Homework 1: Finding Similar Items: Textually Similar Documents

Group 33

Group Member: Xitao Mo, Shihao Xu

1. Solution

We found the textually similar documents based on Jaccard similarity using the shingling, minhashing, and locality-sensitive hashing (LSH) techniques.

There are seven classes in our source codes:

1) In class FileHandler, we read all txt files from the gaven folder, and get the string of documents;
2) In class Shingling, we construct k–shingles of a given length k from the string of documents, compute a hash value for each unique shingle, and get an ordered set of its hashed k-shingles;
3) In class CompareSets, we compute the Jaccard similarity of two sets of integers, and get the jaccardSimilarity;
4) In class Minhashing, we transform the hashed k-shingles into a minHash signature, and compare the signatures to get the jaccardSimilarity;
5) In class Permutation and, we create the Permutation matrix;
6) In class LSH, we divide the minHash signature into some bands, compare the hashing of bands in different signatures, and finds all candidate pairs of signatures that agree on at least fraction t of their components;
7) In class Main, we read four novel files, shingling them with k=9, and calculate the jaccardSimilarity. Next, we transform the shingling into minHash signatures with permute number 1000, and calculate the similarity. Finally, we use LSH algorithm with bands=50, rows=20 to find candidate pairs, and calculate the Jaccard similarity with threshold=0.8 to find similar pairs.

2. How to build, run the code and command-line parameters

Run the main.java to calculate Jaccard similarity using the shingling, minhashing, and locality-sensitive hashing (LSH) techniques. The dataset is in the folder /src/resource.

3. Results

```
The Jaccard similarity of two sets is:0.0074487895716946
> generating characteristic matrix...
> generating signature matrix...
Time consumed: 0.594 sec
> Computing document-wise similarities:
1.0 0.008 0.006 0.023 0.006 0.008 0.004 0.005 0.005 0.005 0.0 0.0 0.002
0.008 1.0 0.006 0.007 0.004 0.016 0.008 0.03 0.002 0.009 0.0 0.0 0.003
0.006 0.006 1.0 0.002 0.002 0.007 0.006 0.004 0.007 0.0 0.003 0.003 0.004
0.023 0.007 0.002 1.0 0.002 0.013 0.012 0.016 0.003 0.004 0.0 0.0 0.006
0.006 0.004 0.002 0.002 1.0 0.0 0.007 0.009 0.003 0.001 0.002 0.002 0.006
0.008 0.016 0.007 0.013 0.0 1.0 0.017 0.006 0.003 0.003 0.0 0.0 0.005
0.004 0.008 0.006 0.012 0.007 0.017 1.0 0.005 0.006 0.001 0.0 0.0 0.005
0.005 0.03 0.004 0.016 0.009 0.006 0.005 1.0 0.01 0.003 0.001 0.001 0.005
0.005 0.002 0.007 0.003 0.003 0.003 0.006 0.01 1.0 0.005 0.002 0.002 0.002
0.005 0.009 0.0 0.004 0.001 0.003 0.001 0.003 0.005 1.0 0.002 0.002 0.004
0.0 0.0 0.003 0.0 0.002 0.0 0.0 0.001 0.002 0.002 1.0 1.0 0.003
0.0 0.0 0.003 0.0 0.002 0.0 0.0 0.001 0.002 0.002 1.0 1.0 0.003
0.002 0.003 0.004 0.006 0.006 0.005 0.005 0.005 0.002 0.004 0.003 0.003 1.0
CandidatePairs are:[[10, 11]]
1.0
SimilarPairs are:[[10, 11]]
```