HW1 - FINDING TEXTUALLY SIMILAR DOCUMENTS

Syed Arif Rahman - sarahman@kth.se
Group-10

Short Explanation of the program:

My program is written in Java and consists of five different classes, including the main class. These classes are Shingling, CompareSets, MinHashing, and CompareSignatures. Here's a brief overview of each class:

1. Shingling:
   This class includes the ShingleHashMap() function, which returns a HashMap<Integer, String> of stored keys. The function also generates a hash value for individual shingles using the Java hashcode() function.

2. CompareSets:
   The CompareSets class features the GetJaccardValue() function, which calculates the Jaccard similarities of two sets obtained from the Shingling class.

3. MinHashing:
   The MinHashing class incorporates the MinHashSignature() function with five parameters. These parameters are passed from the main class to include AllSet, n value, the current set, and a set containing random values (sets a and b). In this class, first save the current set and create a Boolean list for that set. An ArrayList of signatures is then created and filled with maximum values. Subsequently, iterate through each key of AllSet, compare it with the current set, and mark it using the Boolean List. Finally, traverse the Boolean list, and if it's true, then create a signature using predefined rules and save it to the signature set before returning it.

4. CompareSignatures:
   Similar to Jaccard similarities, the CompareSignatures class includes the similarities_of_sign() function, which returns the comparison between two given signatures, providing the desired data.

How to build:

The datasets are stored in a file named "dataset" under the src(source) folder of java project with each dataset saved in the format (1.csv, 2.csv, etc.). If you wish to experiment with a new dataset, ensure it follows the same format, and then simply click the "run" button in the Main class. Here datasets are sourced from the Economic Freedom dataset for the years 2017 to 2021, with each year having its own CSV file. You can access the dataset via the following link: https://www.fraserinstitute.org/economic-freedom/dataset?geozone=world&min-year=2&max-year=0&filter=0&page=dataset

Results:

I utilized five CSV files to develop our program, assigning a numeric value n = 500. Additionally, I generated two random lists, a and b, each containing 500 random data points using a specific formula. The following showcases a snippet of shingles from the first CSV, where each shingle comprises six characters:

Shingle hash of CSV 1: {-1476220926=a,"Fre, 1566727522=54,10., 1502435725=3,8.20, 1502435727=3,8.22, 1447056962=1.91,9, 1502435726=3,8.21, 1674011586=9,18.6, ...}

Here's a brief excerpt of shingles from CSV 5, where each shingle consists of six characters:

Shingle hash of CSV 5: {-1476220926=a,"Fre, 1502435725=3,8.20, 1566727522=54,10., 1447056961=1.91,8, 1674011589=9,18.9, 1502435727=3,8.22, 1502435726=3,8.21, 1445484071=1,AUT,, ...}

Output in milliseconds of 5 csv file Shingle is : `[116, 386, 37, 170, 127]`

Next, the Jaccard similarities are computed:

CompareSets set1 and set1: 1.0
CompareSets set1 and set2: 0.5088043932430013
CompareSets set1 and set3: 0.48499307825984506
CompareSets set1 and set4: 0.47022299956274594
CompareSets set1 and set5: 0.4631450700548255

The code then compiles MinHash signatures, preceded by the generation of two random lists, a and b. A concise overview of MinHash signatures for the first two CSV files is presented:

Signature of MinHash

Signature for CSV 1: [1, 1, 1, 13, 6, 1, 1, 3, 1, 0, 3, 0, 1, 1, ...]
Signature for CSV 2: [3, 0, 1, 13, 6, 1, 1, 1, 1, 0, 1, 2, 1, 1, ...]

Output in milliseconds of 5 csv file Minhash Signature is : `[3309, 4382, 1686, 1739, 1662]`

Finally, the comparison of signatures is conducted. Here is an overview of CSV 5 signatures with all other CSV files:

Signature for CSV 5: [0, 6, 0, 3, 0, 78, 78, 2, 4, 6, 0, 1, 2, 0, 0, ...]
Signature for CSV 1: [0, 0, 0, 2, 0, 78, 78, 0, 0, 6, 0, 0, 6, 0, 1, ...]
Similarity of signature for CSV 5 and CSV 1: 0.666
Signature for CSV 2: [1, 0, 1, 0, 0, 78, 78, 0, 1, 6, 0, 1, 0, 1, 0, ...]
Similarity of signature for CSV 5 and CSV 2: 0.754
Signature for CSV 3: [0, 0, 1, 3, 0, 78, 78, 0, 0, 6, 0, 1, 1, 1, 0, ...]
Similarity of signature for CSV 5 and CSV 3: 0.772
Signature for CSV 4: [0, 1, 1, 2, 0, 78, 78, 0, 0, 6, 0, 1, 2, 1, 0, ...]
Similarity of signature for CSV 5 and CSV 4: 0.792
Signature for CSV 5: [0, 6, 0, 3, 0, 78, 78, 2, 4, 6, 0, 1, 2, 0, 0, ...]
Similarity of signature for CSV 5 and CSV 5: 1.0

Conclusion:

In conclusion, the analysis reveals that the Jaccard similarity and signature similarity exhibit minimal discrepancies. Nevertheless, utilizing the MinHash technique yields better improved results and a more favorable output but takes slightly more time. The smaller value of k contributes to a higher percentage of similarities.