Homework 2

Data Mining Technology for Business and Society

Deadline: **25** May 2016

Exactly Two students for each group

Data and software are available at:

http://www.diag.uniroma1.it/~fazzone/Teaching/Data_Mining_Technology_for_Business_and_Society_2016_2017/DMT4BaS_2016_2017.html

Description:

In this homework you will solve recommendation problems using link-analysis techniques. In particular, you have to perform graph transformation and use Topic-Specific-PageRank on the transformed graph to obtain recommendations.
You will test your methods on MovieLens 100K Dataset. This dataset contains 100000 ratings of 943 users on 1682 movies. Each rating is a natural number in [1, 5]. This dataset is already splitted in five (training_set, test_set) couples. You will use these five (training_set, test_set) couples to evaluate the quality of your recommendation method.

The homework is splitted in two parts:

> In the first part, you have to implement a movie recommendation method using a particular link-analysis procedure (explained below). For this part, it is also required to evaluate the quality of the provided method. For this purpose, you will use the average Normalized Discounted Cumulative Gain metric (explained below, **because it is different from the one used in the first homework**).
>
> In the second part, you are invited to create a movie recommendation method for **groups** of users, by extending the method implemented in the first part. For this part it is not required to evaluate the quality of the recommendation method.

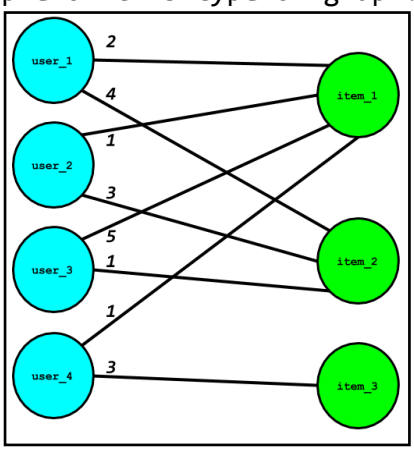
To complete the homework, you have to fill the files
`Network_Based_Recommendation_System_FUNCTIONS.py` and
`Network_Based_Recommendation_System_FOR_GROUPS.py`.
The file `Network_Based_Recommendation_System.py` contains the software that must be run to complete the first part of the homework. This software calls all functions stored in the file `Network_Based_Recommendation_System_FUNCTIONS.py` to perform and test the solution provided to complete the first part of the homework.
To complete the second part, you have to write your code directly inside the file `Network_Based_Recommendation_System_FOR_GROUPS.py` and run the file by itself.
In particular, you must write your code where the tag '`# Your code here ;)`' is located.
VERY IMPORTANT RULE:
Do not remove or change what is already written in the files. You can only add new code.

To complete the homework, you have to complete these steps:

1. Download and decompress the zip file "Homework_2.zip" on your machine.
   You can find what you need at this link http://www.diag.uniroma1.it/~fazzone/Teaching/Data_Mining_Technology_for_Business_and_Society_2016_2017/DMT4BaS_2016_2017.html in section "Homework_2".
2. Fill the empty methods and parts inside the files `Network_Based_Recommendation_System_FUNCTIONS.py` and `Network_Based_Recommendation_System_FOR_GROUPS.py`, by writing your code where the tag '`# Your code here ;)`' is located.
3. Create a final report **in PDF**.
4. Create a zip file with this content:
   1) The filled version of `Network_Based_Recommendation_System_FUNCTIONS.py` and `Network_Based_Recommendation_System_FOR_GROUPS.py`.
   2) The original version of `Network_Based_Recommendation_System.py`.
   3) A single human-readable file containing the final report **in PDF**.
   4) Nothing less, nothing more.
5. The file name must have this format:
   DMTfBaS_Homework_2__StudentID_StudentName_StudentSurname__StudentID_StudentName_StudentSurname.zip
6. Finally, you must to send this zip file to fazzone@diag.uniroma1.it with this email subject:
   DMTfBaS_Homework_2__StudentID_StudentName_StudentSurname__StudentID_StudentName_StudentSurname.

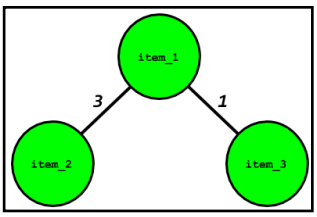
Zoom On Part One:

In this part of the homework you will implement a particular link-analysis procedure for movie recommendation.
First of all, it is fundamental to notice that you can model recommendation system data as a weighted bipartite graph, where nodes are users and items (movies for MovieLens dataset), and an edge between nodes represents a rating of a user for an item.
In the picture you can see an example of this type of graph:



First step (user independent part):
The first step of the method consists on the creation of a reduced version of this bipartite graph. This reduced graph is a weighted graph containing only item nodes of the original bipartite graph (it is reduced in the number of nodes).
An edge between two item nodes in the reduced graph is present when they are both connected to the same user node in the original bipartite graph. In other words, this means that in the original bipartite graph there is at least one user node that has an edge to both these item nodes. Equivalently, this means that in the dataset there is at least one user that has rated

both items. This reduced graph is a weighted graph, where the weight on edges represents the number of distinct user nodes connected to both item nodes in the original bipartite graph.
This reduced graph is called **Item-Item graph**.
In the picture is represented the corresponding Item-Item graph of the previous bipartite graph:



Pay attention to the fact that the value of the ratings is not involved in the reduced Item-Item graph construction process.
You must implement this method by filling the python function "`create_item_item_graph(graph_users_items)`" contained in "`Network_Based_Recommendation_System_FUNCTIONS.py`".

Second step (user dependent part):
The second step consists in performing Topic-Specific-PageRank on the reduced Item-Item graph, by considering as nodes of the topic all items rated by the input user. To increase the personalization of the recommendation, the teleporting probability distribution among all nodes in the virtual topic must be biased using the rating values that the input user has associated to each rated item.
For example, the teleporting probability distribution for `user_3` on the nodes of the reduced Item-Item graph is the following: `{'item_1': 5/(5+1), 'item_2': 1/(5+1), 'item_3': 0}`.
You must implement the method that creates the teleporting distribution vector by filling the python function "`create_preference_vector_for_teleporting(user_id, graph_users_items)`" contained in "`Network_Based_Recommendation_System_FUNCTIONS.py`".
For the Topic-Specific-PageRank computation **you must** use the method "`pagerank`" contained in "`Network_Based_Recommendation_System_FUNCTIONS.py`".

Third step (refinement):
This step consists in producing a sorted list of items in a descending order of Topic-Specific-PageRank score. Of course, this sorted list of items must not contain the items already rated by the input user. This sorted list is the output of the recommendation process.
You must implement this method by filling the python function "`create_ranked_list_of_recommended_items(page_rank_vector_of_items, user_id, training_graph_users_items)`" contained in the file "`Network_Based_Recommendation_System_FUNCTIONS.py`".

Fourth step (Test):
To test the quality of the recommendation method, you will use the Average Normalized Discounted Cumulative Gain metric (average **nDCG**).
The version of DCG that you must use in this homework is different from the one used in the first homework.
First of all, to evaluate the nDCG for a particular user we need to compute the NCG for that user, and after, normalize the value dividing by the maximum DCN achievable for that user.
More formally:

$$nDCG(user) = \frac{DCG(user)}{MaximumDCG(user)}$$

In the experiment you will compute the DCG on all items in the test set of the user, sorted according to the ranking computed by the recommendation algorithm.
Let's clarify this with an example.

Let's assume now that, for a generic user 'u', we have in the **test-set** only these **(item, rating)** data: set([(4, 5), (7, 5), (23, 3), (1, 2), (5, 2)]).
Let's assume also that, always for the user 'u', the recommendation system has produced this recommended sorted list of items:
"[38, 5, 40, 29, 1, 15, 7, 9, 23, 4, 15, 6, 43, 21, 34, ...]"
(the length of this list is equal to the total number of item nodes in the training graph minus the number of items associated to the user 'u' in the training set).
As described before, we have now to sort all the (item, rating) elements associated to the user 'u' in the **test-set**, according to the order imposed by the recommended list of items: [(5, 2), (1, 2), (7, 5), (23, 3), (4, 5)].
This final list is that one to consider for the DCG evaluation, let's call this list **"evaluation_list"**.

The DCG of a particular user is defined as follow:

$$DCG(user) = \sum_{i=1} \frac{rating_{user}(item_i)}{\log_2(i+1)}$$

Where the index of an item is the position of the item itself in the **evaluation_list**.
From the previous example, we have:
DCG('u')= 2 + 2/lg(3) + 5/lg(4) + 3/lg(5) + 5/lg(6).
You must implement this method by filling the python function "`discounted_cumulative_gain(user_id, sorted_list_of_recommended_items, test_graph_users_items)`" contained in the file "`Network_Based_Recommendation_System_FUNCTIONS.py`".

As explained before, to obtain a normalized version of DCG you have to divide the DCG of a user by the maximum DCG achievable for that user.
You must to fill the python function "`maximum_discounted_cumulative_gain(user_id, test_graph_users_items)`" contained in the file "`Network_Based_Recommendation_System_FUNCTIONS.py`", to calculate the maximum DCG for the input user.

The software contained in "`Network_Based_Recommendation_System.py`" calculates automatically the nDCG and Average-nDCG over all users in the graph and also over all (tarining_graph, test_graph) couples :)

Zoom On Part Two:

In this part you have to create a movie recommendation method for **groups** of users, by **extending the method implemented in the first part**, form a single user, to a group of users. The method must be able to handle scenarios in which group members have different importance levels. For this reason, each member of the group has a weight that represents his importance inside the group. Inside the file "`Network_Based_Recommendation_System_FOR_GROUPS.py`", you have some examples of groups of this type.
You must implement your method putting the corresponding python code inside the file "`Network_Based_Recommendation_System_FOR_GROUPS.py`", where the tag '`# Your code here ;)`' is located. The group recommendation will work on the entire MovieLens 100K Dataset.
For this part it is not required to evaluate the quality of the recommendation method :)

What To Put in the Report:

The final report must contain:

1. A table for each couple (Training-Graph, Test-Graph) used in the experiment (five tables in total). The tables must have exactly this format:

	Training Graph_1	Test Graph_1	Compressed Item-Item Graph_1
Num User Nodes			
Num Item Nodes			
Num Edges			

2. The Average-Normalized-Discounted-Cumulative-Gain of the the implemented method.

3. A short, but complete, description of your method for movies recommendation for groups.

For any problem/doubt/question, you must use Piazza.
Please, don't be shy :)