INF 553 – Spring 2017 Assignment 5

Deadline: 04/24/2017 11:59 PM PST

Assignment Overview (9 points)

In this assignment, you are asked to implement the Girvan-Newman algorithm using the Spark Framework to detect communities in a graph. You can use any Spark version that is greater or equal to 1.6. You will use only *ratings.csv* dataset to find users who have the similar movie taste to construct the graph. The goal of this assignment is to help you understand how to use the Girvan-Newman algorithm to detect communities 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!!

Submission Details

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

- **For Python**, your submission must be a **<Firstname>_<Lastname>_hw5.zip** file containing two python scripts **<Firstname>_<Lastname>_community.py** which contains both betweenness and modularity functions and **<Firstname>_<Lastname>_betweenness.py** which only contains betweenness function.

- **For Scala**, your submission must be a **<Firstname>_<Lastname>_hw5.zip** file, in which includes **community.jar** which contains both betweenness and modularity functions, **betweennes.jar** which only contains betweenness function and **Scala source codes**.

Execution Example

The program that you will implement should take two parameters. One is the location of input file (i.e., rating.csv). The other is the path to the output file followed by the name of the output file.

The name of the output file must be **firstname_lastname_communities.txt** for <Firstname>_<Lastname>_hw5.py and community.jar.

For <Firstname>_<Lastname>_betweenness.py and betweenness .jar, the output file name should be **firstname_lastname_betweenness.txt**. The **main class** name for betweenness .jar should be **Betweenness** (capitalize the first letter). The **main class** name for community.jar should be **Community** (capitalize the first letter). The example of execution example is as follows:

For Scala:

./bin/spark-submit –-master local[*] –class <main_class_name> <jar_file> <inputfile_path>/<input_file> <outputfile_path>/ firstname_lastname_communities.txt

For Python:

./bin/spark-submit –-master local[*] <python_script> <inputfile_path>/<input_file> <outputfile_path>/ firstname_lastname_communities.txt

Details

Dataset

rating.csv is in *ml-20m.zip*, which can be downloaded from the following link: https://grouplens.org/datasets/movielens/

Construct the graph

Each node represents a user. Each edge is generated in following way. In rating.csv, you will count the number of times that two users rated the same movie. If the number of times is **greater or equivalent to three**, there is an edge between these two users.

Represent the graph

For Python, you can use GraphFrame, you can learn more about GraphFrame from the link: https://graphframes.github.io/user-guide.html

For Scala, you can use GraphFrame or GraphX, you can learn more about GraphX from the link: http://spark.apache.org/docs/latest/graphx-programming-guide.html

Betweenness and Modularity

You are required to implement betweenness and modularity according to the lecture by yourself. The betweenness function should be calculated **only once from the original graph**. You can write a betweenness function and a modularity function and call the two functions several times until you find the max modularity value.

Output Format

- For firstname_lastname_communities.txt

Each list is a community, in which contains userIds. In each list, the userIds should be in ascending order. And all lists should be ordered by the first userId in each list in ascending order. An example is as follows: (the example just shows the format, is not a solution)

```
[1,5,6,7,9]
[2,3,10,11,14]
[4,8,12,13]
```

- For firstname_lastname_betweenness.txt

Each line is a tuple, the format is like (userId1, userId2, betweenness value). The file is ordered by the first element in ascending order and if the first element is the same, ordered by the second element. The example is as follows: (the example just shows the format, is not a solution)

```
(1,2,3.0)
(1,4,4.5)
(2,3,5.0)
(2,4,1.5)
(3,4,3.0)
```

Grading

Your codes will be tested with the rating.csv file.

The grading guideline is as follows:

- The output does not follow the format, there will be 20% penalty.
- The codes cannot be run in the terminal by using the command mentioned above, there will be 20% penalty.
- 20% penalty is for the late submission.
- The execution time is longer than thirty minutes, your codes will be test on a small dataset, and there will be 40% penalty for the long execution.
- If the result of firstname_lastname_communities.txt is not correct, but firstname_lastname_betweenness.txt is correct, 50% points will be subtracted.
- There will be 20% penalty for late submission.

Bonus (1.8 points)

There are some libraries containing the community detection function. Please detect communities by using community detection function in SparklingGraph. Here is the link to learn more about it: http://sparkling-graph.readthedocs.io/en/latest/comunities.html. The graph is constructed in the same way as mentioned above.