# 数据库专题训练 Similarity search

## 主要数据结构与算法

### unordered_map建立的哈希表

利用*unordered_map*建立由`string`到`vector<int>`的哈希对应关系，从而可以很方便地快速建立倒排表。

具体地，维护三个这样的map：

```
unordered_map<int, set<string>> lines_indexes;
unordered_map<string, vector<int>> jaccard_list;
unordered_map<string, vector<int>> ed_list;
```

其中`lines_indexes`维护了每一行对应的单词序列（用于jaccard计算时求交集），`jaccard_list`维护了每个单词对应的倒排列表，`ed_list`维护了每个qgram对应的倒排列表。

### Merge Opt

从query字符串分词（或分qgram）得到对应的string后，对每个string查找倒排列表，得到整个query串的倒排列表序列。

通过获取阈值算法得到`thredhold`对应的查询集合交集的最小值，设为`T`.

取倒排列表中长度最长的`T-1`个，其余的为短列表。

对短列表进行**scanout**搜索，获得每一个行号及其在短列表中出现的次数。接着对每个短列表中出现的行号，在每个长列表中二分查找，记录它出现的次数。若出现次数 + 剩余长列表数 < T，则不可能达到阈值，因而直接舍弃。否则若执行到最后次数大于等于`T`，则加入到备选列表里。

最后，对备选列表进行精细的DP计算，得到准确的编辑距离，再做进一步的取舍，得到最终的结果。

## 各个函数的详细实现

### createIndex

按行读入存储到对应的数据结构中。接着分别建立倒排表：

- 对于jaccard，需要根据空格分隔开各个单词，需要注意的是要去重，因此首先用`set<string>`存储一行的单词，接着再将这些单词建立倒排表。
- 对于ED，直接每q个为一组作为单词，也需要去重，然后再建立倒排表。

  去重的目的是为了避免某行在一个单词的倒排表中重复出现，导致计算结果出现偏差。

### search

利用MergeOpt算法进行查找。具体流程如下：

- 建立query串的词对应的倒排表，需要注意的是jaccard与ED不同：**jaccard需要在建立倒排表序列时先对单词去重，但ED在建立倒排表序列时不能对单词去重。**（这一点也是我调了一整天才发现的大坑）
- 计算得到交集个数的下界`T`：
  - jaccard:（num为query串得到去重词集合后的集合大小）

    ```
    double t1 = threshold * num;
    double t2 = (num + min_line_size) * threshold / (1 + threshold);
    return ceil(t1) > ceil(t2) ? ceil(t1) : ceil(t2);
    ```

  - ED:（query_len为query串的长度，q为q_gram大小）

    ```
    double t1 = query_len - q + 1 - threshold * q;
    return ceil(t1) > 0 ? ceil(t1) : 0;
    ```

- 利用数组下标排序，避免对原数据结构vector的搬运，从而可以得到数组长度为前`T-1`的长串对应的倒排列表序列的下标。
- 若`T`为0，直接把所有行作为备选集然后进行精确计算。
- 若`T`不为0，则首先在短列表中进行*scanout*算法得到短列表中各个行号出现的个数，然后对每一个行在长列表中进行二分查找，记录出现次数。若出现次数 + 剩余长列表数 < T，则不可能达到阈值，因而直接舍弃。否则若执行到最后次数大于等于`T`，则加入到备选列表里。在计算ED时若查询串与备选行的长度差距大于`threshold`，则直接舍弃该行。（因为不可能通过小于等于`threshold`次编辑得到另一个串）
- 最终在备选集中精确计算。

## calculate

- 计算jaccard：
  - 直接利用stl库对两个set取交集即可。
- 计算ED：
  - 利用O((2τ+1)*n)复杂度的动态规划求解，每行与每列最多只需要计算2τ+1次。