Report **107062546 楊仲愷**

Java File, Class Name: **Shingle.jar, Shingle**
Minhashing.jar, Minhashing
LSH.jar, LSH
Similarity.jar, Similarity

First, please execute Shingle.jar, and we will get the shingle file. Total input files are in the path, /hw02/Input, and the output path we set /hw02/Output001.

```
[root@sandbox-hdp target]# yarn jar Shingle.jar Shingle  /hw02/Input /hw02/Outpu
t001
```

Then, we will get all the files which are processed into 3-shingle.

```
File:001
Claxton hunting first
hunting first major
first major medal
British hurdler Sarah
hurdler Sarah Claxton
Sarah Claxton is
```

The function shingle will divide string into several parts, and num can be input by the shingles we want.

```
public static ArrayList<String> Shingle(ArrayList<String> string1, int num) {
    ArrayList<String> str = new ArrayList<String>();

    for(int i = 0; i + num <= string1.size(); i++) {
        String string = "";

        for(int j = i; j < i + num; j++) {
            if(j == i) {
                string = string1.get(j);
            }else {
                string = string + " " + string1.get(j);
            }
        }
        str.add(string);
    }

    return(str);
}
```

Second, we take advantage of minhashing to check the relations with different shingled file with permuted by different hash functions.

```
[root@sandbox-hdp target]# yarn jar Minhashing.jar Minhashing /hw02/Output001 /h
w02/Output002
```

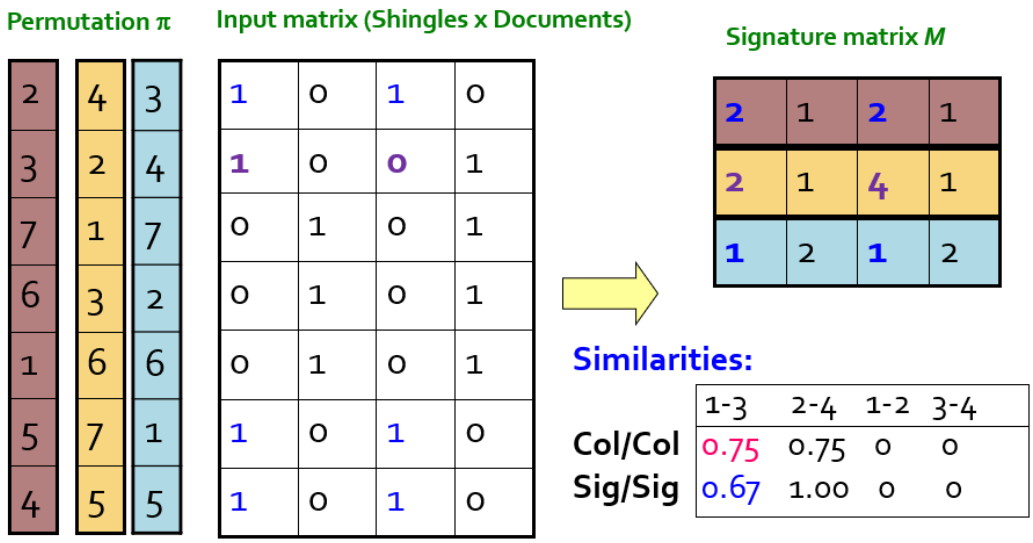
We will get files which are processed with permuted shingles. And 1 means the document contains the shingle.

```
I think there1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
it comes. As1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
the AAAs title.1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
over 60m hurdles1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
medal at next1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Claxton hunting first1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
the European Indoors1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
has re-focused her1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
the 25-year-old also1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
campaign over the1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

And we also get a file named total.txt processed by minhashing. It is the signature matrix.

```
1,0,0,43,0,170,0,0,170,43,0,0,196,0,19,0,0,0,19,0,196,0,19,0,0,0,0,118,0,0,0,0,
2,0,0,0,0,0,0,130,0,119,125,67,0,31,0,0,90,0,0,16,0,16,0,0,13,16,0,0,119,125,0,
3,84,0,0,0,0,97,0,0,28,69,0,0,199,103,0,0,0,20,118,0,0,0,0,0,242,0,0,202,69,167
4,0,0,0,0,0,0,0,180,0,0,0,0,0,0,0,0,0,0,0,135,0,0,0,89,0,0,0,0,0,0,0,0,37,0,0
5,143,0,0,0,0,0,0,143,0,0,0,0,0,0,0,151,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9,0,0,0,0
6,0,0,24,0,0,0,0,0,45,39,0,0,0,0,0,0,0,11,127,0,0,39,0,0,0,0,0,0,24,137,0,41,0,
7,0,12,0,0,0,0,0,0,0,0,0,0,0,83,0,10,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
8,168,0,0,174,168,0,0,0,0,94,0,0,211,62,0,0,0,0,0,0,0,50,0,0,0,0,0,0,0,0,0,0,0,
9,11,86,11,0,0,354,0,0,0,13,0,95,187,0,0,97,95,269,306,95,76,299,95,0,0,0,0,86,
10,0,60,17,0,0,15,0,114,13,0,1,0,0,175,0,104,0,0,0,0,0,15,0,0,1,0,0,74,17,0,1,1
11,0,36,0,0,0,0,0,0,0,14,0,0,49,0,0,0,0,0,40,0,16,110,0,0,4,0,0,0,0,175,14,0,0,
12,21,0,0,0,0,0,0,0,20,0,0,0,0,0,0,0,20,22,51,0,0,22,0,0,0,51,27,118,0,28,0,0,0
```

Minhashing:



I permute every shingle file and find every column which contains the shingle.

```
for(int i = 1; i <= 50; i++) {
    Map<Integer, String> map1 = new HashMap<Integer, String>();

    doublemap.put(i, permute(map.get(i), map1));
    orimap.put(i, map1);
}

int a[][] = new int[doublemap.get(filenum).size()][doublemap.size()];


for(int i = 0; i < doublemap.get(filenum).size(); i++) {

    String word = doublemap.get(filenum).get(i);
    System.out.print(i + word);
    outputStream.writeBytes(word);

    for(int j = 1; j <= doublemap.size(); j++) {

        int ans = 2;
        if(doublemap.get(j).containsValue(word)) {
            ans = 1;
        }else {
            ans = 0;
        }

        System.out.print(ans);
        a[i][j-1] = ans;
        outputStream.writeBytes(ans + " ");
    }
```

Then, I build the matrix and find the smallest number of shingle and represent

```
for(int j = 0; j < a[0].length; j++) {
    int temp = 100000;

    for(int i = 0; i < a.length; i++) {
        if(a[i][j] == 1) {

            if(getAllKeysForValue(orimap.get(filenum), doublemap.get(filenum).get(i)) < temp) {

                temp = getAllKeysForValue(orimap.get(filenum), doublemap.get(filenum).get(i));
            }
        }
    }

    b[j] = temp;
    if (b[j] == 100000) {
        b[j] = 0;
    }
}

String stringtotal = filenum + ",";

for(int i = 0; i < b.length; i++) {

    stringtotal = stringtotal + b[i];

    if(i < b.length - 1) {
        stringtotal = stringtotal + ",";
    }
```

The way I generate different hash functions.

How to pick a random hash function h(x)?
Universal hashing:
$h_{a,b}(x)=((a\cdot x+b) \bmod p) \bmod N$
where:
a,b … random integers
p … prime number (p > N)

The way I generate different prime numbers of hash functions. I search the next 100 numbers of input and determine whether it is a prime number or not with the function isPrimeBruteForce.

```
public static int generateprime(int num) {
    int prime = 0;

    for(int i = num + 1; i < num + 100; i ++) {
        if(isPrimeBruteForce(i)) {
            prime = i;
            break;
        }
    }

    return(prime);
}

public static boolean isPrimeBruteForce(int number) {

    for (int i = 2; i < number; i++) {

        if (number % i == 0) {
            return false;
        }
    }

    return true;
}
```

Let’s go to the third part, locality sensitivity hashing. We use the results which processed from the second part.

```
[root@sandbox-hdp target]# yarn jar LSH.jar LSH  /hw02/Output002 /hw02/Output003
```

Then, we put the candidate pairs into the hash function and show the buckets which contain over 2 files.

```
1       3:0,3,24,41;5:10,29;6:12,19,20,21,25;7:11,37,38;8:27,32;9:14,18,22,28;
10      1:1,19;2:18,22,37;4:2,3,42;5:12,20,29;6:25,26,33,38;7:6,13,48;8:9,16,17,21,23,28,30,43;9:0
11      1:4,25,26;2:29,47;3:0,2,13,21;4:19,30;5:40,43;6:9,32,34,42;7:12,18,20,22,24,28,33,48;8:1,1
12      0:16,21;1:14,45;2:2,9,13,17;3:19,31,33,49;4:3,30,35,38;5:18,27,28,40,47,48;7:0,26,41;8:12,
13      1:0,16,28,44;2:13,21,31,34,45,49;3:24,27,35;5:4,19,38;6:10,18,39;7:12,20,40;9:3,23,33,48;
14      0:24,34,38,42;1:32,44;2:2,9,17,21,40;3:16,18,27;4:3,31;5:0,1,49;7:14,28;8:12,20,25;
15      1:11,34,47;2:2,19;3:32,40,43;4:3,12,20,41;5:13,37,48;7:10,14,24,45;8:6,38;9:0,18,22;
16      1:3,6,9,29;4:2,5,13,18,38,45;5:11,14,25;6:10,28;
17      1:3,14,38,40,41,43;2:22,26,35,48;3:19,21,34;4:9,11,13,16,17,42;5:8,28,33,47;6:4,25,30;7:0,
18      1:0,23,33,40,48;3:16,26,49;
```

I set the band number to key and set the row of signature matrix to value.

```
for(int i = 1; i <= 50; i++) {

    if(i <= 49) {

        string = string + a[i] + ",";
    }else {

        string = string + a[i];
    }
}

file1.set(filenum+"");
file2.set(values);

context.write(file1, file2);
```

The locality sensitivity hashing I used is I add the two of rows in a band and moderate 10 into different buckets.

```
int hash = (a[0][i] + a[1][i]) % 10;

if(hash == 0 && a[0][i] != 0 && a[1][i] != 0){
    list0.add(i);
}else if(hash == 1) {
    list1.add(i);
}else if(hash == 2) {
    list2.add(i);
}else if(hash == 3) {
    list3.add(i);
}else if(hash == 4) {
    list4.add(i);
}else if(hash == 5) {
    list5.add(i);
}else if(hash == 6) {
    list6.add(i);
}else if(hash == 7) {
    list7.add(i);
}else if(hash == 8) {
    list8.add(i);
}else if(hash == 9) {
    list9.add(i);
}
```

Finally, we execute Similarity

```
[root@sandbox-hdp target]# yarn jar Similarity.jar Similarity  /hw02/Output002 /
hw02/Output003
```

We will get the answer.txt which represent the top 10 candidate pairs and similarities.

```
12,20=1.0
15,46=0.84
5,15=0.84
5,36=0.82
7,36=0.82
15,45=0.78
4,15=0.78
4,36=0.78
15,44=0.76
37,46=0.74
```

We get the signature matrix firstly.

```
 for(int i = 0; i < 50; i++) {
     String string = "";
     string = br.readLine();

     String total[] = string.split(",");

     for(int j = 0; j < total.length ; j++) {
         if(j < total.length - 1) {
             signa[i][j] = Integer.parseInt(total[j + 1]);

         }
     }
}
```

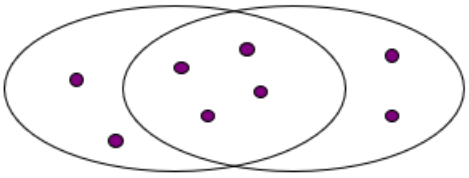
Then, we count the similarities of candidate pairs and get the highest one.

```
for(int n = 0; n < 25; n++) {
    String temp[] = br2.readLine().split("\t");

    String temp1[] = temp[1].split(";");

    for(int i = 0; i < temp1.length ; i ++) {
        String temp2[] = temp1[i].split(":");
        String temp3[] = temp2[1].split(",");

        for(int j = 0; j < temp3.length ; j ++) {

            for(int k = j + 1; k < temp3.length; k++) {

                double sim = countsim(temp3[j], temp3[k], signa);
                String set = (Integer.parseInt(temp3[j]) + 1) + "," + (Integer.parseInt(temp3[k]) + 1);

                if(map.get(set) == null) {
                    map.put(set, sim);

                }else if(map.get(set) != null && sim > map.get(set)) {
                    map.put(set, sim);

                }
            }
        }
    }
}
```

This is how we calculate similarities. I get the one set of candidate pairs and get the signature matrix of the two columns and use Jaccard similarity.

Jaccard similarity:

$$sim(D_1, D_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$



```
public static double countsim(String a, String b, int siga[][]) {
    int count = 0;
    double sim = 0.0;

    for(int i = 0; i < 50; i++) {

        if(siga[i][Integer.parseInt(a)] == siga[i][Integer.parseInt(b)]) {
            count++;
        }
    }

    sim = count / 50.0;

    return(sim);
}
```

And sorted by value; get the top 10 pairs and similarities.

```
Set<Entry<String, Double>> set = map.entrySet();
List<Entry<String, Double>> list = new ArrayList<Entry<String, Double>>(set);

Collections.sort( list, new Comparator<Map.Entry<String, Double>>()
{
  public int compare( Map.Entry<String, Double> o1, Map.Entry<String, Double> o2 )
  {
    int result = (o2.getValue()).compareTo( o1.getValue() );

    if (result != 0) {
      return result;

    } else {
      return o1.getKey().compareTo(o2.getKey());

    }
  }
} );
```