Project 3-A

Ben Zipes, Hugo Chapado and Gonzalo Prats

https://github.com/gonprats/finalprojDSA

Video: https://www.youtube.com/watch?v=__M00ATymJQ

Figure 1. Project Logo[1]

1. Extended proposal

1.1. Problem

We want to find out how close a specific person is to a public figure through social media. That is, how far in terms of layers of friends (connections) separate a regular person from a socially known one.

1.2. Motivation

In the media industry, most businesses want to have an insight on how much of an impact a certain person can make through social media and for that, it helps to know how close that person is to publicly known figures. Then, if a company like this one was to hire someone new, they would want to look at that kind of information. This kind of influence is crucial nowadays since a strong presence on the internet is vital for any business to grow. And knowing that their staff is well connected and present in social media grants a higher status and, therefore greater success in the long term.

1.3. Features

We know we have solved the problem when we get a clear path of friends(friend of a friend , and so on) until we get to our target, which in this case is a public figure. A public figure will depend on the minimum number of followers we set (if user number of followers> number of followers we set , then he is a public figure). We also have a maximum degree of separation so in the case we don't find a famous person in that degree of layers, then the program will stop and it will meant that we didn't find any user of our interest.

1.4. Data

The dataset we will be using is an excel spreadsheet and comes from kaggle (https://www.kaggle.com/hwassner/TwitterFriends).

This data set is divided in 10 columns , the vertices in our case will be individual twitter accounts , which includes(id, username, number of followers, number of follows), there also other data such as their last tweet, avatar ,tags , that we will be deciding if they are useful to implement in the project while we are working on it. Then the edges will be the connections between friends/follows, this means that every twitter account has a list of friends, so every user will have an edge to the twitter accounts that are in their respective list of friends.

1.5. Tools

We will be using python3 in Visual Studio Code and implementing a GUI with Tkinter. Visual studio code offers a great variety of extensions to help us develop in an easier and faster way our code. And Tkinter is the easiest tool we found to make a professional GUI with an easy programming.

1.6. Visuals

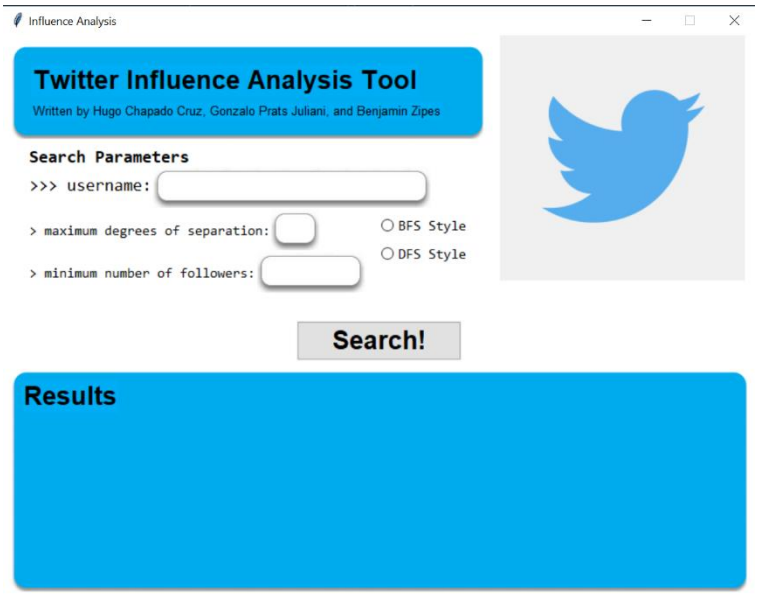

Figure 2.Mock-up interface[1]

1.7. Strategy

We will implement the graph as a Node List as it replicates the actual format of the Twitter social network with lists of followers. We plan on comparing a BFS algorithm to DFS algorithm for finding a valid path to an account with the desired number of followers. This way we can analyze each node as we visit it with its own attributes. The idea for that is to implement a regular BFS/DFS and then make the necessary changes to it so that it can traverse a list of Nodes, that is, we want the function to be able to search according to the number of followers and the list of friends. These kinds of changes are easier with the node structure as we can simply reference the search to that structure and once we

arrive there, we only need to access the attributes and compare them to the values we want to compare.

1.8. Distribution of Responsibility and Roles:

The first we will do as a group is to implement the graph, so that everyone agrees the names of the variables and how the graph is implemented so it is easier to work on through the project for all. Then have thought on splitting the work in the following way, Gonzalo Prats will implement the first algorithm of the graph(BFS), Hugo Chapado the second algorithm(DFS) and Ben Zipes will implement the user interface. Then if one of us finishes their respective part , then that person will help the one that is struggling the most . Then we will also through every week we will have some meetings to talk about the project and see how it is going.

2. Analysis

2.1. Changes to the proposal

As expected, the proposal was just a placeholder for the final project. As we started working on in, we realized how some of the things could be done in an easier or faster way. What we first did was create the full structure: we created a Node class to store all the information from a certain user. This included its id, its username, number of followers, number of friends and the complete list of friends (people that this person follows in twitter). With all this information we just had to figure out how to insert them into a graph so that then we could perform both searches in it. For that, we came up with a basic graph class, with an adjacency list that would store the Node structure defined earlier. An adjacency matrix made no real sense as the graph is rather sparse. Within the graph function, we had to develop useful functions just to manage the graph such as *findnode* or *findid*. Both look for a node in different ways (through username and through the id, respectively).

Another problem we faced was not knowing how to properly import the data from the excel spreadsheet. We used the pandas library and its functions were very helpful, as *read_excel* creates a dataframe so that the data manpulation gets easier. Up until here, all was expected. But when it came to extract the information, we realized that it was “dirty”, meaning that there could be extra whitespaces, quotes that made an integer into a string and other minor changes that had to be addressed before trying to create the Nodes. After trial and error we ended up with a simple function that would get rid of all this noise by selecting parts of the strings we wanted to change

Then, going into the function itself, we discussed different approaches to it. First, we tried to divide it in three parts: DFS, BFS and a general function that would handle both two and work with Nodes at the same time, which is what we were looking for. Then we tried to make DFS and BFS fully functional for our purposes, that is, we tried to adapt the algorithms so that just by using them we would get the desired output. Both these two approaches became messy in terms of programming,

with many variables and we had trouble combining both into the same program (losing track of the variables was our main problem). Our final approach, and the one we end up choosing was to make a single function for everything. As parameters, this function has a Node (a user), the search algorithm choice, the threshold for the number of followers (what is the minimum number of followers for the user to be considered socially known), and the maximum number of layers to look for, as sometimes we would want just to look for a certain proximity to the user.

2.2. Complexity of the main functions

- *finalfunction:* this is the function that performs the search (DFS or BFS) according to the given parameters. Its complexity in its worst case is $O(n^2)$. We first traverse the whole list of users and then traverse the list of friends that this user has.
- *findnode* and *findid:* both are simple functions that search for a certain user. For that, we traverse the whole node list and for that, its complexity is O(n).
- Reading the data: although this is not really a function itself, we believe that it is worth mentioning that, for creating the node list, we must traverse the whole dataset and then, for each user, also traverse the whole friends list. This means a complexity of $O(n^2)$
- GUI: the GUI has a complexity of O(1) as it only takes the results and plots them in a graphic way

3. Reflection

3.1. Overall experience

We believe that the whole project was a very good way of learning how a team is supposed to work and develop leadership and teamsmanship skills. Orgainzation became key when it came down to meeting dedlines. We found out that working on a professional environment like this one meant that we had to be able to mix the individual work with the team's interests, which was not always easy as we all have different working styles. Pursuing all of us the same goal, however, made that task easier and our motivation for the project was a central part for the correct development of the project, we were all very excited to see the code and the user interface working. Overall, it was a very positive experience we are sure we have learned a lot from

3.2. Challenges

Logistical challenges came first. We are three students, and our schedules are not always compatible, so in one of the first meetings we had to agree to meet at certain times so that we could maintain our involvement in the rest of the classes at the same time we advanced on the project.

Coding came along with more challenges, nevertheless. As it is usual, the first version of a coding project is never the good one and we ran into many debugging and compiling errors. One of the issues was with the search, as the Node structure had to be improved several times due to lack of accuracy (we would misscount the followers or get the friends name or id wrong). But with Visual Studio Code, debugging became very easy and the errors could be traced in a matter of minutes. That is why the

challenges cannot be described as specific ones but rather as general coding problems. Again, through trial and error we were able to solve the problems one by one and, ultimately, we were able to complete a code without errors.

3.3. Changes we would make if we did the project again

If we could redo the project, we would have partitioned the problems in smaller ones (like in a Divide and Conquer algorithm). Like we said, one of the challenges was to track each error through the code and that could have been partially avoided if we debugged at each stage instead of writing the whole code at once and then debugging the whole code. If we had divided the project into smaller sections (for example, going function by function and checking that each one of them worked sepparately before putting the whole code together) we would have saved time, which we could have used for a more professional GUI or a most sophisticated development.

3.4. Personal learning

Hugo Chapado: *I learned a lot about GUIs and how they are implemented. They always seem complicated to develop but with the right framework it is much easier and interesting to work on*

Benjamin Zipes: *I learned some best practices for coding a user interface and learned the tkinter package extensively. I also learned to work with python in an object-oriented style to connect the user interface to a working graph and search algorithms*

Gonzalo Prats: *I would have never said that learning how to manage my own time would be key for the good develoment of a group project. Also, I learned that brainstorming is not just about cominng with the right ideas, but coming up with simple ideas that might develop later*

4. References

1. Data set Link https://www.kaggle.com/hwassner/TwitterFriends

We change some names and number of followers of some accounts so it looks better on the video