**Bot Detection Code**

- README for *bot detector code*-

What files in this repository:

**README\_bot\_code.txt**

The present readme

**MPI\_graphCut.py**

The main file to run the bot detector

**networkClassifierHELPER.py**

The main Helper file for the bot detector

**ioHELPER.py**

Auxiliary input/output stream methods

1. **Introduction: what is the Bot Code for?**

The code implements a bot detection method that classifies accounts as bots or humans, given the retweet graph of their interactions on a given event.

1. **What output to expect?**

The code will create a folder named **network\_piBots\_DB\_NAME** wher **DB\_NAME** is the name of the database/event under study. This ‘keyword’ will be used in most paths to input/output files, so be consistent.

The folder will contain a CSV file with a name that looks like:

**ntwk\_piBot\_mu\_1.0\_alpha1\_50.0\_alpha2\_500.0\_lambda1\_0.8\_lambda2\_0.6\_epsilon\_0.001\_mode\_normal\_iteration\_0\_SEED\_0.csv**

This is a CSV with two columns, the first one being the USER\_ID, the second one the Bot score between 0 and 1 (0=human, 1=bot). Here is an example file produced:

**Example:**

*30408820;0.42*

*47186038;0.51*

*3082813560;0.73*

*762706392398467072;0.5*

1. **What input to feed the code?**

**First step**

The first step is to collect a database, call it **DB\_NAME,** of tweets and accounts on a given event (cf. search and stream files that create sqlite databases). Once this is done, one must construct the retweet graph and store it into a folder called **RT\_graphs**, with path /**RT\_graphs/DB\_NAME\_G0\_RT\_GRAPH.csv**. This is a csv file with 3 columns:

**User1\_ID** (int)

the user ID of a User1

**User2\_ID** (int)

the user ID of a User2 (retweeted at least once by User1)

**Number of retweets** (int) from User1 to User2

**Example**

If User1 with ID 12345 retweeted User2 with ID 54321 6 times, then there will be a line

12345; 54321;6;

In the csv file. Note that if User2 retweets User1 three times; there will be another line

54321;12345;6;

However, if User2 never retweeted User1, there will **NOT** be a line:

54321;12345;0;

Then, given that CSV, the code will produce a NetworkX object to apply efficient graph methods.

Once the Retweet Graph part is done, you must also specify hyperparameters to the classifier.

**Hyperparameters Inputs:**

To run the code, you must specify the following hyperparameters as arguments in the command line.

**mu** (float)

The value of the gamma parameter (cf. paper/thesis), usually I take gamma=1. In the code this parameter is called mu (in accordance with Chris Mark’s code)

**alpha1** (float)

The value of the alpha1 parameter (cf. paper/thesis), usually I take alpha1=100, but this can vary (cf. paper/thesis)

**alpha2** (float)

**T**he value of the alpha2 parameter (cf. paper/thesis), usually I take alpha2=100, but this can vary (cf. paper/thesis)

**Iterations** (int)

The number of iterations (cuts): taken equal to 1 in paper experiments

**db** (string)

The database/event name to match folder paths: **DB\_NAME**

**mode** (string)

The parameter “mode” describes which kind of prior one wants to use for node potentials. If no value specified, this will be a no prior (0.5 for all accounts).

**alambda1** (float)

The value of the lambda11 parameter (cf. paper/thesis), called alambda1 in code. Usually I take alambda1=0.8.

**alambda2** (float)

The value of the lambda00 parameter (cf. paper/thesis), called alambda2 in code. Usually I take alambda2=0.6.

**epsilon** (float)

the value the exponent such that the delta parameter (cf. paper/thesis) equals 10^-(epsilon). We want delta to be close to 0, so I usually take epsilon>=3.

**SEED** (int)

a random seed for sampling experiments

1. **Running the code:**

Before your run the code, make sure to install all required libraries, and get all helper scripts in the right folders.

**Pre-installing dependencies**

Make sure to install the python libraries:

* NetworkX (handles graphs, graph cut…)
* MPI4py (handles MPI to parallelize on multiple processors)

**Run code from command line**

Once you have navigated to the relevant folder, you can run the code with the following command:



Assuming you are working the Yellow Vest movement with (from left to right):

* Using mu=1
* Alpha1 = 100
* Alpha2 = 100
* Iteration = 1
* DB\_NAME keyword (for retweets graph, and valid paths) called YELLOW\_VEST
* Mode=’normal’ meaning no prior (every account at 0.5)
* Lambda11=0.8
* Lambda00=0.6
* Delta=10^(-3), hence lambda10=0.8+0.6-1+10^(-3)=0.401
* SEED=0

Finally, if you are running the code on a cluster to parallelize it on say 10 nodes, you want to add the **mpirun** command at the beginning of the command, where you specify the number of processor to parallelize on: here **np=10**

