小实验一 <近似查询> 实验报告

最终效果

ID	Homework	Upload Timestamp	Status	Memory(GB)	Time(s)	Comment	Ops
2878(Marked)	exp1-final	2017/04/11 14:05:48	Correct.	11.563	11.811		
2694(Marked)	exp1	2017/04/10 19:05:15	Correct.	5.43	2.103	MergeOpt	

编辑距离<近似查询>

算法

采用Q-gram算法，在创建索引的时候，预先设置一个 Q，对于每次查询字符串 S 和编辑距离上限 τ，可以计算出它和满足条件的字符串之间至少有 T 个完全相同的Q-gram，其中：

$$T = |S| - Q + 1 - Q \times \tau$$

根据这种思想，在创建索引的时候，对于数据集中的每一个字符串，取出其每一个Q-gram插入到一颗 `Trie` 树中，建立反向列表，即实现 $gram \rightarrow IDlist$ 的查询;需要注意的是，同一个字符串可能具有相同的Q-gram，对此只需要对相同的Q-gram建一个索引即可。

在查询时，对于查询的字符串 S，可以根据上面的公式计算出 T，再从 `Trie` 树中取出 S 的每个Q-gram（此处相同的Q-gram要取出多次）对应的 $IDlist$，而 T 即表明，所有满足编辑距离限制的字符串在 $IDlist$ 中至少出现了 T 次，所以我们只需要将在对应的 $IDlist$ 中出现次数不小于 T 的字符串取出来，再计算其和查询字符串 S 之间的编辑距离即可。

对于找出在若干个 $IDlist$ 中出现次数不小于 T 的 ID，我们可以在创建索引的时候对每个 $IDlist$ 内部的元素排序，然后将这些 $IDlist$ 中较长的 $T-1$ 放到一边，对于剩下的 $IDlist$，使用类似计数排序的方式统计每个 ID 出现的次数，然后再将这些 ID 在较长的 $T-1$ 个 $IDlist$ 通过二分查找的方式统计出现的次数，由此可以统计出出现次数不少于 T 的 ID，然后再进行进一步的判断即可。

对于计算出的 $T < 0$ 的情况，则无法使用上面的方法，但是导致 $T < 0$ 情况出现唯有 $|S|$ 较小，所以这部分直接判断即可。

优化

在实践中，发现 $Q = 8$ 速度最快，且满足内存限制;

回顾 T 的计算公式：

$$T = |S| - Q + 1 - Q \times \tau$$

可以发现 Q 越大，则越可能导致 $T < 0$，但是 Q 越小，则取出的 $IDlist$ 越大;故对于每个查询，应该选取最大的使得 $T \geq 0$ 的可能的 Q，所以在可能的情况下，我们应该对更多的 Q 建立反向列表;

注意到我们是使用 `Trie` 树来储存索引，所以我们可以直接在同一个 `Trie` 树中建立不同 Q 的索引，而不会增加太多的空间;此方法即使我在程序中使用的方法。

Jaccard<近似查询>

Jaccard近似查询和编辑距离的近似查询非常类似，Jaccard的定义如下：

$$\frac{|Q \bigcap S|}{|Q \bigcup S|} \geq \tau$$

其中 Q 是查询的单词集合, S 是数据库中的某个记录的单词集合;将公式进行推到，即可得到：

$$T = |Q \bigcap S| \geq |Q \bigcup S| \times \tau \geq |Q| \times \tau$$

即：

$$T \geq \lceil |Q| \times \tau \rceil$$

其中 T 表示查询字符串和数据库中某记录完全相同的单词个数。

所以，对于数据库中的每个记录，将其单词插入到 `Trie` 树中，建立 单词 $\rightarrow$ $IDlist$ 的反向列表;

对于每个查询，从 `Trie` 树中取出对应若干个的 $IDlist$ ，然后用和编辑距离相同的方法统计出出现次数不小于 T 的 ID （由于方法完全相同，故在此不再赘述），在使用 `Trie` 树（此处的 `Trie` 树和建立索引的 `Trie` 树不是同一个）判断一下每个取出的 ID 和查询是否满足集合即可。