拼音输入法实验报告

李晨昊 2017011466

2019-4-13

目录

1 项目简介

本项目采用 rust 语言编写，只需要 rust 配套的工具链即可编译运行 (`cargo run --release`)。我**没有**按照实验指导书中说明的一样来组织文件格式，而是按照通常的 rust 项目所遵守的惯例。

我使用了一些第三方的库，简要分类如下：

1. 编码相关：因为给出的输入文件和训练文件是 gbk 编码的，而 rust 要求所有的字符串都以 utf-8 编码
2. 数据读入/持久化相关：读入 json 格式的训练数据，将训练结果以 bincode 格式保存
3. 并发相关：为了加快训练/测试速度，使用多线程训练/测试
4. 命令行界面相关：支持用命令行参数指定输入文件，输出文件，训练结果文件，训练文件夹，拼音文件；实现了一个简单的 REPL 界面

训练结果 `cache.bin` 已经保存在项目根目录下，如果希望重新训练的话，可以删除它再重新运行一次 (这就需要指定训练文件夹了，因为它太大了，并没有附在项目根目录下)。得益于多线程的加速，在我的计算机上使用提供的 `sina_news` 训练，大概 12s 即可完成训练。

具体使用方法可以执行 `cargo run --release -- -h` 查看。注意用 cargo 传递命令行参数给应用的时候需要额外加一个`--`，因此如果希望从 `in.txt` 读入，输出到 `out.txt`，应该输入 `cargo run --release -- in.txt -o out.txt`。

2 算法分析

我基本上只使用了基本的的二元模型。在此基础上还进行了一点很小的修正：我统计了每个字后面跟着 stop word 的情况，即标点符号和 “的”，“是” 之类的虚词，利用这个信息，对于每个拼音输入的最后一个字，如果它之后接 stop word 的频率较大，则给它一定的加成。(这个修正是来源于输入样例 “jin tian hui jia bi jiao wan”，如果不采用这个修正，则会识别成 “今天回家比较完”，因为 “比较完善” 这个词组出现的次数比 “比较晚” 多许多，按照现有的模型，“完” 的确是一个更好的选择。加入了这个修正后，因为 “完” 很少单独出现在句尾，这样就能正确地选择出 “晚”)

算法的流程如下：

1. 读入训练数据，用散列表记录每个汉字和每个相邻的汉字对出现的次数信息，还统计每个汉字后解 stop word 的频率信息。三者分别记为 u_map，b_map，$last\_correction$
   - 为了优化速度，下面的一些取对数等操作实际上已经在这一步做了
2. 读入拼音信息，用散列表记录每个拼音对应的所有汉字可能，记为 py
3. 读入输入数据，对每个拼音句子 S，维护一个 dp 数组和 ans 数组，其中

$$len(dp) = len(ans) = len(S)$$

$$len(dp[i]) = len(ans[i]) = len(py[S[i]]),\ for\ i\ in\ \{1, ..., len(S)\}$$

$$dp[i][j] = \max_{k=1..len(py[S[i-1]])}(dp[i-1][k] + \ln(\frac{b\_\_map[(py[S[i-1]][k], py[S[i]][j])]}{u\_\_map[py[S[i-1]][k]]}))$$

$$ans[i][j] = argmax_{k=1..len(py[S[i-1]])}(...)$$

$$for\ i\ in\ \{2, ..., len(S)\},\ j\ in\ \{1, ..., len(py[S[i]])\}$$

$$dp[1][j] = \ln(u\_map[py[S[1]][j]]),\ for\ j\ in\ \{1, ..., len(py[S[1]])\}$$

使用动态规划，可以在 $O(\Sigma_{i=1}^{n-1} len(py[S[i]])len(py[S[i+1]]))$ 时间内计算出这两个数组

4. 在动态规划结束后，为最后一行的 dp 数组加上 $last\_correction$

$$dp[len(S)][j] = dp[len(S)][j] + last\_correction[py[S[len(S)]][j]]$$

$$for\ j\ in\ \{1, ..., len(py[S[len(S)]])\}$$

5. 找出 dp 数组中最后一行的最大值，并依照 ans 数组不断向前查找上一个最佳答案位置，组成答案字符串。

3 举例分析

3.1 好的例子

1. `shang hai shi ren min jian cha yuan di yi fen yuan zai qi su shu zhong zhi kong`

- 上海市人民检察院第一分院在起诉书中指控

2. `wo men yao jin mi tuan jie zai yi xi jin ping tong zhi wei he xin de dang zhong yang zhou wei`

- 我们要紧密团结在以习近平同志为核心的党中央周围

3. `shen ru guan che luo shi xi jin ping zong shu ji shi cha bei jing zhong yao jiang hua jing shen he jing jin ji xie tong fa zhan gui hua gang yao`

- 深入贯彻落实习近平总书记视察北京重要讲话精神和京津冀协同发展规划纲要

4. `que bao dang tuan jie dai ling ren min bu duan kai chuang zhong guo te se she hui zhu yi shi ye xin ju mian`

- 确保党团结带领人民不断开创中国特色社会主义事业新局面

5. `mo su er ji qi zhou bian di qu li shi shang yi zhi shi xun ni pai a la bo ren de tian xia`

- 摩苏尔及其周边地区历史上一直是逊尼派阿拉伯人的天下

6. `gao du fa da de ke xue ji shu he jing ji shui ping ba ren men dai ru le xin de bu tong yu yi wang de sheng huo huan jing zhong`

- 高度发达的科学技术和经济水平把人们带入了新的不同于以往的生活环境中

3.2 不好的例子

1. `huang pu qu shi chang jian guan bu men zai yan ge zhi fa de tong shi bing mei you yi dao qie`

- 正确：黄浦区市场监管部门在严格执法的同时并没有一刀切
- 预测：黄埔区市场监管部门在严格执法的同时并没有一刀切
- 分析：显然光靠这句话本身输入法没办法判定这里的 huang pu qu 是上海市的黄浦区还是广州市的黄埔区，而在训练数据中的确是后者的出现频率更大 (虽然按照我的生活经验，其实是前者使用更多)

2. `qi zhong zui zhu yao de yuan yin qi shi ye wu fei jiu shi zai wo men shen bian ji shao kan dao zhe zhong`

- 正确：其中最主要的原因其实也无非就是在我们的身边极少看到这种
- 预测：其中最主要的原因其实业务费就是在我们的身边际少看到这种
- 分析：因为业务费和边际这两个词比较常见，输入法没能看出这两组词可以分开；我认为这可能需要基于词的识别才能做到

3. `jia shang yi xie jian ce ji gou shi fou zai jian ce ji zhu shang neng gou ti gong xiang ying zhi chi dou shi ge wen ti`

- 正确：加上一些检测机构是否在检测技术上能够提供相应支持都是个问题
- 预测：加上一些监测机构是否在监测技术上能够提供相应支持都是个问题
- 分析：检测和监测读音相同 (不考虑音调)，意思接近，而前者的频率更高，那么可以预料几乎所有的监测都会被识别成检测

4. `you shi hai hui fang da`

- 正确：有时还会放大

- 预测：优势还会放大
- 分析：这句话非常依赖于前文的语境，只单独看这句话的话输入法的识别结果也是完全合理的

4 测试结果

我将提供的新闻中 11 月的新闻作为测试数据 (它不作为训练数据)，按照标点符号划分句子，舍弃长度小于 5 的句子，得到了如下的结果

```
# 注：这是开发中的某个版本，现在这样运行并不会执行测试，而是会启动REPL
$ time cargo run --release
TestRes { n_ch: 4181885, n_sen: 351693,
  ch_acc: 0.8759435039461869, sen_acc: 0.4314359398680099 }

real    0m33.517s
user    8m34.982s
sys     0m0.272s
```

简要说明如下：

1. 关于正确率

- 如果要求每个句子的长度更长，整句的正确率的正确率显然会下降，但是这已经完全足够日常使用的长度。经过我自己的观察，我在书写这篇报告的时候绝大多数输入都在 1-4 个字符之内
  - 如果舍弃长度小于 10 的句子，则整句的正确率在 35% 左右
  - 如果不舍弃任何句子 (当然，没有空句子)，则整句的正确率在 51% 左右
- 字正确率基本上都稳定在 88% 左右。事实上，如果舍弃长度小于 10 的句子，字正确率反而比现在高一些
- 个人认为这个正确率还是相对不错的，这可能是因为输入数据 (11 月的新闻) 与训练数据 (其它的新闻) 主题和句式都非常相似，如果换成其它的输入则准确率很有可能无法保证
- 因为我的数据生成方式为将多音字直接对应为它的任一个拼音，所以以上正确率可能还有一定偏差 (应该是偏低)
  - 例如，对于 “听取了其委托的辩护人的意见” 一句，我生成的输入是 “ting qu le ji wei tuo de bian hu ren de yi xian”，导致输入法将 “其委托的” 识别成了 “纪委托的”；如果我正确地输入 “ting qu le qi wei tuo de bian hu ren de yi xian”，我的输入法是可以正确识别的

2. 关于性能

- 上面的测试结果是在我的平台 (R7-2700) 上得到的，在其它平台上肯定会有一些出入
- 得益于 rust 优秀的语言模型，利用它来进行并行计算非常容易。上面的结果中 `user` 耗时大约为 `real` 耗时的 16 倍，符合 R7-2700 的参数 (8C16T)
- 占用内存峰值大约在 150MB 左右

5 参数调整

如果去掉关于 stop word 的修正，其它条件不变，则结果为

```
TestRes { n_ch: 4181885, n_sen: 351693,
  ch_acc: 0.8752474063729634, sen_acc: 0.4285328397210067 }
```

其实差距很小。显然，这个方法基本上只是一个为了解决样例的 ad hoc 的方法，很难期望它对于其它的句子有同样的适用性。我目前还是保留着这个修正，毕竟它还是有一定的正面作用的。

这个结果说明了利用这种人为的特征提取实现更高级的人工智能的困难之处，也许它对某一个特定的输入比较有用，但是对于其它的输入则并没有什么意义。为了达到更好的效果，还是需要更加系统化，更有理论依据的方法。

6 总结

我并没有实现太复杂的算法，大部分的精力还是放在怎么样优化目前的算法，让它的识别率和速度更加优秀，尤其是速度。虽然速度并不是要求的考察标准，但是我仍然希望能尽可能优化它。毕竟纵观人工智能的历史，现在常用的大多数算法基本都是很久以前就有的想法，在此基础上这些年来的改进方向无非就是优化效果和时间常数。虽然一个新颖的算法的价值很大，但是如果不讨论常数，它也很难在生活中或者工业界发挥作用。

如果希望进一步追求识别率，我还是认为直接一步到位使用神经网络比较合适，这些二元/三元模型的近似程度都太过于粗浅了，而且随着模型复杂程度的加大，现代 CPU 这孱弱的计算能力未必就能把传统模型算的比其它硬件加速的神经网络快。