

HW1

PB18000268 曾勇程

1 计算题

1.1 请推荐如下查询的处理次序。

(tangerine OR trees) AND (marmalade OR skies) AND (kaleidoscope OR eyes)

其中，每个词项对应的倒排记录表的长度分别如下：

词项	倒排记录表长度
eyes	213 312
kaleidoscope	87 009
marmalade	107 913
skies	271 658
tangerine	46 653
trees	316 812

记 $len(x)$ 表示词项 x 的倒排记录表的长度。

$$\therefore len(tangerine \text{ OR } trees) \leq 46653 + 316812 = 363465$$

$$len(marmalade \text{ OR } skies) \leq 107913 + 271658 = 379571$$

$$len(kaleidoscope \text{ OR } eyes) \leq 87009 + 213312 = 300321$$

∴ 若仅从估计的 OR 操作后的结果大小来看，推荐的处理次序为

(kaleidoscope OR eyes) AND (tangerine OR trees) AND (marmalade OR skies)

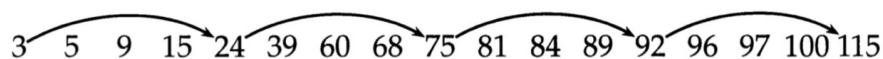
(PS: 这里还有一种思路(仅作为思路开发不作为题答案): 分析数据可知 46653 和 316812 差了 2 个数量级即

tangerine 和 trees 的相同文档数量可能会很少; 而 107913 和 271658 是相同数量级的, 即 marmalade 和 skies 的相同文档数量可能会很多, 而在它们 OR 操作^{多的}上限^{的情况下}不够的情况很可能导致的结果

是: $len(marmalade \text{ OR } skies) < len(tangerine \text{ OR } trees)$, 因此一种处理次序为 (kaleidoscope OR eyes) AND (marmalade OR skies) AND (tangerine OR trees)

1.2 考虑利用如下带有跳表指针的倒排记录表

3 5 9 15 24 39 60 68 75 81 84 89 92 96 97 100 115



和一个中间结果表（如下所示，不存在跳表指针）进行合并操作。

3 5 89 95 97 99 100 101

采用基于跳表指针的倒排记录表合并算法，请问：

- 1) 跳表指针实际发生跳转的次数是多少？
- 2) 当两个表进行合并时，倒排记录之间的比较次数是多少？
- 3) 如果不使用跳表指针，那么倒排记录之间的比较次数是多少？

采用如下算法：

```
INTERSECTWITHSKIPS( $p_1, p_2$ )
1   $answer \leftarrow \{\}$ 
2  while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$ 
3  do if  $\text{docID}(p_1) = \text{docID}(p_2)$ 
4      then  $\text{ADD}(answer, \text{docID}(p_1))$ 
5           $p_1 \leftarrow \text{next}(p_1)$ 
6           $p_2 \leftarrow \text{next}(p_2)$ 
7  else if  $\text{docID}(p_1) < \text{docID}(p_2)$ 
8      then if  $\text{hasSkip}(p_1)$  and  $(\text{docID}(\text{skip}(p_1)) \leq \text{docID}(p_2))$ 
9          then while  $\text{hasSkip}(p_1)$  and  $(\text{docID}(\text{skip}(p_1)) \leq \text{docID}(p_2))$ 
10             do  $p_1 \leftarrow \text{skip}(p_1)$ 
11             else  $p_1 \leftarrow \text{next}(p_1)$ 
12      else if  $\text{hasSkip}(p_2)$  and  $(\text{docID}(\text{skip}(p_2)) \leq \text{docID}(p_1))$ 
13          then while  $\text{hasSkip}(p_2)$  and  $(\text{docID}(\text{skip}(p_2)) \leq \text{docID}(p_1))$ 
14             do  $p_2 \leftarrow \text{skip}(p_2)$ 
15             else  $p_2 \leftarrow \text{next}(p_2)$ 
16  return  $answer$ 
```

1) 一次, 从 24 \rightarrow 75

2) 用二元组 (a, b) 表示一次比较, a 在顺序排记录表中, b 在中间结果表中.

所有的比较: $(3, 3), (5, 5), (9, 89), (15, 89), (24, 89), (75, 89), (75, 89), (92, 89), (81, 89),$
 $(84, 89), (89, 89), (92, 95), (115, 95), (96, 95), (96, 97), (97, 97), (100, 99),$
 $(100, 100), (115, 101)$

两次 $(75, 89)$ 比较的原因: 一次 `then if` 中, 一次 `then while` 中 (重复了一次)

\therefore 比较次数为 19 次 (可以去掉第 8 行的 `then if` 代码, 直接用 `then while` (第 9 行), 这样的话仅需比较 18 次, 即 $(75, 89)$ 不会重复比较)

3) 不使用标志指针: $(3, 3), (5, 5), (9, 89), (15, 89), (24, 89), (39, 89), (60, 89),$
 $(68, 89), (75, 89), (81, 89), (84, 89), (89, 89), (92, 95), (96, 95), (96, 97), (97, 97),$
 $(100, 99), (100, 100), (115, 101)$

比较次数: 19 次.

1.3 写出倒排记录表 (777, 17743, 294068, 31251336) 的可变字节编码。在可能的情况下对间距而不是文档 ID 编码。写出 8 位块的二进制码。

文档ID	777	17743	294068	31251336
间距	777	16966	276325	30957268
二进制码	0011, 0000, 1001	0100, 0010, 0100, 0110	0100, 0011, 0111, 0110, 0101	0001, 1101, 1000, 0101, 1110, 1101, 0100
VB编码	6 0000, 0110 137 1000, 1001	1 0000, 0001 4 0000, 0100 198 1100, 0110	16 0001, 0000 110 0110, 1110 278 1110, 0101	14 0000, 1110 97 0110, 0001 61 0011, 1101 212 1101, 0100

2 问答题

2.1 基于机械分词的常见方法中对于“最大匹配”的依赖，可能导致什么隐患？如何利用 N-最短路径缓解这一隐患？如何选择一个恰当的 N 值？

隐患：对词典的依赖性很强，这将导致：维护高质量词典需要极大的开支；永远难以应对新生词汇（存在未登录词识别问题）；词汇频率/重要性往往对结果不产生影响，这也导致切分出来的结果有着很大的可能性存在切分歧义比如FMM会把“使用户满意”划分为“使用/户/满意”。

N-最短路径方法首先根据词典，找出字符串中所有可能的词，然后构造词语切分有向无环图，最终保留N条最短路径，因为保留了很多种可能的路径和分词方案，故能起到歧义排除和未登录词识别的作用。另一方面，N-最短路径仅在对句子进行分词时要用到词典，后面的过程是基于概率统计的，对词典的依赖性很小。

N-最短路径结合权重/概率之后（统计粗分模型），可视作基于统计的分词方法，字与字相邻共现的频率或概率能够较好的反映成词的可信度。故结合权重/概率后，具有无词典分词结合上下文识别生词、自动消除歧义的优点。

如何选择恰当的 N 值？

一种想法是根据路径的数量而定，路径较多N就大点，否则N就小点，比如取 $N = \min\{10, 10\%N_{\text{总}}\}$ ， $N_{\text{总}}$ 表示全部的路径数，同时确保最短路径集合中至少有10条。

另一种想法是结合路径的概率去考虑，即

设原始输入字符串为C，可能划分的词串为 $W_i, i = 1, 2, \dots, m$ ，表示对字符串C有m中可能的划分。我们要求的即使概率 $P(W_i|C)$ 最大的N个。根据贝叶斯定理，

$$P(W_i|C) = \frac{P(C|W_i) \cdot P(W)}{P(C)}$$

因为在给定字符串的情况下， $P(C)$ 是一固定值， $P(C|W_i)$ 为1，故所求转化为 $P(W_i)$ ，设 $W_i = w_1 \dots w_m, m \geq 1$ ，假设上下文无关，即词与词独立，则 $P(W_i) = \prod_{j=1}^m P(w_j)$ ， $P(w_j)$ 可以根据训练语料库通过极大似然估计统计得出，故概率可以求出，进而整个有向图的路径可以求出，最终可以得到一个候补的N种字符串划分结果。

这时最短路径集合取概率最大的前N条路径。可以先求出每条路径相对于所有路径的相对概率，则此时 $\sum_{i=1}^{N_{\text{总}}} P_i = 1$ ，其中 P_i 是相对概率。然后决定N的选择确保最终集合内所有元素的相对概率之和大于某个常数，比如说10%，或者其他比例。

2.2 如何结合查询词项的分布细节，设计相对合理的跳表指针步长？

设计方法：可以先根据分布细节，计算出倒排记录表中文档ID的间距，然后根据间距将所有节点切分为多个节点集群（一个节点集群内的节点是连续且间距较小的，而两个连续的节点集群的交界处间距比较大），然后在节点数量比较多的节点集群处多设置一些跳表指针（比如每隔3个节点设置一个跳表指针），而在节点数量比较少的节点集群少设置一些或者不设跳表指针，举个例子：

3→4→6→7→8→9→10→11→12→60→61→100→101→102→103→104

现根据间距，将上面的文档ID分为3个节点集群（这里默认间距大于20的话就切割节点集群，比如11→60，间距为49>20，故11为上一个节点集群的最后一个节点，而60为下一个节点集群的第一个节点。切分间距的大小要根据倒排索引表的总体间距大小(或文档频率)而定，有些词项出现次数比较少，文档ID的平均间距较大，这时选20可能就不太合适）：(3,4,6,7,8,9,10,11,12)、(60,61)和(100,101,102,103,104)，故跳表指针可以设置为：

3→8→12→100→104

中间的节点集群(60,61)节点数只有2个，故不在这个节点集群中设置跳表指针。

这样设计的原因是：我们设置跳表指针的目的是为了加快合并，而在间距较小且比较密集的节点集群中一般比较次数比较多而且进展缓慢，故可以考虑多设置一些跳表指针，加快跳转的速度；而在节点数量比较少的节点集群中，考虑到两节点集群间本来间距就比较大，故在分界节点处的跳转幅度已经足够大，若在这种节点处设置跳表指针，该跳表指针的成功率一般不高（这很可能会增加不必要的比较，拖慢合并速度），加上在这种节点集群中节点数量又不多，比较次数也不可能太多，故可以选择不设跳表指针，减少存储空间消耗。

2.3 在信息检索系统中，如何同时使用位置索引和停用词表？潜在问题有哪些，如何解决？

同时使用：对于每个文档和任何查询，删除停用词并合并剩余词项的位置索引，通过确定相对位置来查找准确的短语匹配(文档和查询都经过相同处理的话，原来能匹配的文档现在仍能匹配)。

潜在问题：1.有些停用词在特定场景下是有意义的，例如：“非”、“不”表示否定、“较”、“稍微”表示程度等；2.有些停用词的组合是有意义的，例如，“的士”、“To be or not to be”等。对于一个查询，先删除停用词再去位置索引能查询到所需要的信息，但同时也会查询到很多不需要的信息，即正确率会受到很大影响。比如“的士”，先去除“的”，查询“士”，由于在原文档的处理中也去除了停用词表，故该查询能查到与“的士”相关的文档，但同时也会查到很多仅与“士”有关的文档，而这些是与信息需求无关的。

解决方法：

- 减少停用词的使用；
- 减少停用词表中的词项的数量；
- 引入词项权重，使得那些常用的包含停用词的并且停用词有意义的短语得以整个保留；
- 可以根据词典和基于统计，观察在停用词的一个近邻领域内，停用词和连续的几个字能否组成一个高频词，比如“的”附近组成“的士”，若能则保留停用词，否则去除，不过这种方法会导致处理文档所需的时间和查询所需时间大大增加。