DSCI 553: Foundations and Applications of Data Mining

Fall 2020

Assignment 3

Deadline: 10/22/2020 11:59 PM PDT

1. Assignment Overview

In this assignment, you will use twitter data along with networkx and gephi to form social networks and do analysis on the created network. You will also implement a community detection algorithm and apply it to the created network.

2. Requirements

Please pay attention to these requirements as they will play an important role in your grades!

2.1 Programming Requirements

You must use Python 3.6 and networkx to implement all tasks. You could use numpy, scipy, spark, sklearn, and pandas.

If you need to use additional libraries, you must have approval from TA and declare it in requirenments.txt

There will be a 20% penalty if we cannot run your code due to Python version inconsistency or undeclared 3rd party library.

2.2 Write your own code

Do not share code with other students!!

For this assignment to be an effective learning experience, you must write your own code! We emphasize this point because you will be able to find Python implementations of some of the required functions on the web. Please do not look for or at any such code!

TAs will combine all the code we can find from the web (e.g., Github) as well as other students' code from this and other (previous) sections for plagiarism detection. We will report all detected plagiarism.

2.3 What you need to turn in

Your submission must be a zip file with name: **firstname_lastname_hw3.zip** (all lowercase). You need to pack the following files in the zip file:

a. three Python scripts, named: (all lowercase)
   firstname_lastname_task1.py, firstname_lastname_task2.py, firstname_lastname_task3.png

   Notice how for task 3, you only need to turn in a screenshot of your created network in gephi (we already can check your ability to create the network in tasks 1 and 2. Here we want to check your ability to use gephi).

   You could have additional files, but these three files are required.

b. You don't need to include your results (except for task 3). We will grade your code with our separate testing data (data will be in the same format).

3. Gamergate Dataset

In this assignment, you will work with the Gamergate.json dataset (same as in assignment 1) The Gamergate.json contains metadata for the gamergate twitter dataset which is a data collected from twitter. Each line has information about a specific tweet with different metadata associated. Note: Each line in the data file contains a tweet in json format. Note you will only need Gamergate.json from assignment 1 and not tweets dataset.

4. Tasks

4.1 Task 1: Creating Retweet Network and Analyzing it (29 points)

For this task, you will create a retweet network and perform analysis on it. Notice our test case will be different, but you can use the Gamergate.json dataset to create and debug your code as the format of our test set will be similar to Gamergate.json.

4.1.1 Retweet Network

Retweet network is a weighted and directed graph in which nodes are representing users and there are edges between users based on the retweet relation (retweet relation == X retweets Y). The edges between two users should be weighted (if the retweet relation holds between two same users increase the weight). The edges should also be directed based on the retweet relation meaning that if "X retweets Y" then there should be an edge from X (the source) to Y (the target).

For Task 1 you need to answer the following questions:

A. Given a json file (similar in format to Gamergate.json) create the retweet network for it and save the network as a gexf. (7 points)

B. How many nodes does this network have? (5 points)

C. How many edges does this network have? (5 points)

D. Which user's tweets get the most retweets? We need the screen name of this user. (5 points)

E. What is the number of retweets the user in part D received? (1 points)

F. Which user retweets the most? We need the screen name of this user. (5 points)

G. What is the number of retweets the user in part F did? (1 points)

Input format: (We will use the following command to execute your code)

```
python firstname_lastname_task1.py <input_file_name> <gexf_output_file_name> <json_output_file_name>
```

Param: input_file_name: the name of the input file (the json data that you should create the network on, would be similar in format to Gamergate.json), including file path

Param: gexf_output_file_name: The name of the output gexf file, including file path

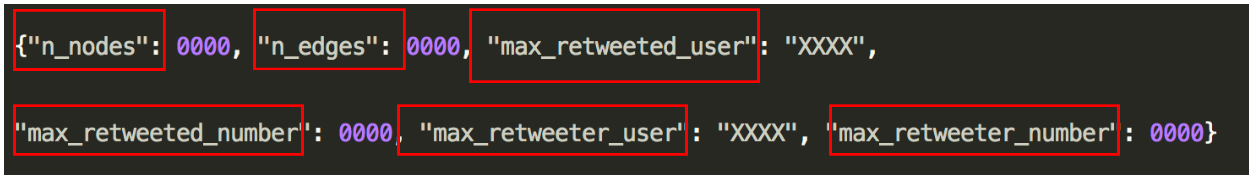
Param: json_ouutput_file_name: The name of the output JSON file, including file path

Output format:

IMPORTANT: Please strictly follow the output format since your code will be graded automatically.

a. The output for Questions B/C/E/G will be a number. The output for Question D/F will be a string (screen name of the user). The output of A would be the gexf network created.

b. You need to write the results for parts B/C/D/E/F/G in the JSON format file. You must use **exactly the same tags** (see the red boxes in Figure 1) for answering each question.



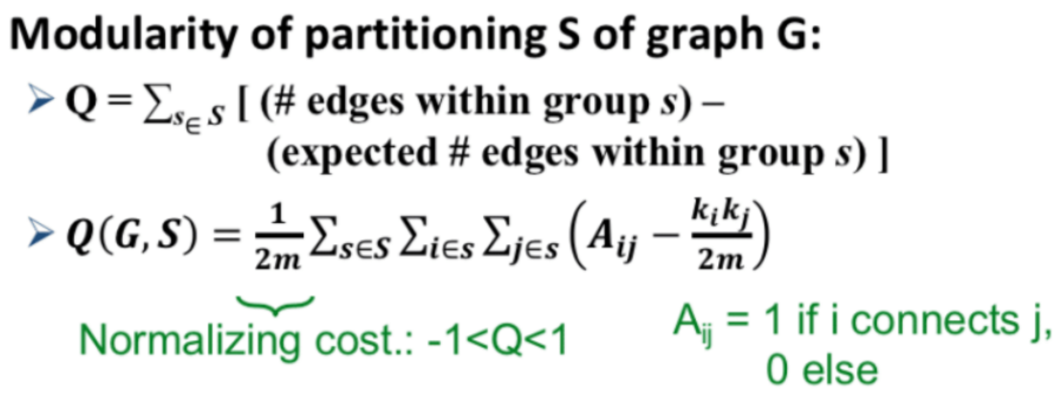
```
{"n_nodes": 0000, "n_edges": 0000, "max_retweeted_user": "XXXX",

"max_retweeted_number": 0000, "max_retweeter_user": "XXXX", "max_retweeter_number": 0000}
```

Figure 1: JSON output structure for task1.

4.2 Task 2: Community Detection (14 points)

For this task, you will be implementing the CLAN algorithm discussed in class to detect communities on the gamergate dataset (on the same network you created in the previous task). CLAN is a two-step community detection method that uses node attributes to debias previous community detection methods that tend to create too many singleton communities without taking content of the node into consideration. For the first step of CLAN, you will implement an unsupervised community detection method (the Grivan-Newman algorithm). For the second step, you will train a classifier to use node attributes in the graph on the major communities detected by Grivan-Newman and classify non-significant communities into one of those major communities.

For Task 2 you need to perform the following tasks:
A. Partition the graph into communities that maximize the modularity objective as shown below unsupervisely. (4 points)



Modularity of partitioning S of graph G:

- $\mathbf{Q} = \sum_{s \in S}$ **[(# edges within group s) – (expected # edges within group s)]**
- $$Q(G,S) = \underbrace{\frac{1}{2m}}_{\text{Normalizing cost.: } -1<Q<1} \sum_{s \in S} \sum_{i \in s} \sum_{j \in s} \left(A_{ij} - \frac{k_i k_j}{2m} \right)$$

$A_{ij} = 1$ if i connects j, 0 else

You should also consider the singleton communities as a valid community (singleton community has only one member).
Save your results as a txt file where the **first line reports the modularity value of the best split and the following lines are the detected communities** where each line represents one community with the following format:
‘node1_Screen_name’, ‘node2_Screen_name’, ‘node3_Screen_name’,…

Sort the results based on the community size in ascending order and then the first node screen name in the community in lexicographical order. The screen names of nodes in each community should also be in the lexicographical order. **(For details of formatting follow the toy test case provided to you)**

B. For this part, you will take the two largest communities detected in part A and train a **Multinomial Naïve Bayes** classifier based on TFIDF features of the nodes in the detected largest communities and text used by them (remember each node is a user and each user has a set of tweets they tweeted, so take all the tweets for a specific user and use TFIDF features on that text to train the classifier). After training the classifier, now classify each of the nodes from the other smaller communities to one of these significant communities. Imagine these users in small communities as a test instance that you are labeling them based on their features (text tweeted by them) on a classifier that was trained on nodes from the two significant communities. Create a txt file with the two communities and the nodes in them following the same format as in part A **except you do not need to report the modularity score this time in the first line.** (Note for this case you will end up having two communities, the threshold in the CLAN algorithm can be considered as 2 in this case, so your output will have two lines each representing the communities with all the nodes involved in them) (7 points)

C. Train a **Multinomial Naïve Bayes** classifier this time using the count-vectorizer features and repeat the task in part B. Create a new txt file and report the communities in it following the same format as in part B. (3 points)

Input format: (We will use the following command to execute your code)

```
python firstname_lastname_task2.py <input_file_name> <taskA_output_file_name> <taskB_output_file_name> <taskC_output_file_name>
```

Param: input_file_name: The name of the input json file, including file path. You will take this file create the retweet network same as in task 1. And apply tasks in this section on it.

Param: taskA_output_file_name: The name and path of the output txt file for task A.
Param: taskB_output_file_name: The name and path of the output txt file for task B.
Param: taskC_output_file_name: The name and path of the output txt file for task C.

Output format:

Text file for each of the A, B, and C tasks are described under each task in detail.

4.3 Task 3: Using Gephi and Visualization (7 points)

Use Gamergate.json and create the retweet network for it (no need to submit the code for this part! We can check your skills for this in tasks 1 and 2). Use gephi (with force atlas 2 layout) and visualize the created network. Then apply modularity on the network and partition the network based on modularity class. Color two major components of the network blue and orange. Submit a screenshot of this network. Notice for this task there is no separate test case. Use the given Gamergate.json data to perform the task. (7 points)

Input format:
There will be no input format for this task.

Output format:

Submit a screen-shot of the created network in gephi following the exact color coding assigned (orange and blue for the two major components.)

5. Grading Criteria

Perfect score for this assignment is 50 points.

Assignment Submission Policy
Homework assignments are due at 11:59 pm on the due date and should be submitted in Blackboard. Every student has ***FIVE free late days*** for the homework assignments. You can use these five days for any reason separately or together to avoid the late penalty. There will be no other extensions for any reason. You cannot use the free late days after the last day of the class. You can submit homework up to one week late, but you will ***lose 20% of the possible points*** for the assignment. After one week, the assignment cannot be submitted.

(% penalty = % penalty of possible points you get)

1. You can use your free 5-day extension separately or together.
2. If we cannot run your programs with the command we specified, there will be 80% penalty.
3. If your program cannot run with the required Python versions, there will be 20% penalty.
4. If our grading program cannot find a specified tag, there will be no point for this question.
5. We can regrade on your assignments within seven days once the scores are released. No argue after one week. There will be 20% penalty if our grading is correct.
6. There will be 20% penalty for late submission within a week and no point after a week.
7. There will be no point if the total execution time exceeds 15 minutes.