INF 553 – Spring 2017
Assignment 4: LSH For Similar Movies

Deadline: 04/03/2017 11:59 PM PST

Assignment Overview

In this assignment you are asked to implement the LSH technique using the Spark Framework in order to identify similar movies based on the ratings of the users. **You can use any Spark version that is greater or equal to 1.6 but you need to specify the version in your description file.** You will use only one dataset in order to find the similar movies as described in the following sections. The goal of this assignment is to help you understand how you can use the LSH algorithm to identify similar items in an efficient way by programming it within a distributed environment.

Write your own code!

For this assignment to be an effective learning experience, you must write your own code!
Do not share code with other students in the class!!
Here's why:

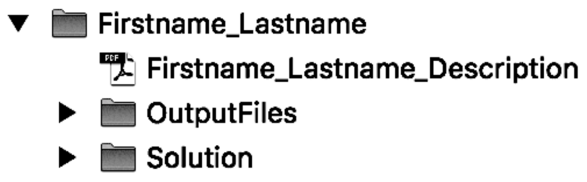
- The most obvious reason is that it will be a huge temptation to cheat: if you include code written by anyone else in your solution to the assignment, you will be cheating. As mentioned in the syllabus, this is a very serious offense, and may lead to you failing the class.
- However, even if you do not directly include any code you look at in your solution, it surely will influence your coding. Put another way, it will short-circuit the process of you figuring out how to solve the problem, and will thus decrease how much you learn.

So, just don't look on the web for any code relevant to these problems. Don't do it.

Submission Details

For this assignment you will need to turn in a Python, Java, or Scala program depending on your language of preference. We will test your code using the same dataset in order to verify its correctness. This assignment will surely need some time to be implemented so please plan accordingly and start early!

Your submission must be a .zip file with name: **<Firstname>_<Lastname>_hw4.zip**. The structure of your submission should be identical as shown below. The *Firstname_Lastname_Description.pdf* file contains helpful instructions on how to run your code along with the answers to the questions of the following sections. The *OutputFiles* directory contains the required deliverable output files and the *Solution* directory contains your source code.



```
▼ Firstname_Lastname
      Firstname_Lastname_Description
  ▶ OutputFiles
  ▶ Solution
```

LSH Algorithm

In this assignment you will need to develop the LSH technique using the *ml-latest-small/ratings.csv* file from the MovieLens dataset. The dataset is provided to you under the */data* folder of the bundled .zip file of the assignment. The goal of this assignment is to find similar movies according to the ratings of the users. In order to solve this problem, you will need to read carefully the sections 3.3 – 3.5 from Chapter 3 of the Mining of Massive Datasets book.

In this problem, we will focus on 0-1 ratings rather than the actual ratings of the users. To be more specific, if a user has rated a movie, then his contribution to the characteristic matrix is 1 while if he hasn't rated a movie his contribution is 0. **Our goal is to identify similar movies whose Jaccard Similarity is greater or equal to 0.5.**

Implementation Guidelines - Approach

The original characteristic matrix must be of size [users] x [movies]. Each cell contains a 0 or 1 value depending on whether the user has rated the movie or not. Once the matrix is built you are free to use any collection of hash functions that you think would result in a more consistent permutation of the row entries of the characteristic matrix.

Some potential hash functions could be of type:

$f(x)= (ax + b) \% m$
or
$f(x) = ((ax + b) \% p) \% m$

where p is *any prime number* and m is the *number of bins*.
You can use any value for the *a, b, p or m* parameters of your implementation.

Once you have computed all the required hash values you must build the Signature Matrix. Once the Signature Matrix is built you must divide the Matrix into b bands with r rows each, where *bands x rows = n (n is the number of hash functions),* in order to generate the candidate pairs. Remember that in order two movies to be a candidate pair their signature must agree (i.e. be identical) with at least one band.

Once you have computed the candidate pairs, your final result will be the candidate pairs whose Jaccard Similarity is greater or equal to 0.5. After computing the final similar items, you must compare your results against the provided ground truth dataset using the precision and recall metrics. The ground truth dataset contains all the movies pairs that have Jaccard similarity above or equal to 0.5. The ground truth dataset is located under the */data* folder of the assignment's bundled .zip file, and named as *SimilarMovies.Goundtruth.05.csv.*

Example of Jaccard Similarity:

	user1	**user2**	**user3**	**user4**
movie1	0	1	1	1
movie2	0	1	0	0

Jaccard Similarity (movie1, movie2) = ***#movies_intersection / #movies_union = 1/3***

Execution Example

The program that you will implement should take two parameters as input and generate one file as an output. The first parameter must be the location of the *ratings.csv* file and the **second one must be the path to the output file followed by the name of the output file. The name of the output file must be *firstname_lastname_SimilarMovies.txt*.** The content of the file must follow the exact same directions of question 1 in the *Questions & Grades Breakdown section* below. If your program does not generate this file or it does not follow the specifications as described in the following section, there would be a penalty of 50%.

Java/Scala Execution Example:

```
./bin/spark-submit --master local[*] --class <main_class_name> <jar_file> <input_file> <output_path>/firstname_lastname_SimilarMovies.txt
```

Python Execution Example:

```
./bin/spark-submit --master local[*] <python_script> <input_file> <output_path>/firstname_lastname_SimilarMovies.txt
```

Finally, inside the description file of your submission write how to run your code, which Spark version you used and if you use Java or Scala, specify the language versions as well.

Questions & Grade Breakdown

1. **A file that contains the pairs of similar movies that your algorithm has computed using LSH followed by their Jaccard similarity**.
   (40 Points)

   Example Format:
   movie$_1$, movie$_2$, Jaccard-Similarity$_{12}$
   movie$_1$, movie$_3$, Jaccard-Similarity$_{13}$
   …
   movie$_n$, movie$_k$, Jaccard-Similarity$_{nk}$

   The file must be sorted first by the left-hand side movie and then by the right-hand side movie. For example, the first row on the above snippet should be shorted firstly by movie$_1$ and secondly by movie$_2$.

   The file should be located under the *OutputFiles directory* of your submission with the name:
   firstname_lastname_SimilarMovies.txt

2. **For Jaccard Similarity >= 0.5 compute the precision and recall for the similar movies that your algorithm has generated against the ground truth data file under the data folder in the .zip file.**
   (40 Points)

   Expressions:
   Precision = tp / (tp + fp)
   Recall = tp / (tp + fn)

   tp: true positives
   fp: false positives
   fn: false negatives

   In order to get full credit for this question you should have *precision >= 0.9 and recall >= 0.75*. If not, then you will get partial credit based on the formula:

   *(Precision/0.9) * 20 + (Recall/0.75) * 20*

3. **In your description file you must include a table similar to the following table from Section 3.4.2 of Chapter 3 from the Mining Massive Datasets book (Figure 3.8).**
   (15 Points)

   The table should include the values of the S-Curve for the number of bands (b) and rows (r) you have divided your signature Matrix.

   For instance, the following table is computed using b=20 and r=5

s	**$1-(1-s^r)^b$**
0.2	0.006
0.3	0.047
0.4	0.186
0.5	0.470
0.6	0.802
0.7	0.975
0.8	0.9996

4. **In your description file, for the table you have computed above, answer to the following question.**
   (5 Points)

   How does the value of *b* and *r* affect the value of the threshold for the movies dataset?

General Instructions:

1. Make sure your code compiles before submitting
2. Make sure to follow the output format and the naming format.

Grading Criteria:

1. If your programs cannot be run with the commands you provide, your submission will be graded based on the result files you submit and 20% penalty for it.
2. **If the input parameters are more than the ones requested there will be 50% penalty.**
3. **If the output file does not follow the specifications requested there will be 50% penalty.**
4. **If your program generates more than one file, there will be 20% penalty.**
5. **If you don't provide the source code and just the .jar file in case of a Java/Scala application there will be 60% penalty.**
6. **If your submission does not state inside the Description pdf file how to run your code and which Spark version you used there will be a penalty of 20%.**
7. There will be 20% penalty for late submission.

Appendix A

A.1 There is no limit on the number of hash functions you can use.
A.2 You could resemble the number of bands with the number of partitions of your RDD.
A.3 If you want to improve your algorithm's efficiency you could use the Vectors from Spark MLlib to perform your operations.