Assignment4 Report

Name: Ke Wang NetID: kw427

1. Implementation

Shingles.java: Specify the structure of the shingles, including its label id, content, and an array list of document id that contains the shingle. I removed all the non-letter words.
Doc.java: Store the document id and the Jaccard coefficient.
NumberPair.java: Specify the pair of random numbers a and b.
ReadFile.java: Read the files from test file, label the shingles and store the document id into *ArrayList<Shingles> shingles*, the corresponding structures.
NearDuplicates.java: the main class of the core code. It calculates 25 random permutations of the 3-grams, calculates the sketch of each document, and compares all the pairs of documents in the test data set to find the pairs with J(d1,d2) > 0.5.

PartI: calculate 25 random permutations of the 3-grams
getRandom() generates 25 pairs of random numbers a and b for random permutations.

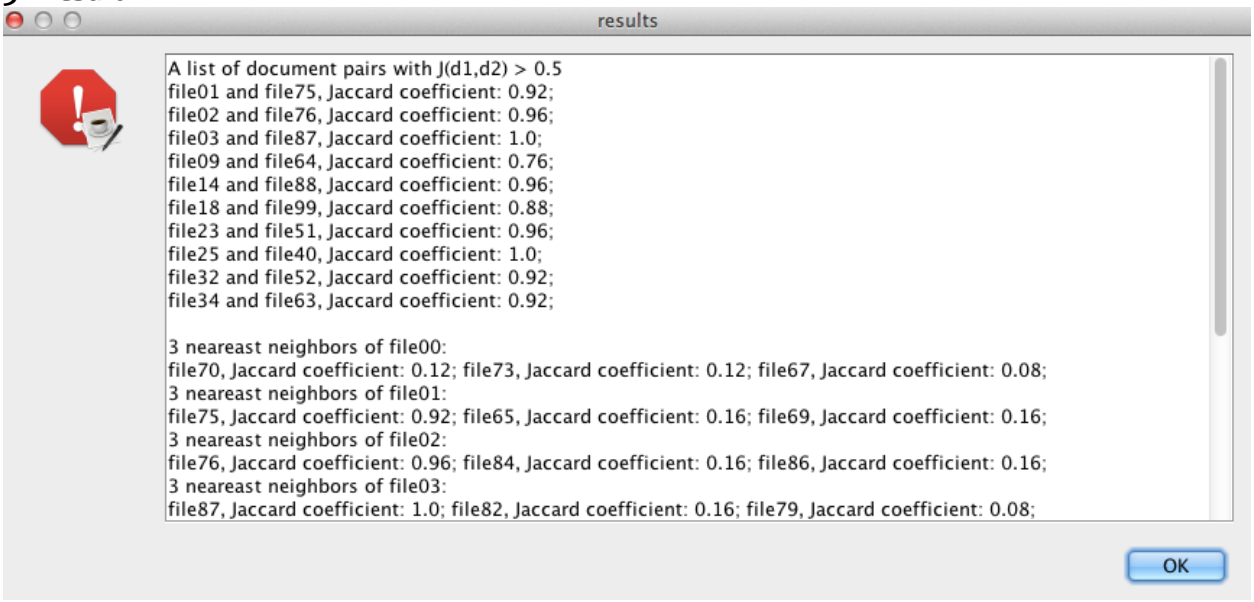
PartII: calculate the sketch of each document
getSketch() applies the function f(x)=(ax + b) mod p to each label, and retain only the smallest resulting value, store the matching shingles into the sketch of each document.

PartIII: compare all the pairs of documents in the test data set
compare() counts the number of corresponding sketch values that are equal of both documents, and calculate the Jaccard coefficient. Find the pairs with J(d1,d2) > 0.5.
Test.java: Test the project and show result dialog.

2. How to run

Method1: Run the runnable file *NearDuplicates.jar* by double clicking or in command line.
Method2: Import the project "NearDuplicates" into Eclipse, and run Test.java.
Note: Since we read files from local files this time instead of online data, we need to keep the test file and runnable jar file in the same directory, which is included in my whole submitted file.

3. Result



Since the sketches are generated randomly, the project will get results that are slightly different from each other. The results includes

1. A list of document pairs with J(d1,d2) > 0.5
2. For each of the first ten documents file00.txt to file09.txt, return the "three nearest neighbors".

An example is as below:

A list of document pairs with J(d1,d2) > 0.5
file01 and file75, Jaccard coefficient: 0.92;
file02 and file76, Jaccard coefficient: 0.96;
file03 and file87, Jaccard coefficient: 1.0;
file09 and file64, Jaccard coefficient: 0.76;
file14 and file88, Jaccard coefficient: 0.96;
file18 and file99, Jaccard coefficient: 0.88;
file23 and file51, Jaccard coefficient: 0.96;
file25 and file40, Jaccard coefficient: 1.0;
file32 and file52, Jaccard coefficient: 0.92;
file34 and file63, Jaccard coefficient: 0.92;

3 neareast neighbors of file00:
file70, Jaccard coefficient: 0.12; file73, Jaccard coefficient: 0.12; file67, Jaccard coefficient: 0.08;
3 neareast neighbors of file01:
file75, Jaccard coefficient: 0.92; file65, Jaccard coefficient: 0.16; file69, Jaccard coefficient: 0.16;
3 neareast neighbors of file02:
file76, Jaccard coefficient: 0.96; file84, Jaccard coefficient: 0.16; file86, Jaccard coefficient: 0.16;
3 neareast neighbors of file03:
file87, Jaccard coefficient: 1.0; file82, Jaccard coefficient: 0.16; file79, Jaccard coefficient: 0.08;
3 neareast neighbors of file04:
file65, Jaccard coefficient: 0.16; file66, Jaccard coefficient: 0.16; file72, Jaccard coefficient: 0.16;
3 neareast neighbors of file05:
file83, Jaccard coefficient: 0.12; file79, Jaccard coefficient: 0.08; file84, Jaccard coefficient: 0.08;
3 neareast neighbors of file06:
file79, Jaccard coefficient: 0.24; file77, Jaccard coefficient: 0.2; file78, Jaccard coefficient: 0.2;
3 neareast neighbors of file07:
file78, Jaccard coefficient: 0.12; file83, Jaccard coefficient: 0.12; file81, Jaccard coefficient: 0.08;
3 neareast neighbors of file08:
file77, Jaccard coefficient: 0.12; file81, Jaccard coefficient: 0.08; file84, Jaccard coefficient: 0.08;
3 neareast neighbors of file09:
file64, Jaccard coefficient: 0.76; file67, Jaccard coefficient: 0.2; file71, Jaccard coefficient: 0.2;