Dynamic Representation of Fake News Proliferation in Twitter Subpopulations

Application Deployment Plan

Group 10

Justin Prez
Justin Rosner
Akil Hamilton
Harshil Modi
Zac Demelo

Table of Contents

SECTION I - LIST OF SOFTWARE USED FOR APPLICATION

The entire project can be found at:
https://github.com/HarshilModi10/Twitter_Fake_New_Model

The java code is contained within the “Back-End” branch, and the supporting programs are contained within the “master” branch.

Java

Java code was used to implement the algorithms and logic of the application for the analysis of fake news proliferation among a sample twitter population of 456626 users. Ensure that the java project is deployed within an IDE such as Eclipse.

MongoDB

MongoDB was used to store the data from the database website onto a noSQL database. The code for the database collection adding and querying is provided. For the purposes of submission the final software will work without the use of the Database. This was done to ensure that no errors occur for the TA’s for grading purposes. However, TA’s can still run the database softwares provided if they wish.

Python & iGraph

iGraph is a python library that was used to generate the force directed graph representation of the directed twitter network. Using this library, the states of the graphs were physically depicted overtime and then spliced together to make a video.

Shell Script

Bash was used to splice together a series of static frames of the graphs generated using iGraph. The splicing functionality was accomplished using ffmpeg.

HTML & CSS/SASS

HTML was used to create the static web page format. CSS compiled from the SASS code and was used to create the styling for the web page.

SECTION II - INPUT AND OUTPUT

TEXT FILES FOR JAVA

1. The input for the Java program will be in the form of text files of the following format:

$$\begin{array}{ll} N_1 & E_1 \\ N_2 & E_2 \\ N_3 & E_3 \\ . & . \\ . & . \\ . & . \\ N_n & E_n \end{array}$$

   Where N_i is a node in the graph, and E_i is the node that it is connected to (With N being to tail of the edge and E being the head). These two values are separated by a space, and then following the second node there is a new line symbol. Essentially each line in the input file is representing an edge in the graph. The data used for this project is contained within the **data/** folder in **retweet_data.txt**.

2. The output from the Java program will be in the following format:

$$\begin{array}{llll} N_1 & & & \\ N_2, & N_3, & \ldots & , N_k \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ N_m, & N_{m+1}, & \ldots & , N_x \end{array}$$

   The first line is the randomly selected source node representing the initial fake news tweet. Each line thereafter in the output file represents an hour that the tweet has been 'alive' for, and all of the nodes on that line are the nodes that were retweeted within that hour timeframe. The output file is located within the **data/** folder in **output.txt**. Please note that when viewing the output file in Eclipse there seems to be an error due to the large size of the output, so some lines appear to be blank. If this happens for you please view the output file in another third party app such as notepad or notepad++.

INPUT FROM DATABASE

The input to the database output from the Java program will be in the following format:

N_2: {M_0 ...   , M_k}
.   .   .
.   .   .
.   .   .
N_m: {M_0, ... , M_k}

Each line in the database represents a user from the database file and M_0, ... , M_k represent the other users that the users are connected too.

INPUT & OUTPUT TO iGraph PROGRAM

Using the output generated by the text file created using java, the states of the graph could be plotted overtime and saved as a series of image files. This was accomplished using Python and the iGraph library. The iGraph library was not built-in so had to be installed via the PyPi package manager. The input format of the text file is as follows:

N_1
N_2, N_3, ...   , N_k
.   .   .
.   .   .
.   .   .
N_m, N_{m+1}, ... , N_x

This python file would scrape this text file line by line and perform the transitions on a directed graph using the original edgelist file provided as well. This script would place these transitions into a "./states/" folder in the following format "graph_state_(:04d).png" which would range 1 to x (x being the number of state transitions recorded).

SECTION III - DEPLOYMENT OF APPLICATION

Java Deployment

Once the java project is set up in your local environment, you can use the **Retweet.java** file to execute the project code and attain a sample output. As discussed above, you can view the sample output within the **output.txt** (***please note the viewing error that may occur within Eclipse due to the large file size***).

JUnit testing files are included for each of the algorithm classes: **BFSTest.java**, **BinarySearch.java**, **InsertionTest.java**. These can executed independently to verify the functionality of the algorithms.

iGraph Deployment

This program could be run via the command line with the following command:

```
> python disp_graph.py
```

This would generate all the requisite files in the folder, this video could then be created through running the shell script command as follows:

```
> ./generate.sh
```

MongoDB Deployment

To deploy MongoDB start your local MongoDB server and run the Mongo.java code provided within the database folder. To read the data from the database, run reading.java within the database folder.

Website Deployment

Since the web page is static, you can access the website from the index.html file or the webpage can be accessed using this link being hosted from github: https://harshilmodi10.github.io/Twitter_Fake_New_Model/