拼音输入法：实验报告

刘泓尊　2018011446　计 84

目录

本项目完整文件与模型已经上传到清华云盘 https://cloud.tsinghua.edu.cn/d/e9d7a28fd1174a1689da/，总大小约 5GB(其中语料集 2.1GB)。如果您想跳过约 30min 的模型构建时间，请从云盘上下载训练好的模型。运行/bin 下的 pinyin.py 即可开始测试。具体操作与功能见本报告。请不要更改文件路径，以免错误；**如果您想重新构建训练模型，请先运行“clean.py”**将 json 文件转化为纯文本，之后使用**“python pinyin.py -p”**实现预训练，训练结束后，您可以使用**“python pinyin.py inputfile outputfile [answerfile]”**进行测试。如:

“python pinyin.py ../input/input.txt ../output/output.txt ../answer/answer.txt”

1 原理简介

我分别实现了 2-Gram 和 2-Gram,3-Gram 结合的方式进行，基于 Viterbi 算法进行时间复杂度的优化。

1.1 问题建模

基于统计知识，由一个拼音串转化为汉字串的模型可以抽象为求解使得

$$P(S) = P(w_0 w_1 \cdots w_n) = \prod_{i=1}^{n} P(w_i \mid w_0 w_1 \cdots w_{i-1})$$

最大的候选句子序列 $w_0 w_1 \cdots w_n$。根据大数定律，条件概率 $P(w_i \mid w_0 w_1 \cdots w_{i-1})$ 可以使用频率近似代替，即

$$P(w_i \mid w_0 w_1 \cdots w_{i-1}) = \frac{Count(w_0 w_1 \cdots w_{i-1} w_i)}{Count(w_0 w_1 \cdots w_{i-1})}$$

上述模型是隐马尔可夫模型 (HMM)。

1.2 2-gram 模型

2-gram 模型假设句子中某个单词的出现取决于其前面一个单词，对应于概率模型中的

$$P(S) = \prod_{i=1}^{n} P(w_i \mid w_{i-1}) = \prod_{i=1}^{n} \frac{Count(w_{i-1} w_i)}{Count(w_{i-1})}$$

例如对于句子“<s> I am Sam </s>”，上述模型为
P(”<s> I am Sam </s>”) = P(i | <s>)P(am |i)P(Sam |am)P(</s> |Sam)

1.3 Backoff and Interpolation

考虑到可能存在 $Count(w_{i-1} w_i) = 0$ 的情况，会产生概率为 0 的情形。将上式修正为:

$$P(w_i \mid w_{i-1}) = \alpha \cdot \frac{Count(w_i)}{\sum_i Count(w_i)} + (1-\alpha) \cdot \frac{Count(w_{i-1} w_i)}{Count(w_{i-1})}$$

有论文称之为“Backoff and Interpolation”(退化与插值) 方法。经过多次权衡，我取 $\alpha = 0.05$.

1.4 3-gram 模型

3-gram 模型假设句子中下一个单词的数先取决于前面两个单词。即

$$P(S) = \prod_{i=2}^{n} P(w_i \mid w_{i-2} w_{i-1}) = \prod_{i=2}^{n} \frac{Count(w_{i-2} w_{i-1} w_i)}{Count(w_{i-2} w_{i-1})}$$

为了防止状态转移过程中出现概率为零，将上式修正为:

$$P(w_i \mid w_{i-2}w_{i-1}) = \alpha \cdot \frac{Count(w_i)}{\sum_i Count(w_i)} + \beta \cdot \frac{Count(w_{i-1}w_i)}{Count(w_{i-1})} + \gamma \cdot \frac{Count(w_{i-2}w_{i-1}w_i)}{Count(w_{i-2}w_{i-1})}$$

其中 $\alpha + \beta + \gamma = 1$. 经过多次测试，最终取 $\alpha = 0.05, \beta = 0.85, \gamma = 0.10$.

1.5 Viterbi 算法

Viterbi 算法是求解 HMM 的有效优化算法，该算法基于动态规划思想。记 dp[t][i] 为第 t 个字取候选汉字序列第 i 个字的概率。那么有

$$dp[t][i] = \arg \max_{j \in words(t-1)} (dp[t-1][j] \times P(w_i \mid w_j))$$

根据此思想，设每个拼音的候选汉字最大为 T，计算第 t 层 T 个节点概率的复杂度为 $O(T^2)$, 那么求解整个句子 $(length(S) = n)$ 的时间复杂度为 $O(nT^2)$

1.6 Viterbi 算法扩展到 3-gram

3-gram 模型的每个状态和前面两层有关，所以可以增加一个维度信息，记 dp[t][j][i] 为第 t 个位置的汉字在上一个字为 j 的条件下，取 i 的概率。那么 Viterbi 算法可以修正为:

$$dp[t][j][i] = \arg \max_{k \in words(t-2)} (dp[t-1][k][j] \times P(w_i \mid w_k w_j))$$

由于综合考虑了前面两步的信息，所以该算法的复杂度提升到了 $O(nT^3)$. 但是因为 T 是固定的，对于每个拼音，其对应的汉字一般只有 10-20 个，所以该算法可以看做是 $O(n)$ 的.

为了节省模型空间，我只保留了出现频率前 10^6 的三元组 (w_k, w_j, w_i). 而不是一个三维矩阵。为了可以使用矩阵存储，我将字按字频进行排序并编号，字频统计结果放在./sina_news_vocab/vocab.txt 中。

1.7 一些优化

1.7.1 加入句子起始符 <bos> 与结束符 <eos>

为了保证句子的完整性，准确预测具体开头与结尾的汉字，我引入了起始符 <bos> 和结束符 <eos>. 并在统计过程中，以标点符号作为句子起始结束的标志，进行 2-gram 和 3-gram 的统计。在跑 Viterbi 算法时，计入 <bos> 和 <eos> 的影响。经过测试，该改进有效提升了句子第一个字和最后一个字的准确率。如将“禁用的武侠小说非常精彩”改进为“金庸的武侠小说非常精彩”。

1.7.2 Beam Search

朴素的 Viterbi 算法状态数非常多，3-gram 模型高达 $O(nT^2)$. 为了改进时间复杂度，加快搜索速度，我使用 Beam Search 进行优化。Beam Search Decoding 的思路是: 在第 t 层不保存所有的候选节点，而是只保存概率最大的前 s 个节点，剩余的节点不被传播到下一层。如果只保留概率前 $T/2$ 个节点，那么 2-gram 的时间复杂度可以减小为 $O(nT^2/4)$.

1.7.3 Unknown Words

对于不在“一二级单词表”中的汉字，我将其标记为 <unk>，以提高算法的普适性。该改进对算法的准确率提升并无作用，但是却保留了句子的完整性。

1.8 其他尝试

1.8.1 Laplace 平滑

我还尝试了其他的平滑化方法，比如 Laplace 平滑，该平滑使用

$$P_{Laplace}(w_i \mid w_{i-1}) = \frac{Count(w_{i-1}w_i)+1}{Count(w_{i-1})+V}$$

其中 V 为词汇表的大小。但是测试发现效果不如 Backoff and Interpolation 平滑，我最终没有采用此方法。

1.8.2 语料库的扩展

为了观察语料库对模型的影响，我爬取了一部分微博语料作为训练集。经过测试发现，加入微博语料之后，口语测试集的准确率有了明显上升 (约 5% 的改进)，但是书面语测试集的准确率有所下降。我查阅了相关文献，发现在基于统计的框架下，这两个领域暂时无法“优雅地”兼顾。因此，作为一种取舍，最终我使用新闻数据集，保证了书面语的准确率最高。

1.8.3 Interpolation(插值法) 参数的确定

我使用了可以有效确定 α, β, γ 的比例的方法，称为 Deleted Interpolation Algorithm(删除插值法). 该方法可以通过统计词频来调整三个参数的比例。具体流程如下:

Algorithm 1: Deleted Interpolation Algorithm

Input: *corpus*;

Output: α, β, γ;

$\alpha = 0, \beta = 0, \gamma = 0$;

for *each 3-gram* w_1, w_2, w_3 *with* $Count(w_1w_2w_3) > 0$ **do**

  depending on the maximum of the following 3 values:

  case $\frac{Count(w_1w_2w_3)-1}{Count(w_1w_2)-1}$: increment γ by $Count(w_1w_2w_3)$

  case $\frac{Count(w_2w_3)-1}{Count(w_2)-1}$: increment β by $Count(w_1w_2w_3)$

  case $\frac{Count(w_3)-1}{N-1}$: increment α by $Count(w_1w_2w_3)$

Normalize α, β, γ

return α, β, γ

2 具体功能与使用流程

运行过程包括 (1). 数据预处理, (2). 模型训练和 (3). 测试三个阶段。在预处理和训练过程中会新建文件作为存档，在测试阶段会加载这些文件，用于计算输出结果。具体流程为:

2.1 语料预处理

由于给出的数据时 `json` 格式，并且带有很多冗余信息，所以我先将语料文件转换为“纯文本”，以便后续训练过程。该过程将字典中`"html"` 项的内容抽取出来，并替换其中的“标点符号”为句子的开始符 `E` 和结束符 `B`。例如: “清华大学计算机系。”→“`E` 清华大学计算机系 `B`”

您可以在`./sina_news_gbk/`下执行 `python clean.py` 进行上述过程。新得到的“纯文本”文件命名为 `2016-xx_con.txt`,仍然放`./sina_news_gbk/`下。为了识别标点符号,我使用了 `zhon.hanzi` 库下的 `punctuation` 变量。

2.2 模型训练

我分别实现了 2-gram 模型和 2-gram 与 3-gram 结合的方式。所以预处理过程主要是统计 2-gram 和 3-gram 的频率。在本目录下运行 python pinyin.py -p 可以开始模型构建过程。注意，对应的语料 2016-xx_con.txt 需要放在./sina_news_gbk/下。对于课程提供的语料库，构建过程约为 30min，最坏情况下占用 5GB 内存资源.

2.3 模型测试

模型测试需要使用 python pinyin.py inputfile outputfile [answerfile] 格式，其中 inputfile 是拼音文件，每行一句拼音。转换结果将输出到 outputfile。如果您需要验证正确率，请将标准答案放在 answerfile 下，并在命令中加入此参数，转换完成后将自动开始正确性的检验，分别输出字正确率和句正确率。

2.4 从 stdin 输入测例

您也可以从 stdin 输入合法的拼音句子，得到对应的输出。请使用 python pinyin.py -i 开启该功能。注意，您同样需要在./sina_news_gbk/下准备好 2016-xx_con.txt 语料。

2.5 文言文模型

我本来想利用给定的语料实现对文言文和诗歌的翻译功能，但是查阅文献后得知，古文和现代汉语是两个不同的语言体系，其语言特征差距较大，很多时候不能达到“鱼和熊掌兼得”的效果。因此，我单独实现了文言文拼音翻译模块。我爬取了古诗文网下的几万首古诗文，经过同样的预处理过程得到了古文语料 poem.txt. 之后基于此语料进行训练，模型保存在./vocab_poem 文件夹下。您可以使用此模块进行对文言文的测试，效果很好。

2.6 查看帮助

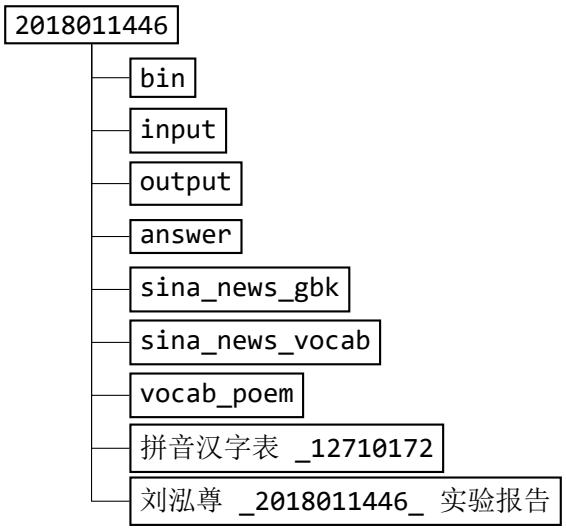
为了帮助使用者尽快掌握使用流程，您可以使用 python pinyin.py -h 查看 help 信息。具体介绍为:

help

```
$ python pinyin.py -h

If you HAVE NOT preprocessed data, please run:
    python pinyin.py -p

Attention:
This Preprocess will takes for about 30 mins and 5GB Runtime memory!!!

If you HAVE preprocessed data, please run:
    python pinyin.py inputfile outputfile [answerfile]

A simple demo is:
    python pinyin.py ./input.txt ./output.txt ./answer.txt

If you want to run from stdin, please input
```

```
    python pinyin.py -i

If you want to run from stdin which translating [poems and classical chinese], please input
    python pinyin.py -c
```

文件结构



3 准确率统计

二元模型： 正确例子

- 请大家选择你觉得可以的时间
- 美军方称不承认中国东海防空识别区
- 特朗普希望不久和中国国家主席面对面会晤

二三元结合： 新增正确例子

- 金庸的武侠小说非常精彩
- 智能技术与系统国家重点实验室
- 对染色体人工合成的工作给予了高度评价
- 你的世界会变得更精彩

二三元结合： 不好的例子

- 他养了**一致青瓦**当宠物
- **他**是我的母亲
- 今天回家比较**完**
- 中国**诗人**人民民主专政的社会主义国家

古文模型： 好的例子

- 举头望明月低头思故乡
- 君不见黄河之水天上来奔流到海不复回

错例分析 第 1 个错误和第 3 个准确体现了“语料特征”，因为语料是新闻语料，“青瓦”的概率远大于“青蛙”的概率，而“比较完”的概率也明显多于口语化的“比较晚”；第 2 个错误在基于统计的模型下难以解决，因为“他”的出现概率高于“她”，实际上，搜狗输入法面对多个“ta”的问题，也是给出了若干候选，不能根本上保证概率最高的结果就是正确的；第 3 个错误则体现了 2-gram 对模型的过度影响，模型对汉字的选择明显偏向于“二元词”而非“单字词”。

准确率 我统计了不同模型和参数的准确率，效果如下：

模型	二元模型	三元模型	二三元结合
口语集：字准确率	0.9259	0.6011	**0.9264**
口语集：句准确率	0.5661	0.4221	0.5666
书面语集：字准确率	0.9608	0.7142	**0.9748**
书面语集：句准确率	0.7678	0.5102	**0.8170**

二元模型下参数 α 的选择 (书面语集)：

alpha	0.2	0.1	0.05	0.01	0.001
字正确率	0.9412	0.9543	**0.9608**	0.9601	0.9532
句正确率	0.7518	0.7614	**0.7678**	0.7678	0.7564

注： 口语集使用微博语料，书面语集使用历年政府工作报告。对应的 answerfile 使用 pypinyin 进行转换。规模约为 10W 个汉字，1W 个句子。可以看到，二三元结合的准确率最高，仅使用三元模型的准确率最低。书面语测试集准确率明显高于口语测试集。二三元结合的方式甚至可以与搜狗输入法媲美。

4 改进思路

尽管二三元结合的模型已经达到了极高的准确率，甚至可以与搜狗输入法媲美，但是我还是思考了若干改进思路。我希望将来有时间实现与验证这些改进。

4.1 Bidirectionality

原始的算法仅仅考虑到了下一个单词取决于上文的汉字，却没有考虑到下文的汉字。而汉语体系中，一个汉字是基于上下文 (Context-Related) 的。所以我认为可以考虑双向的模型，综合“过去”与“未来”的信息。

最简单的思路是，分别统计“从左到右”和“从右到左”两个方向的 2-gram 和 3-gram。使用 Viterbi 算法分别从两个方向计算句子概率，选择概率最大的那个结果。

经过查阅文献，另一个著名且有效的算法是 Conditional Random Field(条件随机场)。CRF 通过构建非有向图模型，将“未来”的句子信息引入当前位置汉字概率的计算。但是该模型的时间复杂度较高，计算过程比 HMM 缓慢。

4.2 语料库的适当扩充

经过实验发现，模型的输出结果和语料特征强相关。由于本模型主要采用新闻语料进行训练，对于口语测试集的准确率明显不如书面语测试集；针对古文的训练也可以看到这个现象。可以采取多个类型的文本，使得模型适用于多种不同场合。

4.3 尝试更高元模型

由于汉语中二元词频率占了极大部分，所以本模型使用二三元结合的方式已经可以胜任正常需求。为了进一步提高准确率，可以考虑 4-gram 模型或者更多元的模型。这样做可能会提高对一些俗语和成语的准确率。

4.4 基于“词”的 n-Gram 模型

可以考虑建立词典，使用分词算法保存出现的多字词，以“词”为单位进行训练和计算。

4.5 基于 Sequence Processing Networks

从拼音到汉字的转换问题是一个典型的“序列到序列”的问题。因此，可以使用处理序列的 RNN，LSTM，GRU 等神经网络模型得到语言模型。比如 bi-LSTM 可以很好的实现序列翻译与 Context-Related 的功能。

5 总结

这是我第一次接触人工智能算法，我收获良多。我实践了 NLP 领域十分著名的 N-Gram 模型，了解了马尔科夫过程在人工智能领域的广泛应用，领略了 Viterbi 算法的明显优势，最终达到了 90% 以上的准确率。

同时，我阅读了大量论文，了解了如何使用 Beam Search 对 Viterbi 算法进行加速，了解了“条件随机场”在马尔科夫模型中的应用，并使用 Deleted Interpolation Algorithm 来确定参数比例，学习了 Backoff and Interpolation 的平滑化方法。同时，我了解了 Sequence Processing Networks 的前沿知识，这些知识都对我的实现有很大启发，也是对我科研之路的一个启蒙。

写完本项目后，我便开始关注我使用的搜狗输入法。在我写本报告的过程中，搜狗输入法用的算是比较顺手，但是也难免出现预测错误的情况，尤其是对于“专有名词”。但不得不说，拼音输入法确实给人带来了巨大的便利，我也期待着更多的改进！

感谢马老师和助教的悉心指导！