·PinyinInput_HMM 是项目工程，通过 Pycharm 编译，源代码在 src 中，shurufa.py 是 GUI 运行文档。

一、实验目的

写一个基于 HMM 的拼音输入法，可以是汉语也可以是你知道的其他语言。

二、原理

汉语有很多字拼音相同。可以根据上下文来挑选概率大的字。如“wo”可对应汉字“我、窝、握、卧”等，“de”也可以对于“的、得、地、德”等。但是“wode”对应的概率比较大的应该属于“我的”。

我们把拼音作为观察值，把汉字当作状态，那么可以用 HMM 来建模拼音输入。转移概率可从语料库训练而得，生成概率则需考虑多音字（在不知道多音字概率的情况下可假设等概率）。

三、实验步骤

3.1 训练

我下载了 10 几本小说，作为扩充语料库，并下载了一个汉字-拼音文件。作为基本汉字的输入语料。

1. 将文本变成句子 src/train/get_sentence.py
   为了得到转移概率，那就要知道一个汉字后买面跟另外一个汉字的概率，为此，先从语料库中得到句子，这样，就能通过分析单个句子，知道一个汉字后跟另外一个汉字的概率。
   这里设计文件的读写问题。需要打开一个文件夹下的所有文件，然后一个个读取，最后写入一个文件中。
   句子的体提取是通过把汉字变为 unicode 码，然后根据汉字的编码范围（ u4e00-u9fff），到遇到非汉字即标点符号时，就截取成一个句子。

2. 根据一个基本的汉字库（带拼音），得到所有的汉字集合，拼音集合，以及每个拼音对应的所有汉字。src/train/get_hzpy.py
   这一步，通过读取一个汉字-拼音对照文件 hanzipignyin.txt(自己下载),
   得到所有的汉字集合 all-states.txt,
   得到所有的拼音集合 al-observations.txt
   得到一个拼音对应的所有汉字 pingyin2hanzi.txt

   这一步需要注意的是，要简化拼音，因为 hanzipignyin 中带有声调，所有要去掉。做法就是先将拼音小写化，然后去声调，因为汉语拼音的声调在一些固定的拼音上，所

以进行替换即可。

3. 得到三个矩阵的值 src/train/get_base.py

- 得到每一个汉字作为开头的次数 base_start.txt
  这是通过统计所有句子中出现在第一个的汉字得到的
- 得到一个汉字对应的各种拼音在语料库中出现的次数 base_emmission.txt
  通过 hanzipingyin.txt 文件，首先得到一个基本的统计，然后读取 sentence.txt 文件，将一个句子变为拼音（通过引入库文件 PininHelper 做到），然后更具汉字语拼音的对照关系，继续优化 base_emmisson.txt
- 得到一个汉字后面接的汉字以及对应的次数 base_transition.txt
  读取 sentence.txt 文件，按照如下形式错位统计，从而得到转移次数，即一个汉字后跟另外一个汉字的次数:

  输入句子：人生何处不相逢
  将其变为如下形式：
  ['人生何处不相', '生何处不相逢']
  人：生
  生：何
  ...

  相： 逢
  这样就能统计得到一个汉字后面跟的另外一个汉字以及它的次数

4. 将次数转变为概率 src/train/get_probability.py
   主要是处理 3 中得到的几个文件，将其中的次数变为概率。
   需要注意的是，计算概率时的基数问题，这几的基数就是统计所有的情况出现的所有次数。以处理 base_start.txt 为例
   将所有汉字的出现次数设置为 1，然后再加上已经统计出来的所有汉字出现在句首的次数，这样就构成了基数 count。 然后将数据分成两部分，在语料库中出现 data 和没有出现的 default. default 概率为;1.0/count。 data 的概率为 start[hanzi]/count.

3.2 测试

1. 编写 hmm 类

类接口如下:

```
    # coding: utf-8
class AbstractHmmParams(object):

    def start(self, state):
        """
         得到初始状态（汉字的概率）
        """
        pass

    def emission(self, state, observation):
```

```
        """
        得到汉字到拼音的概率
        """
        pass

    def transition(self, from_state, to_state):
        """
        得到上一个汉字到当前汉字转移概率
        """
        pass

    def get_states(self, observation):
        """
        得到当前观测值（拼音）的状态值（汉字）
        """
        pass
```

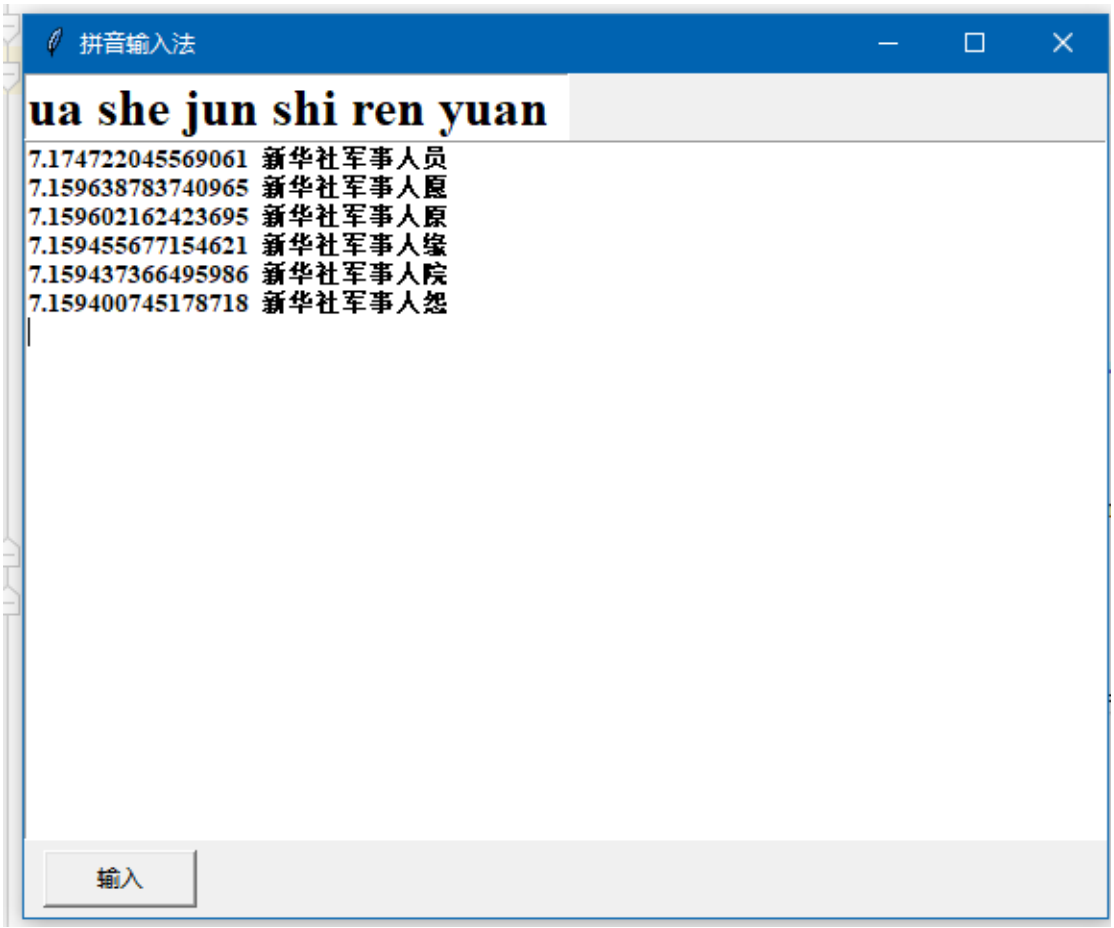
2. 实现 viterbi 算法　src/hmm/..

viterbi 算法是一个动态规划算法，这里需要注意的是每次得到前 n 个最优解，然后得到最优的结果。

3. UI 设计　src/shurufa.py

通过 tkinter 实现。

4. 结果展示


拼音输入法
ua she jun shi ren yuan
7.174722045569061 新华社军事人员
7.159638783740965 新华社军事人厦
7.159602162423695 新华社军事人原
7.159455677154621 新华社军事人缘
7.159437366495986 新华社军事人院
7.159400745178718 新华社军事人怨
输入


拼音输入法
ni hao ya ni shi
5.040508102980495 你好呀你是
5.03963977796016 你好鸦你是
5.039544666758206 你好押你是
5.039543272730293 你好压你是
5.039511346350063 你好垭你是
5.039511346350063 你好蓥你是

四、关于本实验

1. 编译器
   pycharm+python3.6
2. 花费时间大概 30h，主要用于
   - 查资料学习 hmm
   - 编写基本框架
   - 后期改正
3. 遇到的问题
   - 读写问题
     因为设计很多文件的读写，该开始特别不顺利。后来熟练了，就快了。
   - 汉字编码问题
     因为在训练阶段，我有将文本材料变成句子，为了得到争取的答案，在网上找到一种方法，就是将汉字编码变为 unicode 编码，这样，就能很好的解决。
   - 汉字变拼音问题
     在计算发射矩阵时，为了计算汉字对应的拼音，需要得到每个句子的拼音，这里用到 PininHelper 包
     汉字变拼音，导入了
   - viterbi 算法编写问题
     每次选出最优的一部分，而不是仅仅一个。
     这里引入了 heapq 包，采用堆排序的数据结构，按照 score 每次选出最优的 6 个解