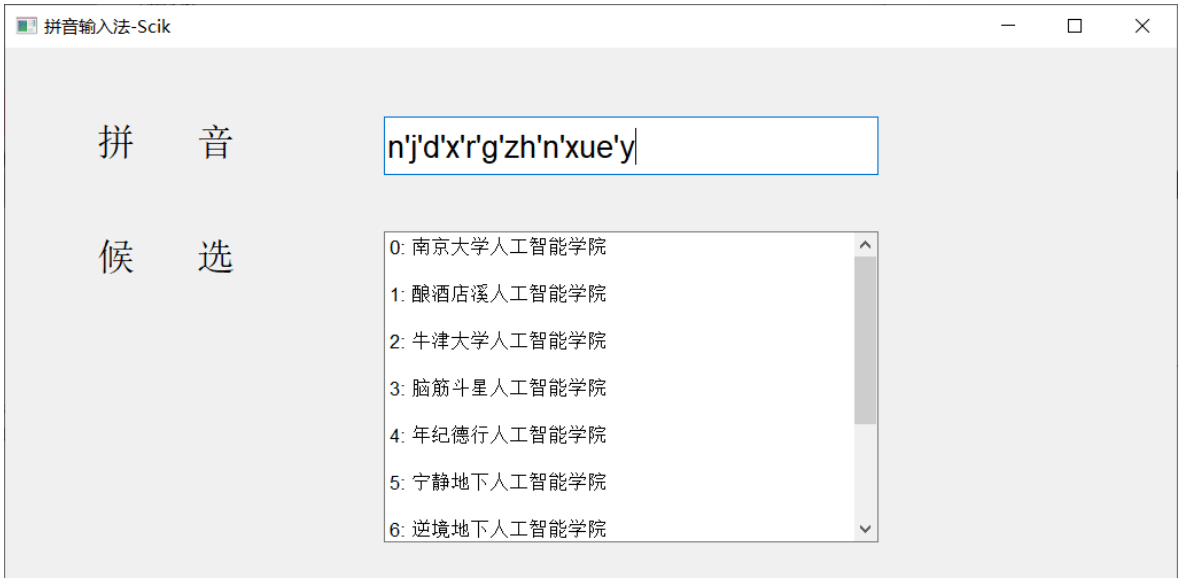
6.1 全拼效果



6.2 首字母效果



6.3 混合拼音效果



7 遇到的问题

`python`解释器找不到自建模块，最后解决方法是：

将`path\mypinyin.pth`文件放入目录`D:\Users\86176\anaconda3\Lib\site-packages`，文件内容是

```
D:\_WangKe\0\mypinyin\src
```

即将项目路径加入运行环境，使解释器能够找到自建模块。