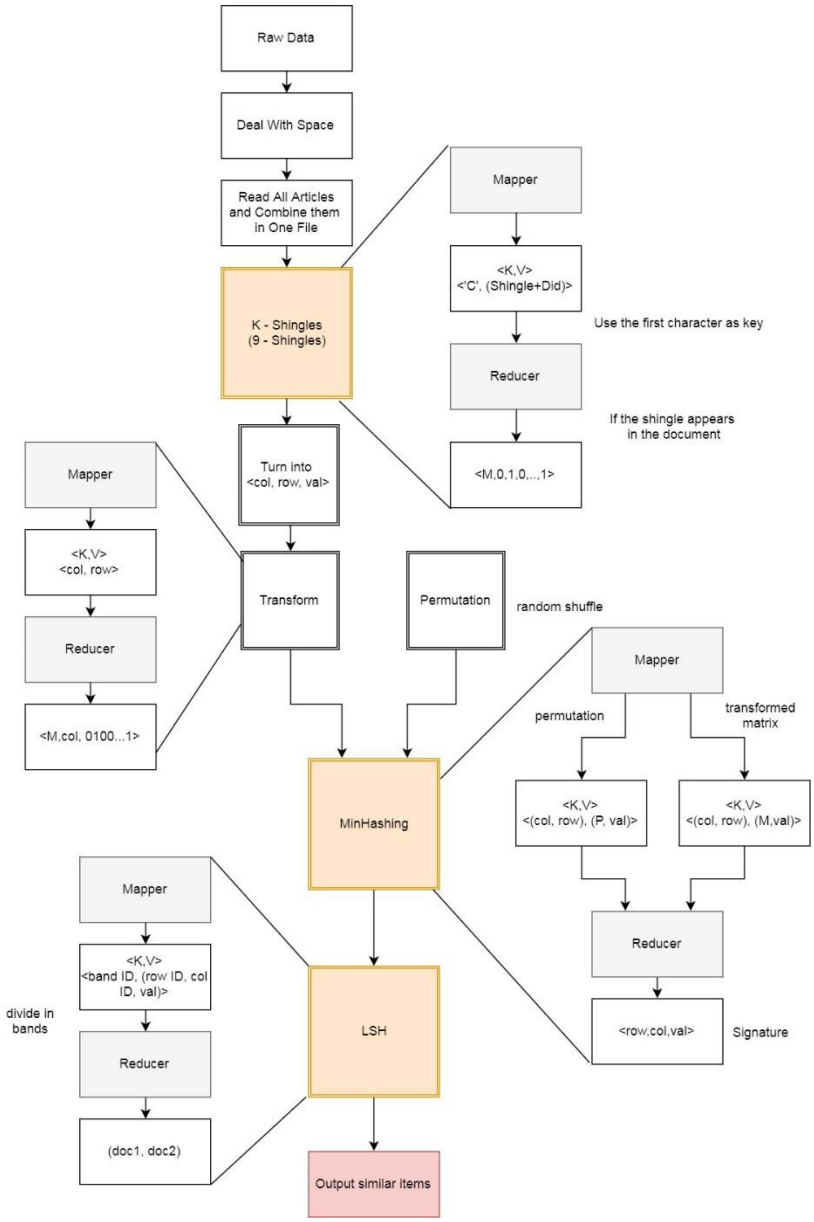
Final Project (Finding Similar Items)
Members:
103062308 陳祖培
103062218 鍾祐霖
103062205 劉凌君

Introduction:

We want to use LSH algorithm to find similar articles. For some articles, we set a threshold and show the similar articles as results. In the following, we will issue our implementation and the experiment results of different coefficients.

Flow Chart:



Implementation:

a. Dealing the raw text:

Since we want to generate shingles by characters, we need to filter the line symbols and all the space symbols into only one space symbol.

```
1,(CNN)Simona Halep declared on the eve of the French Open that about "15 players" were the favorite but the way she has played at Roland Garros in the last week, the number has dwindled to one -- and it is the Romanian herself. The 2014 finalist would have been the heavy favorite had she not sustained an ankle injury in the final of the Italian Open last month against Elina Svitolina -- a week after defending her crown at the Madrid Open. However, the third seed has brushed aside any fitness concerns by reaching the last eight without conceding a set. READ: Halep ... and chocolate mousse On Monday she crushed Carla Suarez Navarro -- the 21st-seeded Spaniard who had beaten Halep in their four previous clay duels -- 6-1 6-1 on a blustery day in Paris to set up a rematch with Svitolina, who rallied from 5-2 down in the third set to end the run of Croatian qualifier Petra Martic 4-6 6-3 7-5. After Venus Williams and Svetlana Kuznetsova lost Sunday, the French Open -- which is missing Serena Williams, Maria Sharapova and Victoria Azarenka -- is guaranteed of producing a first-time grand slam winner. Halep hopes it will be her. Her great form, she said, can be attributed to a more positive attitude on court since early April. "I work hard and I changed," Halep told reporters, Monday. "I changed pretty fast." Why fast? Halep revealed that her highly respected coac
```

b. Shingling:

We use k-shingles and count it with characters. Hence, we implement the first MapReduce job to generate all shingles.

In the mapper, we just generate all shingles and emit them to reducer. We use the first character of the shingle as key. Then, we use the shingle and document Id as value. In this way, the amount of reducers will decrease.

```
String keyName = shingle.toString();
context.write(new Text(String.valueOf(keyName.charAt(0))),new Text(keyName+key.toString()));
```

In the reducer, each will get shingles from every document, and then we need to generate a matrix representing whether the shingle appears in the document.

```
M,0,0,1,0,0,0,0,0,0,0
M,0,0,0,0,0,0,1,0,1,0
M,0,0,0,0,0,0,0,0,0,1
M,0,0,0,0,0,1,0,0,0,0
M,0,0,0,0,1,0,0,0,0,0
M,0,0,0,0,1,0,0,0,0,0
M,0,0,0,0,0,0,0,1,0,0
M,1,1,1,1,0,0,0,0,0,0
M,0,0,0,0,0,0,0,1,0,0
M,0,0,0,0,0,1,0,0,0,0
M,0,0,0,0,0,0,0,0,1,0
M,1,1,1,1,1,1,1,0,0,0
M,0,0,0,0,0,0,1,0,0,0
M,0,0,0,0,1,0,0,0,0,0
M,0,0,0,0,0,0,1,0,0,0
M,0,0,0,0,0,0,0,1,0,0
M,0,0,0,0,0,0,0,0,1,0
M,0,0,0,0,0,1,0,0,0,0
M,0,0,0,0,0,1,0,0,0,0
M,0,0,0,0,0,0,0,1,0,0
M,0,0,1,0,0,0,0,0,0,0
M,0,0,0,0,0,0,0,0,1,0
M,0,0,0,1,0,0,0,0,0,0
M,0,0,0,0,0,0,0,1,0,0
```

c. Pre-work of Transform

We need to transform the result of previous stage into the form of <row, column, value>. Otherwise, we don't know the id of shingles. We can't transpose the matrix.

d. Transform

The output of the previous stage is stored in the form like this (use shingle as column and document as row):

```
0,2,1
1,6,1
1,8,1
2,9,1
3,5,1
4,4,1
5,4,1
6,7,1
7,0,1
7,1,1
7,2,1
7,3,1
8,7,1
9,5,1
10,8,1
11,0,1
11,1,1
11,2,1
11,3,1
11,4,1
11,5,1
11,6,1
12,6,1
13,4,1
14,6,1
```

However, we need to transform it into the form like this (use document as row and shingle as column) so that it could be used in the next stage.

```
M,0,00000001000100000000000000000000010100100000
00000001100000000000001000000001000000000010000
00000000000010000011000010000100100000110001000
10010010000000000000011010000101100001000011100
00000100000110101001000000000010101000111000001
01000011000010000000100011000000000000101101100
10000001000100000000000100010000000000001000000
00000000000000001010000010000010000000010000000
01000011100000010000110000000100000000000011000
00100000000000000110110000000011000000000000000
10000000001000000110110100000000000000100000001
10100010100010100010001001000000000000000100000
01000000100010000011000000010001100101000000000
00000000110000000000000101000000000100000001011
11000010000000000110000000100000001000001000000
```

Hence, we need to design a MapReduce job. First, in the mapper, we just emit it in this form: <K, V> = <columnID, (rowID,value)>. Next, in the reducer, we will get the entries from each column. Finally, we could output the matrix in the form using documents as row index.

e. Minhashing

In this stage, we need to generate permutation first and compress the amount of shingles to the amount of permutation times. To generate the permutation, we use shuffle function and store all the possibilities in a text file. Then, we utilize it and the transformed matrix as the input of the job.

```
for (int i=0;i<sizeOfSignature;i++){
  Collections.shuffle(solution);
  IOUtils.write("P,"+Integer.toString(i)+",", os);
  for (int j=0;j<solution.size();j++){
    IOUtils.write(Integer.toString(solution.get(j)), os);
    if (j!=solution.size()-1){
      IOUtils.write("-", os);
    }
  }
  IOUtils.write("\n", os);
}
os.close();
```

In the mapper, we need to determine the input comes from the permutation file or matrix file. If it belongs to the permutation file, we emit it for every document.

```
if (matrix.contains("P")){
  for(int i=0; i<numOfDocuments;i++){
      context.write(new Text(data[1]+","+Integer.toString(i)), new Text("P," + data[2]));
  }
}
```

If it belongs to the matrix file, we throw it for every result of permutation.

```
if (matrix.contains("M")){
  for(int i=0; i<numOfSignatures; i++){
    context.write(new Text(Integer.toString(i)+","+data[1]), new Text("M," + data[2]));
  }
}
```

In the reducer, we could receive it for every entry of the signature (result of minhashing). For each reducer, we could get one result of permutation and one row of document. Then, we just do hashing according to the permutation and get the signature of this entry.

```
0,0,7
0,1,1
0,2,7
0,3,7
0,4,7
0,5,7
0,6,5
0,7,14
0,8,4
0,9,2
1,0,3
1,1,3
1,2,3
1,3,3
1,4,3
1,5,1
1,6,5
1,7,5
1,8,5
1,9,4
10,0,3
10,1,3
10,2,3
10,3,3
10,4,3
10,5,3
10,6,3
10,7,3
10,8,1
10,9,2
```

f. LSH

In this stage, we need to divide the matrix into bands, so we design a MapReduce job for it.

In the mapper, we divide the signature id by the band size and use this as key so that we could classify it by band.

```
String rowPerBand = conf.get("rowPerBand");
String[] data = value.toString().split(",");
int bandNum = Integer.parseInt(data[0])/Integer.parseInt(rowPerBand);
int newRowId = Integer.parseInt(data[0])%Integer.parseInt(rowPerBand);
context.write(new Text(Integer.toString(bandNum)), new Text(Integer.toString(newRowId)+","+data[1]+","+data[2]));
```

In the reducer, we compare all the signatures in this band, if all the signatures of two documents in the band are the same, and we will regard they are the similar items. In the details, we use a hashmap to determine it. We concatenate all the signature values as hash key, so if there are more than two values in one entry, that means these two items are identical in this band.

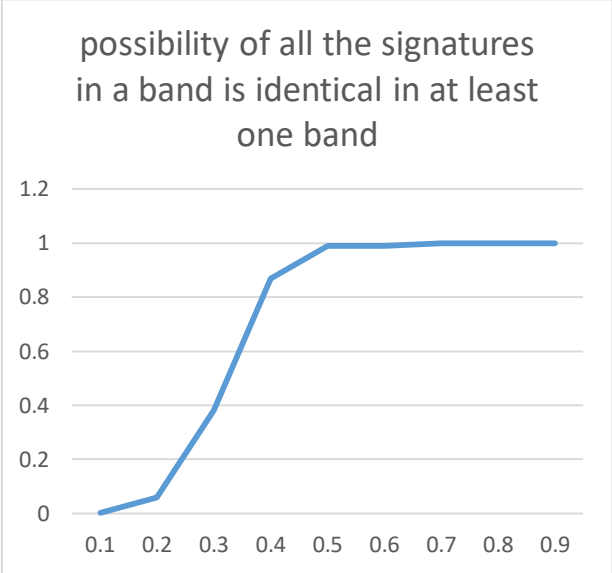
Mathematic Analysis:

a. Choosing number of bands (permutation times)

According to the Mathematic analysis in the content of the ppt, we derive a formula stating that the possibility of all the signatures in a band is identical in at least one of *n* bands, where a band contents *r* signatures, is $1-(1-P^r)^n$. Where P = the similarity ratio of the signatures between documents. We than analyze the filtering threshold in different settings of rows per band (*r*).

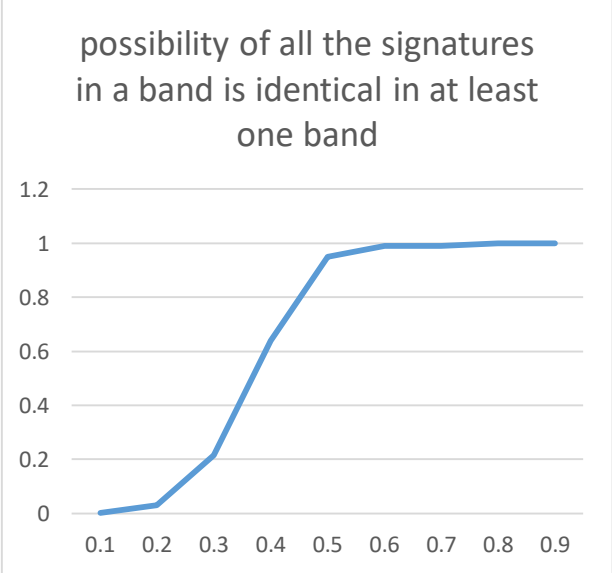
#Rows = 1000, r = 5

similarity ratio of the signatures between documents.	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*
possibility of all the signatures in a band is identical in at least one band	*0.001*	*0.06*	*0.38*	*0.87*	*0.99*	*0.99*	*1*	*1*	*1*



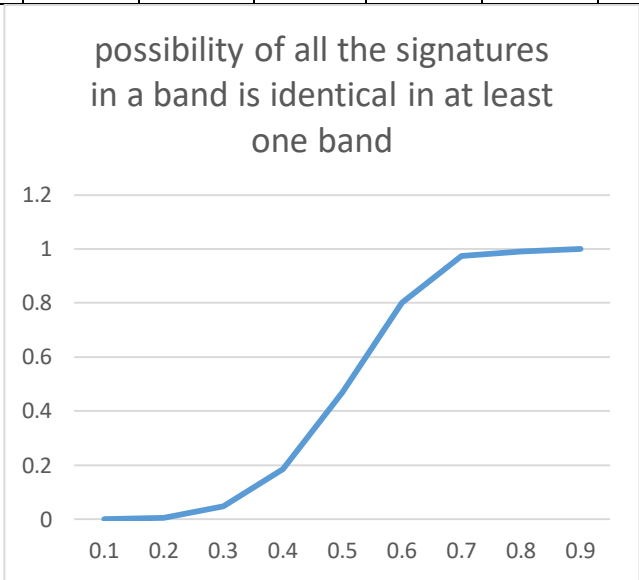
#filtering threshold≈0.3~0.4
#Rows = 500, r = 5

similarity ratio of the signatures between documents.	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*
possibility of all the signatures in a band is identical in at least one band	*0.0009*	*0.03*	*0.215*	*0.64*	*0.95*	*0.99*	*0.99*	*1*	*1*



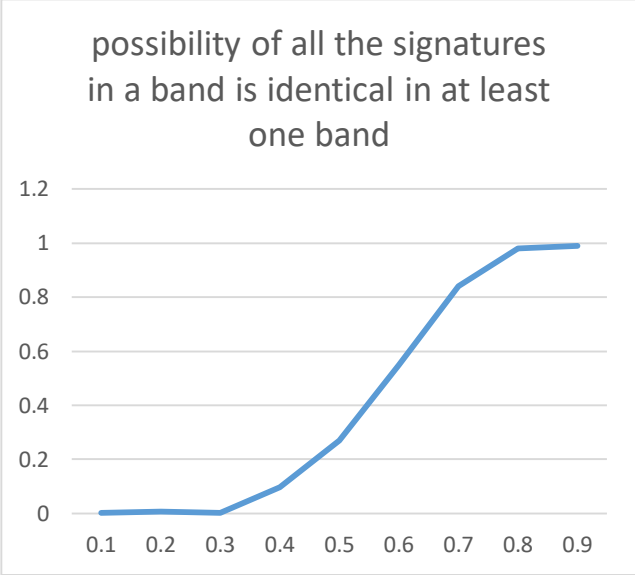
#filtering threshold≈0.4
#Rows = 100, r = 5

similarity ratio of the signatures between documents.	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*
possibility of all the signatures in a band is identical in at least one band	*0.001*	*0.006*	*0.047*	*0.186*	*0.47*	*0.802*	*0.975*	*0.99*	*1*



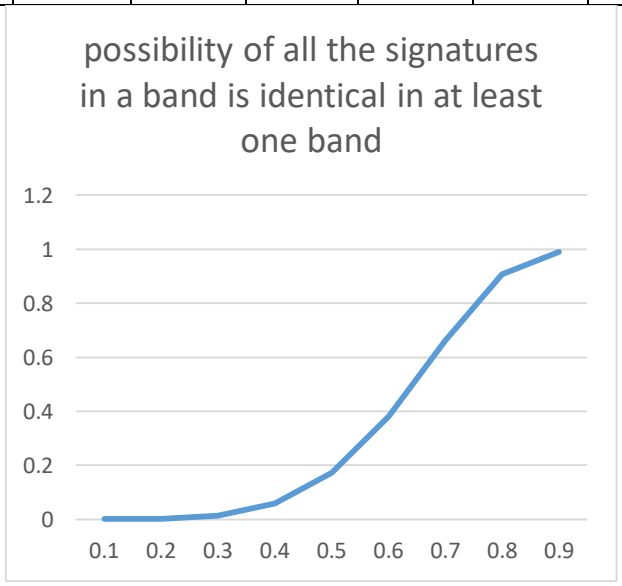
#filtering threshold≈0.5~0.6
#Rows = 50, r = 5

similarity ratio of the signatures between documents.	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*
possibility of all the signatures in a band is identical in at least one band	*0.001*	*0.006*	*0.003*	*0.0978*	*0.27*	*0.55*	*0.84*	*0.98*	*0.99*



#filtering threshold≈0.6
#Rows = 30, r = 5

similarity ratio of the signatures between documents.	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*
possibility of all the signatures in a band is identical in at least one band	*0.001*	*0.0019*	*0.014*	*0.059*	*0.173*	*0.38*	*0.66*	*0.907*	*0.99*



#filtering threshold≈0.7

Thus, we can change the filtering threshold by changing the number of bands (permutations) to acquire the corresponding result we want. If we're working on duplicated detection or plagiarism, we can set the permutation time to 30 or lower, while if the case is similar items recommendation, we can set the permutation time to 500 or even 1000.

In the experiment, we make up 10 documents in different content similarity to show the different results.

b. Choosing shingle length

We observe that the length of shingle we choose will affect the similarity between the original document and its according signatures. And how long is appropriate for a document depends on how long the document is.

We experiment two different shingle length in our experiment to show the difference.

c. Things we observe about document length

We observe that the different length of documents will affect total number of shingles generated and thus affect the permutation result, so we make up our testcases to make them at roughly the same length.

Design of test cases:

We use one article as base. Then, the others are composed of certain percentage of the article and different kinds of articles. There are almost the same number of words in ten text files.

File Name	Index	Content
News1	0	The whole article
News2	1	90% of news1 and 10% of different article
News3	2	80% of news1 and 20% of different article
News4	3	70% of news1 and 30% of different article
News5	4	60% of news1 and 40% of different article
News6	5	50% of news1 and 50% of different article
News7	6	40% of news1 and 60% of different article
News8	7	30% of news1 and 70% of different article
News9	8	20% of news1 and 80% of different article
News10	9	10% of news1 and 90% of different article

Experiment Result:

5-Shingle with permutation time 30

```
1  [0, 1, 3, 4]
2  [1, 2]
3  [0, 2]
4  [0, 1, 2, 3]
```

bucket 1: the smallest similarity of articles is 60%

The threshold of testcase is 70%, but the result appears 60%. According to the mathematic analysis, the possibility that 60% similar article appears is 38%.

5-Shingle with permutation time 50

```
1  [1, 2, 3, 4, 5, 6]
2  [1, 2]
3  [0, 1]
4  [0, 2]
```

bucket 1: the smallest similarity of articles is 40%
The threshold of testcase is 60%, but the result appears 40%. According to the mathematic analysis, the possibility that 40% similar article appears is 9%.

5-Shingle with permutation time 100

```
1   [2, 3]
2   [0, 1, 2]
3   [0, 1, 2, 3]
4   [5, 7]
5   [0, 1, 2, 3, 6]
6   [0, 1, 3]
7   [0, 1, 2, 3, 4]
8   [3, 4]
9   [3, 5]
10  [0, 2]
11  [1, 2]
```

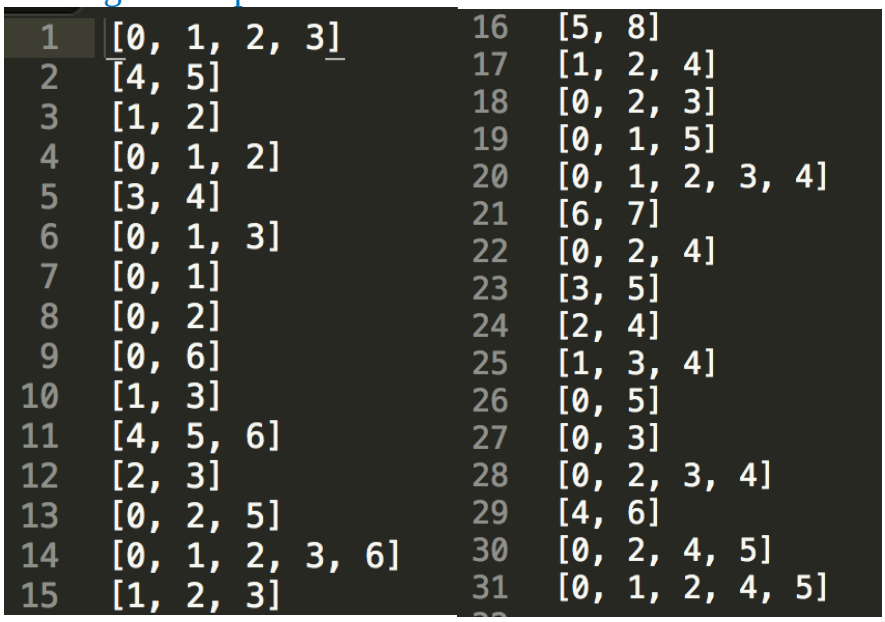
bucket 5: the smallest similarity of articles is 40%
The threshold of testcase is 50%~60%, but the result appears 40%. According to the mathematic analysis, the possibility that 40% similar article appears is 18%.

5-Shingle with permutation time 500

```
1   [0, 1, 2, 4]
2   [0, 1]
3   [1, 2]
4   [0, 3]
5   [0, 1, 2, 3, 4]
6   [0, 1, 2]
7   [1, 2, 3, 4]
8   [1, 3]
9   [0, 2]
10  [0, 1, 2, 3]
11  [0, 1, 3, 5]
12  [4, 6]
13  [0, 1, 3]
14  [1, 4]
15  [4, 5]
16  [0, 2, 3]
17  [3, 4]
18  [2, 3, 5]
19  [0, 5]
20  [3, 5]
21  [0, 2, 4]
22  [5, 7]
23  [0, 2, 4, 6, 7]
24  [0, 3]
25  [1, 2, 3]
26  [0, 3]
27  [2, 3]
28  [0, 1, 4]
29  [1, 2, 3]
30  [1, 2, 4]
```

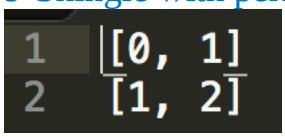
bucket 23: the smallest similarity of articles is 30%
The threshold of testcase is 40%, but the result appears 30%. According to the mathematic analysis, the possibility that 30% similar article appears is 21%.

5-Shingle with permutation time 1000



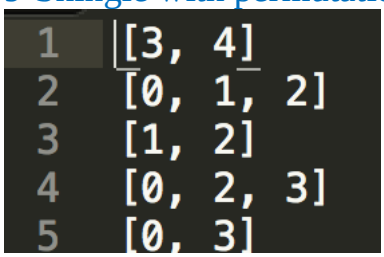
bucket 14: the smallest similarity of articles is 40%

9-Shingle with permutation time 30



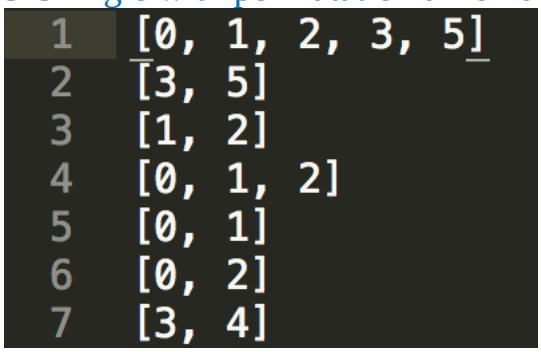
the similarity of articles in the buckets is 90%
The threshold of testcase is 70%~80%.

9-Shingle with permutation time 50



bucket 4: the smallest similarity of articles is 70%
The threshold of testcase is 60%.

9-Shingle with permutation time 100



bucket 4: the smallest similarity of articles is 50%
The threshold of testcase is 50~60%.

9-Shingle with permutation time 500

```
[0, 2, 3]
[0, 1]
[1, 2, 3]
[0, 2, 4]
[1, 2]
[0, 1, 2, 6]
[0, 1, 2]
[0, 1, 2, 3]
[0, 1, 2, 3, 4]
[2, 3, 5]
[0, 1, 3]
[0, 2]
[0, 1, 2, 4]
[3, 4, 5]
[0, 3]
[3, 4]
[2, 3]
[1, 5]
[1, 2, 3, 5]
```

bucket 6: the smallest similarity of articles is 40%
The threshold of testcase is 40%.

9-Shingle with permutation time 1000

```
[1, 2]
[1, 2, 4]
[0, 1, 3]
[0, 2]
[0, 1]
[0, 2, 3, 4]
[2, 3]
[0, 3]
[2, 5]
[1, 3]
[0, 1, 2]
[3, 4]
[0, 4]
[1, 2, 5]
[1, 2, 3]
[2, 4]
[0, 1, 2, 3]
[1, 2 ,3]
[5, 7]
[6, 7]
[0, 2, 3, 5, 6]
[0, 1, 2, 4]
[0, 3, 4]
[0, 1, 2, 3, 5]
[0, 2, 3]
[6, 8]
```

bucket 21: the smallest similarity of articles is 40%
The threshold of testcase is 30%~40%.

Conclussion:

The results of 9-shingles are better than the results of 5-shingles. Our documents are probably more suitable for 9-shingles.