拼音输入法大作业

梁业升 2019010547（计 03）

2022 年 4 月 10 日

实验环境

系统：macOS 12.0
环境：Python 3.9.10

1 算法与实现概述

1.1 预处理

使用课程提供的 2016 年新浪新闻语料库。

1.1.1 分割语句

由于需要获取汉字之间的出现关系，需要排除汉字之外的字符。我们使用汉字的 Unicode 范围来排除非汉字字符，并根据这些字符将原文分割为若干有效汉字组成的句子。由于长度为 1 的“句子”对本实验使用的算法无意义，我们只保留长度不小于 2 的句子。

分割结果保存到临时文件./res/segments.txt。

关键代码如下：

```
for c in content:
    if c < u'\u4E00' or c > u'\u9FA5':
        content = content.replace(c, " ")
all_content += "".join([s + "\n" for s in content.
   split() if len(s) >= 2])
```

1.1.2 统计出现频次

对1.1.1中得到的有效句子进行二元和三元的统计。对于二元模型，记录关于每个出现的汉字的数据；而对于三元模型，对于关于每个出现的二元词的数据。

在对每个句子进行遍历的过程中，记录：

- `start`：出现在开头的次数
- `total`：出现的总次数
- 与后一个汉字组成词所出现的次数

然后，将二元和三元模型的数据保存到临时文件`./res/stats.json` 中。其格式如下：

```
{
    "binary": {
        "编": {
            "start": 24244,
            "total": 150332,
            "者": 1842,
            "组": 775,
            ...
        },
        ...
    },
    "ternary": {
        "编者": {
            "start": 959,
            "total": 1821,
            "按": 1449,
            ...
        },
        ...
    }
}
```

1.2 基于字的二元模型实现

输入法本质上是根据拼音输出正确概率最大的汉字串 $w_1w_2\cdots w_n$，即求：

$$\max P(w_1w_2\cdots w_n) = \max \prod_{i=1}^{n} P(w_i|w_1\cdots w_{i-1}) \tag{1}$$

我们使用 `viterbi` 算法实现。记第 i 个拼音对应汉字 w_j 为节点 $v_{i,j}$，并定义两个相邻节点 $v_{i-1,j}$ 和 $v_{i,k}$ 的距离为：

$$D(v_{i-1,j}, v_{i,k}) = -\log(P'(w_k|w_j)) \tag{2}$$

其中，P' 为平滑处理后的条件概率（$i = 2, \cdots, n$）：

$$P'(w_k|w_j) = \lambda \frac{\texttt{occurance}[w_jw_k]}{\texttt{occurance}[w_k]} + (1-\lambda)P(w_k) \tag{3}$$

其中

$$P(w_k) = \frac{\texttt{occurance}[w_k]}{\sum_j \texttt{occurance}[w_j]} \tag{4}$$

特别地，对于 $i = 1$，我们不使用汉字出现的总次数，而是使用汉字出现在首位的总次数来计算：

$$P'(w_k)\Big|_{i=0} = \frac{\texttt{occurance\_first}[w_k]}{\sum_j \texttt{occurance\_first}[w_j]} \tag{5}$$

求概率 $\prod_{i=0}^{n} P(w_i|w_1\cdots w_{i-1})$ 的最大值，即求 $\sum_{i=1}^{n} D(v_{i-1,k_{i-1}}, v_{i,k_i})$ 的最小值，即最后一层节点到原点距离的最小值。

实现上，在预处理（1.1.2）中已经计算出了式 3、式 4 和式 5 中需要的 `occurance` 和 `occurance_first` 字典。根据上述公式进行计算，得到每个位置对应每个汉字到原点的距离及其对应的前一个位置的汉字，根据最后一层距离最小的汉字向前回溯即可得到所求的字符串。

1.3 基于字的三元模型实现

三元模型与二元模型原理上基本相同，区别在于进行动态规划的节点变成了两个汉字组成的词语而非单个汉字。

使用 $D(v_{i-1,j,k}, v_{i,k,l})$ 表示从第 $i-1$ 到 i 个拼音，由字符串 $w_{i,k}$ 到 w_l 的距离。公式2、3分别变为：

距离定义：

$$D(v_{i-1,j,k}, v_{i,k,l}) = -\log(P'(w_l|w_{j,k})) \tag{6}$$

概率计算：

$$P'(w_l|w_{j,k}) = \lambda \frac{\texttt{occurance}[w_{j,k,l}]}{\texttt{occurance}[w_{j,k}]} + (1-\lambda)P(w_k) \tag{7}$$

实现方法与二元模型类似。

1.4 基于字的二元模型和三元模型混合实现

由于中文中大量的词语为二字词，因此猜测，只使用三元模型可能在某些情况下相比二元模型更不准确。

于是，在三元模型的基础上，计算 $P'(w_{k,l}|w_{j,k})$（式7）时，使用三元和二元计算得到概率的加权平均：

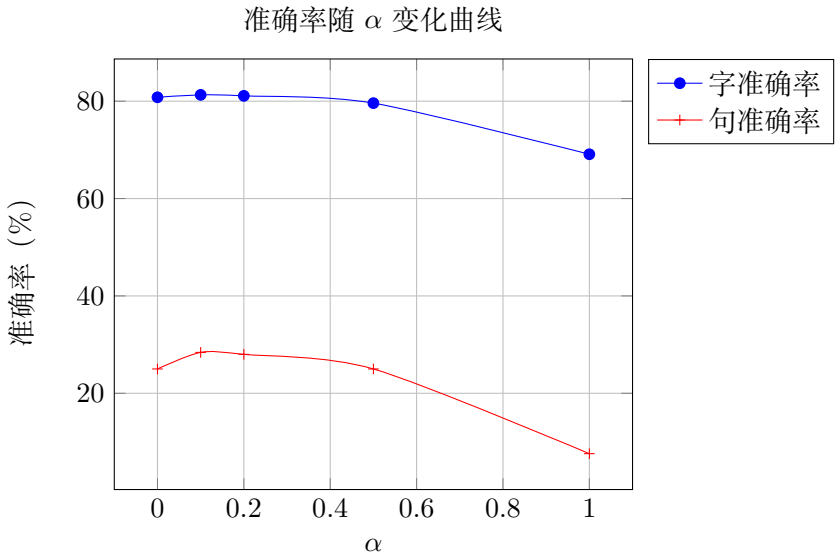
$$P'(w_l|w_{j,k}) = (1-\alpha)P'_{\text{ternary}}(w_l|w_{j,k}) + \alpha P'_{\text{binary}}(w_l|w_k) \tag{8}$$

其他计算和实现与前面所述相同。

2 实验结果

2.1 输入法性能

在混合模型（$\lambda = 0.9$）中对参数 α 进行准确率测试，结果如下：



$\alpha = 0$ 时，相当于三元模型；$\alpha = 1$ 时，相当于二元模型。可以看到，三元模型效果明显优于二元模型，而对三元模型进行微小的二元模型修正（$\alpha = 0.1$）后，其效果优于三元模型，这与 1.4 中的猜测所吻合。

2.2 输入法效果

选取几个比较有代表性的例子，其中第一行为一元模型，第二行为三元模型，第三行为 $\alpha = 0.1$ 的混合模型：

```
shi jie yi liu da xue
世 界 一 流 大 学
是 界 一 流 大 学
世 界 一 流 大 学

zhong guo ren min gan xie ni men
中 国 人 民 感 谢 你 们
中 国 人 民 感 谢 你 们
中 国 人 民 感 谢 你 们

xin xing guan zhuang bing du
新 行 关 装 并 都
新 行 关 状 病 毒
新 行 关 状 病 毒
```

```
bei jing wai gui yu da xue
北   京   外   国   于 大   学
北   京   外   国   语 大   学
北   京   外   国   语 大   学

qing shan lv shui jiu shi jin shan yin shan
清   单   旅 说   就   是 进   山   因   山
青   山   绿 水   就   是 金   山   银   山
青   山   绿 水   就   是 金   山   银   山
```

可以看到，对于基本由二元词语组成，且上下文对词语影响不大的句子，二元模型能够较好地完成转换。而对于含有超过 2 个字组成的词语（如“外国语”），或上下文对词意影响较大（如“青山绿水”）时，二元模型的局限性比较大。

三元模型由于掌握了更大窗口的信息，因此在二元模型局限性较大的多元词语等方面表现较好。同时，三元模型的信息包括了一部分二元模型的信息，因此对二元词的转换效果也不错，但会有出错的时候（如“是界一流大学”）。这时，混合模型的特点就显现出来了：它既能够改善二元模型仅限于二元词语的局限，又修正了三元模型对二元词语的转换偏差。

另外，语料库对于输入法的效果影响明显。由于本次试验使用 2016 年的语料库，而“新型冠状病毒”是在 2020 年出现的新词语，因此输入法无法准确处理这个词语是在情理之中的。