拼音输入法作业报告

吴宸昊 2019011338 计96

零、如何复现实验

在src下运行python main.py

输入文件放在../data/input.txt 输出文件放在../data/output.txt

一、算法介绍

我们要将输入的n个拼音尽可能地转化为n个汉字，使之连贯成为一个句子。在这个问题中我使用了**基于二元字和三元字模型**的算法，该算法一共有以下几步：

- 得到n个拼音对应的n个合法汉字集合
- 从这n个汉字集合中挑出最可能和前后汉字组成句子的n个汉字
- 输出这n个汉字组成的句子

那么对于算法的这三步，我们一共要做哪些工作呢?

- **预处理：建立拼音与汉字集合的对应关系：**我们根据所提供的拼音汉字表，将（每一种拼音，对应合法汉字列表）存储在一个词典中，并对该词典进行持久化存储。之后每次进行翻译任务时将该词典读取到内存之中，以防止每次计算的时间开销。

> 详细代码可见main.py init_pinyin_table函数

- **预处理：基于已有语料训练二元字、三元字模型：**我们在翻译一句话时，为什么会选择某一种汉字组合呢？原因是这种汉字组合在已有语料中的某种**指标**较高；而这种指标可以直观的理解为出现的概率较高。所以我们要统计字组合在语料库中出现的次数。

由于三元模型中包含了二元模型，所以我们只介绍三元模型的统计过程。我们的语料库给了一些句子（当然需要进行一定的预处理），我们拿出句子中的每一个三元组，统计最后一个字出现在前面字串后的次数，并进行相应的递归统计（即在统计三元的同时统计二元）。

> e.g. 我爱你啊->依次对一下次数进行统计
>
> 你 appears after 我爱
>
> 爱 appears after 我
>
> 啊 appears after 爱你
>
> 你 appears after 爱

处理完所有语料之后，我们就得到了一个包含语料字组合频次信息的词典，对该词典进行持久化存储。之后每次进行翻译任务时将该词典读取到内存之中，以防止每次计算的时间开销。这一步的计算时间花销较大。

> 详细代码可见main.py train_model函数

- **翻译过程：HMM模型与Viterbi算法：**这一步是翻译过程的核心，这一部分借用了隐马尔科夫模型

与维特比算法的思想。

隐马尔科夫模型告诉我们，在n元模型中，第i个字出现的概率只与这个字前n-1个字有关。因此我们计算第i个字时只用考虑前n-1个字，使得计算过程变得简单。

而Viterbi算法的思想：**为了找到最优解，要先找到局部最优解。**即如果最终路径会经过某一节点，则从起点到该节点的路径一定局部最优。 因此我们从初始字符节点的概率开始逐层计算，找出到达每一层每一节点的概率最大路径，直到找出最后一层。则最后得到的所有路径中概率最大的一条就是所需的结果。那么我们应当如何计算某一字组合的概率可能呢?

记$P(w_{i,j})$表示第i位字为j的概率。$P(w_{i,j})$的计算公式为：

$$P(w_{i,j}) = \lambda\frac{model[w_{i-2}w_{i-1}][w_i]}{model[w_{i-2}w_{i-1}][total]} + \lambda\frac{model[w_{i-1}][w_i]}{model[w_{i-1}][total]} + 2\times(1-\lambda)\frac{model[w_i]}{model[total]} \tag{1}$$

这个概率计算公式综合考量了三元模型、二元模型以及单字出现的概率（对于第一个字与第二个字的概率需要一定特殊计算），是为了防止单独使用某一模型时所有字组合概率均为零的情况。（关于参数的取值需要不断实验找到最优）

则一个字组合$w_1w_2\dots w_n$出现的概率$P=\prod_{i=1}^{n}P(w_{i,j})$，我们从第一个字的概率计算起，逐层计算到达每个节点的所有路径概率。则到达某一节点中概率最大的一条就是该节点的局部最优路径；如果最终的最佳路径经过该节点，则最佳路径一定包含该局部最佳路径。

二、一些优化

1. **@ magic**

   在预处理训练语料的时候，需要对原始文本进行一些处理，比如将句子中所有非法字符统一替换成@，并且在句首句末统一加上@字符，最后利用正则消去连续重复的@，最后使得两个@之间都是合法文本。举例如下：

   > 我对他说：“今晚月色真美。”
   >
   > => @我对他说@今晚月色真美@

   这样处理有几个原因

   - 原本语料含有大量非法字符，如果不去除会对训练模型的过程造成干扰
   - 关于**句首与句末**字符出现的概率

     如果不在句首与句末加入@字符，则在字组合的概率计算过程中，我们处理句首第一个字出现的概率是以它在整个语料中出现的概率代替。这会给某些特定情况下的概率计算带来误差。比如：

     > de guo：P(的国) > P(德国)

     原因是的在整个语料中出现的次数太多了，如果只以它出现的总概率代替出现在句首的概率，会造成如上的错误。而加入@之后，**德**字出现在句首的概率高于**的**，所以我们可以打出正确的字。在这里我们引入@字符来标志一句话的开始与结束。这类似于一种定界符，可以提高我们输入法的正确率。

2. **二元到三元**

最开始只使用二元模型的情况下，虽说正确率差强人意，但是一些**显然需要**上三元模型的字组合则打不出来，比如

> 机器学习及其应用
>
> 马六甲海峡

在使用三元模型之后，这些名词可以正常显示。但是又发现了另外的问题，比如：

> yi zhi ke ai de xiao hua mao
>
> wrong answer：亿支票〉得哓骅冒
>
> better：一只漂亮的小花猫

这个拼音组合打出了一些奇怪的字符组合，检查之后发现是因为在使用三元模型的过程中**没有综合使用二元概率**，导致在概率计算过程中出现了所有句子组合概率均为零的情况！这促使我去检查代码中的疏漏，对概率计算过程进行改进。我的解决方案是综合二元与三元进行计算，因为有可能三字拼音组合在语料中从来出现过，使得所有字组合概率都为0，从而显示出不合理的结果。

在重新调整概率计算模型后，某测试集的字准确率由`0.845`提高至`0.909`

3. **综合三元、二元与单字**

   在综合考虑了三元与二元之后，我们为什么还要加入**单字**的概率呢?

   因为在某些情况下，字与字之间的联系并不那么强烈；如果不考虑单字模型，则字与字之间会被强行加上联系。因此在我们的概率模型中，我们要一定程度上消减这种字之间的连续性；所以在概率计算过程需要考虑单字出现的概率。比如：

   > bi sai di yi you yi di er
   >
   > wrong answer: 比赛第一优异的儿
   >
   > right answer: 比赛第一友谊第二

   上面的答案错误的原因就是因为第一优异这四个字的关联性本没有那么强。因此我们可以写出式(1)来降低字与字之间的关联性，这里就需要对λ的不同情况进行实验。

λ	**准确率**
0.8	0.908
0.85	0.913
0.9	0.9131
0.95	0.9136
1.0	0.9097

   经过对不同λ进行测试之后，选择了$\lambda = 0.95$作为最后的参数，且该测试集准确率由`0.909`上升至`0.914`。从中我们可以发现，虽然这个策略有一定的优化效果，但是优化效果已经不太好。

   而且优化之后，有些原本正确的句子变得错误，这说明任何优化策略都是有利有弊的。

   由于时间算力有限，调参工作做的也不够细致，还可以有更大的调参空间。

三、Good case

> ji qi xue xi ji qi ying yong
>
> 机器学习及其应用

这个句子在二元情况下怎么打都是"机器学习机器应用"，在上三元模型之后就变成了“机器学习及其应用”。

> you yi di yi bi sai di er
>
> 友谊第一比赛第二

这个例子在上面已经介绍过了，是弱化字与字之间关联性的结果。

> qing shan lv shui jiu shi jin shan yin shan
>
> 青山绿水就是金山银山

四、Bad case/可能的解决方案

1. **多音字情况**

   > wo qu gei ni mai yi ge ju zi
   >
   > bad case: 我去给你买一个句子
   >
   > better: 我去给你买一个橘子

   这是典型的对多音字处理不当的情况。

   思考的改进方法是对字出现时的拼音进行统计，给拼音也**加权**：计算字出现在每种拼音下的概率，则在计算字组合概率的时候能避免这种情况的出现。但是在本次作业中无法提供相应支持。

2. **对句意的理解**

   > ta shi wo de mu qin
   >
   > bad case：他是我的母亲
   >
   > better：她是我的母亲

   这里的她是需要在对句意的理解下才能正确打出，而按照我们的概率模型计算答案会给出他。而这种情况很多成熟的输入法都没有解决。需要更加智能的模型在对句子结构、句意进行分析的基础上才能正确打出。

五、实验总结

这次拼音输入法的实验让我们手动实现了一个搜索过程，与课程前几章内容相吻合。运用到了隐马尔科夫过程和维特比算法相关知识，最后得到了一个准确率还不错的拼音汉字转化程序，并在实验过程中学到了许多新知识。

实验过程要综合考虑时间与空间效应，则没有采用4元以上的模型，这不仅耗费大量空间，还延长程序运行的时间。

另外对于这次实验有一些建议：比如可以在前期增加一些辅导力度，因为要从零开始还是有一定困难；还有可以解决一下文件编码的问题（似乎一直有同学在吐槽这个问题）。

附：程序使用方法

参数选项

```
optional arguments:
  -h, --help            show this help message and exit
  --verbose             Whether to print more information
  --input-file INPUT_FILE
                        Path of file that will be translated
  --output-file OUTPUT_FILE
                        Path of file that will be written into
  --load-model LOAD_MODEL
                        Path of model that will be loaded
  --save-model SAVE_MODEL
                        Path of model that will be saved
  --words WORDS         Path of word-table that will be saved
  --pinyin-table PINYIN_TABLE
                        Path of pinyin-table that will be saved
  --init-words          Task: init legal words table
  --init-pinyin-table   Task: init pinyin-table
  --train               Task: train model
  --translate           Task: translate sentence
  --encoding ENCODING   Coding method of input files
  --file FILE           Path of file to train from
  --dir DIR             Path of directory to train from
  --n-gram N_GRAM       using n-gram model
```