拼音输入法：实验报告

陈鑫圣　计算机科学与技术系 计13　2021010760

算法简介

本次实验采用了基于字的二元模型和三元模型。

问题描述

本实验需要解决的问题是，给定汉字拼音序列 $S = s_1 s_2 \cdots s_n$，求符合该拼音序列的最佳汉字序列 $W = w_1 w_2 \cdots w_n$。

模型建立

在拼音序列为 S 的前提下，汉字序列为 W 的概率为 $P(W|S) = \dfrac{P(W)P(S|W)}{P(S)}$。因为对于不同的 W 而言，$P(S)$ 是常数，而 $P(S|W) \approx 1$，所以 $P(W|S) \approx P(S) = \prod_{i=1}^{n} P(w_i|w_1 \cdots w_{i-1})$

因此，任务即找到 $W = w_1 w_2 \cdots w_n$ 使得 $\prod_{i=1}^{n} P(w_i|w_1 \cdots w_{i-1})$ 最大，也就是使得 $\sum_{i=1}^{n} \ln P(w_i|w_1 \cdots w_{i-1})$ 最大。

进一步地，再将上式简化为 $\sum_{i=1}^{n} \ln P(w_i|w_{i-k} \cdots w_{i-1})$（当 $i \le k$ 时，取 $w_1 \cdots w_{i-1}$，即每个字只考虑其前面的 k 个字。本实验尝试了 $k = 2, 3$，即二元模型和三元模型。

式子中的概率，基于语料库中相应字词的频率进行估计。

Viterbi 算法

在二元模型中，上式求最大值问题可以转化为一个多层图上从起点层到终点层的最长路径问题，采用 Viterbi 动态规划算法解决此问题。

具体而言，第 i 层的第 j 个节点为 $w_{i,j}$ 为拼音 s_i 对应的第 j 个可能的汉字，而 $w_{i,j}$ 与 $w_{i+1,k}$ 之间的边长定义为：在前一个字是 $w_{i,j}$的 前提下，后一个字是 $w_{i+1,k}$ 的概率。

在三元模型中，我对上述的式子最大化问题做了进一步简化，从而在三元情况下依然使用 Viterbi 动态规划算法解决。

具体而言，在计算到第 i 层结束时，该层的每个节点 $w_{i,j}$ 都记录了从起点层开始到该节点的最长路径的前驱，将这一操作拓展为，记录 `MAX_PREV` 个最优前驱。

于是，$w_{i,j}$ 与 $w_{i+1,k}$ 之间的边长如下定义：对于 $w_{i,j}$ 的每个最优前驱 $w_{i-1,l}$，计算在前两个字是 $w_{i-1,l}$ 和 $w_{i,j}$的前提下，后一个字是 $w_{i+1,k}$ 的概率。所得的这 `MAX_PREV` 个概率取最大值。如此简化后，使得三元模型的时间复杂度相对于二元模型没有太大增加。

实现细节

平滑处理

在实现中，为了避免某个字在词频表中未出现，而导致概率为 0 的极端情况，进行了以下平滑处理。

单字出现的概率依然是

$$P(w_i) = \frac{\text{count}(w_i)}{total}$$

其中 $\text{count}(w_i)$ 表示 w_i 在语料库中出现的概率，$total$ 表示语料库中所有单字出现的概率。

连续二元组出现的概率平滑为

$$P(w_i|w_{i-1}) = x \cdot \frac{\text{count}(w_{i-1}w_i)}{\text{count}(w_{i-1})} + (1-x)\frac{\text{count}(w_i)}{total}$$

连续三元组出现的概率平滑为

$$P(w_i|w_{i-2}w_{i-1}) = x \cdot \frac{\text{count}(w_{i-2}w_{i-1}w_i)}{\text{count}(w_{i-2}w_{i-1})} + (1-x)\left(y \cdot \frac{\text{count}(w_{i-1}w_i)}{\text{count}(w_{i-1})} + (1-y) \cdot \frac{\text{count}(w_i)}{total}\right)$$

系数 x, y 的值见“实验效果”部分。

句首处理

上述模型中，句子的第一个字只能通过该字在词频表中的频率计算，在三元模型中的第二个字也只能通过前两个字的信息来计算。这可能导致句首误差较大。

为弥补这一点，在生成词频表时专门生成了句首词频表。在二元模型中，第一个字使用句首词频表计算上述概率；在三元模型中，第一、二个字使用句首词频表计算上述概率。

实验环境

实验硬件环境为

- AMD Ryzen 7 5800H with Radeon Graphics (3.20 GHz)
- 64 位操作系统, 基于 x64 的处理器

实验使用的系统为 Ubuntu：

- Distributor ID: Ubuntu
- Description: Ubuntu 20.04 LTS
- Release: 20.04
- Codename: focal

实验使用的编程语言为 Python ， 主要使用的库有 pathlib, argparse, tqdm, collections, Numpy, json, numpy, matplotlib 等。

语料库和数据预处理方法

本实验采用了下发的新浪新闻语料库。在拓展部分，还使用了 GitHub 项目 nlp_chinese_corpus （https://github.com/brightmart/nlp_chinese_corpus） 中的 baike2018qa 语料库。语料库和 README 的下载地址：https://cloud.tsinghua.edu.cn/d/df510cc2fba34aaf8ad9/。

数据预处理由 `data_preprocess.py` 完成。该程序读入原始数据，并进行数据提取，然后生成词频表。此外，该文件也完成了拼音汉字表的预处理，把拼音汉字表转化为 json 文件。

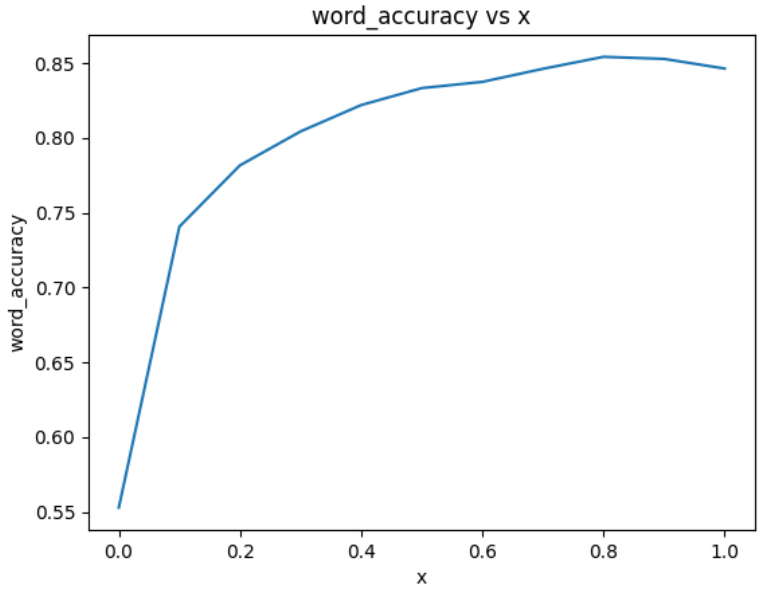
数据预处理时，首先将文件中相应的键对应的值（最为一个“句子”）读入，忽略所有长度为 1 的句子。选取所有单字、相邻的两字、相邻的三字，都只保留全由汉字组成的情形。然后根据一二级汉字表，将其中不全由一二级汉字组成的串剔除掉。为避免中间文件过大，句首一元、二元词频只保留前 4000、2000000 个，全文一元、二元词频只保留前 6000、2000000、2000000 个。

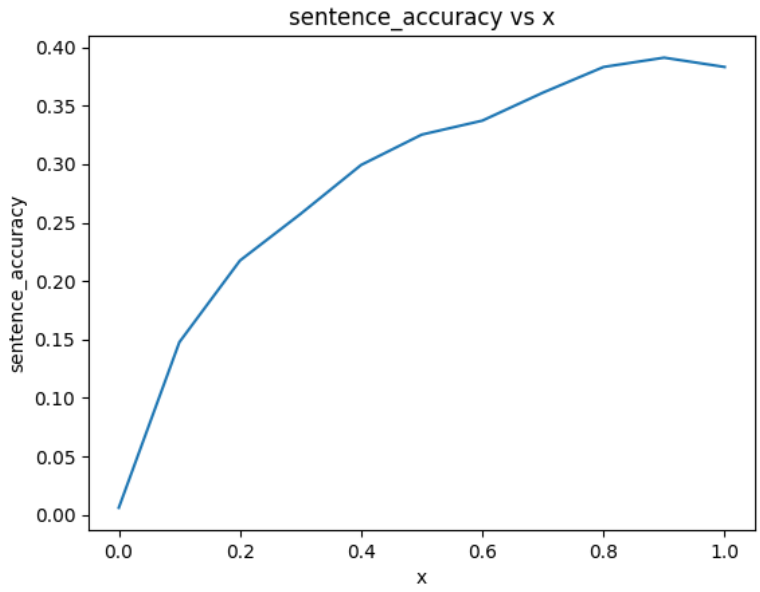
实验效果

基础部分

在基础部分，即基于新浪新闻语料库的字二元模型部分，实验效果如下。

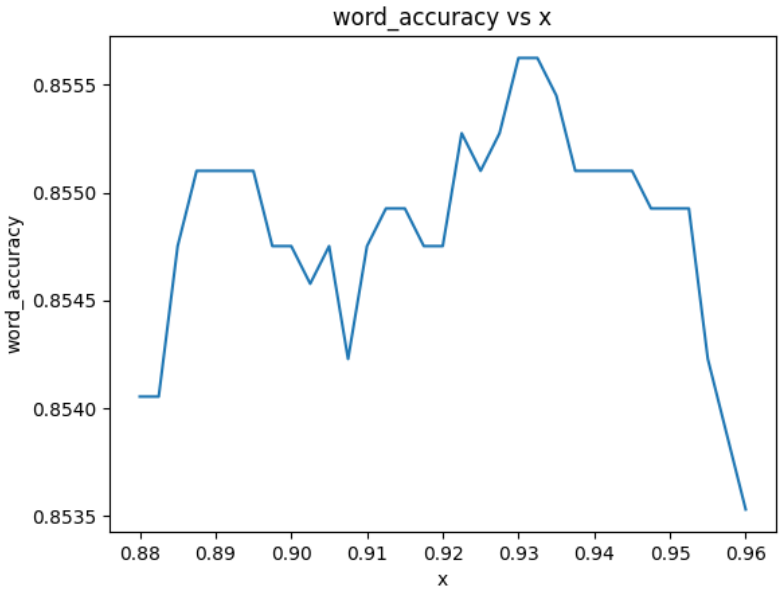
我对这部分涉及到的参数 `x` （参见“平滑处理”部分）进行了枚举调整。所使用的 python 脚本是 `scr/test_parameters.py` 里的 `model_2` 函数。结果如下（具体数据见 `parameters/sina_2_rough.txt`）。

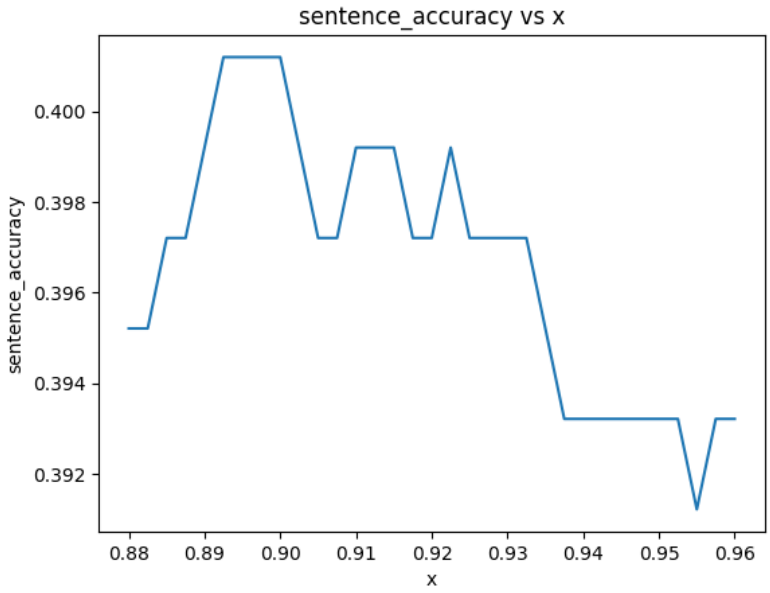




这一结果符合实验预期，即参数 `x` 较小时，模型退化为一元模型，相邻字之间的联系对结果的影响随 `x` 下降而减弱，故准确率下降。而当参数 `x` 过于接近 1 时，平滑效果减弱，输入法会较为显著地受到语料库的二元词频的有限性的制约，因而效果也会下降。

从数据中估计，参数 `x` 可能在 0.92 附近最佳，故在 $[0.88, 0.96]$ 区间进行进一步尝试，结果如下。（具体数据见 `parameters/sina_2_refined.txt`）





当 $x = 0.9000$ 时，字准确率为 0.8548，句准确率为 0.4012。

拓展部分

拓展部分我增加了 baike2018qa 语料库，并使用了基于字的三元模型。

准确率

当 $x = 0.9000, y = 1.0000$ 时，字准确率为 0.8950 ，句准确率为 0.5409。

改进尝试：超参数的调整

在本部分，我调整了超参数 `MAX_PREV` （见“Viterbi 算法”部分）。

较出乎意料的是，该参数在取 1 的时候据准确率比取较大值的时候高。我设计这一参数时，本是出于在不过多牺牲效率的前提下为下一层提供更多选择的想法，希望以此提高句准确率，但事实恰好相反。可能是因为在前面留下了一些前面部分结果已经不佳的句子，但这些句子可能在后续计算的部分表现较好，反而造成干扰。

MAX_PREV	字准确率	句准确率
1	0.8950	0.5409
2	0.8971	0.5369
3	0.8943	0.5250
4	0.8949	0.5209
8	0.8954	0.5170

案例分析

正确案例

- 高考成绩在今天上午公布
- 互联网产业有巨大潜力
- 十三届全国人大一次会议
- 全面报道和分析
- 假如给我三天光明

错误案例

- 青藏大甩卖
  - 清仓大甩卖
- 北京市首个举办过夏奥会与冬奥会的城市
  - 北京是首个举办过夏奥会与冬奥会的城市
- 自强不息后的再无
  - 自强不息厚德载物
- 量子就产作为量子通讯的重要资源
  - 量子纠缠作为量子通讯的重要资源
- 整合下西洋为青花瓷的迅速崛起提供了历史契机
  - 郑和下西洋为青花瓷的迅速崛起提供了历史契机
- 要把我们的祖国建设成为一个富强民主德国家
  - 要把我们的祖国建设成为一个富强民主的国家

分析

- 语料库的局限性
  - 该拼音输入法在涉及文言文（如“自强不息厚德载物”）、专业术语（如“量子纠缠”）、人物或地点名称（如“郑和”）等表现不佳。这可能反映出，这类基于语料库直接提取词频进行训练的方式，对在一般语境中（例如本次选取的新闻、百科问答）不容易出现的表达（如前述的文言文、专业术语等）具有较大缺陷。
- 语料标注方式的局限性
  - 在语料库中，只提供了汉字文本，而没有汉字对应的读音，这使得输入法在计算时，容易出现多音字的错误。特别是当某个常见多音字有不常见读音的时候，或者某个不常见读音组合有读音不同但汉字相同的常见汉字组合的时候，例如错误案例中的“青藏大甩卖”。
- 三元模型本身的局限性
  - 该输入法不具备判断句子成分，并基于此进行计算的能力，因而出现了“北京市首个举办过夏奥会与冬奥会的城市”这样的错误。在以上错误案例中，输入法被“北京市”“德国”这些高频短语

误导，这反映出三元模型本身对于较大语境（句层次）的理解不足。

实验总结

改进思路

通过这次实验，我对拼音输入法有了初步而粗浅的体会，相较于本实验的输入法，我认为拼音输入法还有一些改进的方向。

- 语料库选取应足够大，在内容上尽量包罗万象，而在语言风格上也不能局限于完全书面或口语，在语料库的产生时间上也应该尽量保持较新。
- 在拼音汉字表的基础上，可以录入拼音词典表，以词语为单位进行计算。
- 分析语句成分以及各词语的词性，在句子的层次上对结果进行评价和筛选。
- 即使在模型确定之后，通过调整和选择参数，可能对结果也会有较大影响。

实验收获

在完成本次实验之后，我对借助 python 完成项目有了更深的体会，包括 python 的语言特性，以及 python 丰富而便捷的包（例如本次实验中用到的 argparse、collections、tqdm ）等。我也更加强烈地感受到语言的便捷性和“使用轮子”对于提高效率的作用。

此外，这也是我第一次接触并完成人工智能相关项目，我了解到了人工智能的一些数学基础，以及如何将这些数学理论结合到实际的具体问题中，体会到了问题分析、建立模型、调整参数等一系列过程。