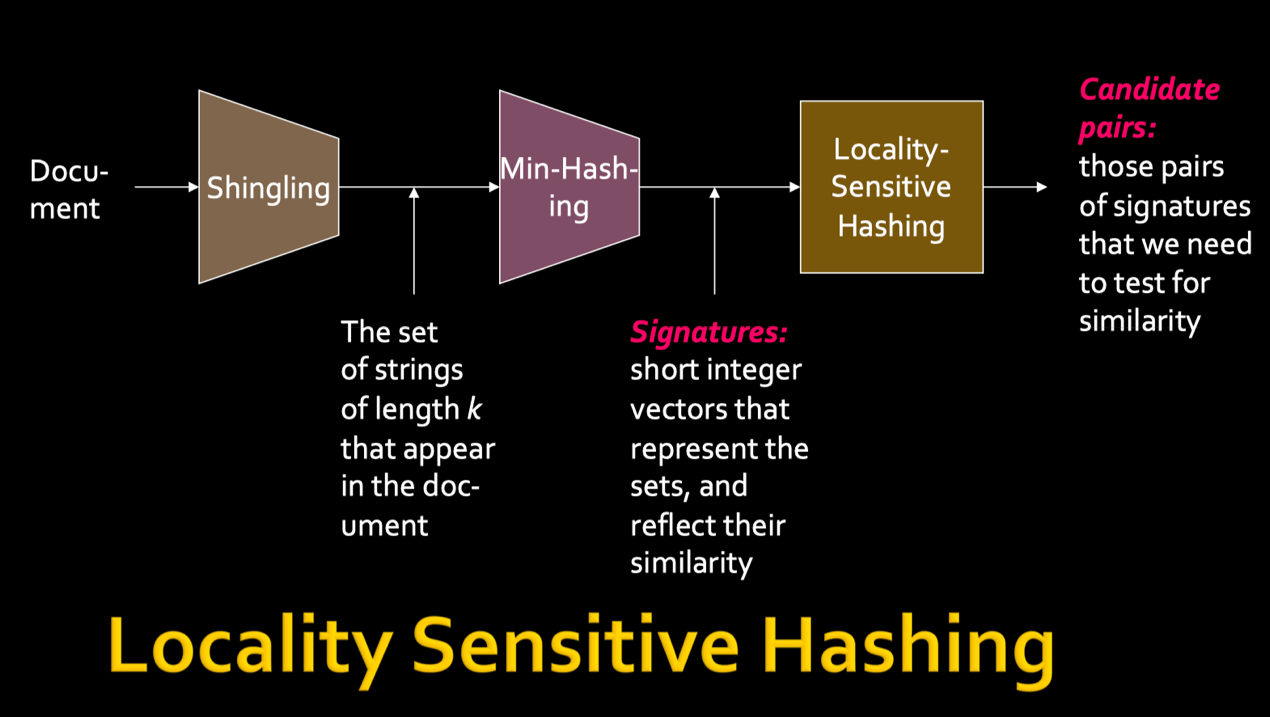
**第12周上机实验报告**

# 一、题目：

实现min\_hash和locality\_sensitve\_hashing



# 二、算法思路：

**鉴于课程中已经教授过本算法，在此不再赘述算法思路。**

**改为着重介绍实现过程中各个参数的选取。**

Min\_hash

**在第十二次作业中，计算得到**

**令，可得k = 255.94**

**因此，令k = 256**

**于是，min\_hash得到的signature有256行，通过256个哈希函数得到。**

**此处的哈希函数属于哈希族其中n是vocabulary中的单词总数，p是一个大于n的素数。因为n = 28102，令p = 28277。a、b是属于[1,p]的随机数。**

Locality Sensitive Hashing

**由于k=256，所以b\*r=256。**

**经过多次尝试以后，为了降低LSH结果的false positive, 我选择令b = 8, r = 32。**

**S = = 0.93**

**AND操作时，为了达到的目的，我参考了BKDR hash算法，从而使band中r行里的每一行都参与哈希值的计算，并且符合雪崩效应（即改变r行中任意一行的值都会对哈希值产生较大影响）**

**令bucket\_size = 2000000, 这是调参很久以后得到的比较好的值。从算法的思路不难看出bucket\_size越大越好，经过反复试验，本实验中bucket\_size大于1000000时，效果比较好。**

# 三、程序设计框架：

核心类的名称和功能

Min\_hash

* Min\_hash实现了min hashing部分
* Num\_Docs 记录documents的数量
* Hash\_func\_num 存储使用的hash函数的数量，也就是随机重排permutation的数量，即上文中的k。
* Hash\_p 存储大于N的素数，测试样例中N=28102,所以可以选择28277
* Hash\_a, hash\_b分别存储k个哈希函数中的参数a与参数b
* 类初始化时指定hash\_func\_num
* Read\_file读取文件
* Gen\_signature生成siganature

1. **class** min\_hash
2. {
3. **private**:
4. **int** num\_Docs;                  //number of documents
5. **int** hash\_func\_nums;            //min hash的hash func的个数
6. vector<vector<**int**> > signature; //存储signature
7. vector<vector<**int**> > data;
8. **const** **int** hash\_p ; //hash func的参数 (ax+b)mod p mod N 中的p，选择一个大于N的prime number，测试样例中N=28102，所以可以选择28277
9. vector<**int**> hash\_a; //hash func参数a
10. vector<**int**> hash\_b; //hash func 参数b
12. **public**:
13. min\_hash(**int** K);
14. ~min\_hash();
15. **void** read\_file(string file\_name);
16. **void** gen\_signature(); //生成signature
17. vector<vector<**int**> > get\_signature() **const**;
18. vector<vector<**int**> > get\_data() **const**;
19. };

LSH

* b代表一共b个band
* r代表一个band有r行
* s是
* min\_hash\_data存储min\_hash产生的signature
* buckets存储bucket
* hash\_a和hash\_b存储计算bucket\_index时用到的哈希函数的参数
* query\_list存储所有文档在LSH后得到的相似的文档
* LSH初始化是指定b,r,s
* Get\_min\_hash,读入了min\_hash的signature
* Generate\_buckets生成buckets，并且根据所有bucket的情况，将每个文档的candidate pair存放在query\_list中
  + AND操作的时候，使用BKDR Hash，将每一行的数值对哈希值整体的影响放大，进而达到**的目的**
  + OR操作，遍历每一个size大小大于2的bucekt，将其中的元素作为condidate pair添加到query\_list中
* Get\_query\_list对于给定的DocID返回LSH中它的candidate pairs
* Output\_query\_list将query\_list输出成文件“LSH\_buckets.txt”

1. **class** LSH{
2. **private**:
3. **const** **int** b; //b 个 band
4. **const** **int** r; //r rows per band
5. **const** **double** s; //阈值s
6. vector<vector<**int**> >min\_hash\_data;
7. **int** k;// k 个 buckets
8. vector<vector<set<**int**> > >buckets;
9. vector<**int**> hash\_a; //b个哈希函数的参数a
10. vector<**int**> hash\_b; //b个哈希函数的参数b
11. **int** hash\_p; //b个哈希函数的参数p
12. vector<set<**int**> >query\_list; //doc i 相似的文档存在vector[i]中的set中
14. **public**:
15. LSH(**int** bb, **int** rr, **double** ss);
16. ~LSH();
17. **void** get\_min\_hash(vector<vector<**int**> >data);
18. **void** gennerate\_buckets(); //生成buckets并且生成query\_list
19. set<**int**>  get\_query\_list(**int** index)**const**;
20. **void** output\_query\_list(string file\_name)**const**;
21. };

各个核心类与核心函数的关系

Main.cpp中，先使用min\_hash获得signature，再使用LSH获得buckets，接下来随机挑选了6个docID，返回他们的LSH中的candidate pair，并且使用JS函数计算Jaccard Similarity验证正确性，将结果输出到屏幕上。

1. **int** main(**int** argc, **const** **char**\*\* argv) {
3. min\_hash test1(256);// epsilon = 0.15 delta = 0.1
4. test1.read\_file("./docword.enron.txt");
6. **time\_t** start = clock();
8. test1.gen\_signature();
9. vector<vector<**int**> > data = test1.get\_data();
10. vector<vector<**int**> > sig = test1.get\_signature();
12. LSH test2(8,32,0.93);  //0.93 = (1/8)^(1/32)
13. test2.get\_min\_hash(sig);
14. test2.gennerate\_buckets();
16. **time\_t** end = clock();
17. cout <<"Min-hash and LSH costs " << (**double**)(end - start) \* 1000 / CLOCKS\_PER\_SEC << " ms" << endl<<endl;
19. test2.output\_query\_list("LSH\_buckets.txt");

22. cout<<"query and calculate Jaccard Similarity"<<endl;
23. **int** queries[6] = {0,92,5083,7366,14087,29556};
24. **for**(**int** i = 0;i < 6;i++){
25. set<**int**>ans = test2.get\_query\_list(queries[i]);
26. cout<<"query Doc ID "<<queries[i]+1<<" :"<<endl;
27. **for**(auto x : ans){
28. cout<<"doc "<<x<<" Jaccard Similarity: ";
29. cout<<JS(data,queries[i],x)<<endl;
30. }cout<<endl;
31. }
32. **return** 0;
33. }

# 四、实验结果说明：

实验结果截图

* + Buckets

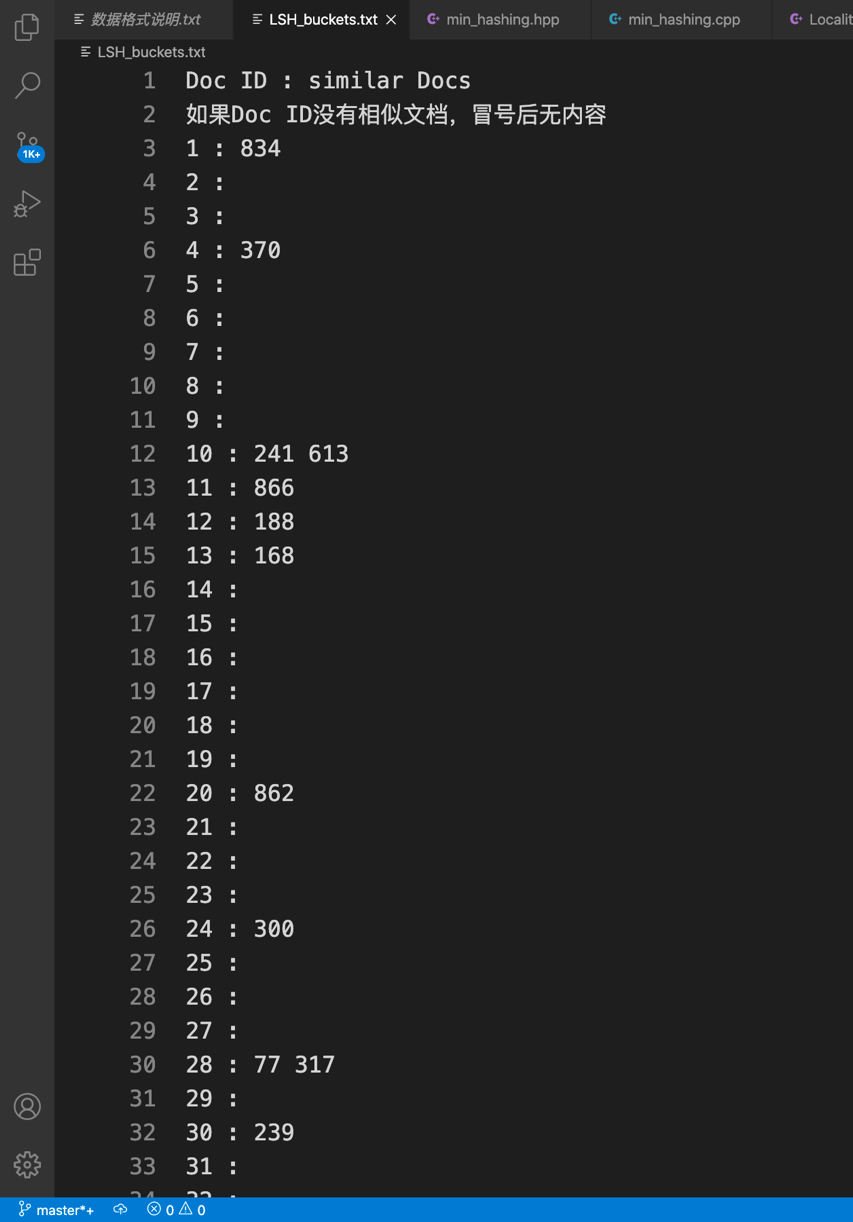


Figure 1

* + 运行结果

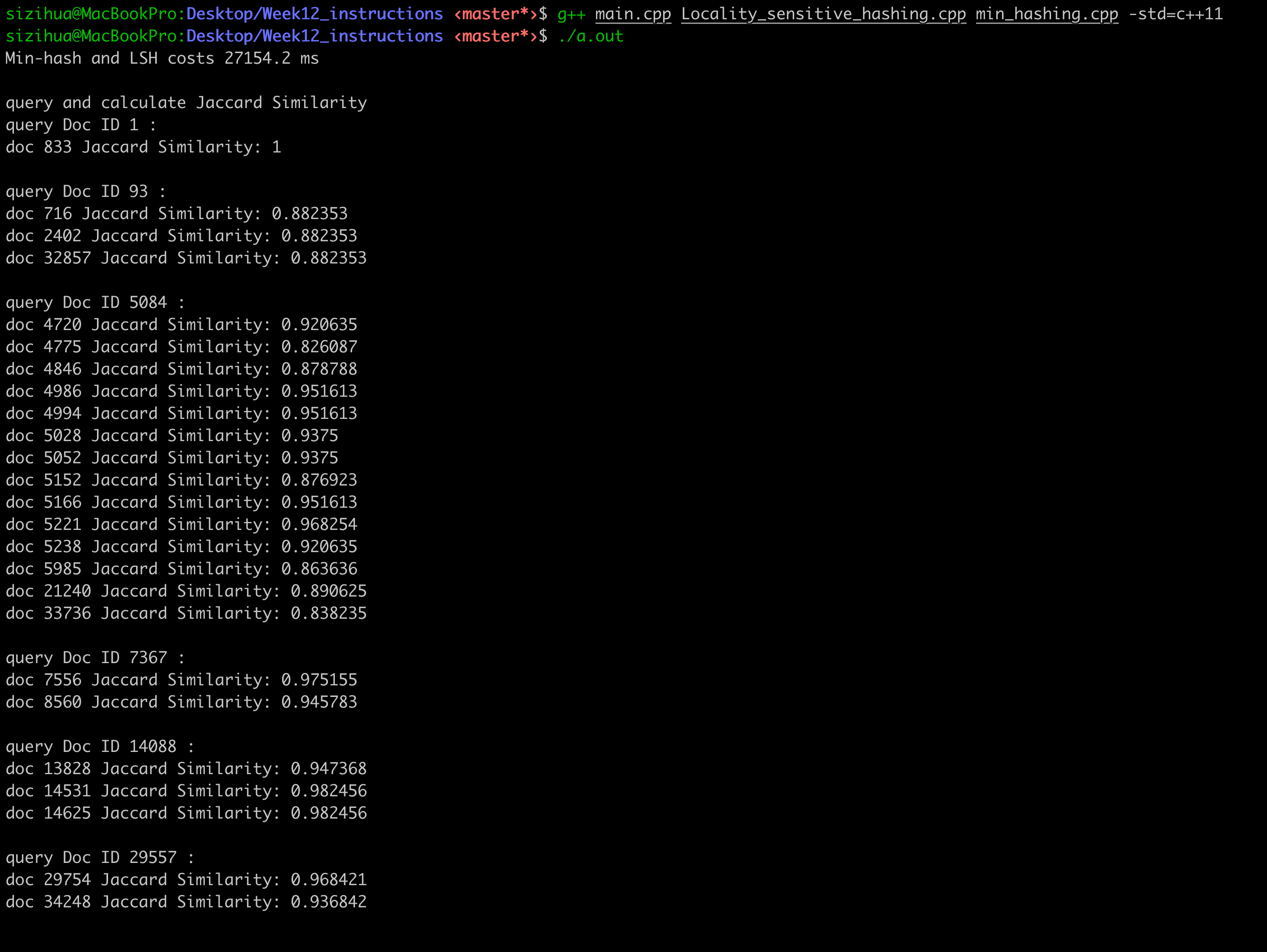


Figure 2