2021 Introduction to Massive Data Analysis

Term Project

Deadline: 2022.01.05 (Wed.)

Question 1: Finding Similar Articles

> Given a set of BBCSports articles, Implement LSH using MapReduce to find out articles similarity.

According to Ch.3, 3 essential steps are needed for similar docs. Thus, this assignment should contain these 3 parts.

1. **Shingling :** Convert documents to sets. Ch.3 part1 p.17~25 The first part of this assignment is to implement the k-shingle, but this time we are going to use "**words**" instead of "**characters**". You should read in all the articles of the topic, and then translate them into shingles. After this part, your output may look like this example, which depends on how you implement your own Shingling.
   Note that you should use **3-shingles**.

Shingles \ Document				
	1	1	1	0
	1	1	0	1
	0	1	0	1
	0	0	0	1
	1	0	0	1
	1	1	1	0
	1	0	1	0

2. **Min-hashing** : Covert large sets to short signatures, while preserving similarity.
   The goal in this part is that you should hash your shingles into smaller size to get "**signatures**".
   As we know, permuting rows even once is prohibitive because the cost is too high. However, there is an implementation trick.

As described in Ch.3 part1 p.43~p.44, instead of permuting rows, you can simply generate different hash functions to simulate the permutation step according to the algorithm.
Note that you need **100 different hash functions**.

3. **Locality-sensitive hashing** : Focus on pairs of signatures likely to be from similar documents.
   Finally, we can apply LSH to the signatures. You should first partition your signature matrix M into b bands and r rows, where **b = 50** and **r = 2**.
   Then, for each band, you should hash its portion of each column to a hash table with **k buckets**.
   The details are all in Ch.3 part1 p.53~60.
   Note that candidate pairs are those that **hash to the same bucket for ≧ 1 band**. After getting all the candidate pairs, you can calculate the **Jaccard Similarity** and then output the **top 10**.

Data format:

Input: Several files in English and number.

Output: "(%s, %s): %f"

Top 10 indices of candidate pairs and their similarities in **decreasingly** order (increasingly ordered by index number if similarities are equal.).
Each line should be: filename1, filename2, similarity (round to the two non-zero digit after decimal point), and filename1 should be smaller than filename2

```
(001, 002): 95.00%
(003, 005): 95.00%
(001, 010): 90.00%
(001, 007): 80.31%
(002, 010): 65.26%
```

Question 2: Recommendation System: Item-item Collaborative Filtering

There are 2 sections of question 2. Finish basic section will get 70 points only.

1. **Basic Section**: Similarity

   From lecture Recommendation Systems P.27 ~ 30

   Similarity: cosine similarity with subtract mean

$$sim(x, y) = cos(r_x, r_y) = \frac{r_x \cdot r_y}{\|r_x\| \cdot \|r_y\|}$$

   Output format: (item, item), similarity

2. **Advanced Section:** Rating Predictions
   Select **top 10 similarity** to calculate the movie rating for each user.
   i.e., N = 10

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

   Output format: (user, item), rating

Data format:

Input: (userID, movieID, rating)

ml-latest-small.zip From MovieLens: https://grouplens.org/datasets/movielens/

It contains 610 users and 9742 movies.

Output: "(%d, %d): %f"

Output all result sorted by item/user ID in increasing order.

Assignment Requirement:

Select 1 topic above. Finish all requirements of the topic will get 90 points, you can explain your additional work to get more points.

Part1 Code

Upload the code.
Should implement all mapper and reducer in one file.
Please make sure that your file is named as
Term_Project_studentID.java or **Term_Project_studentID.ipynb**

Part2 Report ()

Explain how do you design your mapper and reducer.
Named as **Report.pdf** or write the markdown in .ipynb file.

Please pack the above files into a zip file. Name it as
"Term_Project_studentID.zip"