ID2222 Data Mining

Niklas Barth and Yu Gao

**Lab Report of Finding Similar Items: Textually Similar Documents**

**Abstract**

In this lab, we use Jaccard similarity to identify textually similar documents by the shingling, minhashing, and locality-sensitive hashing (LSH) techniques. All the code is written in Python which is also adopted in Jupyter Notebook.

**Code Structure**

**Class Shingling**

This class takes a string document and size of k as the input and it returns a Hash integer set as its output. Here the document could be a short sentence like a header of news, or the full text of a report. k is the length of elements that each shingle contains. But it should be noted that we decided to use words instead of chars to speed up computation on our machines which is achieved by class DocumentCollection.

For example, Input: A= This is a document , B= This was , k = 2;

Output: Ahash={0,1,2,3}, Bhash={0,4}

**Class CompareSets**

This class takes two sets of integers which were created by class shingling. As the output, it will return the value of intersection between two sets divided by their union. It s also called Jaccard similarity which is the main criterion of similarity identification in our project.

For example: Ahash={0,1,2,3},Bhash={0,4}

Jaccard similarity = 1/5 = 0.2

**Class MinHashing**

Here we use a Hash integer set and the length of vector n as the input. The MinHashing function generates a n-length integer vector MinHash signature. This vector consists of the minimum values from (a + b\*x) % p for each element x in Hash integer sets. All the values of a, b, p are randomly generated n times and applied for each element in Hash integer sets by a for loop.

For example: Ahash={0,1,2,3}, n=6:

Return a 1x6 vector Asig={0,2,5,1,2,3}, the same with Bhash => Bsig={1,4,5,1,3,5}.

**Class CompareSignatures**

This class is aimed to calculate the similarity of two integer vectors that are MinHash signatures.

For example: Asig = {0,2,5,1,2,3}, Bsig = {1,4,5,1,3,5}: similarity = 2/6 = 0.33

**Class LSH (locality-sensitive hashing)**

This class uses a matrix of MinHash signatures as well as a integer bandwidth as the input. Document pairs which have at least one identical band in their MinHash signatures would be seen as candidate pairs.

For example: Asig = {0,2,5,1,2,3}, Bsig = {1,4,5,1,3,5}, bandwidth = 2:

there are 3 subset for each minhash signature,

Asig:{0,2},{5,1},{2,3}, Bsig:{1,4},{5,1},{3,5}.

Asig and Bsig are candidate pairs because the second subset between them are equal.

**Instructions for use**

1. Put your test documents as a csv file within the same folder of project
2. Start the main class
3. Check the length of csv file you have and decide the number of documents you want to test
4. Decide the value of k, n, threshold s and bandwidth
5. Run the main function

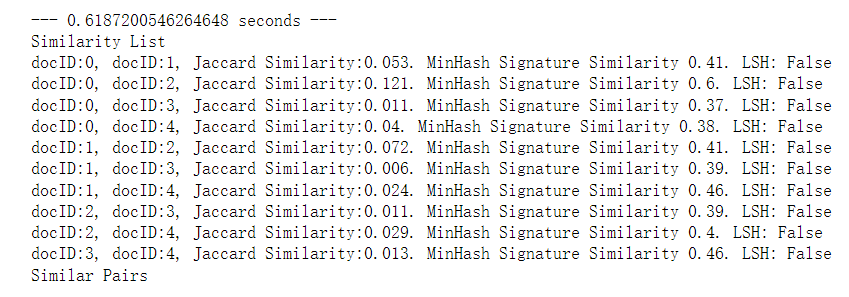
The document pairs with a Jaccard similarity which is higher than s as well as return True in LSH test will be seen as similar documents. And the test results of all pairs will also be shown in our project.

**About Data**

In this project, we copied 7 pieces of new of USA midterm election from different medias online. And Based on the first article, we created 3 new articles that have some changes from article 1 as the potential similar documents for it. Article 6 is created by omitting two sentences of article 1 at the beginning and middle part. Article 7 is created by replacing all of “it” with “them”. Article 8 is created by adding two more sentences to article 1 from article 2.

**Evaluate implementation's scalability**

We test the scalability of our project by counting the execution time with two different number of input articles.



*Figure 1: execution time for 5 articles*



*Figure 2: execution time for 10 articles*

As we can see, the time for testing 10 articles is 4 times more than that for 5 articles. Here we only change the number of input articles, but the value of k, n as well as bandwidth will also have impacts on the execution time.