INF 553 – Spring 2017

Homework #3: LSH

Due: March 10, Friday
100 points

In this assignment, we will consider the problem of using LSH to find similar users, based on the fraction of the movies they have watched in common. You are asked to write a Spark Python program which uses LSH to efficiently find similar users.

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.

- Apply minhash to obtain **a signature of 20 values** for each user. Recall that this is done by "logically" 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.
- 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**.
- Finding similar users: 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). If there aren't 5 similar users, e.g. U2 is only similar with U5, U9, U10, just output U2:U5,U9,U10.
- Computation of signatures, LSH, and Jaccard similarities of similar users need to be done in parallel.
    - Hint: While computing LSH, you could take the signatures of one band as key, the user ID as value, and then find the candidate pairs.

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 their **top-5** similar users as follows.

```
U2:U9,U10
U3:U8
U8:U3
…
```

Users should be output in the **increasing** order of the integer part (e.g., 1, 5, 10) of their IDs (e.g., U1, U5, U10).

Similar users for each user are ordered in the same way. If there are not 5 similar users after applying LSH, i.e. U2 is only similar with U9, U10, output all them (U9, U10).

Format of execution:

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

D. Time limit

Your program will be graded on a c4.2xlarge EC2 instance, and need to output result within 1 hour for each test case. The largest test case would be as large as the attached examples_test_cases/case_l/input.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. **DO NOT** use fixed input/output path. Read them from program arguments. The program takes 2 arguments:
    a. Path to input file
    b. Path to output file
2. You must use the minHash functions, b, and p values provided in the instructions, or your output would be different from the standard solution.
3. Make sure you 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. Do not copy from each other or you will face serious consequences!**