

近似查询 设计文档

计算机系 13 班 王庆 2011011239

目录

一.	实验目的	1
二.	设计思路	1
三.	遇到的困难	4
四.	代码结构	4
五.	实验结果	5
六.	感谢语	5

一. 实验目的

给你一个文档和一个查询字符串，在这个文档中找出与查询字符串满足相似度阈值的所有字符串。

二. 设计思路

DevidedSkip 算法

```
Input: set of RID lists and a threshold  $T$ ;  
Output: record ids that appear at least  $T$  times on the lists.  
1. Initialize a result set  $R$  to be empty;  
2. Let  $\mathcal{L}_{long}$  be the set of  $L$  longest lists among the lists;  
3. Let  $\mathcal{L}_{short}$  be the remaining short lists;  
4. Use MergeSkip on  $\mathcal{L}_{short}$  to find ids that appear at  
   least  $T - L$  times;  
5. FOR (each record  $r$  found) {  
6.   FOR (each list in  $\mathcal{L}_{long}$ )  
7.     Check if  $r$  appears on this list;  
8.   IF ( $r$  appears  $\geq T$  times among all lists)  
9.     Add  $r$  to  $R$ ;  
10. }  
11. RETURN  $R$ ;
```

1. 建立索引，将文档的每行字符串进行 gram 划分，建立倒排列表。如果同一行字符串中出现重复的 gram，假设重复出现 5 次，则将这个 substr 标记为 substr, substr1, substr2, substr3, substr4，即将重复的 gram 进行改造，当成不同的 gram,建立倒排列表。
2. 对于给定的 query，我们可以得到两个参数 T 和 L:

ED :

$$T = \text{query.length}() - q + 1 - q * \text{threshold}$$

jaccard:

$$T = \max(\text{threshold} * \text{gramNum}_{\text{query}}, (\text{gramNum}_{\text{query}} + \text{gramNum}_{\text{docShortestString}}) * \frac{\text{threshold}}{\text{threshold} + 1})$$

$$L = \frac{T}{\mu * \log_{10}(M) + 1} \quad \mu = 0.0085$$

<1> 将这个 query 进行 gram 划分，获取划分后的 gram 对应的倒排列表中的 ids，这样一个 query，可以得到一个小型的倒排列表 query_indexed_list.

<2> M=query_indexed_list 中 gram 对应的 ids 串长度的最大值，计算 T 和 L 值

<3> 如果 T 和 L 小于等于 0，那么将文档里面所有的字符串与 query 算 ed 和 jaccard 值，将符合要求的加入 result; 否则，将 query_indexed_list 排序(ids 串长度, 从小到大), 假设共有 W 个 ids 串，则将 W 个 ids 串划分成前(W-L)

和 L 两部分。如果 $W \leq L$, 那么将文档里面所有的字符串与 query 算 ed 和 jaccard 值, 将符合要求的加入 result.

<4> 前($W-L$)部分, 使用 MergeSkip 下列算法:

该 $T = \langle 2 \rangle$ 中计算出来的 $T-L$

```

Input: a set of RID lists and a threshold  $T$ ;
Output: record ids that appear at least  $T$  times on the lists.
1. Insert the frontier records of the lists to a heap  $H$ ;
2. Initialize a result set  $R$  to be empty;
3. WHILE ( $H$  is not empty) {
4.   Let  $t$  be the top record on the heap;
5.   Pop from  $H$  those records equal to  $t$ ;
6.   Let  $n$  be the number of popped records;
7.   IF ( $n \geq T$ ) {
8.     Add  $t$  to  $R$ ;
9.     Push next record (if any) on each popped list to  $H$ ;
10.  }
11.  ELSE {
12.    Pop  $T - 1 - n$  smallest records from  $H$ ;
13.    Let  $t'$  be the current top record on  $H$ ;
14.    FOR (each of the  $T - 1$  popped lists) {
15.      Locate its smallest record  $r \geq t'$  (if any);
16.      Push this record to  $H$ ;
17.    }
18.  }
19. }
20. RETURN  $R$ ;

```

<5>在后 L 个 ids, 使用 MergeOpt 算法

- MergeSkip 算法得到一群候选者 ids, 以及这些 ids 在 query_indexed_list 排序后前 $W-L$ 个 id list 出现的次数。
- 遍历这些候选者 ids, 如果在 query_indexed_list 排序后的后 L 中, id 出现的次数大于等于 $T-n$ (n 为该 id 在前 $W-L$ 个 id list 出现的次数), 则该 id 加入最终候选者队列。
- 计算最终候选者队列里面 id 对应字符串与 query 的 ed 和 jaccard 值, 符合阈值, 加入 result。

三. 遇到的困难

由于缺乏对算法本质的了解，导致在调试的过程中遇到了很多的困难，最后在我们班李震同学的帮助下，了解了算法这样设计的原因，仔细检查自己的实现，找到了 bug。


四. 代码结构

文件名称	作用
SimSearcher.cpp	实现对应.h 中的函数 min(): 获取两个数的最小值 max(): 获取两个数的最大值 BinSearch(): 返回 vector 中大于等于 value 的最小值对应的 index BinSearchValue(): 返回 vector 中值等于 value 对应的 index
SimSearcher.h	createIndex(): 输入文件，创建索引 searchJaccard(): 输入查询 query, 返回 \geq threshold 的 result searchEd(): 输入查询 query, 返回 \leq threshold 的 result getEditDistance(): 返回两个字符串的编辑距离 getJaccardDistance(): 返回两个字符串的 Jaccard 距离
VectorItem.cpp	实现对应.h 的函数
VectorItem.h	封装一个 <code>vector<int></code> insert(): 插入一个 int 元素 getVector(): 返回封装的 vector

	<code>sort()</code> :将封装的 <code>vector</code> 排序 <code>size()</code> :返回封装 <code>vector</code> 的大小
SmallHeap.h	封装一个堆顶为最小值的堆 <code>size()</code> : 返回堆中元素的个数 <code>shiftup(s,e)</code> :从 <code>s</code> 这个位置开始向上维护最小堆, 找到 <code>e</code> 的位置, 插入 <code>e</code> 。 <code>poll()</code> :弹出堆顶元素 <code>shiftdown(s,e)</code> : 从 <code>s</code> 这个位置开始向下维护最小堆, 找到 <code>e</code> 的位置, 插入 <code>e</code> 。

五. 实验结果

Online 测试结果, 几乎同一份代码 (微小改动):

History 					
ID	Homework	Upload Time	Result	Memory Usage(GB)	Time Usage(s)
3771	exp1: Similarity Search	Tue, 10 Jun 2014 20:14:03 +0800	Correct.	0.124	48.385
3770	exp1: Similarity Search	Tue, 10 Jun 2014 20:12:01 +0800	Correct.	0.124	48.089
3769	exp1: Similarity Search	Tue, 10 Jun 2014 20:09:55 +0800	Correct.	0.124	48.347
3768	exp1: Similarity Search	Tue, 10 Jun 2014 20:07:23 +0800	Correct.	0.124	48.202
3767	exp1: Similarity Search	Tue, 10 Jun 2014 19:13:19 +0800	Correct.	0.124	48.865
3766	exp1: Similarity Search	Mon, 9 Jun 2014 17:33:25 +0800	Correct.	0.124	49.289

代码: 压缩包 `exp1_2011011239` 直接提交到在线系统即可测试。

六. 感谢语

非常感谢老师非常认真地教授给我们知识, 我了解了很多新的搜索算法, 学习了很多paper, 收获很大。非常感谢助教帮我们搭建了测试平台, 帮助我们更好的了解算法的性能!

祝愿李老师身体健康, 工作顺利, 一切都好!

祝愿助教学业进步, 多中 paper, 一切顺利!