Social Network Analysis for Twitter

Student Name: [redacted]

Supervisor Name: [redacted]

Submitted as part of the degree of M.Sc. MISCADA to the

Board of Examiners in the Department of Computer Sciences, Durham University

Abstract —

Context: The extensive use of social media, such as Twitter, generates a significant amount of relational data, which provides opportunities for people, from researchers to entrepreneurs, to gain insight by analyzing social networks on a large scale. Building networks with highly relevant data and choosing the appropriate analysis framework plays an essential role in uncovering a group's potential attributes. To make this analysis technique replicable in modular form for more disciplines, proposing a general and comprehensive method for such replicability and compatibility is necessary.

Aims: The aim of this project is to establish a complete social network analysis pipeline for Twitter. We aim to build modules covering the whole process of the analysis, including APIs connecting, data mining and storage, network generation, and analysis framework, with each step being easily modified and applied to various disciplines.

Method: A relational and an attribute dataset are built as the fundamental prerequisite of the network generation by utilizing the Twitter APIs (Application Programming Interfaces) with the help of a Python package tweepy. Two network graphs of different scales and levels of detail are established, and a structured analysis with selected metrics is applied for both graphs. A community detection algorithm based on the map equation is implemented to explore potential groups in the network. Additionally, all the programs are executed on a self-deployed remote Linux server with an independent environment to meet the requirements of cross-terminal operations and solve the package controversy.

Results: Two ego graphs are created: a 2-degree ego graph (233,083 nodes and 641365 edges) and a 1.5-degree ego graph (492 nodes and 28148 edges) with node weights and edge weights representing the extent of activity and interaction. Structured analysis comprising eight metrics is applied for graphs and results saved in the Python package pandas data-frame. A community detection algorithm is implemented on the 1.5-degree ego graph, and the detected result is visualized in a colorful format. Additionally, the requirement of being easily transplantable is achieved by integrating all the modules in a Python-class.

Conclusion: Our pipeline conducted a thorough social network analysis for Twitter and generated a class with all the modules integrated. A similar complete analysis can be performed by merely calling the methods with self-defined parameters. Extracting some modules as a tool is straightforward and efficient, allowing this research to contribute to other fields by allowing other researchers to replicate and use these ready-made modules.

Keywords — Social Network Analysis, Graph Theory, Twitter, Community Detection, Linux

I INTRODUCTION

From the emergence of human society, we are living in a world of connection. Individuals, groups, and organizations act as nodes connecting with others through various types of relationships, including kinship, social contact, exchange (trade or information sharing), conflict, and collaboration, building a series of broad and complex networks, which are the social networks. The social network exists in every aspect of our lives, such as relationship networks in the neighborhood, traffic networks in the city, organization networks in the enterprise, and most extensively, the social media networks on the internet. Therefore, to gain an in-depth insight into the world filled with complicated networks, it is essential to analyze them with a scientific and systematic method, which is Social Network Analysis (SNA).

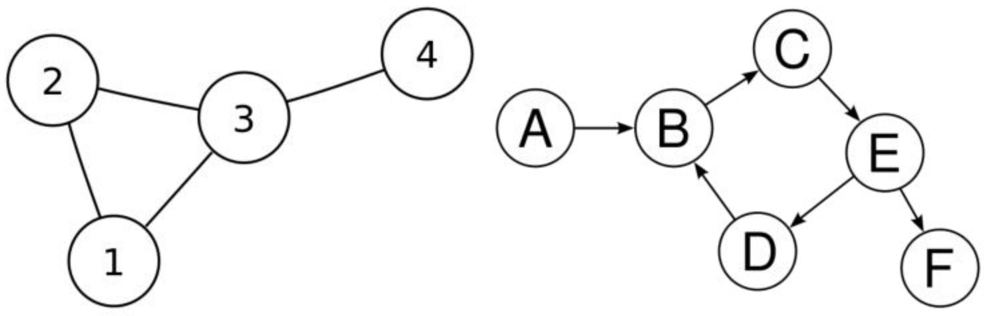
The SNA mainly focuses on structures and patterns of the connection between individuals rather than their characteristics. In other words, an individual's position in the network is more crucial than its attributes (age, gender, income). It shares a similar idea in the real estate profession that a house's location always occupies a higher weight than the interior decoration in the total price (Hansen et al. 2020). Through the SNA, the analyzer can uncover many potential properties of the network, including identifying the central players, finding the bridge nodes, and monitoring the information flow.

In the past few decades, various social media have entered people's daily lives and fundamentally changed how the entire society communicates and interacts. Twitter is one of the most widely used social media with millions of users. From the perspective of social media classification, Twitter is a microblog type platform that allows users to post personal content limited to 280 characters, and in addition to posting individual tweets, *retweet*, *comment*, and *quoate* are also common activities of Twitter users. Every day, millions of users generate information and interact with others, creating billions of networks with different topics, patterns, and scales. Analyzing these networks would uncover many potential characteristics of society and help people build a better world. Additionally, Twitter has developed a friendly application programming interface (API) and a relatively powerful python package *tweepy*, which allows users to access these massive data and conduct large-scale analysis.

A Background

The network size in social media is always large, and focusing on an individual's perspective is a good starting point for analyzing the network. In Hansen et al.'s (2020) work, network analysts call the individual in focus *ego*, and the people connected to it *alters*. One *ego* and all its *alters* can form a network, called an egocentric network. Precisely, this network that is only composed of *ego* and its *alters* is the basic *1-degree ego network*, which can extend to a large scale. The *1.5-degree ego network* is the extension of *1-degree ego network* by involving the connection between all of the alters. Similarly, the *2-degree ego network* extends the *1.5-degree ego network* by including all of the alters' own alters. A higher degree of expansion is possible, but the network's scale will rapidly expand, and it will not be easy to track nodes in the distance, which is not consistent with the original intention of focusing. On the other hand, the knowledge of analysis *1.5 degree ego network* and *2-degree ego network* is valuable enough to find out the group's general trend, the main direction of information flow, and the real hub of the entire network.

A scientific and objective analysis of the network is the most basic requirement, and graph theory is the theoretical cornerstone behind it. Graph theory is a branch of mathematics providing a set of concepts and methods for graph representation and analysis (Giorgos 2010). It has a long history, which Leonhard Euler may first study in 1736 (Hansen et al. 2020) and has now become one of the rationales for the SNA. In the context of graph theory, a *graph* or a *digraph* based on its symmetry is the representation of a social network, while the graph implies symmetric relations, and digraph implies asymmetric relations (Bandyopadhyay et al. 2011). As shown in Figure 1, for both graphs, a node or a vertex represents a unit that can be any type of independent entity, including an individual, a family, a village, an institution, and a country. As for edge, it represents the tie or relationship between two nodes. Notably, in a digraph, links are edges with direction termed *arcs*, which illustrates the flow of the information or the tendency in the relationship. These asymmetric relations are widely existing in the social media world, and Twitter is a canonical representative of which the following relationship is not mandatorily reciprocal. Additionally, both nodes and edges can add weights to involve more attributes in the network, such as distance, amount of information, degree of activity, and other quantitative indexes.



(a) Unirected Graph

(b) Directed Graph

Figure 1: Graph Example

B Objectives

This project aims to build a pipeline that covers the entire process of Twitter SNA and integrates all methods and analyses into an easy-to-call module. To achieve this goal step by step, we divide the objectives into the minimum, intermediate, and advanced.

The minimum objective is to grab the relational data through Twitter APIs and generate two egocentric networks, the *1.5-degree ego network* and *2-degree ego network*, and then perform overall analysis and centrality analysis on these two networks. All these operations are implemented on a self-deployed remote server with an appropriate Python Jupyter Notebook virtual environment, and all results are displayed in the data frame for further comparison and analysis.

Based on the minimum goal, the intermediate objective is to obtain attribute data from Twitter and use these data to add weights to the *1.5-degree ego network*, including node weights and edge weights. Then the community detection algorithm is applied to this weighted graph, and the result is visualized in a colorful format to show the detection effect.

Finally, the advanced objective is to integrate all modules into a Python class. Other research can directly conduct similar analysis through this class, or call certain methods to achieve a specific purpose.

II RELATED WORK

Initially, social network analysis (SNA) was merely a research method in sociology (Scott 1988). With the addition of graph theory and computer science, it has gradually become an independent methodology and has a wide range of applications in many fields. At the same time, the widespread use of social media such as Twitter provides trillions of machine-readable relation data, which makes large-scale social network analysis feasible and meaningful. In this section, we give an overview of the social origins and social media's shaping and influence on the current SNA, as well as its application in different fields.

A Sociology Origin

"Social network" was originally a metaphor in sociology. This metaphor aimed to make the complex concept "social world" more comprehensible by relating it to the common webs in daily life, such as textiles and fishing nets (Scott 1988). Moreno (1953) is the first sociologist to develop the metaphor into practice. He studied the relationship and interaction in a small group by conceptualizing it in what he called "sociometry" and found out the most influential person among the people, which can be known as the "hub" of the group. Later, many sociologists in America and Britain developed maturer methods that concern the structural properties of networks and then use them to study the relationship between family members, village communities, and tribal societies. In these works, social psychologists pay particular attention to the "centrality" of the different individuals in a small communication group. At the same time, the sociologists and anthropologists were more interested in the overall properties of a relatively large group, such as "density" and "connectedness"(Scott 1988). Although after decades of development, the theories and techniques of social network analysis have been greatly enriched and improved, exploring its origin would still help reseachers understand the network's essence and provide inspiration for current research.

B Role of Social Media and Computer Science

In addition to the efforts of sociologists and graph theory, the widespread use of social media and the rapid development of computer science have also played a huge role in promoting SNA. Traditionally, to obtain the relational data, the two most common ways are questionnaires and interviews (Serrat 2017). Unfortunately, these methods have natural disadvantages that the size of the data collected by hand is limited, and the accuracy of the answers decided by human memory is in doubt. In this case, it would lead to the results that the analysis scale is hard to extend, and the conclusion is not convincing enough. However, with the widespread use of social media, data acquisition is no longer an issue. Every day, billions of people create trillions of messages through various devices, including desktops, laptops, tablets, phones, and all these messages act as ties directly or indirectly linking each individual through all kinds of social media platforms, such as Wechat, Facebook, Twitter, Instagram, Email and others, which build numerous networks of a considerable scale.

Most importantly, the data from the social media platforms are machine-readable (Hansen et al. 2020) and can be exported and analyzed by various tools. Currently, there are many well-developed software and packages such as Pajek, UCInet, Netdraw, Mage, GUESS, R packages (sna and igraph), Python packages (networkx) and Gephi (Butts et al. 2008, Giorgos 2010) involved in the entire process of SNA, including network generation, metrics calculation, and results' visualization, which significantly improves the research quality and efficiency. Additionally, the modern SNA is computationally intensive, and the research on optimization of computing resources and improvement of computing efficiency (Ediger et al. 2010, Butts et al. 2008) has dramatically shortened the computing time, which makes large-scale analysis achievable.

C General Application

In the past few decades, the research of SNA has advanced by leaps and bounds, and it has an increasingly wide range of applications. In this subsection, we will study the general application of SNA in society from the perspectives of individuals, companies, and governments. For individuals, SNA can draw an informative map representing a person's social circle and reveal their relative location to others within a group. According to this map, people can expand their resource utilization capabilities by connecting to someone with more considerable influence or finding the shortest path to a target person (Hansen et al. 2020).

As for enterprise, the organization structure is not a simple hierarchical system that only reflects who works where and who reports to whom. Instead, it is a network structure containing valuable business information. With the help of SNA, the entrepreneurs can locate critical employees with vital expertise, discern the bridges or brokers linking separate departments, and check the staffing and collaboration patterns' appropriateness. The SNA is like an X-ray diagnosis that provides the decision-makers with invisible details of the company and supports them to optimize the management, which would significantly enhance efficiency and innovation (Serrat 2017, Hansen et al. 2020).

When it comes to government, SNA is deeply involved in various public management works and plays an increasingly irreplaceable role. For example, tracking the information flow in the misinformation diffusion networks guides the police in finding out the instigators and eliminating the adverse effects (Ahmed et al. 2020). Monitoring the real-time changes in epidemic networks provide early warnings of the outbreak and offer support for cutting off the virus transmission route (Zhou et al. 2010). Analyzing the social media networks in disaster areas uncovers the distribution of victims and helps the rescue team arrange the relief fund and supplies more smartly (Kim & Hastak 2018, Chatfield & Brajawidagda 2012).

D Application in Twitter

The content on Twitter comes from people of different ages, genders, religions, and cultural beliefs worldwide, which truly and comprehensively reflect the full picture of social life. Studying the characteristics of the Twitter network and excavating valuable information is of great significance to various public undertakings. In terms of the Twitter network, the information flow is one of the most important features. According to Myers et al. (2014), the action "follow" in Twitter is essentially an information consumption behavior, and the Twitter network is more of an information network. In 2017, Himelboim et al. proposed six different information flow patterns and described how people communicate in these information structures. Therefore, tracking information flow will reveal the structure of complex networks and internal personnel dynamics, which have many practical applications in our daily lives. For example, Beguerisse-Díaz et al. (2014) applied the flow-based community detection on the users who participated in the 2011 London riots and found out the most influential people, which would help the government improve the censorship. Tsunami warning system (Chatfield & Brajawidagda 2012), information delivery service (Haythornthwaite 1996) and smoking cessation (Prochaska et al. 2012) and other studies also play an important role in improving the quality of social life.

E Summary

Prior research of this field either focused on fundamental theory or application in a specific context. The theoretical work is relatively comprehensive but lacks practical results, while the applied work is worthy of practicing, but the coverage is partial. This work aims to use Twitter as an example to establish a pipeline covering the entire process, which is comprehensive, practical, and easy to customize for research in other domains. Additionally, since information flow is essential to Twitter, a flow-based community detection method, *map equitation* (Bohlin et al. 2014) is also included in our project.

III SOLUTION

This section is the solution to establishing a complete SNA pipeline for Twitter. We divide the whole process into six steps, environment preparation, data mining, network generation, community detection, network analysis, and integration. Firstly a Linux remote server is prepared with an independent virtual Jupyter Notebook environment and all the necessary packages installed. Secondly, the parsed data from the Twitter API will create two datasets, and two ego graphs are generated from them. Then a community detection algorithm and other analyses are applied to these two networks. Finally, all the modules are integrated into a Python-class.

A Environment Preparation

We use programming language Python and several Python libraries to build modules and perform analysis in this project. To execute each process smoothly, we need to install all the necessary packages in advance. However, as we intend to apply our results to people in different disciplines, we have to consider the compatibility for each user. Therefore, we adopt the method of configuring a virtual environment, which only installs the selected packages and has no impact on the local environment. Additionally, if the user runs a large-scale analysis, the calculation would occupy most of the computing resources, and the waiting time may be too long to perform other operations. To solve this problem, we chose to deploy a remote server to separate long-term computing tasks, free users from waiting and enable them to check progress anytime, anywhere.

The operating system we use in this project is a Linux system, and the specific version is Ubuntu 18.04.1. We choose Linux because, compared to Windows, it has some natural advantages, such as friendly environment configuration and concise package management. Also, some special packages can only execute on Linux. For example, the package *infomap* for community detection (discussed in the following section) needs a dedicated compiler and related packages (mapequation.org 2020-05-28 Access). The installation in Linux is a one-step operation, but it takes many steps to complete in Windows.

In this Linux system, the programming environment of this project is the Python Jupyter Notebook, a web-based platform with good interactivity and compatibility, which lays a good foundation for the deployment of the remote server. As mentioned above, we need to configure a virtual environment for this Jupyter Notebook with an appropriate Python version and all the required packages installed. The Python version we use is 3.6.9, and the libraries used are *tweepy*, *networkx*, *json*, *pandas*, *infomap*, *time*, *tqdm* and *matplotlib*.

The final step of preparation is the remote server deployment. Firstly, we choose SSH to connect to the server, because SSH is a safe and convenient connection method. Unix users only need to type the *ssh* command in the terminal to connect, and Windows users can also easily connect through PuTTY. We then want to make Jupyter Notebook on the remote server available on the local device. The technology we use is *SSH Tunneling*, a port forwarding method that forwards network requests running on the remote port to a designated local port. Finally, the user can use the corresponding token to open the Jupyter Notebook in the local browser on the related port.

B Data Mining

Mining Data from Twitter is the prerequisite step before the network construction and corresponding analysis. In the beginning, we use the Python library *tweepy* (tweepy.org 2020-08-02 Access) to connect to the Twitter APIs and get the raw data, which are various Twitter objects. The structure of the Twitter object is intricate, which contains a lot of redundant information. Therefore, to obtain target data, we need to parse these Twitter objects, extract valid data, and then reorganize them into a concise format. Finally, the data is saved in several JavaScript Object Notation (JSON) files, which are easy to modify and import.

B.1 Twitter APIs and Tweepy

Twitter APIs are Twitter company's channels to provide users with programmatic access to Twitter data. According to the official documents (developer.Twitter 2020-08-26 Access), there are three steps to establish the connection:

- Sign in a Twitter account and apply for a developer account.
- Create an application.
- Get the authentication details, which are *consumer key, consumer secret, access token key, access token secret*.

Afterward, a direct connection is available, but it requires to deal with a lot of low-level details. Fortunately, the Python package *tweepy* solves the problem, which bypasses most of the low-level techniques and provides users with more natural instructions. Through *tweepy*, we can use our keys and tokens to establish a connection and obtain the required data from various Twitter API channels.

B.2 Twitter Objects

The raw data acquired from the APIs are a series of Twitter objects containing all the details of the tweets and related Twitter users. However, the Twitter objects are not standard Python data types, which don't support the direct manipulation, and most of the information they encompass is of no use for us. Thus, to build a concise and personalized dataset for this project, parsing the Twitter objects and further data extraction is inevitable.

The Twitter objects are encoded using JSON, which is based on key-value pairs, with named attributes and associated values. These attributes are the carriers containing all the information. The types of Twitter objects are various, and we only use two of the most important objects, *Tweet* and *User*. Each object has several core attributes that support direct extraction. For example, each *Tweet* object has an author, a message, a unique ID, a timestamp, and other related users' information. At the same time, each *User* object has a Twitter name, an ID, followers, and an account bio (developer.Twitter 2020-08-26 Access). However, there are some data hiding in the deep layers of the object, which one can't access directly, such as retweet origin, quote source and mentioned users. The technique we adopt to solve the problem is to parse the object data into a clear and intuitive JSON format, manually find the position of target attribute, and get the data with specialized instructions.

B.3 Limitation Handling

The quantity of data in this project is enormous, and automatically acquiring data is an essential requirement. However, two limitations would disrupt the acquisition process and prevent us from continuously obtaining a large amount of data. The first limitation is the rate limits of Twitter APIs, which is a protective measure set by Twitter developers to maintain servers' stability. We list some crucial rate-limit types and their details in Table 1. When the number of requests reaches the limit, the program will interrupt. To avoid the interruption, we designed a *Rate_Limit_Handler* module to force the process to stop when hitting the limit, and continue running from the breakpoint once the restriction lifted. Another limitation is invalid data. When the owner removes the data we request, or the target account is locked, a null data error would raise and interrupt the process. The solution to this problem is similar to the *Rate_Limit_Handler* module, which is to build an *Invalid_Error_Handler* module to skip invalid points and record the error position for the future check.

Table 1: Rate Limts List

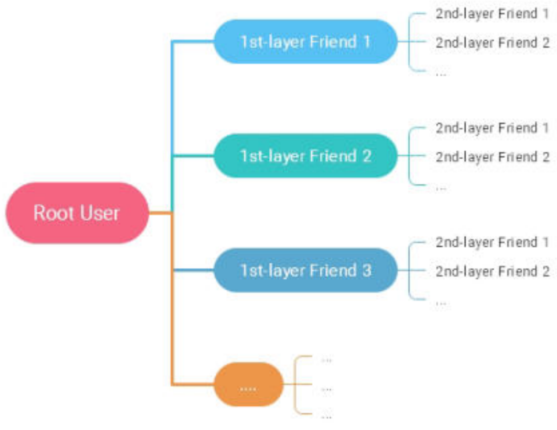
Type	**Rate limits**	**Description**
Accounts	100,000 / 24-hours	The total request limit.
Friends / list	300 / 15 min	Return user's *Following*, also called *Friends*
Friends / id	75,000 ids / 15 min or 15 users / 15 min	Return id of the *Following* or *Friends*
Statuses / User_timeline	36,000 tweets / 15 min	Return up to 3,200 most recent tweets the given user

B.4 Dataset Structure and Saving

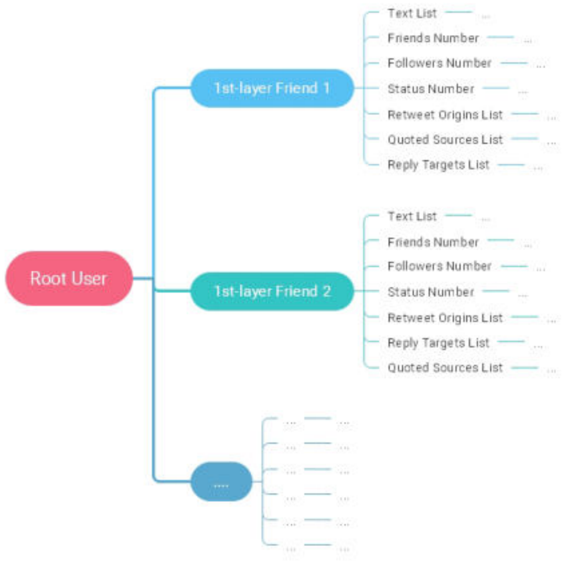
In this project, we need to prepare two different types of datasets, a relational dataset and an attribute dataset. The relational dataset is for the network generation, so its structure should match the network architecture. As the type of graph to be generated is an ego graph, which is equivalent to a hierarchical structure, the process of building the dataset is to find a starting point, and extend it layer by layer. The initial point is called the root user, and the root user's friends are called 1st-layer friends. Similarly, the friends of the 1st-layer friends are called 2nd-layer friends. Then we get the structure of the relational dataset and is shown in Figure 2(a).

Except for the relational data, we still need attribute data to describe the nodes' characteristics and the interaction between nodes. We recorded seven different types of data. The text, friend number, follower number, and status number represent the node's characteristics, and the retweet origin, quote origin, and reply targets reflect its interaction with other nodes. The structure is shown in Figure 2(b). Because the quantity of data is too large to complete the collection in a short time, we only collect the attribute data of the first-layer friends. Nevertheless, the technique can apply to datasets of all sizes.

Eventually, we saved the two datasets into JSON files, because JSON is essentially a hierarchical format that is consistent with the structure of our dataset. In addition, with the Python package *json*, it is fast and straightforward to switch between JSON files and Python dictionaries.



(a) Relational Dataset



(b) Atrribute Dataset

Figure 2: Dataset Strucutre

C Network Generation

Based on the two datasets mentioned above, we can generate graphs representing the network and adding weight to nodes and edges, reflecting the individual characteristics and group interaction. This process is achieved by *networkx* that is a Python package for the graph creation, manipulation, and analysis (NetworkX 2020-08-22 Access). There are two graphs generated in this project, a 1.5-degree ego graph and a 2-degree ego graph, but due to the dataset limitation, we can only add weights to the 1.5-degree ego graph.

C.1 Graph Creation

From the relational dataset, we can create the graph to represent the relationship in the network. To begin with, we import the corresponding JSON file into the programming environment and convert it into a Python dictionary. Then we extract all the users as nodes and add arcs from the root user to 1st-layer friends, creating a 1-degree ego graph of the root user. Afterward, based on the 1-degree ego graph, we continue to add arcs from each 1st-layer friend to its 2nd-layer friends, which would create a 2-degree ego graph. Note that not all the 2nd-layer friends are *pure 2nd-layer nodes* that only connect to the 1st-layer friends but don't connect to the root user. They can be *1st-layer nodes* (directly connect to the root user) or even the root user. Because if a *1st-layer node* is the friend of another *1st-layer node*, even though they are assigned to the 2nd-layer friend category in the dataset, it still belongs to the first layer in the graph, and the only change is to establish a connection between the *1st-layer nodes*. It is the same for the root user. Therefore, when all the *pure 2nd-layer nodes* are removed from the 2-degree ego graph, we can get the 1.5-degree ego graph.

C.2 Weight Addition

In terms of Twitter users, the primary activity is to post statuses to their timeline. Simultaneously, these statuses can be interactive actions with others, such as retweeting messages, quoting tweets, and replying to specific users. To uncover the attribute behind these activities, we need to quantify them into an indicator in our graphs, which are the node weights and edge weights.

The node weights describe the degree of individual activity, and the edge weights reflect the strength of the relationship between two nodes. The formula of node weights $Weight_{node}$ is shown in Eq.(1). For each node, we calculate the status number N_{status} in the recent history and then multiply the result by a coefficient c_{node} to get the node weights. The c_{node} is a manually set coefficient to limit the product in a reasonable range for the subsequent calculation.

$$Weight_{node} = c_{node} * N_{status} \tag{1}$$

Compared with the node weights, the factors that determine the edge weights are relatively complicated. The basic idea is that the frequency of the interaction between individuals reflects the strength of their relationship, and the importance of different types of interaction varies. In Twitter, three of the most common interactive activities are *retweet*, *qoute*, and *reply*. Based on the usage habits, we can sort them according to their importance to the relationship. The *retweet* is an act of approval, which is a relatively weak interaction. As for *qoute*, in addition to showing approval, it also reflects the trend of users establishing more connections, which is much more interactive than *retweet*. When it comes to *reply*, that is an active point to point mutual behavior, which has the strongest degree of interaction.

Additionally, the personal influence of the user also affects their relationship with others. In a relationship, users with more followers have a relatively significant effect, making the connection initiated from them more meaningful. Besides, if users with a more considerable impact only follow a few other users, which means a higher follower-to-following rate, the relationship derived from them will be more critical.

$$\begin{aligned} Weight_{edge} =&((c_1 * RT_{ab}) + (c_2 * QT_{ab}) + (c_3 * RP_{ab})) \\ &* (1 + c_4 * (FL_a - FL_b)/(FL_a + FL_b)) \\ &* (1 + c_5 * Rff_a) \end{aligned} \tag{2}$$

Combining all the factors above, we develop the formula to calculate the edge weights between the user A and B, and it is shown in Eq.(2). The RT_{ab}, QT_{ab}, RP_{ab} represent the number of A retweets B, A quoted B, and A replies to B. The c_1, c_2, c_3 are manually assigned coefficients to control the relative weight, and we set them to $c_1 < c_2 < c_3$ to reflect the order of importance mentioned above. In additon, the FL_a and FL_b is the follwer count of A and B, and the Rff_a is the follower-to-following rate of A. The c_4 and c_5 are also manully set coefficients to control the range of the product.

D Community Detection

There is a phenomenon called *homophily* in networks that individuals tend to establish connections with others with similar characteristics to form highly connected groups, which are communities. The process of detecting communities is to classify the nodes in the entire network based on certain indicators, which segment a large network into small modules with similar attributes. A simple example is shown in Figure 3. After the community detection, we can simplify the network with complex structure and discover some latent properties. Also, when only specific topics or groups are involved, we can extract such communities and perform detailed analysis, which significantly reduces the scope of calculation and the waiting time.

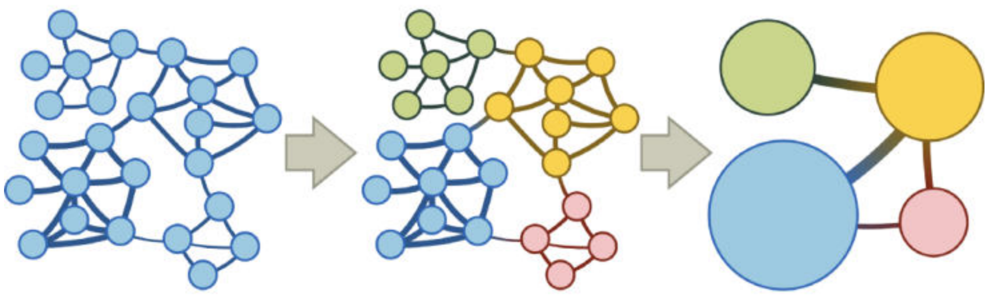


Figure 3: Example of Community Detection

The community detection algorithm in this project is *Infomap*, which is a flow-based algorithm (Bohlin et al. 2014) focusing on the dynamics of the network rather than its topology. The basic idea is to start a random walker traversing the network, moving from one node to another. At each step, the neighbor that the random walker chooses to move to is proportional to the weights of the link to the neighbor, where the weights are the reflection of information flow. If the random walker enters a community, it will stay a relatively long time before it comes to the next community. After the algorithm runs for a period of time, we can distinguish different communities by tracking the trail of random walkers. In our egocentric network, nodes are information carriers making the information flow through interactions with other people, such as retweet, quote, and reply. As described above, we integrate these interactions into the network's edge weights, which meet the requirement of the *Infomap*. Then we manually control the coefficient c_1, c_2, c_3, c_4, and c_5 in Eq.(2) to determine the scale of the weight, which will ultimately affect the quality of the community detection.

E Network Metrics

The network metrics are quantitative measures to inspect the network patterns and the relative position of internal nodes. Some metrics describe the characteristics of the entire network, while others focus on each node. According to the network's constitution in this project, we select several overall metrics and vertex-specific metrics to explore the properties latent in our networks. The overall metrics we choose are *density*, *reciprocity*, *diameter*, *average distance*, as well as *strongly connected components*, and the vertex-specific metrics are *degree*, *clustering coefficient*, *page rank*, and *eccentricity*. We will describe these metrics in detail below.

E.1 Overall Metrics

Density The density measures the degree of interconnection of vertices in the network by calculating the ratio of the actual number of edges to the maximum possible number of edges. The directed graph formula is in Eq.(3), and the numerator should be divided by 2 for an undirected graph.

$$density = \frac{m}{n(n-1)} \tag{3}$$

Where n is the number of nodes, and m is the number of edges in graph G.

Reciprocity The reciprocity is exclusively a directed graph metric, which calculates the percentage of edges pointing in both directions in the total edges of the graph. The formula is shown in Eq.(4).

$$reciprocity = \frac{|\,(u,v) \in G \mid (v,u) \in G\,|}{|\,(u,v) \in G\,|} \tag{4}$$

Where (u,v) is the edge in graph G, representing that it is from vertex u to vertex v.

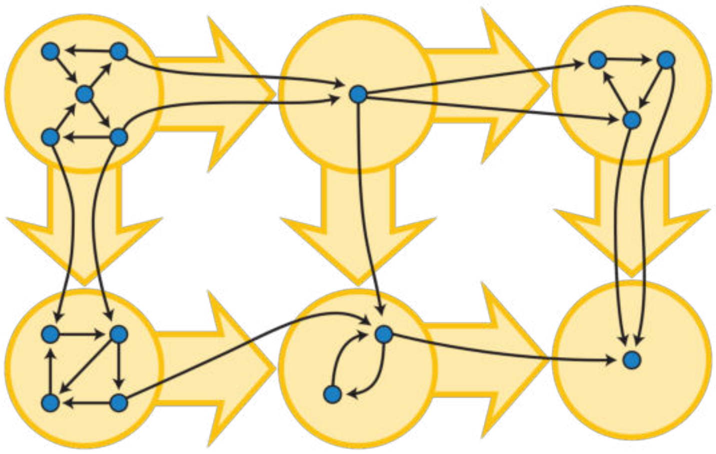


Figure 4: Condensation

Strongly Connected Strong connection is also exclusively a directed graph concept, which means there is a path in each direction between each pair of vertices of the graph. In other words, each node in the strongly connected directed graph can always find a series of edges leading to another node. However, not all graphs are strongly connected, and most of them are composed of many **strongly connected components** that are subgraphs of the original graph. To better investigate the connection between different strongly connected components, we contract each component into a single vertex to form a **directed acyclic graph**, where no strongly connected subgraphs have more than one vertex (Diestel 2005). This directed acyclic graph is the **condensation** of the original graph. The schematic process of generating the condensation is shown in Figure 4.

Diameter In the network, the **distance** between two nodes is the length of the shortest path between them, that is, the minimum number of edges from one node to another. For a node, the maximum distance from it to all other nodes in the network is the **eccentricity** of the node. The **diameter** of the network is the maximum eccentricity in the network, which is the farthest distance between any two vertices.

Average Distance The average distance is the average length of the distance between all vertex pairs, which describes how close the nodes are in the entire network. The formula is shown in Eq.(5).

$$average\_distance = \sum_{s,t \in V} \frac{d(s,t)}{n(n-1)} \tag{5}$$

Where V is the set of nodes in graph G, $d(s,t)$ is the distance from s to t, and n is the number of nodes in G.

E.2 Vertex-specific Metrics

Degree In a directed graph, there are two types of degree, **In-degree** and **Out-degree**. The In-degree is the number of edges pointing to the vertex, while the Out-degree is the number of edges the vertex points toward.

Page Rank The page rank is a core metric in search engines to rank web pages on the Internet. It computes a ranking of the nodes in the network based on the number of incoming links and the tendency of nodes to link to the target.

Clustering Coefficient The clustering coefficient measures the degree of connectivity between vertex neighbors, which is the ratio of the number of edges connecting vertex neighbors to the total number of possible edges between vertex neighbors.

F Integration

There are many program details in each module of the project, some of which will not change with the background difference. Besides, some researchers only want to obtain comprehensive analysis results straightforwardly without customizing the analysis structure. In this case, we integrate all modules into a Python-class that not only supports direct analysis of a given network, but also extracts specific modules for personalized research.

IV RESULTS AND EVALUATION

In this section, we will present the analysis results of the two egocentric networks. According to the metrics classification, we divide the analysis into two parts: overall analysis and vertex-specific analysis. Then we will discuss the process of community detection and evaluate the quality of the detection results. The *ego* of our egocentric network is a reporter named Hu Xijin, the editor-in-chief of the Chinese newspaper Global Times. We choose a reporter as ego because the news reporter group is one of the most active groups in Twitter, and the frequency of information flow in the circle of journalists is very high, which would benefit the process of the flow-based community detection. Additionally, the information about journalists is relatively public, which would facilitate our evaluation of the detection results.

A Overall Analysis

In this subsection, we will compare the overall attributes of the 1.5-degree ego network and the 2-degree ego network as the primary analysis. Then we will further analyze the strong connectivity and explore the characteristcs of these strongly connected components.

A.1 Primary Analysis

The first-layer node of the network that directly connects to the ego is the *Following* of the Hu Xijin in Twitter, and the number of the *Following* is 669. However, some of the users in his *Following List* have too many friends to really "follow" their status, which lacks authenticity in reflecting information flow. Therefore, we remove the users with *Following Quantity* over 3,000. Besides, some of the remaining users lock their accounts so that we can't obtain any related information about them. Although we can establish connections with these users, we can't track their information flow for community detection, nor can we extract their *Following List* to create the 2-degree ego network. Thus, these users also need to be deleted. In the end, there are 491 first-layer nodes left. On this basis, adding one (the ego) is the number of nodes in the 1.5-degree ego network, which is shown in Table 2. As for the second-layer nodes, these two restrictions are no longer necessary because, in this project, we don't track the flow of information in this layer, nor do we further expand the network. As expected, the size of the 2-degree ego network is much larger than the 1.5-degree one.

By definition, we involve all the connections between the first-layer nodes, but not the second-layer nodes. The metric **density** intuitively reflects this detail, where the density of the 1.5-degree ego network is 10,000 times that of the 2-degree ego network. Besides, after the calculation based on **reciprocity**, the reciprocal edges of the 1.5-degree and 2-degree network are 11,822 and 12,185, respectively. There is only about 300 difference between them, which is relatively small compared to the massive difference in total edges. This huge gap is another embodiment of the characteristics of these two networks.

Table 2: Primary Analysis

Ego Network Type	**Nodes Quantity**	**Edges Quantity**	**Density**	**Reciprocity**
1.5-degree	492	28148	0.12	0.42
2-degree	233083	641365	1.18×10^{-5}	0.019

A.2 Strong Connectivity Analysis

It has been verified by calculation, both the 1.5-degree ego network and the 2-degree ego network are not strongly connected. After generating the corresponding condensation of the two networks, the number of strongly connected components in the 1.5-degree network is 5, while the number in the 2-degree network is 232,596, which is about one-third of the total number of nodes. The relatively high proportion in the 2-degree ego network means that many links are missing in the network, which is consistent with the network properties (not all nodes in the second layer are connected). On the contrary, compared with the total number of nodes, only 5 strongly connected components are very few, which makes the relationship between these 5 components very valuable. In this case, we conduct advanced research on the 1.5-degree ego network.

Firstly, we determine the size of each strongly connected component to find one or several main components. Then, we investigate the members of each component and the connections between them. The results are shown in Table 3, which is very exciting. We can find that the largest component 4 contains almost all the network nodes, which is the main component. Simultaneously, all other components are isolated; that is, they consist of only one node. The most important thing is that all the connections between the components come from the main component 4, and there are no other links between other components, which means that the major component 4 is like a hub connecting other components. This finding is very consistent with our expectation that the ego and its neighbors form a highly connected group with very close interior connections. Through some bridges, the whole group is connected with other "ego" that may become the new hub.

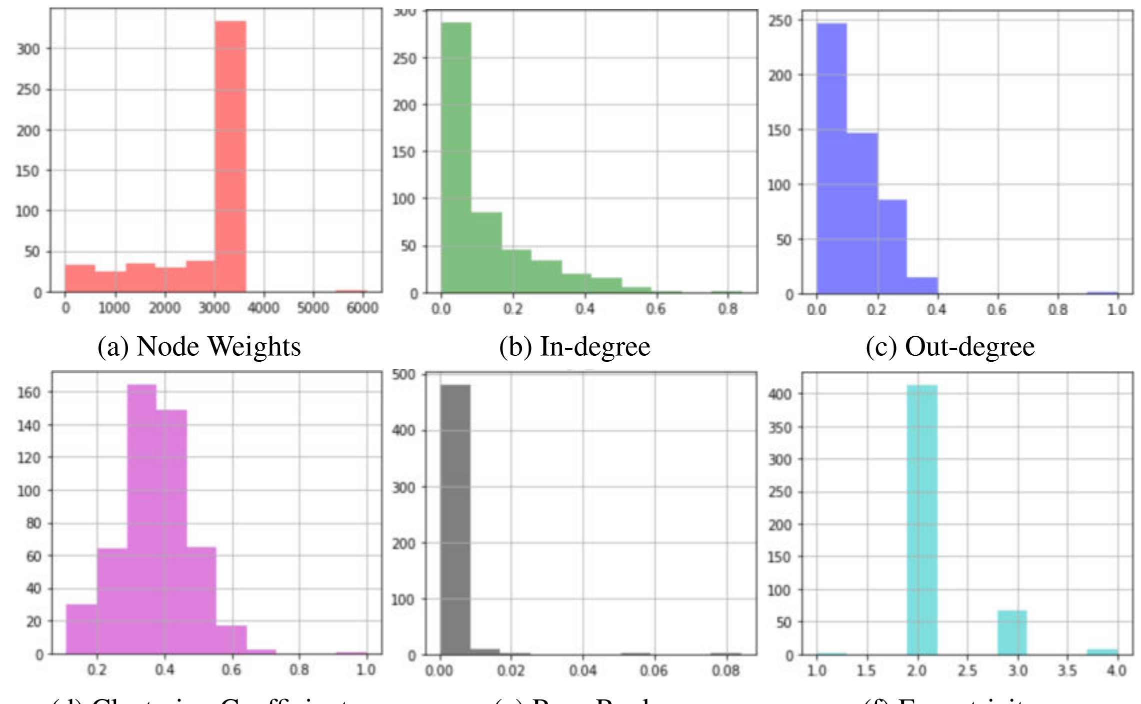
Table 3: Strongly Connected Component Investigation in 1.5-degree ego network

Component	**Size**	**Members**	**Incoming Links**	**Number of Bridges**
0	1	iingwen	(4, 0)	25
1	1	ftchina	(4, 1)	124
2	1	theresa_may	(4 ,2)	32
3	1	RahulGandh	(4, 3)	202
4	488	Others	None	None

Since the largest strongly connected component contains a major percentage of nodes in the network, it is necessary to further study its properties. We calculate the **diameter** and **average distance**, and the results are 4 and 1.8, respectively. This result is fully in line with the theory *"Six Degrees of Separation"*, which shows that even in a large network where most people are not directly connected, you can almost communicate with others in a few steps (usually no more than 6 steps) (Hansen et al. 2020). Aonther evidence of this theory is from Facebook that an average distance is only 4.57 with over 1.59 billion Facebook users in 2016.

B Vertex-Specific Analysis

We select four different vertex-specific metrics to analyze the 1.5-degree ego network and the 2-degree ego network, namely in-degree, out-degree, clustering coefficient, and page rank. We calculate the node weights in the community detection step and the eccentricity in the diameter calculation step. They are also vertex-specific results that represent one of the network features. Therefore, we also include them in this analysis. The analysis results of the two networks are shown in Figure 5 and Figure 6, respectively.

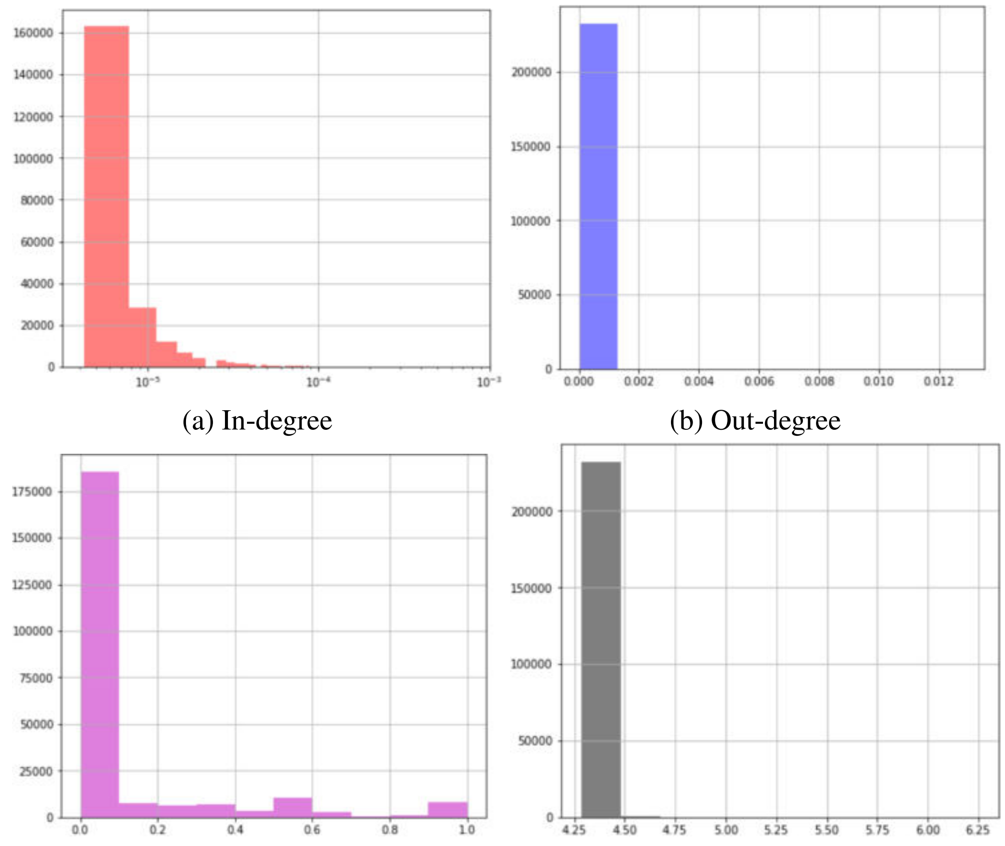

(a) Node Weights (b) In-degree (c) Out-degree

(d) Clustering Coefficient (e) Page Rank (f) Eccentricity

Figure 5: Vertex-specific Analyis of 1.5-Degree Ego Networks

In the 1.5-degree network, the node weights are indicators of users' activity. We can find that most users have node weights over 3,000 indicating that the network is active which is in line with our expectations. For most users, the in-degree value is not very large, but there are still a few users with a higher in-degree value. This means that the network is centered on a few people, and all other users are linked together through these centers. At the same time, except for the ego, most users' out-degree values are at a low level, which means that users in the network tend to only pay attention to a few people around them. It is an implication that there are smaller communities in the network, and we can use community detection algorithms to further detect them. As far as the clustering coefficient is concerned, it exhibits a normal distribution, which reflects the fact that in real life, the connectivity of the user's neighbors is very stochastic. As for page rank, all values are clustered together, which implies the metric fails to distinguish the users in the network. It may be because all users in this network are connected to at least one influential node, the ego. Finally, the eccentricity of most users is 2, which is consistent with the network feature that most users are linked together by the ego.

In the 2-degree network, the in-degree metric distribution is similar to that in the 1.5-degree network. The percentage of users with low in-degree value in this 2-degree network is even much higher because, in addition to the hubs in the 1.5-degree network, all the first-layer nodes in the 2-degree network are also centers connecting their neighbors in the second layer. This result is highly consistent with the 1.5-degree network's conclusion that many centralized nodes are linking other marginalized nodes together in the network. However, because there are many missing edges in the 2-degree networks, the distribution of the out-degree metric and clustering


(a) In-degree (b) Out-degree

(c) Clustering Coefficient (d) Page Rank

Figure 6: Vertex-specific Analyis of 2-Degree Ego Networks

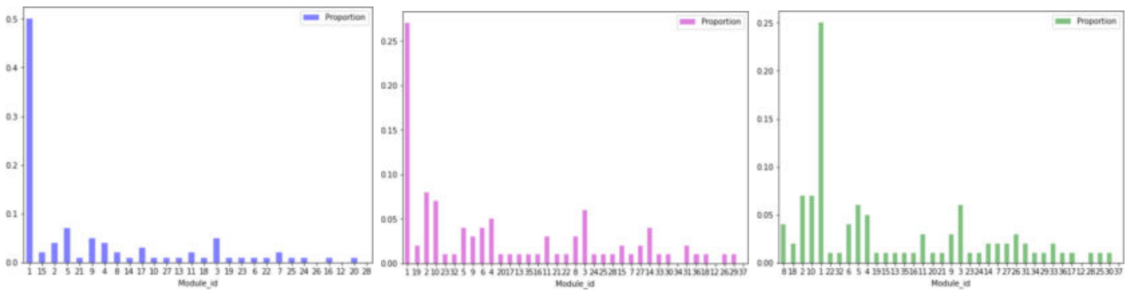
coefficient are abnormal, and they lose the function of reflecting the network attributes. As for page rank, it is still of no use in distinguishing the users in the network.

In general, as the 1.5-degree ego network is complete, the metrics of this network successfully reflect the network's potential properties, while due to the missing edges, the metrics of the 2-degree ego network can only be used as a reference to help us understand the network.

C Community Detection

The basis of the community detection algorithm *Infomap* is the information flow represented by the edge weights in our network. According to the Eq.(2), we can set the coefficient c_1, c_2, c_3, c_4 and c_5 to control the value of edge weigths. The terms controlled by c_4 and c_5 in the formula are two factors that affect the final result by increasing or decreasing a certain percentage, but can't significantly change the scale of the result. On the contrary, c_1, c_2 and c_3 affect the weights of *retweet*, *quote* and *reply* in the formula, which directly determine the magnitude of the result. Therefore, in order to explore how different scales of edge weight affect the community detection results, we set three sets of values for c_1, c_2 and c_3, and keep c_4 and c_5 unchanged, giving three different sets of edge weights, and then execute the community detection algorithm to compare the quality of different detection process. The value of the three sets of c_1, c_2 and c_3 are $(0.1, 0.5, 1)$, $(1, 5, 10)$, $(10, 50, 100)$. The detection results are shown in Figure 7.

We can find that when $(c_1, c_2, c_3) = (0.1, 0.5, 1)$, only 28 communities are detected, and the



(a) $(c_1, c_2, c_3) = (0.1, 0.5, 1)$ (b) $(c_1, c_2, c_3) = (1, 5, 10)$ (c) $(c_1, c_2, c_3) = (10, 50, 100)$

Figure 7: The Community Detection Results of Three Sets of Edge Weights

largest community contains more than 50% of the nodes in the network. This is obviously not an ideal detection result. However, when (c_1, c_2, c_3) becomes $(1, 5, 10)$, the number of communities increases to 38, and the percentage of nodes in the largest community decreases to 25%, indicating that the quality of the detection is improving. Besides, when the (c_1, c_2, c_3) increases to $(10, 50, 100)$, there is only a slight change compred to the results of $(1, 5, 10)$. In this case, we can conclude that when the value of (c_1, c_2, c_3) exceeds the $(1, 5, 10)$, we can get a the high quality detection result, and the nodes in the largest community will remain around 25%.

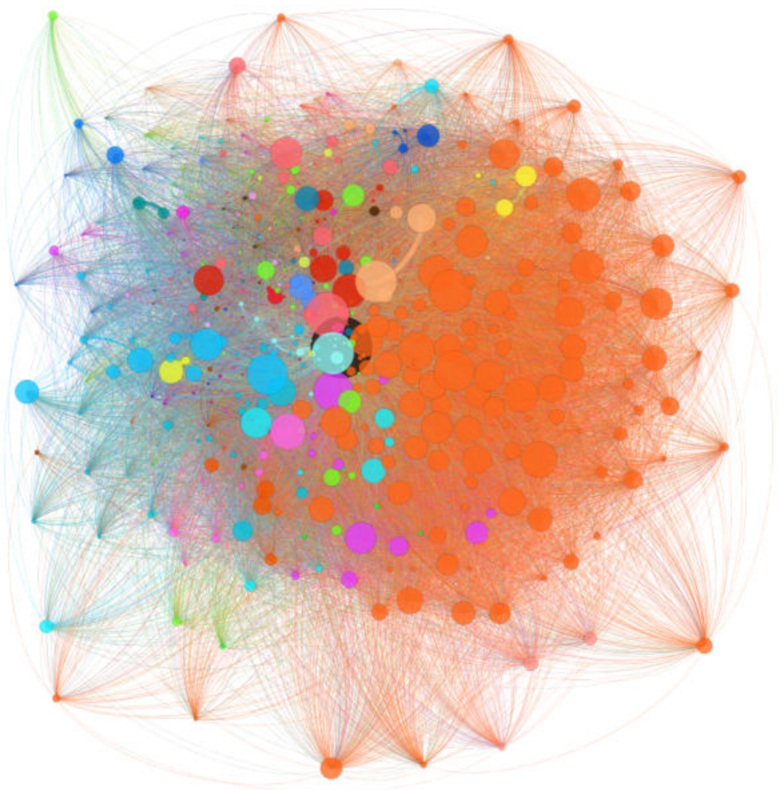


Figure 8: Artwork of Community Detection

The result of the community detection with setting the (c_1, c_2, c_3) to $(1, 5, 10)$ is visualized in Figure 8, which is an artwork. The nodes' different color represents various detected communities, and the black node in the center is the ego. The vertices' size represents the each node's in-degree, and the thickness of the edge represents the edge weights. We can find that many of the nodes with high in-degree compose the largest community. Simultaneously, other communities still have several high-degree nodes, even though their size is much smaller. We can summarize the phenomenon as follows: Although ego and some popular users form a relatively large community, the center of the network, other users far from the center and with certain popularity can still form a small community around themselves.

All in all, in order to perform high-quality community detection, we need to determine the best parameters by constantly adjusting and comparing the detection results of different coefficients. Later, the visualization of the detection results will give us an intuitive understanding of the distribution of communities in the entire network.

V CONCLUSION

In this project, we have successfully established a complete pipeline of social network analysis on Twitter and proved the feasibility by constructing and analyzing two egocentric networks. Firstly, we performed all operations and calculations on a remote Linux server in a custom virtual environment. The attributes of "remote" and "customization capability" will not only contribute to similar projects but also have practical value for any research that requires remote operation or a concise programming environment. Secondly, the APIs we chose to mine relational data and attribute data are the two most common types of Twitter APIs. We have created modules to establish the connection and grab the raw data from these two APIs, and through these modules, other researchers can obtain and save the data predefined by us with one call, or determine the type of data they require.

Additionally, we delved into the powerful software package *networkx* to generate and analyze the networks, and then execute the community detection algorithm on the generated networks. In terms of results, both network analysis and community detection have achieved overall success. That is, all results are consistent with expectations or accurately reflect the underlying attributes. However, this step still has a limitation that the weight addition method is too personal to apply to other types of networks. If others want to perform a similar analysis with custom weights, they should pay attention to the data structure. Finally, our most valuable contribution is integrating all modules into a single Python-class, making our work compatible or reproducible to other research fields.

In order to make the technology in this project applicable to a broader range of fields, further research is needed to optimize the weight calculation model by involving more general and abstract factors. Also, a standardized dataset structure should be designed to meet the requirement of the new model.

References

Ahmed, W., Vidal-Alaball, J., Downing, J. & López Seguí, F. (2020), 'Covid-19 and the 5g conspiracy theory: Social network analysis of twitter data', *Journal of medical Internet research* **22**(5), e19458.

Bandyopadhyay, S., Rao, A. R. & Sinha, B. K. (2011), *Models for social networks with statistical applications*, Vol. 13 of *Advanced quantitative techniques in the social sciences*, SAGE, Los Angeles.

Beguerisse-Díaz, M., Garduño-Hernández, G., Vangelov, B., Yaliraki, S. N. & Barahona, M. (2014), 'Interest communities and flow roles in directed networks: the twitter network of the uk riots', *Journal of the Royal Society, Interface* **11**(101), 20140940.

Bohlin, L., Edler, D., Lancichinetti, A. & Rosvall, M. (2014), Community detection and visualization of networks with the map equation framework, *in* Y. Ding, R. Rousseau & D. Wolfram, eds, 'Measuring Scholarly Impact', Springer International Publishing and Imprint: Springer, Cham, pp. 3–34.

Butts, C. T. et al. (2008), 'Social network analysis with sna', *Journal of statistical software* **24**(6), 1–51.

Chatfield, A. & Brajawidagda, U. (2012), 'Twitter tsunami early warning network: a social network analysis of twitter information flows'.

developer.Twitter (2020-08-26 Access), 'Twitter api documentation'.
URL: *https://developer.twitter.com*

Diestel, R. (2005), *Graph Theory (Graduate Texts in Mathematics)*, Springer.

Ediger, D., Jiang, K., Riedy, J., Bader, D. A. & Corley, C. (2010), Massive social network analysis: Mining twitter for social good, *in* 'Proceedings', CPS and Ieee Computer Society, Los Alanitos, Calif., pp. 583–593.

Giorgos, C. (2010), 'Social network analysis (sna) including a tutorial on concepts and methods', *National University of Singapore, Singapore* .

Hansen, D. L., Shneiderman, B., Smith, M. A. & Himelboim, I. (2020), *Analyzing social media networks with NodeXL: Insights from a connected world*, second edition edn, Morgan Kaufmann, an imprint of Elsevier, Cambridge, MA.

Haythornthwaite, C. (1996), 'Social network analysis: An approach and technique for the study of information exchange', *Library & Information Science Research* **18**(4), 323–342.

Himelboim, I., Smith, M. A., Rainie, L., Shneiderman, B. & Espina, C. (2017), 'Classifying twitter topic-networks using social network analysis', *Social Media + Society* **3**(1), 205630511769154.

Kim, J. & Hastak, M. (2018), 'Social network analysis: Characteristics of online social networks after a disaster', *International Journal of Information Management* **38**(1), 86–96.

mapequation.org (2020-05-28 Access), 'Infomap - network community detection using the map equation framework'.
URL: *https://www.mapequation.org*

Moreno, J. L. (1953), 'Who shall survive? new york: Beacon house'.

Myers, S. A., Sharma, A., Gupta, P. & Lin, J. (2014), Information network or social network?, *in* C.-W. Chung, ed., 'WWW '14 Companion : proceedings of the 23rd International Conference on World Wide Web : April 7-11, 2014, Seoul, Korea', International World Wide Web Conferences Steering Committee, [Place of publication not identified], pp. 493–498.

NetworkX (2020-08-22 Access), 'Networkx — networkx documentation'.
URL: *https://networkx.github.io*

Prochaska, J. J., Pechmann, C., Kim, R. & Leonhardt, J. M. (2012), 'Twitter= quitter? an analysis of twitter quit smoking social networks', *Tobacco control* **21**(4), 447–449.

Scott, J. (1988), 'Social network analysis', *Sociology* **22**(1), 109–127.

Serrat, O., ed. (2017), *Knowledge solutions: Tools, methods and approaches to drive organizational performance / by Olivier Serrat*, Springer, Singapore.

tweepy.org (2020-08-02 Access), 'Tweepy documentation — tweepy 3.9.0 documentation'.
URL: *http://docs.tweepy.org*

Zhou, T., Wang, B. H., Han, X.-p. & Shang, M.-s. (2010), 'Social network analysis and its application in the prevention and control of propagation for public opinion and the epidemic', *Journal of Systems Engineering* **25**(6), 742–754.