INF 553 – Fall 2018

Homework #3: LSH

Due: October 23, Tuesday
100 points

In this assignment, we will consider movie recommendations problem. You are asked to write a Spark Python program which uses LSH to efficiently find similar users and make recommendations. You can use the sample offered to test your code and compare to the output, but I will create a new dataset for final grade.

A. Problem

Suppose **there are 100 different movies, numbered from 0 to 99**. A user is represented as a set of movies. Jaccard coefficient is used to measure the similarity of sets.

- Finding similar users: Apply minhash to obtain **a signature of 20 values** for each user. Recall that this is done by permuting the rows of characteristic matrix of movie-user matrix (i.e., row are movies and columns represent users).

  Assume that the i-th hash function for the signature: **h(x,i) = (3x + 13i) % 100**, where x is the original row number in the matrix. Note that i ranges from 0 to 19.

  Apply LSH to speed up the process of finding similar users, where the signature is **divided into 5 bands, with 4 values in each band**.

- Making recommendations: Based on the LSH result, for each user U, find top-5 users who are most similar to U (by their Jaccard similarity, if same, choose the user that has smallest ID), and recommend top-3 movies to U where the movies are ordered by the number of these top-5 users who have watched them (if same, choose the movie that has the smallest ID).

- Computation of signatures and LSH needs to be done in parallel.
  - Hint: While computing signatures, you can divide users (represented as a set of movies) into partitions and compute signatures for the partitions in parallel.
  - Hint: While computing LSH, you could take the band as key, the user ID as value, and then find the candidate pairs/users in parallel.
  - Hint: You need to compute the Jaccard similarities of similar pairs identified by LSH, based on which you find the top-5 users and top-3 movies.

B. Input format

You are provided with a file where each line represents a user (with ID "U1" for example) and a list of movies the user has watched (e.g., movie #0, 12, 45). Each user has watched at least one movie.

```
U1,0,12,45
U2,2,3,5,99
…
```

C. Output format

For each user with that we could find similar users, output the recommended movies as follows. No recommendations for the user if there is no similar user found.

```
U1,5,20,38
U2,1,5,12
…
```

Users should be output in increasing order. And the movie should be ordered by the number of the top-5 users who have watched them (If same, the movie that has smallest ID comes first).

It is possible to find that the total number of top-5 users watched movies is less than 3. It's fine. Just output all in this situation.

For simplicity, the recommended movies may contain the movie that user has already watched. You don't need to remove it.

D. Format of execution

bin/spark-submit <FirstName>_<LastName>_ lshrec.py input.txt output.txt

E. Submission

Submit a Python script in the form: john_smith_lshrec.py, that is your first and last names followed by the name of program.

F. Notes

1. The program takes 2 arguments:
    - Input file path
    - Output file path
2. You must use the minHash functions and hash functions provided in this instruction, or your output would be different with the standard solution.
3. Make sure to follow the output format and the submission naming format. If you don't follow either one or both of them, 20% points will be deducted.
4. We will be using Moss for plagiarism detection.