Assignment 02 Date: 2nd October, 2019 Last Date of submission: 11th October, 2019
Late submission attracts appropriate penalty

Title: From a bipartite Dataset, use projection to get uni-partite graph and find community. Also compare community (based on structure) with clusters (based on attributes).

Marks : 10 (20 will be scaled down to 10)

Learning goal: *Learn to work with a bi-partite data set*

Team :
1. Same group as assignment 01
2. It will be team marking

Marking Guidelines

Marking scheme	**Timeliness (5 marks)**	**Functionality asked for (5 marks each for community and cluster)**	**Analysis to compare community and cluster (5 marks)**
Max (5)	on time	working code without issue	Insightful analysis
Mid (4)	on time +1-7 days	working with minor issues	Analysis does not conclude anything
Min (3)	delay more than 7 days	working with major issues	no analysis

Submission procedure:

(a) **Update the teaminformation google sheet in your course folder, update team information (if not already done).**
(b) Filename for submission will have the format : **A01<Team_XX>.zip**
(c) The zip should contain the following (all files necessary to check your work)
  a. Any intermediate file (example- list of characters that you used)
  b. Jupyter notebook with python 3.x code
  c. ***Last part of the jupyter notebook should have a brief section highlighting* your key finding in the analysis**
(d) Mail ZIP file to Bhaskarjyoti01@gmail.com on or before due date

Instruction:

Dataset : MovieLens 100K dataset https://grouplens.org/datasets/movielens/100k/ . This data set consists of 100,000 ratings (1-5) from 943 users on 1682 movies when each user has rated at least 20 movies. Additionally you can find the dataset in the assignment folder.

The file u.data has a tab separated list of **user id | item id | rating | timestamp**. The file u.item has got the movie information (item id from u.data is same as movie id from u.item) that is represented by different genre columns. You can then join u.data and u.item based on item id and movie id. u.user has the demographic information about each user represented by user id.

(a) You are advised to extract a <u>**random sample**</u> of **400 users (out of 943 users)** for computational ease and work with that. Use a seed in the random number generation so that you have a result that you can replicate otherwise your analysis will not hold in the next run. While creating smaller dataset of this bipartite graph, **you have to make sure that all movie information for your selected user ids** are retrieved from u.item.
(b) Now the dataset is that of a bipartite graph (user and movie) where attributes of both user (demographic info) and movie (genre) are available.
(c) Through a simple projection mechanism in networkx, create uni-partite **movie network.**
(d) Do a community analysis to find out **community of movie**.
(e) Since the attributes are available for movies, do a **clustering of movies next**
(f) Compare the **clusters vs. community** and comment in the analysis section of your jupyter notebook based on your findings

Pseudo code for creation of bipartite graph

1. There are 943 users. Use random number generator with a seed to create a **userlist** of **400 userid** so that the userid is in the range of user ids in u.data
2. Extract u.datasmall from u.data using the above **userlist**
3. From the u.datasmall, extract the set of item id into **itemlist**
4. Using **itemlist**, extract u.itemsmall from u.item
5. Now you will work with u.datasmall and u.itemsmall. Both together will give you a bipartite dataset
6. Load u.datasmall into Pandas. Every row of this dataframe now represents an edge of the bipartite graph (corresponding nodes are in userId and itemID)
7. Import bipartite from networkx.algorithms.
8. To create bipartite graph from u.datasmall, you need to have a tuple each for a row of u.datasmall (userId, itemId). One possible way of doing this :
    a. user_list = [x for x in df["UserID"]]
    b. item_list = [str(x) for x in df["ItemID"]] # to have string otherwise #bipaprtite wont work
    c. touples = [(user_list[i], item_list[i]) for i in range(len(user_list))]
    d. Instatiate a networkx graph bip
    e. bip.add_nodes_from(user_list, bipartite=0)
    f. bip.add_nodes_from(item_list, bipartite=1)
    g. bip.add_edges_from(touples)
    h. item_nodes, user_nodes = bipartite.sets(bip)
    i. user_proj = bipartite.projected_graph(bip, user_nodes)
    j. item_proj = bipartite.projected_graph(bip, item_nodes)
    k. Now you can use various community detection algorithms