

LAB 4 CHINESE SEGMENTATION WITH DP

胡译文 2021201719

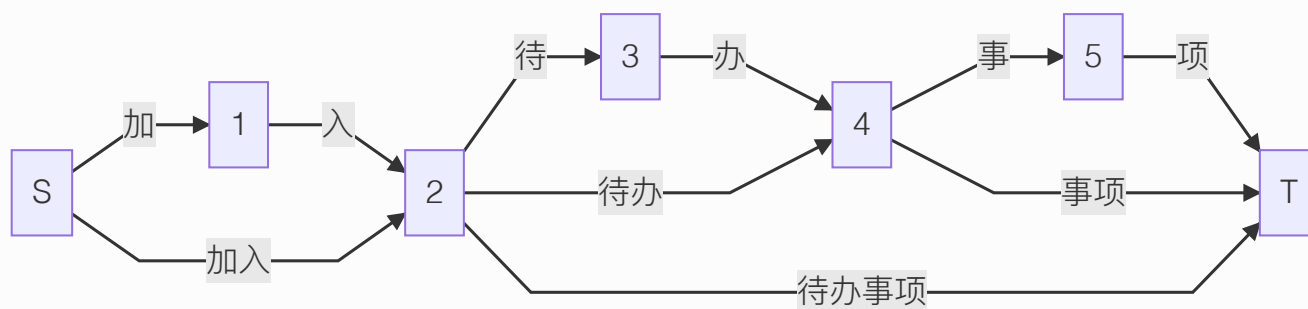
1. 需求分析

本实验要求实现中文分词。目前中文分词有两种方案，一种是经典的统计语言模型方案，另一种是更前沿的基于神经网络的方案。基于课程要求我们选用了统计语言模型的方案。我们使用一个包含一个中文单词、频数和词性的词表来构建统计模型；再利用该模型计算出最可能的分词方案。

| | | | | |
|---|--|---|---------|----|
| 1 | | 的 | 3188252 | uj |
| 2 | | 了 | 883634 | ul |
| 3 | | 是 | 796991 | v |
| 4 | | 在 | 727915 | p |
| 5 | | 和 | 555815 | c |

2. 概要设计

要计算分词，我们可以将各个字和词看成边，每个字之间虚拟一个节点。比如一句话“加入待办事项”用图可以表示成：



我们只需要根据输入构建每个节点间的边权即可。

3. 详细设计

计算一个句子分词的概率，可以采用 BoW 词袋模型的思想，将句子看成分词结果的词的概率之积。由于计算概率的积计算开销较大，一个惯用的思路是转化为对数概率计算和。因此我们将每个边的权重设置为词频的对数。在字典中不会出现频数为 0 的词。

在求解最大值路径的过程中，我们发现我们得到的是一个特殊的有向无环图——每一条边一定从左到右。因此这是一个简单的背包模型，在 $O(nm)$ 的时间复杂度内可以求解。但是我们发现 m 的大小往往远大于 n ——往往标点符号已经是天然的分割，所以一个单句往往长度很小；但最后一个字相同的词平均有 12 个，但中位数仅有 1 个，1/4 位数也仅有 4 个。这表明样本分布极度不平均。如果直接采用背包算法将导致时间复杂度较高。因此我们采用 $O(n^2)$ 的子字符串算法，使用基于哈希的 `unordered_map` 快速查询每个子字符串是否存在以及存在时权重大小。

我们形式化地表示一句话的分词结果和状态：用 $w_{i,j} = c_i, c_{i+1}, \dots, c_j$ 表示由 $j - i + 1$ 个字组成的词， $n_{i,j}$ 表示当前词在词表中出现的次数。为此，我们设 dp_i 为 c_0, c_1, \dots, c_i 子句的最佳分割方法。边的权重采用了 $w_{i,j} = \ln n_{i,j} - \ln N + p_{i,j}$ 方式计算，其中 $N = \sum_{i \in Vocab} n_i$ 、人工设计的罚项 $p_{i,j} = \log_{10} (P \{length = j - i + 1\}) / n_{i,j}$ （推导过程详见 [分词分数计算](#) 部分）。

由此可以得到状态转移方程 $dp_i = \max_{j < i} \{dp_j + w_{i,j}\}$ 。

总时间复杂度为 $O(n^2)$ ， n 为输入单句的长度。

4. 调试分析

中文

本地化（又称国际化）一直是 C++ 软件的一大问题。网上关于本地化的资料并不全面，虽然在大多数机器上都能正常运行，但由于我的电脑在环境变量 `LC_ALL` 和 `locale` 库兼容性上的问题，并不能正常运行。使用以下方法初始化，并采用 `wstring`, `wcin`, `wcout` 的一系列 STL 库，可以成功在文件和命令行读写中文。经测试，Mac 系统在不同语言环境下均可以正常使用。

```
setlocale(LC_ALL, "zh_CN.UTF-8");  
std::wcout.imbue(std::locale("zh_CN.UTF-8"));
```

分词分数计算

在调试过程中遇到了一些问题，比如无法正确分词。具体而言体现在分数计算问题上：

```

测试中文分词
3.57089 试 3.31869 测
    final: 试      2 6.88958 1

0 试中 3.31869 测
5.38595 中 6.88958 测试
    final: 中      3 12.2755 2

0 试中文 3.31869 测
6.48855 中文 6.88958 测试
4.01085 文 12.2755 测试中
    final: 文      4 16.2864 3

0 试中文分 3.31869 测
0 中文分 6.88958 测试
0 文分 12.2755 测试中
4.53983 分 16.2864 测试中文
    final: 分      5 20.8262 4

0 试中文分词 3.31869 测
0 中文分词 6.88958 测试
0 文分词 12.2755 测试中
2.55751 分词 16.2864 测试中文
3.75853 词 20.8262 测试中文分
    final: 词      6 24.5847 5

```

在每个字处切开，都被认为有更高的分数，因此分词结果为“测/试/中/文/分/词”。这是因为最初，我将对数频数 $\ln n_i$ 直接作为了该词的分数。最终，在参阅 BYVoid 的 [基於統計語言模型的拼音輸入法](#)，发现因为不同分词方案词个数不同， $\ln N$ 会对结果产生影响，因此修正了分数计算公式：

$$P^* = -\ln P = -\sum_{i \in S} \ln P(w_i) \approx -\sum_{i \in S} \ln \frac{n_i}{N} = \sum_{i \in S} [-\ln n_i + \ln N]$$

在词表中，长尾词的统计偏差往往很大。因此我们引入“罚项”来弥补数据不足，导致的频率近似概率时，所产生的误差。如果该词没有出现在词表中，则给予较大罚项。为了平滑曲线，使用 $e^{-\log_{10} n_{i,j}}$ 作为系数。

$$p_{i,j} = e^{-\log_{10} n_{i,j}} \times \log(n_{\{length=j-i+1\}}/N) \approx \log_{10} (P\{length=j-i+1\})/n_{i,j}$$

此外，词表中词频过高的词往往会极大影响实验结果，因此我们对于词频在 10000 以上的部分做了根方运算处理，以限制最大词频的影响。

为了尽可能使得分词结果正确，我们反复调整了其中的参数，最终呈现出以上结果。在实验过程中我们深刻体会，可以帮助我们摆脱反复调整数据，也是机器学习算法被高度重视的原因之一。

时间测试

经测试，使用 `unordered_map` 构建索引平均用时 0.2 秒，分词平均用时 0.005 秒。

```
g++ -std=c++17 main.cpp 88 ./a.out
加载词典成功,用时 0.206392 秒
中华人民共和国全国人民代表大会，是中国最高国家权力机关。它的常设机关是全国人民代表大会常务委员会。全国人民代表
大会和全国人民代表大会常务委员会行使国家立法权。全国人民代表大会由省、自治区、直辖市、特别行政区和军队
选出的代表组成。各少数民族都应当有适当名额的代表。
中华人民共和国/全国/人民代表大会/是/中国/最高/国家/权力/机关/它/的/常设/机关/是/全国人民代表大会常务委员会
/全国/人民代表大会/和/全国人民代表大会常务委员会/行使/国家/立法权/全国/人民代表大会/由/省/自治区/直辖市/特
别/行政区/和/军队/选出/的/代表/组成/各/少数民族/都/应当/有/适当/名额/的/代表
分词成功,用时 0.006296 秒
```

5. 用户使用说明

将主体分词库 `SegCN.hpp`、工具库 `utils.hpp`、词典文件 `dict.txt` 以及测试用的 `main.cpp` 放置于同一目录下，使用 C++17 编译并运行 `main.cpp` 文件并运行。输入分词内容（可以包含常见中文标点符号），即可返回以 `/` 分割的分词结果。使用 API 接入 `SegCN.hpp` 可以直接以 `vector` 容器返回分词结果。

6. 测试结果

?

从随机选取的五个句子中发现，分词结果达到较高准确度。