Assignment 8 - Opinion Diffusion Models

# Information

In this assignment, you will implement a few opinion diffusion models from scratch. More information about each of these models can be found in the [reading](https://docs.google.com/document/d/1YKarDDeMRPu6sC478lKbQ1DrmF8UJ5acvCus4nez2Qc/edit?usp=sharing) as well as in [Teamwork 11](https://docs.google.com/document/d/1yC3-Ll1JBIjioxfwRjbX9r-XekIQZKQR-a47ewkJpQs/edit?usp=sharing). The purpose of this assignment is to let you see in greater detail how some of the models from the Teamwork actually function, as well as to give you some additional practice building social network analysis models.

First, go to the [**Github Classroom for Assignment 8**](https://classroom.github.com/a/Gpb9N3_h) and clone the repository there. In it, you will find a python file with several things. The majority of the file is taken up by the definition of the DiffusionModel class. This class has a different method for each different update rule that must be filled in.

# Tasks

Your tasks for this assignment are to implement five different models of opinion diffusion - three binary and two continuous. In addition, you will design your own opinion diffusion model and implement it to see how it operates compared to other popular models. Facilities are already provided for you to visualize the opinion space of the network after a simulation, so your only job for this assignment is to implement the models.

The structure of your python file is as follows:

* get\_binary\_opinions(n): a function that takes a number n as input and returns n values of -1 or 1 selected at random
* get\_continuous\_opinions(n): a function that takes a number n as input and returns n real-valued numbers between 0 and 1 selected at random
* DiffusionModel: the class containing the methods you will need to complete
  + This class requires a networkx graph and a string representing the update rule it is supposed to use. Values for the update rule should be “voter”, “qvoter”, “majority”, “snazjd”, “hk”, “dw”, and a final one that you name.
  + update(): update the opinions of all agents in the network
  + update\_voter(): update opinions based on the Voter model - this method is already defined for you and does not need to be modified
  + update\_qvoter(): update opinions based on the Q-Voter model
  + update\_majority(): update opinions based on the Majority Rule model
  + update\_snazjd(): update opinions based on the Snazjd model
  + update\_hk(): update opinions based on the Hegelsmann-Krause model
  + update\_dw(): update opinions based on the Deffuant-Weisbuch model
  + update\_mymodel(): update opinions based on your own model
  + run\_test\_discrete(): run an experiment on one of the binary opinion models
  + run\_test\_continuous():run an experiment on one of the continuous opinion models
  + run(): a wrapper to run one of the two methods above

Your tasks for the assignment are the following:

1. Implement the Q-voter model

* Choose a random node *i*
* Select *q* of its neighbors
* If they all have the same opinion, give *i* that opinion
* Otherwise, do nothing

1. Implement the Majority Rule model

* Randomly select *q* nodes from the network
* Give each of them the majority opinion in the group

1. Implement the Snazjd model

* Randomly select a node *i*
* Randomly select a node *j* from among *i*’s neighbors
* If their opinions are the same, give that same opinion to all of *i*’s and *j*’s neighbors

1. Implement the Hegelsmann-Krause model

* Randomly select a node *i*
* Collect all of their neighbors whose opinions differ from its own by no more than epsilon
* Calculate the average opinion of those neighbors
* Give *i* the average opinion

1. Implement the Deffuant-Weisbuch model

* Randomly select a node *i*
* Randomly select a node *j* from among *i*’s neighbors whose opinions differ from *i*’s by no more than epsilon
* Calculate mu \* (difference between j’s opinion and i’s opinion)
* Add this value to *i*’s current opinion
* Do the same for *j* (do you need to change the order of opinions when taking the difference?)

1. Implement your own opinion update model!

* For this task, think about the things you take into account when updating your opinion. Do you consider where your information came from? Do you have different levels of trust in different sources? Do some people have more resistance to opinion change than others? Be creative!
* Be sure to include your model’s name in the code where appropriate depending on if you want to use a binary or continuous opinion space so that it can get visualized like the others.

1. Complete the update() method so that it calls each of the methods above appropriately.

Once you have completed all of the tasks above, make sure to **COMMIT** and **PUSH** your code to Github.