Hw3

# PB16080377聂雷海

1. 考虑下面的文档：

**Doc 1**    prediction of whole country sales   
   **Doc 2**    country sales rise in July   
   **Doc 3**    decrease in home sales in June   
   **Doc 4**    July country sales rise prediction

1. 画出该文档集对应的term-document关联矩阵，假定每个单词都作为一个索引词项

（2）画出该文档集对应的倒排索引，假定每个单词都作为一个索引词项。要求每个词项包含  document frequency以及term frequency

Answer:

1. term-document 关联矩阵：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | doc 1 | doc 2 | doc 3 | doc 4 |
| country | 1 | 1 | 0 | 1 |
| decrease | 0 | 0 | 1 | 0 |
| home | 0 | 0 | 1 | 0 |
| in | 0 | 1 | 1 | 0 |
| July | 0 | 1 | 0 | 1 |
| June | 0 | 0 | 1 | 0 |
| of | 1 | 0 | 0 | 0 |
| prediction | 1 | 0 | 0 | 1 |
| rise | 0 | 1 | 0 | 1 |
| sales | 1 | 1 | 1 | 1 |
| whole | 1 | 0 | 0 | 0 |

1. 倒排索引

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| vocabulary: | Doc\_freq | ----------- | posting list |  |  |  |
| country | 3 |  | 1{freq=1} | 2(freq=1) | 4(freq=1) |  |
| decrease | 1 |  | 3(freq=1) |  |  |  |
| home | 1 |  | 3(freq=1) |  |  |  |
| in | 2 |  | 2(freq=1) | 3(freq=2) |  |  |
| July | 2 |  | 2(freq=1) | 4(freq=1) |  |  |
| June | 1 |  | 3(freq=1) |  |  |  |
| of | 1 |  | 1(freq=1) |  |  |  |
| prediction | 2 |  | 1(freq=1) | 4(freq=1) |  |  |
| rise | 2 |  | 2(freq=1) | 4(freq=1) |  |  |
| sales | 4 |  | 1(freq=1) | 2(freq=1) | 3(freq=1) | 4(freq=1) |
| whole | 1 |  | 1(freq=1) |  |  |  |

2. 用C++实现一个内存版的倒排索引：

（1）基本版：dictionary和postings list都放在内存

* + 输入：n个英文txt文档，文档名为1.txt，2.txt，...，n.txt。每个文档的单词都是由一个空格或者标点分割的，不考虑stemming和stop words去除。
  + 输出：1个txt倒排文件。文件中每一行格式为“term 11 12 13 14"，要求按term的字典序排列。其中term为索引词项，4是文档数目，11 12 13 14是四个文档名称。
  + 范围：每个文档包括最后的倒排文件均可在内存容纳
  + 提交：.cpp文件

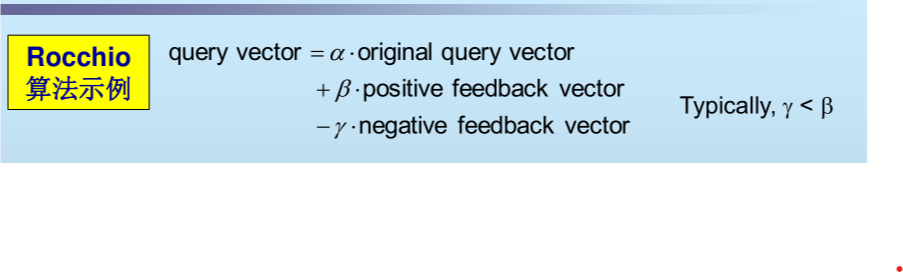
       （2）【选做】升级版：把dictionary放在内存，把postings list存到外存文件中

* + 简单起见postings list只要求ASCII存储，文件名为list.txt。
  + 内存中的dictionary的每个term须有指向其postings list位置的指针（可以用在list.txt中的偏移量表示）
  + 输入：同上
  + 输出：1个dict.txt文件，表示了dictionary，要求term按字典序排列。每个term一行，格式为“term offset”。一个list.txt文件，是由形如"4 12 13 14 15"组成的文本文件，其中4表示文档数目，11 12 13 14为文档ID。所有数字之间以1个空格分割。
  + 范围：单个源文档和dictionary都可以容纳在内存。
  + 提交：.cpp文件

3. 假定初始查询Q为“ extremely cheap DVDs cheap CDs”。文档d1包含词项“cheap CDs cheap software cheap DVDs”，文档d2包含“cheap thrills DVDs”。用户标记d1为相关文档，d2为不相关文档。假定我们直接使用词项频率作为文档向量中词项的权重，并采用Rocchio 1971算法进行相关性反馈，其中α＝1，β＝0.75，γ＝0.25，请给出修改后的查询向量。

Answer:

由Rocchio 算法



可得到

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 词项 | extremely | cheap | DVDs | CDs | software | thrills |
| origin query | 1 | 2 | 1 | 1 | 0 | 0 |
| positive feedback | 0 | 3 | 1 | 1 | 1 | 0 |
| negative feedback | 0 | 1 | 1 | 0 | 0 | 1 |
|  |  |  |  |  |  |  |
| α | 1 |  |  |  |  |  |
| β | 0.75 |  |  |  |  |  |
| γ | 0.25 |  |  |  |  |  |
| origin query\*α | 1 | 2 | 1 | 1 | 0 | 0 |
| positive feedback\*β | 0 | 2.25 | 0.75 | 0.75 | 0.75 | 0 |
| negative feedback\*(-γ) | 0 | -0.25 | -0.25 | 0 | 0 | -0.25 |
| new query | 1 | 4 | 1.5 | 1.75 | 0.75 | -0.25 |
| modified | 1 | 4 | 1.5 | 1.75 | 0.75 | 0 |

4. 查询扩展一般有几种实现方法？请比较一下它们之间的优点和缺点，并说明每一种方法分别适合于什么类型的信息检索应用。

Answer:

* 1. manual thesaurus 人工构建同（近）义词词典

优点：人工处理，命中率高，效果好

缺点：成本高，时间长，效率低

使用：适用于专业名词，新兴网络用语

* 1. automatically derived thesaurus 自动导出同（近）义词词典

优点：效率高，速度快，信息比较稳定

缺点：包含的范围程度，依赖于广泛全面的数据集训练

使用：适用于根据数据集、现有词典，得到词语共现信息从而获取近义词集合。

* 1. refinements based on query log mining 基于查询日志挖掘出的查询等价类

优点：命中率高，支持个性化，基于之前的查询结果

缺点：涉及用户访问网站的隐私

使用：适用于搜索引擎、电商购物等大型网站