# 拼音输入法 实验报告

计53 王润基 2015011279

## 使用说明及文件说明

- 要求的功能：输入拼音序列，输出中文

  打开终端，进入bin文件夹，输入：

  ```
  dotnet ConsoleApp.dll solve ../data/test/input.txt
  ../data/test/output.txt
  ```

- 在官方测试集上测试

  ```
  dotnet ConsoleApp.dll test ../data/test/official_test_in.txt
  ../data/test/official_test_out.txt -f pinyin_chinese -m 1 12l n
  ```

- 其它详见`README.md`

## 实现内容

基于统计语言模型（N-Gram）的拼音输入法

具体地：

- 基于字的一、二、三、高元NGram模型
- 使用SinaNews语料训练，规模10^8字

## 算法介绍

1. 标准N-Gram算法：

   根据词频统计信息，建立【前缀-末字】的频率索引。更严格地说，是【条件-概率分布】索引。

   例如：子串“清华大学”，索引为：dict["清华大"]={'学':0.9, '师':0.1}

   实际查询时，末字拼音也作为条件，即：dict["清华大"+"xue"]={'学':1.0}

   但实现时，出于结构复杂性和文件大小的考虑，内部词典并不根据末字拼音再索引，而是先取出所有字再过滤。

2. 多阶模型混合策略

   - 取有效结果中阶数最大的
   - 将所有有效结果线性混合，目前比例系数为$l = 0.75$。

     令$Dtb(Cond)$表示在$Cond$条件下的概率分布。

则

$$Dtb^*(\text{清华大学}+ji) = Dtb(\text{清华大学}+ji)*l + Dtb(\text{华大学}+ji)*l^2 + Dtb(\text{大学}+ji)*l^3 + L$$

3. 输入法算法：

   拿到一个新的拼音串时，从前往后依次计算前缀的前几个最优解，并以概率分布表示。

   例如：清华大学计算机系

```
dotnet ConsoleApp.dll qsolve
input > qing hua da xue ji suan ji xi
```

```
情(0.3151489)
清(0.1654558)
青(0.1244935)
亲(0.1060068)
请(0.1049221)

清华(0.7430037)
轻化(0.09429821)
青花(0.03987427)
庆华(0.03912253)
情画(0.01957632)

清华大(0.9982271)
清华达(0.0005829963)
庆华大(0.0004541953)
轻化大(0.0002195584)
亲华大(0.0001191592)

清华大学(0.9998255)
清华大雪(7.589365E-05)
庆华大学(6.54778E-05)
亲华大学(1.717826E-05)
清华大血(7.385515E-06)

清华大学期(0.2494571)
清华大学计(0.1964929)
清华大学及(0.1331228)
清华大学机(0.1266883)
清华大学技(0.09348144)

清华大学计算(0.9999789)
清华大学基酸(1.399212E-05)
清华大学期算(2.288649E-06)
清华大学及算(1.221337E-06)
清华大学机算(1.162305E-06)

清华大学计算机(0.9989204)
清华大学计算技(0.0007505873)
```

```
清华大学计算基(0.0001808138)
清华大学计算系(4.075291E-05)
清华大学计算计(3.229387E-05)

清华大学计算机系(0.99954)
清华大学计算机洗(0.0001301397)
清华大学计算机吸(6.763434E-05)
清华大学计算机西(6.564819E-05)
清华大学计算机息(4.775367E-05)

清华大学计算机系
```

可以观察到，后继状态的最优解极有可能是从前驱状态的前几名转移而来的。因此每个状态只需保留前几个最优解，去掉后面的解。（经对比测试，平衡时间和正确率，保留10个解）在保证质量的前提下降低了时空复杂度。

整个过程可用C#代码优雅地表示如下：

```
public override void Input(string pinyin)
{
    distribute = distribute.ExpandAndMerge(str =>
        Model.GetDistribute(new Condition(str, pinyin))
            .Take(10)
            .Select(c => str + c))
        .Take(10).Norm();
}
```

4. 文本统计算法：统计高频子串，动态子串长度

   为了追求模型阶数的提高，需要统计更长子串的频率。而只有那些高频短语才是有意义的。因此需要将统计策略从【固定子串长度，动态子串频率】转为【固定子串频率，动态子串长度】。

   这在理论上可以用Trie树（NFA自动机）实现：仅当一个节点的频率高于某个门槛值时，才扩展新的节点进行统计。

   为此我实现了一个简易Trie树。但实测发现并不比标准库的Dictionary效率更高。因此最终仍然用Dictionary实现统计。

# 效果展示

- 最好结果
  - 参数：
    - 使用SinaNews语料训练，保留最低频率1e-7
    - 高元模型
    - 线性混合策略（0.8）
    - 每步保留最优解10个
  - 具体结果见：`data/test/official_test_out_*.txt`
  - 效果好的：新闻腔，官方表述

```
> 全国人民代表大会在北京人民大会堂隆重召开
5 ＋＋＋＋＋＋＋＋＋＋＋＋＋＋＋＋＋＋＋＋＋＋＋
```

- 效果不好的：不知所云的

```
> 恰似那朵莲花不胜凉风的娇羞
5 卡斯纳多联化部省＋＋＋交休
```

- 效果不好的：专业术语多的

```
> 拼音输入法可以按注音符号与汉语拼音两种汉字拼音方案分成两大类
0 ＋＋＋＋＋＋＋＋属因复好于＋＋＋＋＋＋含自品引＋＋＋＋＋＋＋
```

- 模型间的差异：（详见：`结果分析.pdf`）

  官方测试集上（770+）

  - 语料子串统计的最低频率

| 最低频率 | 文件大小/MB | 字正确率 | 句正确率 |
| --- | --- | --- | --- |
| **1E-03** | 0.003 | 0.051 | 0.003 |
| **1E-04** | 0.031 | 0.480 | 0.034 |
| **1E-05** | 0.316 | 0.616 | 0.087 |
| **1E-06** | 3.7 | 0.681 | 0.147 |
| **1E-07** | 23.3 | 0.735 | 0.266 |
| **1E-08** | 63.2 | 0.753 | 0.304 |

  - 每步取最优解的个数

| 取最优解个数 | 测试用时/s | 字正确率 | 句正确率 |
| --- | --- | --- | --- |
| **2** | 14 | 0.642 | 0.146 |
| **5** | 32 | 0.706 | 0.223 |
| **10** | 53 | 0.735 | 0.266 |
| **20** | 108 | 0.744 | 0.282 |

  - 阶数

| 阶数 | 模型大小/MB | 字正确率 | 句正确率 |
| --- | --- | --- | --- |
| **1** | 0.1 | 0.485 | 0.014 |
| **2** | 7.8 | 0.711 | 0.216 |
| **3** | 21.1 | 0.734 | 0.263 |
| **n** | 34.8 | 0.735 | 0.266 |

# 分析和结论

- 数据方面

  由于训练数据使用了新浪新闻，因此新闻中常出现的官方腔调、政治名词的识别率很高。日常用语的识别率尚可。相对地，出现频率极低的其他领域专有名词基本全军覆没。

  结论：统计语言模型的效果和训练数据密切相关。

- 模型方面

  1. 一元模型就已经能蒙对一半左右的单字，说明我们日常用语中一半的字都是对应音中的最高频字。但由于没有利用任何上文信息，其连贯正确性非常差，整句正确几乎没有。
  2. 二元模型相比一元模型有了质的飞跃。
  3. 三元模型相比二元模型又有了明显提高。
  4. 两种混合策略没有表现出明显差异，仅在个别结果中略有不同
  5. 高阶模型相比三元模型没有什么区别。

     推测原因是：对于高频词而言，三元模型已经足够使其在概率中脱颖而出；对于低频词，高阶模型又没有提供更多的统计信息。

# 一些思考

1. 关于基于词的模型
   - 人在说话的过程中是以词为单元进行思考的。
   - 当一些词（字）以很高频率连续出现时，可以认为出现了新的词。
   - 目前主流的做法是提供词库，但并不包含统计信息。如何把它们融入到基于概率的模型中?

# 改进方案/TODO

- 算法方面：
  - 使用高效的Trie树进行文本统计
  - 自动词抓取：

    若对于子串A，其前后都没有高频字，就认为A成词
  - 支持以字母为单元的输入方式（目前是以拼音为单元）
- 模型方面：
  - 基于词的NGram模型

    - HMM等高级模型
- 功能方面：
    - 基于命令行/GUI的实时交互式输入（真正的输入法）

      目前Inputer类已经设计好了类接口，只需提供UI