Homework 1: Finding similar items

Homework Group 54: Xu Wang(xuwa@kth.se)

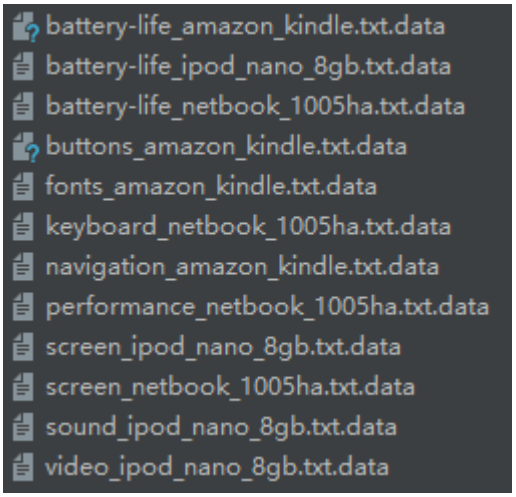
- *Instruction*

The goal of this homework is to find similar documents in a dataset given a similarity threshold s. To run the script, you should first prepare the dataset then simply run the following command in the terminal:

```
python3 main.py
```

- *Dataset & Preprocessing*

The original dataset is composed of user reviews on different products: UCI Machine Learning Repository: Opinosis Opinion／Review Data Set. I selected 12 text files that are relevant in topics, namely:



For data preprocessing, I removed non-ascii characters, punctuations and empty lines in the file. I also converted all characters to lowercase.

- *Shingling*

In this class, we computed the shingles of each input document and hash shingles to integers. The length of shingles is set to **5** since the texts are mainly short comments on websites. Finally all hashed shingles were saved in a list called *dataset_shingles*, whose components are hashed shingles from each document.

Number of shingles in each document:

```
Document 0: 4905 shingles
Document 1: 3263 shingles
Document 2: 12251 shingles
Document 3: 7362 shingles
Document 4: 3982 shingles
Document 5: 5289 shingles
Document 6: 3877 shingles
Document 7: 3408 shingles
Document 8: 3293 shingles
Document 9: 8421 shingles
Document 10: 3983 shingles
Document 11: 7362 shingles
```

- *CompareSets*

After getting the hashed shingles of each document, we can compute the jaccard similarity matrix, where entry (i,j) is the jaccard similarity of hashed shingle sets of document i and document j.

```
jaccard_similarity_matrix:
 [[1.    0.173 0.197 0.209 0.172 0.179 0.165 0.149 0.141 0.187 0.165 0.178]
 [0.    1.    0.146 0.145 0.152 0.166 0.151 0.145 0.227 0.153 0.235 0.212]
 [0.    0.    1.    0.222 0.155 0.227 0.148 0.178 0.126 0.292 0.159 0.218]
 [0.    0.    0.    1.    0.209 0.204 0.22  0.13  0.145 0.22  0.167 0.2  ]
 [0.    0.    0.    0.    1.    0.176 0.182 0.141 0.151 0.172 0.156 0.172]
 [0.    0.    0.    0.    0.    1.    0.172 0.165 0.157 0.277 0.176 0.186]
 [0.    0.    0.    0.    0.    0.    1.    0.14  0.147 0.168 0.153 0.165]
 [0.    0.    0.    0.    0.    0.    0.    1.    0.121 0.163 0.14  0.135]
 [0.    0.    0.    0.    0.    0.    0.    0.    1.    0.15  0.197 0.241]
 [0.    0.    0.    0.    0.    0.    0.    0.    0.    1.    0.169 0.214]
 [0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    1.    0.259]
 [0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    1.   ]]
```

I highlighted several top similarities and the goal of the LSH class is to find these pairs.

- *Minhashing*

In this class I first generated 100 hash functions in the form of (a*x+b) mod r, where r is a large prime number and a,b are random integers ranging from 1 to (r-1). Then

for each document, these hash functions were applied to its hashed shingles and the min value was selected each time. The 100 values became the signature of the document.

- *CompareSignatures*

The similarity of two signatures with the same length is simply the fraction of identical components in the two signatures. We can then compute the signature similarity matrix:

```
signature_similarity_matrix:
 [[1.   0.14 0.21 0.22 0.18 0.19 0.17 0.13 0.15 0.13 0.15 0.21]
 [0.   1.   0.18 0.09 0.17 0.21 0.14 0.13 0.24 0.15 0.24 0.21]
 [0.   0.   1.   0.31 0.24 0.26 0.21 0.16 0.16 0.31 0.19 0.27]
 [0.   0.   0.   1.   0.21 0.24 0.16 0.18 0.17 0.3  0.18 0.22]
 [0.   0.   0.   0.   1.   0.26 0.26 0.24 0.24 0.25 0.17 0.22]
 [0.   0.   0.   0.   0.   1.   0.14 0.21 0.21 0.3  0.19 0.27]
 [0.   0.   0.   0.   0.   0.   1.   0.2  0.19 0.25 0.17 0.21]
 [0.   0.   0.   0.   0.   0.   0.   1.   0.2  0.22 0.18 0.15]
 [0.   0.   0.   0.   0.   0.   0.   0.   1.   0.16 0.17 0.22]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   1.   0.13 0.2 ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   1.   0.29]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   1.  ]]
```

- *(Optional) LSH*

The idea of LSH is to divide the signature matrix into b bands with r rows in each band. For each band, apply a hash function and check whether ‘sub-signatures’ from different documents are hashed to the same busket. The candidate pair is the pair whose signatures have been hashed to the same busket for at least one band. Then if the signature similarity of the candidate pair is beyond the threshold s, we check the true jaccard similarity.

To balance the false positives/negatives trade-off, I wrote a function that finds the (number_of_bands, number_of_rows) pair whose theoretical threshold is the closest to the given threshold. Here I set the threshold to be 0.25 according to the jaccard similarity matrix, and the algorithm yield the following results:

```
Number of bands: 50, number of rows: 2
```

```
Candidate pair:  fonts_amazon_kindle.txt.data screen_netbook_1005ha.txt.data
Signature similarity:  0.25 | Jaccard similarity:  0.172
Candidate pair:  fonts_amazon_kindle.txt.data navigation_amazon_kindle.txt.data
Signature similarity:  0.26 | Jaccard similarity:  0.182
Candidate pair:  battery-life_netbook_1005ha.txt.data keyboard_netbook_1005ha.txt.data
Signature similarity:  0.26 | Jaccard similarity:  0.227
Candidate pair:  battery-life_netbook_1005ha.txt.data video_ipod_nano_8gb.txt.data
Signature similarity:  0.27 | Jaccard similarity:  0.218
Candidate pair:  fonts_amazon_kindle.txt.data keyboard_netbook_1005ha.txt.data
Signature similarity:  0.26 | Jaccard similarity:  0.176
Candidate pair:  buttons_amazon_kindle.txt.data screen_netbook_1005ha.txt.data
Signature similarity:  0.3 | Jaccard similarity:  0.22
Candidate pair:  keyboard_netbook_1005ha.txt.data screen_netbook_1005ha.txt.data
Signature similarity:  0.3 | Jaccard similarity:  0.277
Candidate pair:  sound_ipod_nano_8gb.txt.data video_ipod_nano_8gb.txt.data
Signature similarity:  0.29 | Jaccard similarity:  0.259
Candidate pair:  keyboard_netbook_1005ha.txt.data video_ipod_nano_8gb.txt.data
Signature similarity:  0.27 | Jaccard similarity:  0.186
Candidate pair:  battery-life_netbook_1005ha.txt.data buttons_amazon_kindle.txt.data
Signature similarity:  0.31 | Jaccard similarity:  0.222
Candidate pair:  battery-life_netbook_1005ha.txt.data screen_netbook_1005ha.txt.data
Signature similarity:  0.31 | Jaccard similarity:  0.292
Candidate pair:  navigation_amazon_kindle.txt.data screen_netbook_1005ha.txt.data
Signature similarity:  0.25 | Jaccard similarity:  0.168
```

```
Total runtime is  25.643863916397095
```