Social network analysis of WallStreetBets Subreddit

Group 12

Github repository: https://github.com/MartinSchmauch/network-analysis-for-reddit

Martin Schmauch
University of Oulu
Oulu, Finland
Martin.Schmauch@student.oulu.fi

Arda Tekin
University of Oulu
Oulu, Finland
atekin22@student.oulu.fi

Nikolaos Georgiadis
Group Leader
University of Oulu
Oulu, Finland
ngeorgia22@student.oulu.fi

***Abstract*—Blog communities are used from people to express their opinion and interact with other individuals regarding a specific topic. They consist one of the types of social networks around the web, where people can post content, vote for a post, comment and more. Our analysis will be focused on the interactions of Reddit bloggers through comments of some top posts chosen from a top subreddit community at a specific time. The heart of our analysis consists of a huge graph which represents the interactions of the individuals. After explaining our dataset and our methodologies about data extraction, we will present and discuss some information. We will discuss about how much users are important, about centrality measures and other metrics, we will make some graph representation of data and we will plot some distributions to check if power law can be applied. Finally, we will write some conclusions we can get from the data extracted.**

Keywords—reddit, sna, wallstreetbets, networkx, python

I. Introduction

Reddit is a website that users can post media, links, photos, polls, and video contents for other people to see. It also can be identified as Forum as well as a blog. Use of reddit is dependent on the user. Some people use the website to talk about themselves or experiences while other may use the website to talk about certain topics and discuss within community. The place that these talks happening are subreddits. These subreddits can be identified as small blogs that are within the website for different subject or interests. Users of these subreddit can post whatever they want if it is in community guidelines. Users can vote or comment in the posts that are in these subreddits. Vote system helps users to identify good posts from bad ones while we can also use comments to see reaction and interaction of the users. The most upvoted posts considered as top posts in those subreddits. While number of comments may warry from post to post though the top posts usually have the highest number of comments. We wanted to focus on an active subreddit so we can get fresh data from the users. As of 2021, Reddit has more than 2.8 million subreddits and 130,000 active communities. So, we choose the most active subreddit at 24/04/2022 which was r/WallStreetBets. WallStreetBets is a subreddit for users to talk about the market and trading. Also, WallStreetBets community have a lot of influence in the stock market like GameStop short squeeze [6]. Where stocks of GameStop skyrocketed and became most traded stock in United States. Each day, there are around eight hundred posts and 50,000 comments [7] in WallStreetBets. Even though the top posts are the most upvoted ones Reddit users can also buy PowerUps for their subreddit of choice these PowerUps makes users comment standout and boosts the communities for 5$ per month or users may prefer Reddit Premium for 6$ per month which gives them few other benefits which include Coins. Coins may be used to gift awards; the award makes the posts standout more or just to reward the user and the community for making a good post. We will focus on our goals and how we can solve it in Sec. II. Sec. III will tell about our dataset and how we collected it. Then Sec. IV outlines our general methodology and will detail it more in Sec. V. We will talk about our results and discuss our graph extracted data in Sec. VI. Section VII. will conclude our work.

II. Problem Description

We are needed to create network graph and do analysis for most active blogs in our case, subreddits. Our goal was to find the most influential users in the WallStreetBets, on a specific number of posts in a small-time span. We did not have a ready dataset, so we had to get our own data. So, we got some top posts of last month which were the most upvoted ones. We had two ways to find the most influential user. We could find it by getting upvote counts and the user with the most upvotes can be the most influential user which has already been done in some research or we could look at the comments of the users in top posts and we can decide the most influential user by their interactions with the community. We decided to get the most influential users by their interactions.

III. Dataset Description

For this project, we collected our own data from Reddit using the PRAW (Python Reddit API Wrapper) library[4]. It allows us to access the Reddit API by creating and using a reddit object in Python. In addition, we used mainly NetworkX and Matplotlib libraries to make the graph and plot data. We created submission objects with reddit.subreddit("wallstreetbets").top("month") method, pointing to the reddit page of our desire. These objects can be used to extract data from the specific reddit page and

from a post. Since we are only interested in the relation of the comments on each post, we worked on the submission.comments attribute that provides a list of top level comments (CommentForest). These top-level comments then again contain comment replies (CommentForest) in a forest like structure of trees.

To work with these comments, some preprocessing needs to be done. Some comments of lower levels are mostly hidden to hide complexity of the comment reply feed. On the reddit webpage there is a button stating, "N more replies", which is represented by a MoreComments object in PRAW. Here we needed to unwrap these comments by showing all comments until the lowest level instead of the MoreComments object. We used the replace_more method to achieve this for every top-level comment. This resulted in a list of forests with all top-level comments. The comment object had a lot of attributes from which we have listed the most relevant ones in the following[5].

Attribute	Description
author	Provides an instance of "Redditor" that contains name of the user.
created_utc	Timestamp of comment creation in Unix Time.
is_submitter	Whether the comment author is also the author of the submission.
replies	Provides an instance of "CommentForest".
submission	Provides an instance of "Submission", that the comment belongs to.

From this list of top level CommentForests we created an edgeList, representing comment replies. Everytime an user replied to a comment, an edge is created containing the name of the users as nodes. A directed edge is pointing out from the replier to the author of the comment. We also included meta data like a list of timestamps of the comments creation, for each node-user and weights for each edge. Each element of the edgeList and nodelist looks as following respectively:

(replier (String), author (String))

(user (String), [timestamp1, timestamp2,… (String)])

This edgeList is used to create the graph and analyze the relations of the users replying to each other's comments. Before adding the edges to the graph, we checked, whether a user has replied more than once to a specific other users comments. If that is the case, the weight of this edge will be increased according to the number of replies. The data is stored as .gml file in graph format already. Therefore, we created a graph object and add the edges. An important thing to note is that the dataset's graph contains clusters which correspond to each post-community and nodes at each cluster represent users of the post. Because of the massive node-edge data of the graph, plotting with Networkx and Matplotlib was difficult, so we imported the .gml file at a software which is called Gephi and created a .gephi file which provides a more detailed observation for the graph.

IV General Methodology

The general procedure for our project starts at looking for suitable posts in the "WallStreetBets" subreddit. We decided to choose the top five active posts of the last month during 24/4/2022 regarding user interaction in form of comments. WallstreetBets was the top one subreddit during that day. We took a limit of max three thousand level 0 comments per post (we collected all the replies of these, and the data was collected at 5/5/2022). From this data we created a network by connecting user nodes by their commenting behavior. Furthermore, we used Gephi to make a better visualization of our data graph, giving each cluster a color and making the edges thicker regarding to their weight. Next, we examined the nature of our network and analyzed it using common metrics like diameter, in- and out degree, etcetera. Finally, we summarized our results and drew conclusions based on our observed data.

V Detailed Methodology

In this section, we will explain in more detail, step by step, our procedure for most of the implementations, including reddit data extraction, graph making, visualization, calculation of the global properties of the established network and other graph data extraction.

Reddit data extraction

We have already mentioned in Dataset Description a lot of functions which we used to get data from reddit. What we want, is to get a list with all the nodes (username of every comment) and set as attribute to each node, a list with all the dates that user-node made a comment (list of comment/reply timestamp) and a color, depending on the post it belongs. Also, we need a list of the edges and their corresponding weight. After we have the list of nodes and edges with attributes, we can make the graph. From now on, when we say that we add comments to the list of nodes, we will mean that we add the usernames of the owners of the comments:

-For every of the first 5 submission(post) objects in the list of reddit.subreddit("wallstreetbets").top("month"), we iterate submission.comments. These are the top 5 submissions from the subreddit we are interested in

-For every level 0 comment of a submission:

-We put that comment in the list of nodes

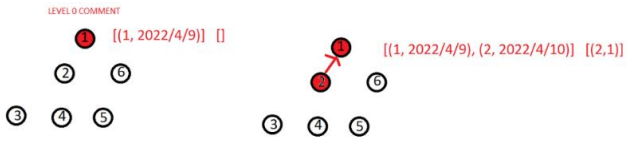
-We add as attributes its time of creation (we use comment.created_utc) and a letter which corresponds to the post color.

-We iterate its replies(comment.replies) and adding them to the list as well. When iterating a reply, we add the following edge in a list: (reply, comment).

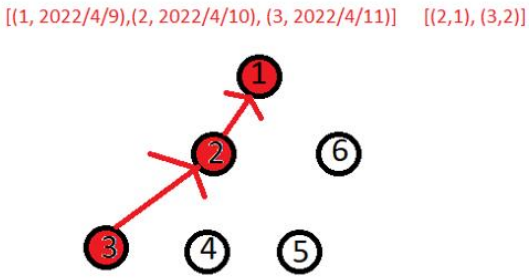
-Continuing from reply to nodes, we recursively repeat these actions in preorder

-After repeating for each level 0 comment of a submission, we combine the list of edges/nodes. Now we have the whole node and edge list of a submission

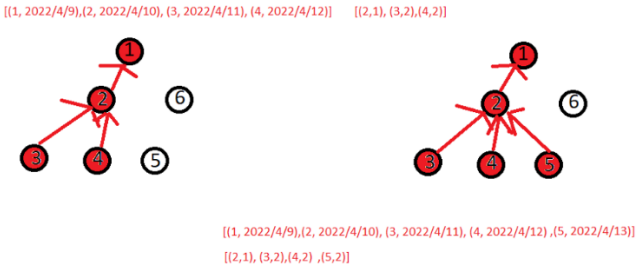
We can observe a simple example bellow for a single comment. The first list is for the nodes and the second is for the edges:



Step 1,2



Step 3



Step 4 ,5

Graph making and visualization

After each node and edge list we get from a submission, we start to add nodes and edges to the graph with the methods graph.add_node() and graph.add_edge()(NetworkX methods to add nodes/edges with/without attributes in a graph object). During iterating the submission lists, we search for node and edge duplicates:

-If we find one node duplicate, we keep the first one only, but we append the date of the second one to the first. So, in the end, we will have only unique users, with the list of the dates they commented as attribute.

-We search how many occurrences each edge has, so we can set its weight equal to how many times appears in the list (how many times a person replied to the same person)

By completing the previous tasks for each submission lists, we have the complete graph as a python Networkx object. Unfortunately, the graph is huge, and it was not possible to plot it in a clear version with matplotlib. Using Gephi, after we exported the graph object at .gml format with the help of the function nx.write_gml() , we were able to import it there. By importing the .gml file at Gephi, we could color the nodes depending on the color attributes we gave them earlier. Furthermore, we runned the layout “ForceAtlas2” to get a better view of the graph.

Note:

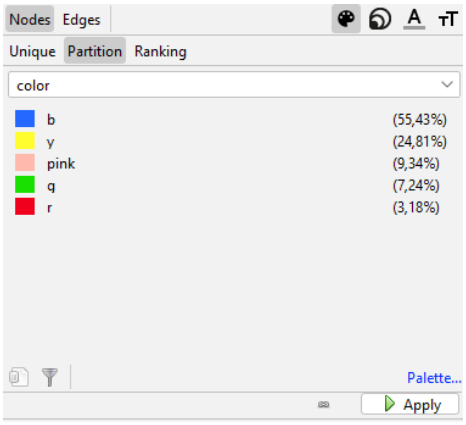
Top 1 post -> red

Top 2 post -> blue

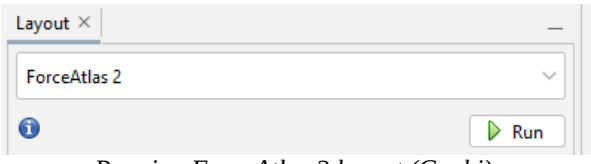
Top 3 post -> green

Top 4 post -> yellow

Top 5 post -> pink



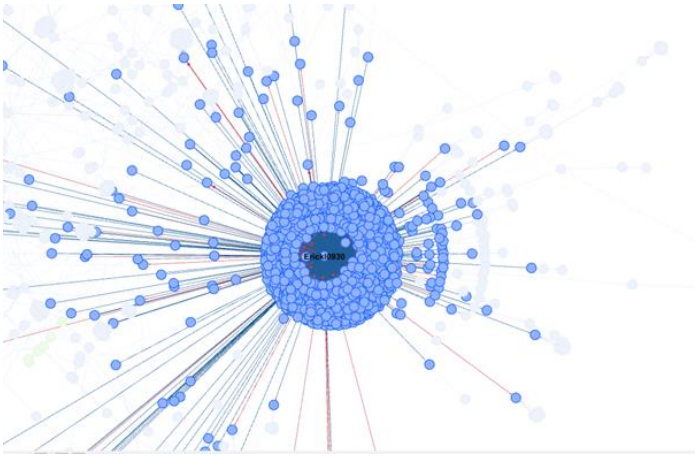
Coloring nodes by attribute (Gephi)



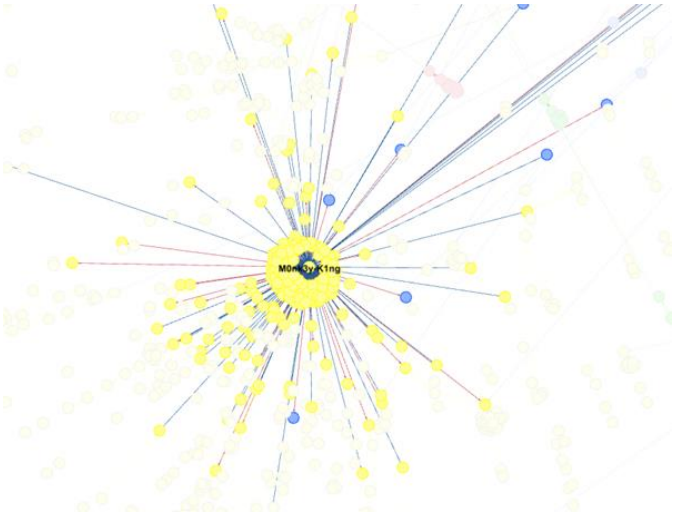
Running ForceAtlas 2 layout (Gephi)

Some important people can be observed bellow. Red arrows indicate the replies that user did.

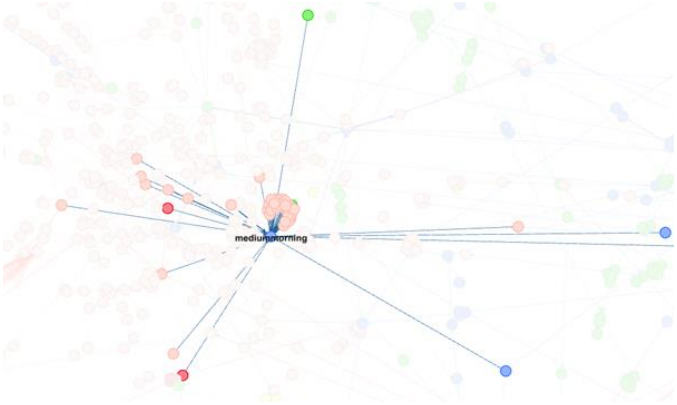
The blue arrows the replies he got.



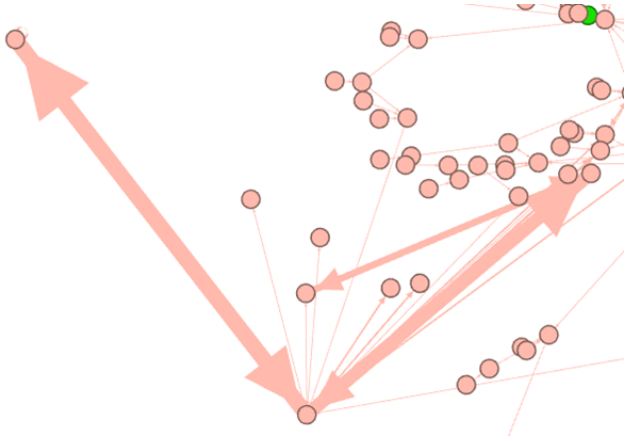
Most influencial user of Top 2 post

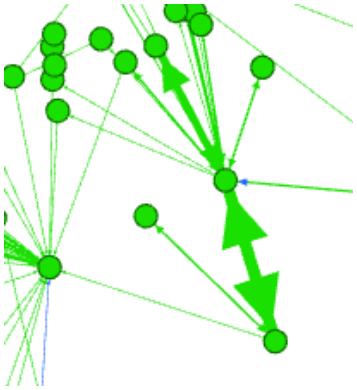


Most influencial user of Top 4 post



Most influencial user of Top 5 post





Big weight edges(people who replied a lot of times to the same person)

Calculation of the global properties of the established network

Most of the properties of the network have been calculated with Networkx library functions. Due to the fact that our graph has a lot of nodes without edges(people who made comments but not replies Figure 23) and a lot of nodes with only self-links(people who commented on themselves only, see Figure 24) we have considered 2 versions of the graph for the most of the calculations, the original undirected version and an undirected subgraph which is not containing the nodes referenced above. Properties like average clustering coefficient and average closeness centrality are being scaled closer to zero if we have a big fraction of those referenced nodes. We want to observe those values for the subgraph that only contains the connections between individuals as well. At this section, we are going to explain which methods we used to calculate these properties and we are not referring at any numerical values yet(see Results and Discussions section for this)

The original directed graph has an amount of **9053 nodes and 5768 edges**(Python, Gephi). If we remove the nodes

without edges and the nodes with only self-loops the nodes remaining are 4657 and the edges 5757 (Python, Gephi). From this, we can observe that there are 11 nodes with only self-loops and 9053 – 4657 – 11 = 4385 nodes without edges. So, almost half of the users of the dataset don't make replies, just posts. We compute the graph with the less nodes, by checking each node's neighbor. If the only neighbor is the same node or if there is not any neighbor we remove the node(you can get the neighbors of v from graph G by calling G[v]). After computing the new graph, we export it as .gml and imported it at Gephi in order to calculate the new amount of nodes/edges.

From now on, we consider the graph undirected.

Clustering coefficient of the graph is a way to tell how much transitivity can be seen in the whole graph. The definition of transitivity is:

''If vertex u is connected by an edge with v, and v is connected
by an edge with w, then u is connected by an edge with w'' [1]

A high clustering coefficient is translated to a network with high transitivity (People belong to tight groups) Clustering coefficient(normalized) of graph can be calculated as the average clustering coefficient of each node(normalized) [1]. We use this formula in order to calculate the value of one node[3]:

For unweighted graphs, the clustering of a node u is the fraction of possible triangles through that node that exist,

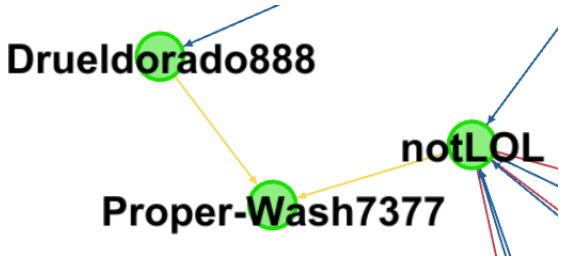
$$c_u = \frac{2T(u)}{deg(u)(deg(u)-1)},$$

where $T(u)$ is the number of triangles through node u and $deg(u)$ is the degree of u.

Node clustering coefficient formula

We are using nx. average_clustering() function which computes the average of the nodes clustering coefficient using the previous formula.

The **diameter** of a graph is "the length of the longest shortest path between any pair of nodes in the graph"[2]. It is a way to tell how many people at most, can be included between the connection of two individuals. For example, if user x and y reply on user z, x and y can be connected via a path of length 2 through z(at this case, if we consider the directions, we can't say that x and y can be connected. Thus, if x replied to z and z replied to y, x and y can be connected).



Two users replied to the same user. If we do not consider directions, any interaction of individual z between x and y(x, y replied to z or x replied to z and z to y) can connect x and y with a path

A **component** in an undirected graph is a subgraph in which there is a path between every pair of nodes inside it[2]. From the size of the largest component, we can observe which is the largest group of users who can be connected through making/receiving replies in the graph. In order to find the larger component in the undirected version of our graph, we used the NetworkX function nx.connected_components() to get a list of sets of the nodes of all the components. We extracted the maximum length set, and we made a subgraph with G.subgraph() function. Then, we exported the component as Giant_component_subgraph.gml. We used Gephi once again, to visualize the component (see Figure 25)

In-degree centrality and out-degree centrality are referring to the in and out degree respectively of each node[1]. We can gain information about how much users are important by observing their in-degrees (in other words, how many replies they got) or about how much users are active in a particular community-post by observing their out degrees(how many replies they made). We used G.in_degree(), G.out_degree() and G.degree() of NetworkX library to calculate the in,out and general degree(in+out) of each node respectively.

Average path length calculated as the sum of all the shortest path lengths between any two pairs of nodes divided by the number of all those pairs. In other words, we can call it **average shortest path length**. We used Gephi to calculate this. From the length of this path, we can observe how many users in average can be between a connection of two other users.

Closeness centrality of a node is a way to measure how much a node is central regarding to every other node in the graph. The bigger the value of a node's closeness centrality is, the less is its shortest path distance between the other nodes [1]. Users with big closeness centrality can be connected through a small amount of in between users with another user.

We used nx.closeness_centrality() function to get a python dictionary with the users as keys and the values of the

centralities as values. The function uses this formula to calculate each node's closeness centrality(normalized)[3]:

Closeness centrality [1] of a node u is the reciprocal of the average shortest path distance to u over all $n-1$ reachable nodes.

$$C(u) = \frac{n-1}{\sum_{v=1}^{n-1} d(v,u)},$$

where $d(v, u)$ is the shortest-path distance between v and u, and n is the number of nodes that can reach u.

Node closeness centrality formula

Betweeness centrality of a node measures how much a node is included in shortest paths that connect any two pairs of nodes. The users that have high betweeness centrality are important on connecting other pairs of users [1]. We used nx.betweenness_centrality() function that returns a python dictionary with the users as keys and the values of the betweeness centralities as values. The function implements the following formula for each node[3]:

Betweenness centrality of a node v is the sum of the fraction of all-pairs shortest paths that pass through v

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

where V is the set of nodes, $\sigma(s,t)$ is the number of shortest (s,t)-paths, and $\sigma(s,t|v)$ is the number of those paths passing through some node v other than s,t. If $s = t$, $\sigma(s,t) = 1$, and if $v \in s,t$, $\sigma(s,t|v) = 0$

Node betweeness centrality formula

Last but not least, an important thing to note is that we used the previously referenced methods to calculate the **average** and the **variance** of all the centrality measures as well as the **variance of the shortest path length**. Average was simply calculated by the sum of all the corresponding values divided by the number of the total nodes. Variance for the centralities was calculated with Numpy var function providing as parameter a list with the values. For the shortest path length variance, we extracted information from the NetworkX function nx.all_pairs_shortest_path_length(). From this we got a dictionary in the form:

{'user1': {'user1': shortest distance from user1, 'user2': shortest distance from user1,

'user3': shortest distance from user1}, ...}

We could add all the shortest distances except the ones with 0 value (from user1 to user1 for example) in a list and then use var function to calculate the variance.

Other graph data extraction

Some additional data we extracted, were the number of comments made per day in total and for every post individually (see graph in next section). The method we followed to get the posts per day is the following:

Analyzing the data, we already have in the graph object, we first making a list of lists. Each individual list contains the dates of all the comments made from a post (we iterate the graph's nodes, and we add to each list the dates of a node in a way so each list will contain the dates of nodes with same color attribute). So, we have 5 lists in total, one for each post and each list contains the corresponding comment dates. After we got all post's submission dates independently, we added them in a list as well. For each of the 5 posts daily comments plot, we used as the first plotting date the submission date of the post and as final plotting date the maximum date found in the corresponding comment date list. For the day interval we got for each plot (last comment date – post submission_date), we counted how many times each date of that interval can be found in the comment date list. Given the dates as x values and the comments count as y values, we made a bar graph, a normal graph, and a normal graph with log scale to check if power law applies. To plot the total comments per day we combined the 5 lists to 1 and we used as first plot date the oldest submission date and as last plot date the newest comment date.

VI Results and Discussions

Using the methods we explained in the previous section, we get a set of numerical results. For every of the Global properties of the graph, we provide the numerical results in a following table. We discuss about power law fitting in the results of in and out degree distribution we made from the data provided. Furthermore, we present the graphs we made about the daily comments, and we discuss about the power law fitting on these as well.

Table of global properties

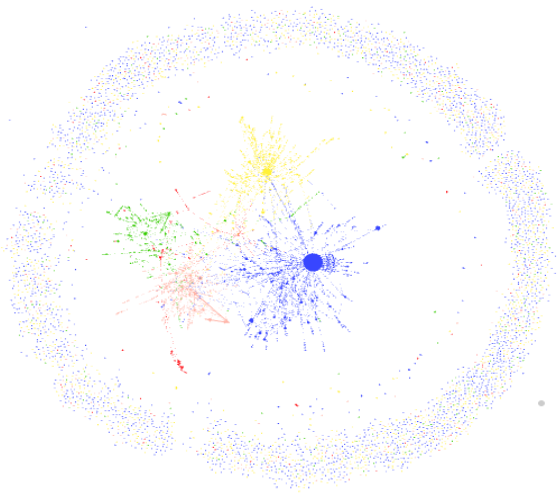


Figure 26: Whole Graph

Our graph has total of 9053 nodes and 5768 edges. The graph shows there are many nodes with multiple edges. There are many users that is replying to multiple users and there are many users getting more than one reply.

The top 10 most influential users according to out-degree

	Username, out-degree
1.	'Erickl0930', 388
2.	'M0nk3y-K1ng', 88
3.	'AutoModerator', 30
4.	'HighlightFantastic64', 15
5.	'MattieShoes', 14
6.	'Angel2121md', 13
7.	'S_Dot_Diggity', 12
8.	'WoolooOfWallStreet', 11
9.	'computery_stuff', 11
10.	'MLXIII', 10

Figure 27: Top 10 Based on Out Degree

Out-degree

The top 10 most influential users according to in-degree

	Username, in-degree
1.	'Erickl0930', 1193
2.	'M0nk3y-K1ng', 216
3.	'sneakersNsadness', 81
4.	'ThetaStoleUrTendies', 64
5.	'mediummorning', 51
6.	'--LiterallyWho--', 51
7.	'HezronCarver', 48
8.	'Newhere84939', 38
9.	'jhillis379', 36
10.	'Sad_Molasses_6753', 31

Figure 28: Top 10 Based on In Degree

The top 10 most influential users according to degree (sum of in- and outdegree)

	Username, degree
1.	'Erickl0930', 1581
2.	'M0nk3y-K1ng', 304
3.	'sneakersNsadness', 81
4.	'ThetaStoleUrTendies', 67
5.	'mediummorning', 53
6.	'HezronCarver', 51
7.	'--LiterallyWho--', 51
8.	'AutoModerator', 44
9.	'jhillis379', 38
10.	'Newhere84939', 38

Figure 29: Top 10 Based on Sum of Degree

When we check degree of our users, we can see that user called 'Erickl0930' has sum of in and out degree 1581 which is the highest, so it is fair to say he is the most popular person in these posts. The reason to his popularity might be related to that 'Erickl0930' is owner of the top second of the posts we selected if we consider that the first post is a community post and not created by the user this puts him to number one post owner and it might explain why he has such high traction. If we check the other owners of the post, we can see user named 'M0nk3y-K1ng' who has sum of in and out degrees 304 which he is after Erickl0930 and lands in being the top two user in figure, but the other owners of the posts are not in our figures of degrees. We can say being the post owner has an effect, but it doesn't guarantee high interactions. Comparing with the total degree of Erickl0930 which is 1581 to total number of edges which is 5768 Erickl0930 had a lot of interactions with the community and most of it getting replies from other users. As in the figure 27 and figure 28 we can see that top 2 of the most influential persons according to in and out degree has the same ranking in both figures. They are also ranked top 2 in the figure 29. It is not hard to say replying to other comments gets you more traction and may lead to getting more replies since the other people might want to reply, though when we look at the figure 27 it doesn't match with the users in figure 28 and figure 29 other than the top 2 users. We can also see that top 2 users also have lot higher degrees than other top users. This might be related to how reddit's comment section works. Reddit default setting for showing comments is set to "Best" so it usually shows the most upvoted and commented post in the top of the comment section but it doesn't only focus on the upvotes or comments it also check the downvotes, age of the comment, how much time the user spent in the community and by these reddit determines to what to show you so usually the top comments always viewed first but sometimes you might also see a comment with less upvotes and comments at the top as well by someone who is entering the post for the first time. This may also tell us that why there is a huge difference between degrees of top 2 users from others. Even between top 2 users there is a huge difference between the number of degrees.

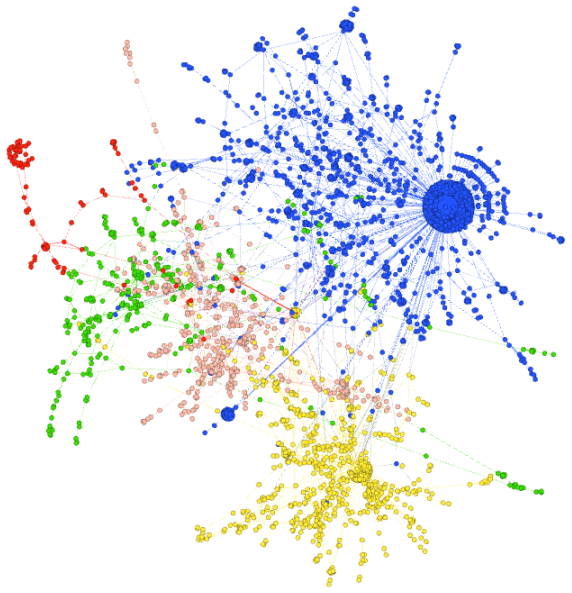


Figure 25: Largest undirected component

The size of the largest connected component is 4352. We can see there are a lot of users connected to each other. Diameter of our graph's undirected version is 21 and directed is 23.

Shortest path length between 2 nodes in the graph:

Average	Variance
5.408344	5.558862

Average clustering coefficient in normalized scale is **0.012274** and excluded nodes without neighbors and with only self-loops is **0.023866**.

In the figures down below, they are values of centralities, in undirected graph and **closeness centrality** is on **normalized scale**:

Average	Value	Variance
In degree Centrality	0.637136	167.157737
Out degree Centrality	0.637136	18.253506
Betweenness Centrality	0.000112	4.504397
Closeness Centrality	0.045653	0.002495

Excluding nodes without neighbors and with only self-loops, undirected:

Average	Value	Variance
In degree Centrality	1.236203	324.205749
Out degree Centrality	1.236203	34.742575
Betweenness Centrality	0.000827	0.000124
Closeness Centrality	0.172542	0.003879

By the values in these figures, we can say there are not many nodes that are between other nodes. Also, from the low closeness centrality we can say nodes are not close to each other.

Degree distributions and power law

Considering the in-degree distribution, you can already observe a curve shape that is typical for the power law. Also, looking at the graphical representation of the distribution on logarithmic scale you can observe a straight line – at least from degree 1 to 12. These assumptions are backed by the results of the power law fitting, as you can see on **Error! Reference source not found.** The fitting returned parameters a = 0.134 and b = 2.165.

$$a \times x^{-b}$$

Although this is good evidence that the in-degree distribution follows a power-law, you need to consider that the number of degrees is quite small and especially at higher degrees the density of data points is very low. This might also be the reason, why the in-degree distribution does not seem to fit a power-law distribution at degrees around 50 and bigger.

The same applies for the out-degree distribution, even though there are even less data points available. The fitting returned a = 0.399 and b = 3.189.

Daily comments and power law

The following figures represent the bar graphs made for daily comments:

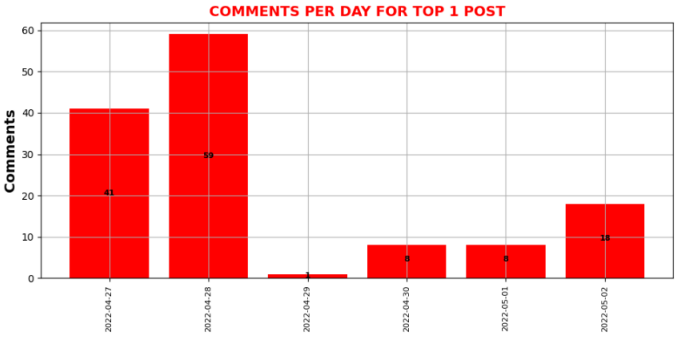


Figure 6: Daily comments for top 1 post

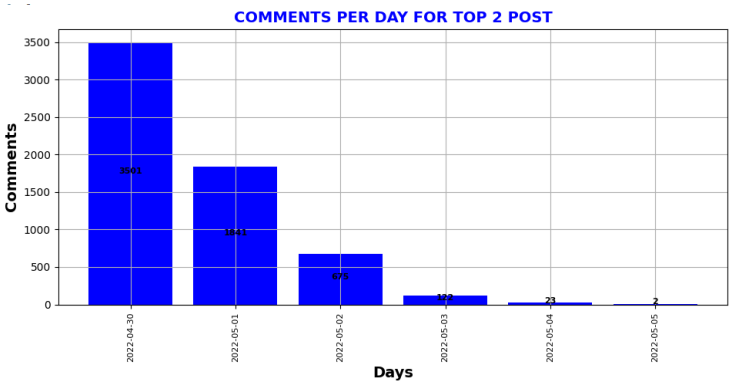


Figure 7: Daily comments for top 2 post

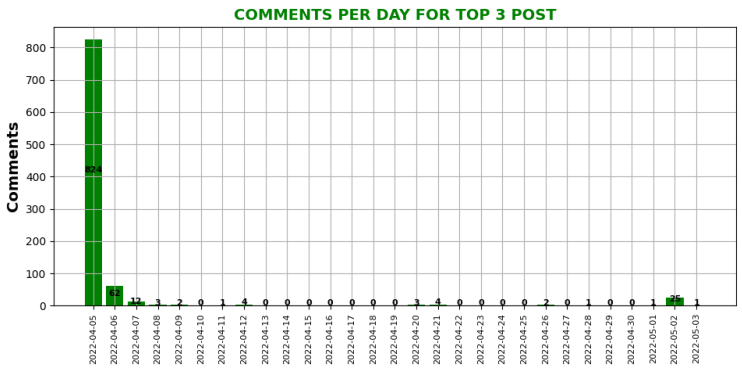


Figure 8: Daily comments for top 3 post

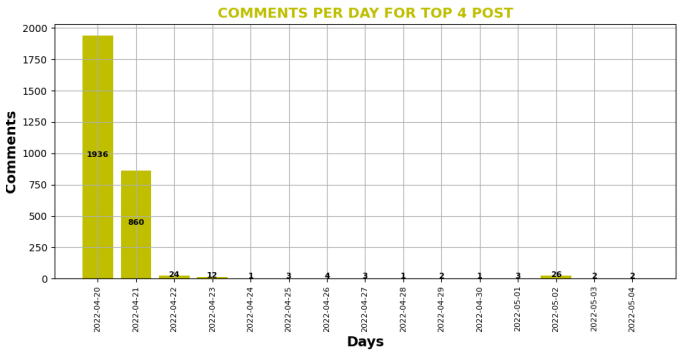


Figure 9: Daily comments for top 4 post

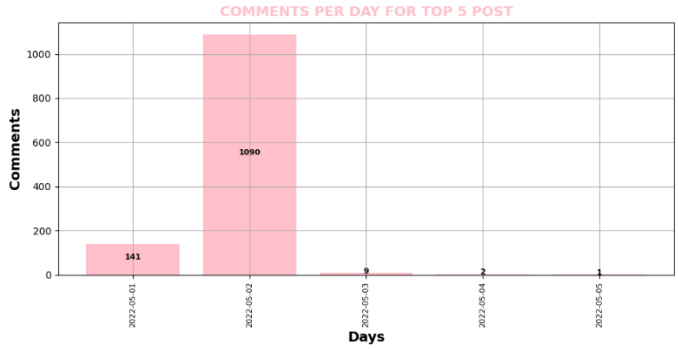


Figure 10: Daily comments for top 5 post

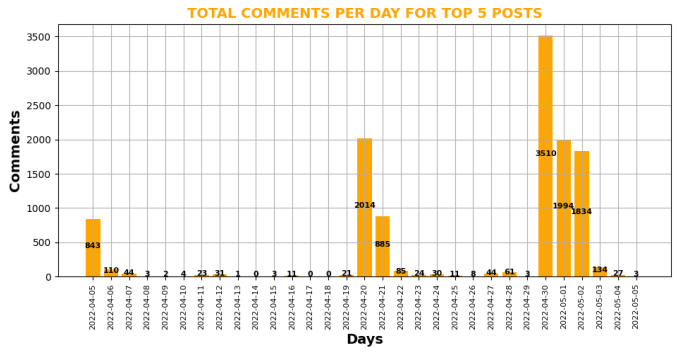


Figure 11: Total daily comments

As in the figures 6 to 10, the posts usually get the most of the comments entered in the first two days and the comments entered in further days seem to be less. We think it is fair to say from these graphs that most posts get most of the user traction in first two days of submission. This fact can also be observed from Figure 11. The total comments were more during the most days of post submissions(days 5/4 - 7/4, 20/4-22/4, 30/4 - 2/5 are close to submission of top 3, top 4 and top 2 post respectively). Using once more the curve fitting function, we could fit a curve for the graph representation version of the daily comments, a curve that approximates best our relation between dates and comments(Figure 12). Using the logarithmic scale graph(we had to exclude the dates which had zero comments because these points couldn't be represented), we can check how much power law applies and for which days, by telling which points of the data are identical with the points of the straight line(fitting function in log scale, Figure 13). From the Figures 13-18, we can say that power law fits more for the top 3 post during the first 6 days. Other daily comments representations seem to be far from power law fitting either because the data was few(new post) or because the amount of daily comments didn't fell with the correct rhythm.

VII Conclusion

We chose WallStreetBets as our focus. Most active subreddit at 24/04/2022. We created a graph that consist of top 5 post of a month that featured comments of the users. They were big groups of people replying to each other and generally replying to the post. After building our graph and looking at our data. We can say solving the problem of finding the most influential user can be solved with checking their interactions, meaning comments. When we look at the number of degrees and total numbers of edges, we can see there are users that excels from other user by just their number of interactions within the community. Especially one user that had way more interactions with the community was 'Erickl0930'. We can also see this in our graph. There were way more actors that were centered around 'Erickl0930' this means he was the main actor that pulled those people in. Based on these we can say last month user named 'Erickl0930' was the most influential user in WallStreetBets.

VIII References

[1] M. Oussalah, "Introduction To Social Network Analysis, Lecture 3. Centrality Metrics", University of Oulu, 2022

[2] M. Oussalah, "Introduction To Social Network Analysis, Lecture 2. Graph Basis", University of Oulu, 2022

[3] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, "Exploring network structure, dynamics, and function using NetworkX", in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008

[4] Data Description: how to extract comments: Bryce Boe, "Comment Extraction and Parsing", praw, 2022. [Online]. Available: https://praw.readthedocs.io/en/stable/tutorials/comments.html

[5] Comment class: Bryce Boe, "Comment", praw, 2022. [Online]. Available: https://praw.readthedocs.io/en/stable/tutorials/comments.html

[6] GameStop short squeeze. (2021, January). Wikipedia: https://en.wikipedia.org/wiki/GameStop_short_squeeze

[7] E. Green, "Analyzing the r/wallstreetbets hivemind," 14 September 2021. [Online]. Available: https://medium.datadriveninvestor.com/analyzing-the-r-wallstreetbets-hivemind-84529946f6e8.

IX Appendix

Top 5 Posts of last month collected at 5/5/2022

Subreddit: WallStreetBets (Top 1 at 24/4/2022)

Top 1

https://www.reddit.com/r/wallstreetbets/comments/txma40/wallstreetbets_predictions_tournament_for_april/?utm_source=share&utm_medium=web2x&context=3

Top 2

https://www.reddit.com/r/wallstreetbets/comments/ufc3eu/alright_boys_time_for_some_classic_wsb_degeneracy/?utm_source=share&utm_medium=web2x&context=3

Top 3

https://www.reddit.com/r/wallstreetbets/comments/twk0qn/our_savior/?utm_source=share&utm_medium=web2x&context=3

Top 4

https://www.reddit.com/r/wallstreetbets/comments/u7wtfh/5k_to_100k_overnight_nflx_put/?utm_source=share&utm_medium=web2x&context=3

Top 5

https://www.reddit.com/r/wallstreetbets/comments/ug4351/suffering/?utm_source=share&utm_medium=web2x&context=3

Degree Distribution

degree: 0 , count: 4385 ; fraction: 0.4843698221584005

degree: 1 , count: 3045 ; fraction: 0.3363525903015575

degree: 2 , count: 844 ; fraction: 0.09322876394565338

degree: 3 , count: 295 ; fraction: 0.0325858831326632

degree: 4 , count: 158 ; fraction: 0.017452778084612836

degree: 5 , count: 88 ; fraction: 0.009720534629404616

degree: 6 , count: 48 ; fraction: 0.0053021097978570636

degree: 7 , count: 38 ; fraction: 0.004197503589970176

degree: 8 , count: 24 ; fraction: 0.0026510548989285318

degree: 9 , count: 18 ; fraction: 0.001988291174196399

degree: 10 , count: 14 ; fraction: 0.0015464486910416436

degree: 11 , count: 9 ; fraction: 0.0009941455870981996

degree: 12 , count: 8 ; fraction: 0.0008836849663095106

degree: 13 , count: 10 ; fraction: 0.0011046062078868884

degree: 14 , count: 8 ; fraction: 0.0008836849663095106

degree: 15 , count: 9 ; fraction: 0.0009941455870981996

degree: 16 , count: 9 ; fraction: 0.0009941455870981996

degree: 17 , count: 4 ; fraction: 0.0004418424831547553

degree: 18 , count: 1 ; fraction: 0.00011046062078868883

degree: 19 , count: 2 ; fraction: 0.00022092124157737766

degree: 20 , count: 6 ; fraction: 0.0006627637247321329

degree: 21 , count: 2 ; fraction: 0.00022092124157737766

degree: 22 , count: 1 ; fraction: 0.00011046062078868883

degree: 23 , count: 1 ; fraction: 0.00011046062078868883

degree: 24 , count: 4 ; fraction: 0.0004418424831547553

degree: 25 , count: 2 ; fraction: 0.00022092124157737766

degree: 26 , count: 3 ; fraction: 0.00033138186236606647

degree: 27 , count: 2 ; fraction: 0.00022092124157737766

degree: 31 , count: 2 ; fraction: 0.00022092124157737766

degree: 34 , count: 2 ; fraction: 0.00022092124157737766

degree: 35 , count: 1 ; fraction: 0.00011046062078868883

degree: 38 , count: 2 ; fraction: 0.00022092124157737766

degree: 44 , count: 1 ; fraction: 0.00011046062078868883

degree: 51 , count: 2 ; fraction: 0.00022092124157737766

degree: 53 , count: 1 ; fraction: 0.00011046062078868883

degree: 67 , count: 1 ; fraction: 0.00011046062078868883

degree: 81 , count: 1 ; fraction: 0.00011046062078868883

degree: 304 , count: 1 ; fraction: 0.00011046062078868883

degree: 1581 , count: 1 ; fraction: 0.00011046062078868883

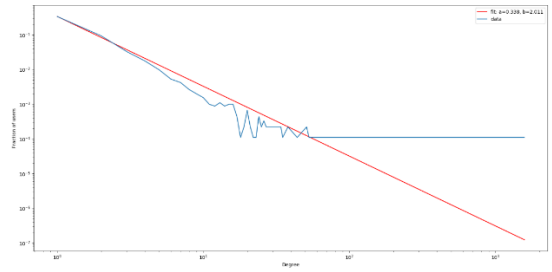


Figure 1: Distribution of Degree on logarithmic scale (power law fitting starts from degree 1, not 0)

In-Degree Distribution

degree: 0 , count: 7222 ; fraction: 0.7977466033359107

degree: 1 , count: 1213 ; fraction: 0.13398873301667955

degree: 2 , count: 270 ; fraction: 0.029824367612945983

degree: 3 , count: 112 ; fraction: 0.012371589528333149

degree: 4 , count: 65 ; fraction: 0.007179940351264774

degree: 5 , count: 33 ; fraction: 0.0036452004860267314

degree: 6 , count: 22 ; fraction: 0.002430133657351154

degree: 7 , count: 21 ; fraction: 0.0023196730365624657

degree: 8 , count: 9 ; fraction: 0.0009941455870981996

degree: 9 , count: 10 ; fraction: 0.0011046062078868884

degree: 10 , count: 7 ; fraction: 0.0007732243455208218

degree: 11 , count: 9 ; fraction: 0.0009941455870981996

degree: 12 , count: 8 ; fraction: 0.0008836849663095106

degree: 13 , count: 5 ; fraction: 0.0005523031039434442

degree: 14 , count: 12 ; fraction: 0.0013255274494642659

degree: 15 , count: 5 ; fraction: 0.0005523031039434442

degree: 16 , count: 2 ; fraction: 0.00022092124157737766

degree: 17 , count: 1 ; fraction: 0.00011046062078868883

degree: 18 , count: 2 ; fraction: 0.00022092124157737766

degree: 19 , count: 2 ; fraction: 0.00022092124157737766

degree: 20 , count: 1 ; fraction: 0.00011046062078868883

degree: 22 , count: 4 ; fraction: 0.0004418424831547553

degree: 23 , count: 4 ; fraction: 0.0004418424831547553

degree: 25 , count: 1 ; fraction: 0.00011046062078868883

degree: 26 , count: 1 ; fraction: 0.00011046062078868883

degree: 28 , count: 1 ; fraction: 0.00011046062078868883

degree: 29 , count: 1 ; fraction: 0.00011046062078868883

degree: 31 , count: 1 ; fraction: 0.00011046062078868883

degree: 36 , count: 1 ; fraction: 0.00011046062078868883

degree: 38 , count: 1 ; fraction: 0.00011046062078868883

degree: 48 , count: 1 ; fraction: 0.00011046062078868883

degree: 51 , count: 2 ; fraction: 0.00022092124157737766

degree: 64 , count: 1 ; fraction: 0.00011046062078868883

degree: 81 , count: 1 ; fraction: 0.00011046062078868883

degree: 216 , count: 1 ; fraction: 0.00011046062078868883

degree: 1193 , count: 1 ; fraction: 0.00011046062078868883

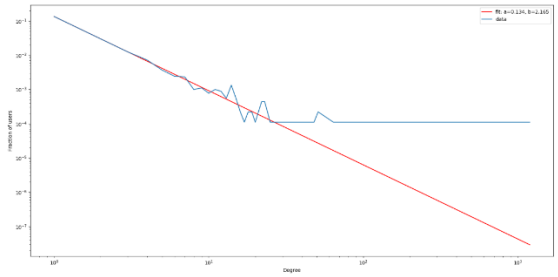


Figure 2: In-Degree Distribution on logarithmic scale. Power Law function fit, using scipy curve_fit function (power law fitting starts from degree 1, not 0).

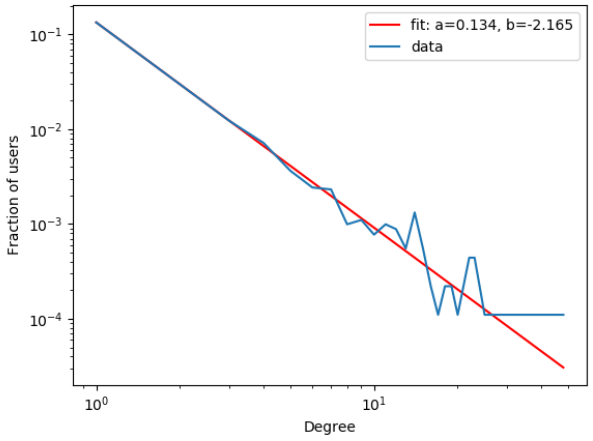


Figure 3: Zoom in from Figure 2 - Degree limited to 50 to focus on data rich region

Out-Degree

degree: 0 , count: 4839 ; fraction: 0.5345189439964653

degree: 1 , count: 3612 ; fraction: 0.39898376228874405

degree: 2 , count: 397 ; fraction: 0.043852866453109464

degree: 3 , count: 106 ; fraction: 0.011708825803601016

degree: 4 , count: 49 ; fraction: 0.005412570418645753

degree: 5 , count: 15 ; fraction: 0.0016569093118303324

degree: 6 , count: 9 ; fraction: 0.0009941455870981996

degree: 7 , count: 7 ; fraction: 0.0007732243455208218

degree: 8 , count: 4 ; fraction: 0.0004418424831547553

degree: 9 , count: 4 ; fraction: 0.0004418424831547553

degree: 10 , count: 2 ; fraction: 0.00022092124157737766

degree: 11 , count: 2 ; fraction: 0.00022092124157737766

degree: 12 , count: 1 ; fraction: 0.00011046062078868883

degree: 13 , count: 1 ; fraction: 0.00011046062078868883

degree: 14 , count: 1 ; fraction: 0.00011046062078868883

degree: 15 , count: 1 ; fraction: 0.00011046062078868883

degree: 30 , count: 1 ; fraction: 0.00011046062078868883

degree: 88 , count: 1 ; fraction: 0.00011046062078868883

degree: 388 , count: 1 ; fraction: 0.00011046062078868883

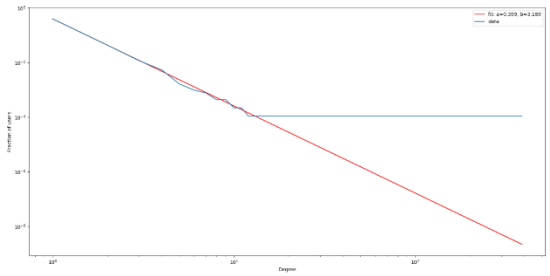


Figure 4: Out-Degree Distribution on logarithmic scale. Power Law function fit, using scipy curve_fit function (power law fitting starts from degree 1, not 0).

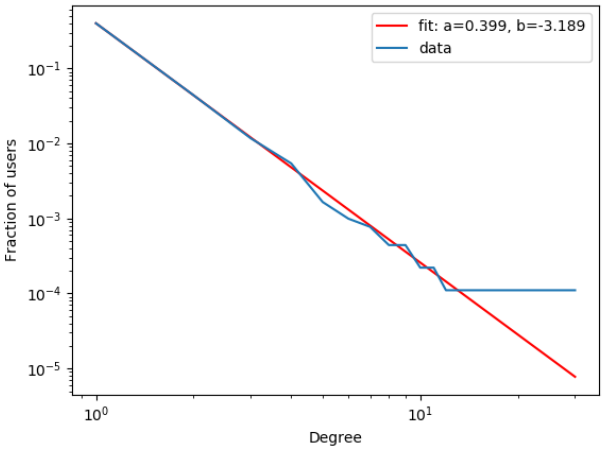


Figure 5: Zoom in from Figure 4 - Degree limited to 50 to focus on data rich region

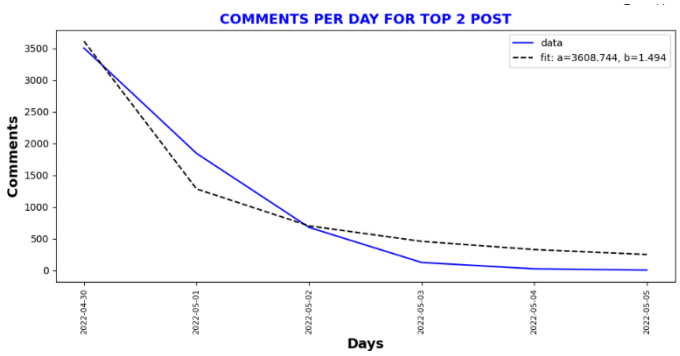


*Figure 12: Daily comments graph representation(blue). The black line represents the fitting curve of the form: a*x^(-b). Note that the date numerical distances are 1.*

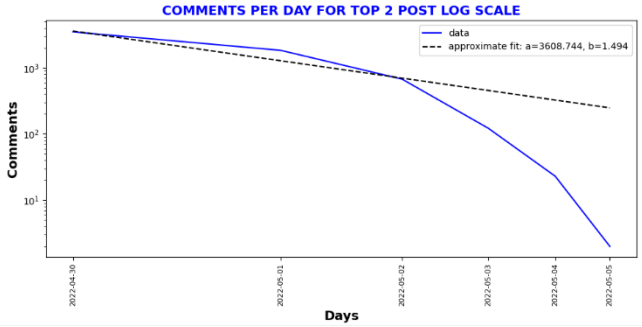


Figure 13: Log scale version of Figure 12.

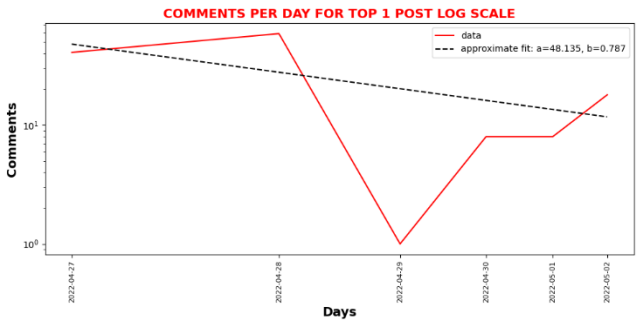


Figure 14: Log scale of daily comments of top 1 post.

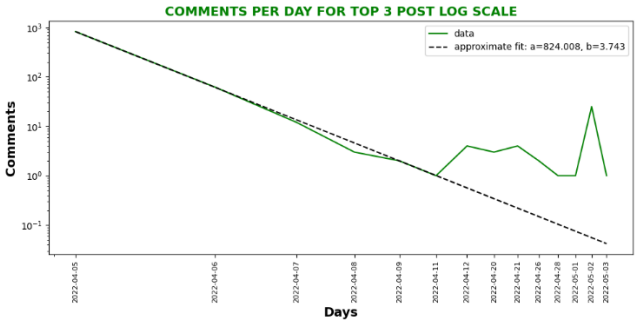


Figure 15: Log scale of daily comments of top 3 post.

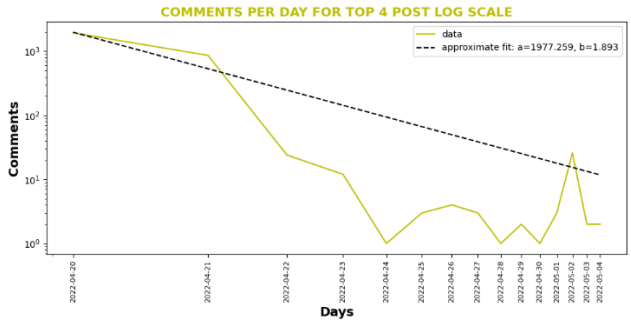


Figure 16: Log scale of daily comments of top 4 post.

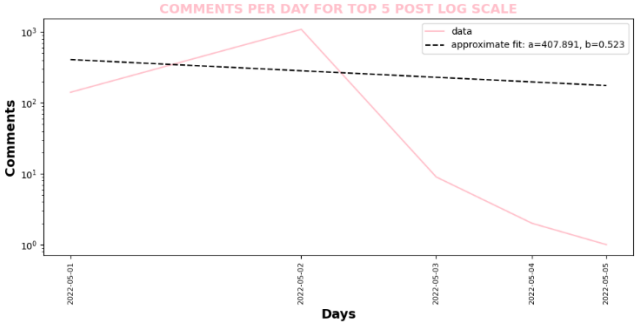


Figure 17: Log scale of daily comments of top 5 post.

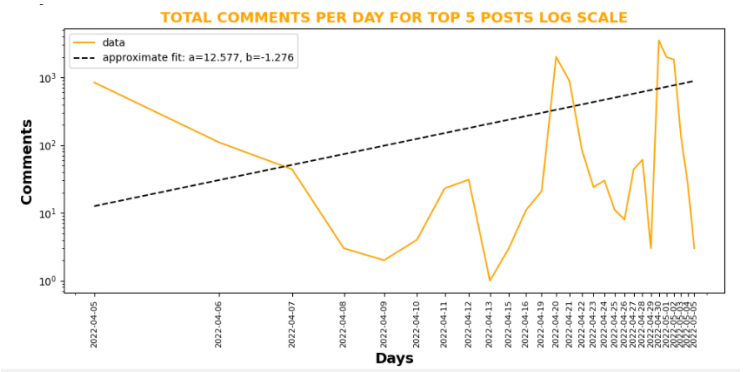


Figure 18: Log scale of total daily comments

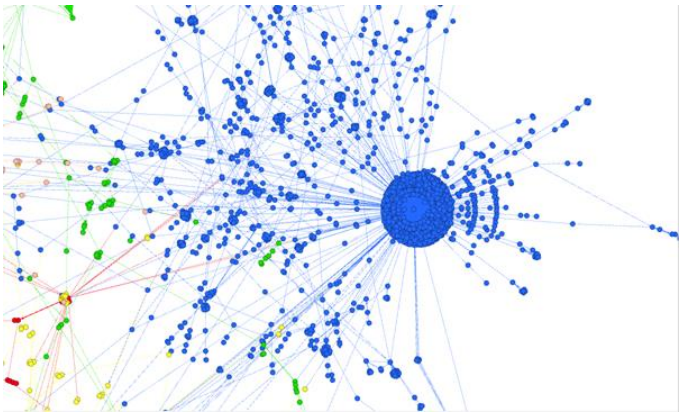


Figure 19: Top 2 post community graph(blue)

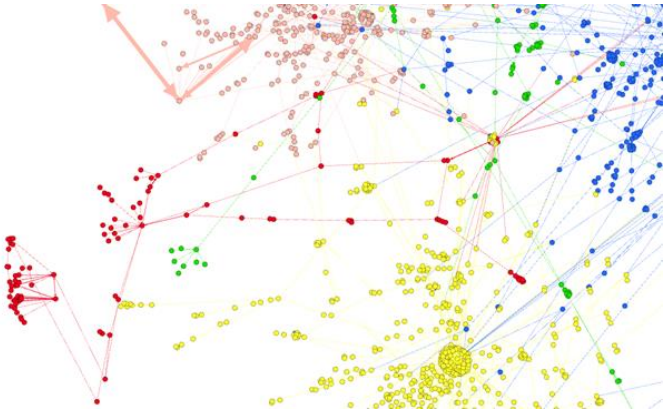


Figure 20: Top 1 and 4 post communities graph(red and yellow)

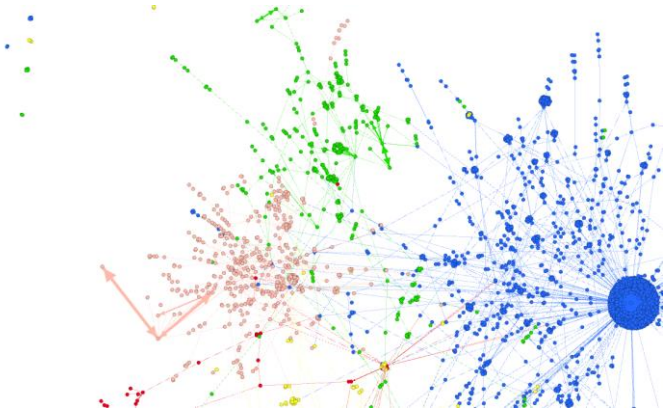


Figure 21: Top 3 and 5 post communities graph(green and pink)

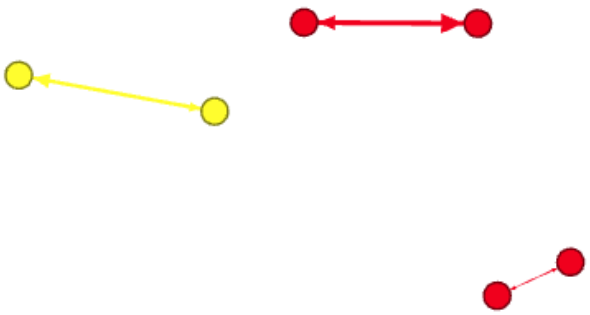


Figure 22: Users who reply to each other

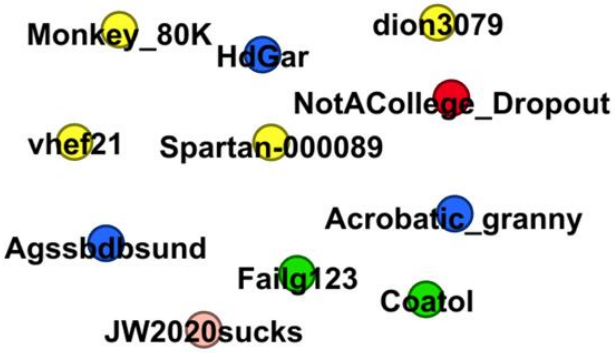


Figure 23: Users who only did comments

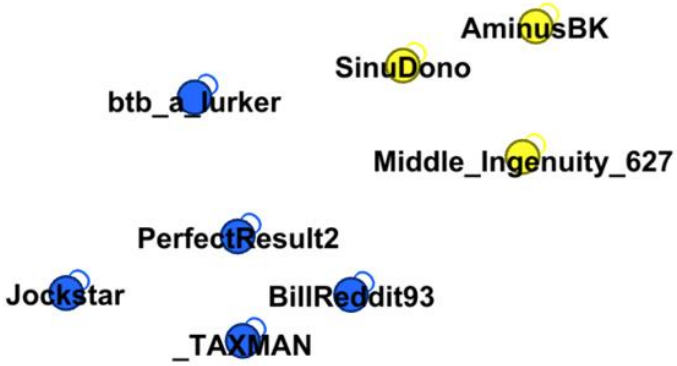


Figure 24: Users who reply to themselves