Athens University of Economics and Business
7th Semester project
Efthymia Kostaki
8170055
"Social Network Analysis on Twitch with Gephi"

February 2021

Table of Contents

Abstract

As a result of increasing time spent online, Twitch.tv has become a successful online-streaming platform. This study is a Social Network Analysis which aims to explore how Twitch.tv has evolved as a growing online community by analyzing relevant aspects which define a network. Especially over the past few months coronavirus has led to an increase in the time spent online and the importance of being part of an online community, therefore a study like this is more relevant in the context of understanding the characteristics of the users who stream content online (streamers) and how they connect with each other in terms of common followers. By collecting meaningful data and using network analysis tools like Gephi leads to crucial implications such as how communities in Twitch function in the midst of a global pandemic, the interests of the users of Twitch and the importance of these online communities.

The project and the resources described are available in a GitHub repository here.

Introduction

Twitch is the website in which millions of people gather every day to discuss, interact and have fun together. The genres which can be found are multiple: Gaming, Music, Talk shows, Sports, Travel, Casual Talking, Food and drinks and special events (Twitch.tv, n.d.).

Twitch.tv was launched in 2011 by Justin.tv which was at that time the Web's largest live video community. Twitch TV was characterized by its cofounders as "the largest competitive video gaming broadcast network in the world" (DiPietro, 2011). Twitch TV focuses on the e-sports community with the aim to support, entertain and inspire them. The website presents gaming competitions from all platforms and games with the greatest quality as well as the greatest gamers in the world competing in the most well-known tournaments. When the site was first launched, Twitch had close to 3.2 million exclusive visitors every month, 4.5 hours viewed per person per month and 45 million total video views per month. In 2012, the website had 20 million unique viewers in each month. In 2014 both of the companies Amazon and Google tried to acquire the site (Geeter, 2019). Eventually, Twitch was acquired by Amazon.com at the price of 970 million in cash, which was around 20% of the company's earnings that year (Soper, 2014).

Twitch today has largely evolved. The number of monthly users has increased to 140 million with 3.8 million streamers and 15 million unique daily users. Regarding Twitch demographics, about 65% of the users are male and 35% are female. Regarding viewership by country, most views come from the US and account for over 23%. In the first three months of 2020, Twitch had total amount of viewership 3.114 billion hours when the total hours which were streamed were 121.4 million. The most viewed game historically, from January 2015 to January 2019, was League of Legends with 29.32 billion views, followed by Fortnite with 16.24 million views. By comparing the channels and viewers heatmap, which the number of viewers or streamers which are active, it is possible to identify the best time to broadcast. In the heatmaps presented in the websites the best days to broadcast are Tuesday and Sunday which was found by comparing the proportion of viewers and the proportion of creators these days. Regarding revenue, Twitch's revenue in 2019 was 1.54 billion, which compares with their competitor, Youtube gaming, who had 1.46 billion revenue while on total it had less users (Iqbal, 2020).

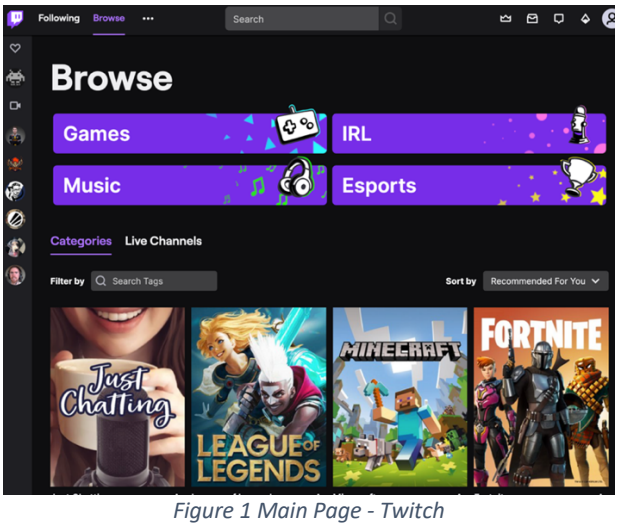


Figure 1 Main Page - Twitch

Coronavirus and Twitch

After getting a glimpse into the world of twitch a next step would be to understand how a global pandemic affected Twitch and how it shaped its online communities. All studies show that Twitch is experiencing a continuous growth during the pandemic. To be more specific there was a significant increase in the number of hours watched on Twitch during the beginning of coronavirus:

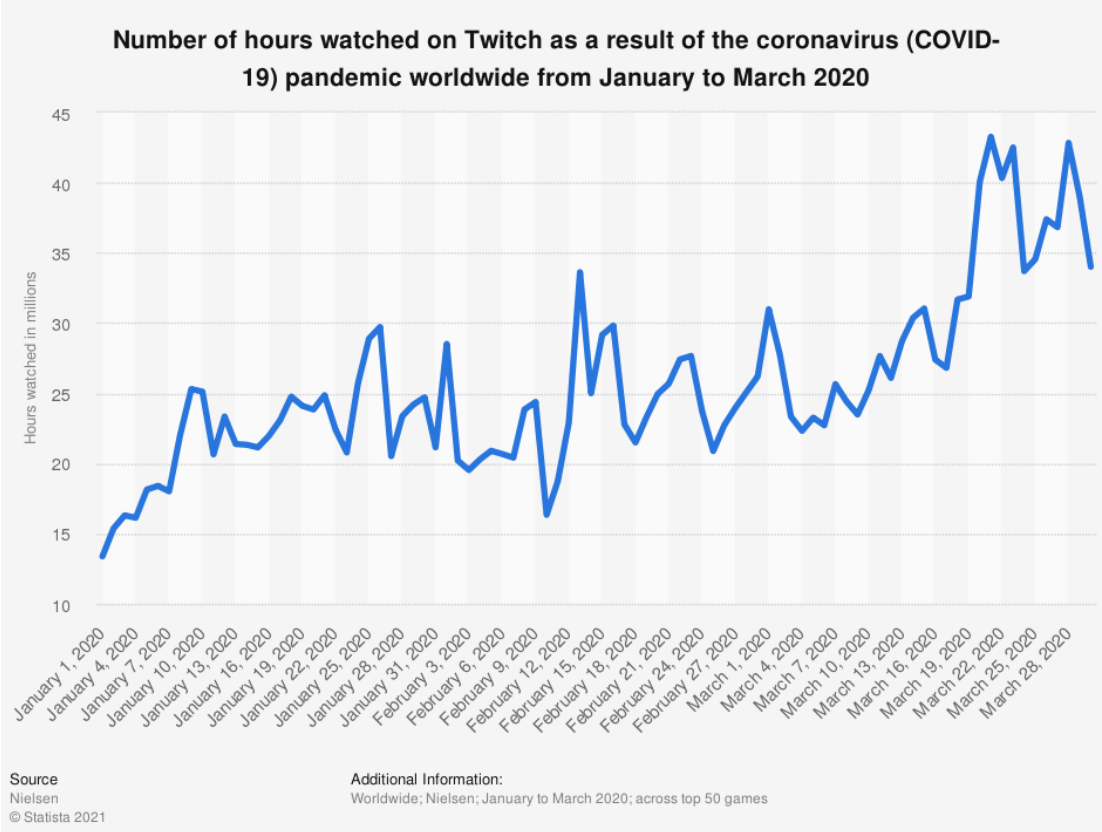


Figure 2 Increase in hours watched on Twitch during the beginning of the pandemic (https://www-statista-com.eaccess.ub.tum.de/statistics/1126326/covid-twitch-hours-watched/)

Moreover, this increase in the number of hours watched on Twitch is imprinted on the next Statista research in which apart from the total hours watched per month, it is clear that on the same time the number of active streamers and also the number of average viewers had an increasing tendency:

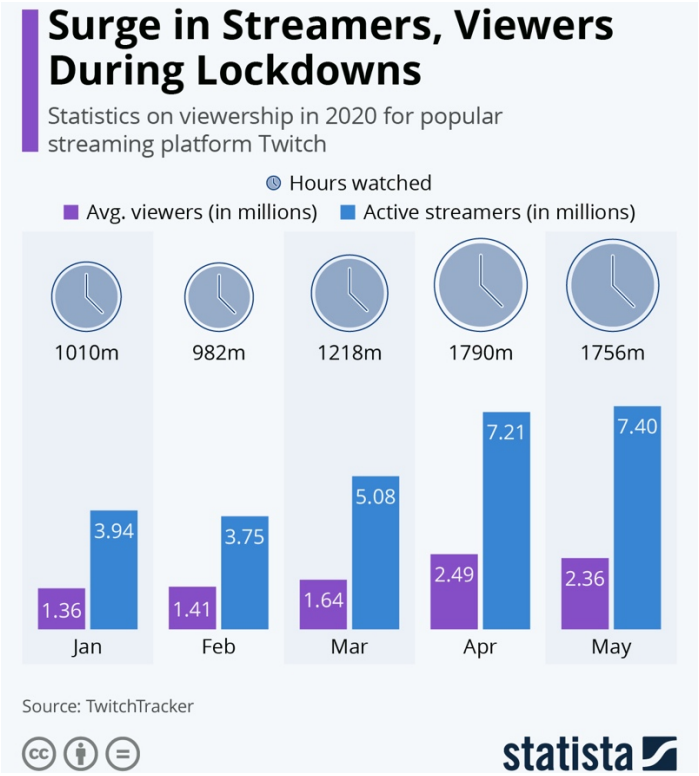


Figure 3 Increase in viewership on Twitch since the beginning of coronavirus. (https://www-statista-com.eaccess.ub.tum.de/chart/21965/twitch-streamers-and-viewers-during-covid/)

Of course, an increase in the amount of time spent watching was expected but with a closer look the increase between 2019 and 2020 was higher compared to the years before:

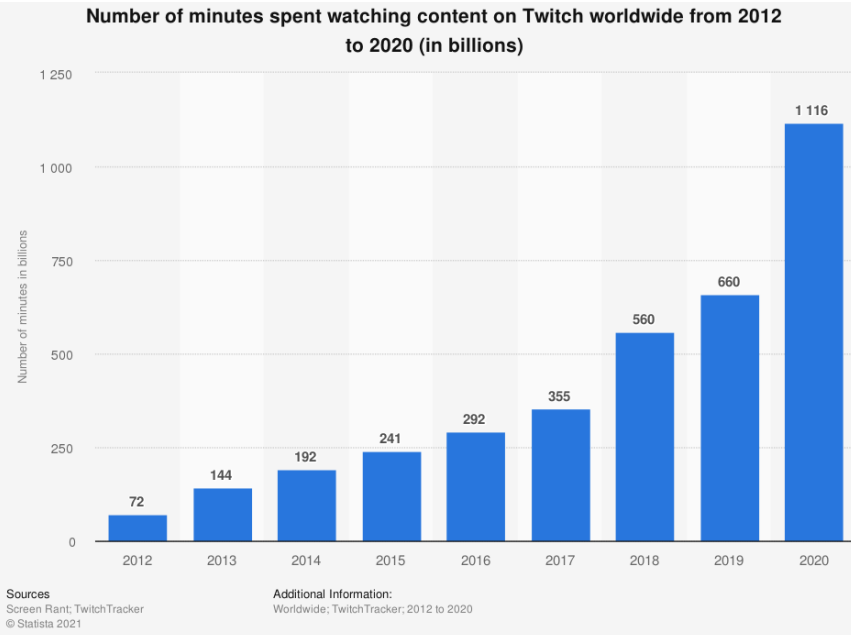


Figure 4 Minutes watched on Twitch on a per year basis. (https://www-statista-com.eaccess.ub.tum.de/statistics/819967/time-spent-watching-twitch/)

Literature Overview

Once the interest in the platform was established, the literature on the topic was researched and investigated so as to identify evidence for the academic interest of such research and how a Social Network analysis on Twitch could look like. In this section, two papers published in the literature are presented and discussed.

The first paper was presented in 2016 by IEEE international conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) by Churchill and Xu. They aim to analyze how different communities which exist online, such as streamers and users interact on Twitch, which is the structure in the social side of these relationships, which game franchises include common audiences. This study was conducted so as to better understand the communities developed by the different games. They identified three categories of players, the Casual Players, the Speed Runners and the Competitive Gamers. This distinction stems from the three communities identified by Twitch, the "Let's Play", "Speed running" and "E-sports" communities. On total they identify 460 streamers due to computational limitations. In the paper they perform two social network analyses. The first one identifies the franchises that attracted similar audience by finding the number of common followers between two streamers, which constitutes and undirected weighted graph. The findings from this research question were the following: some games are more centered than others which means that the streamers in these games have the majority of their followers in common. This leads to the conclusion that viewers who view these types of games tend to watch only relevant content and not explore and familiarize themselves with different type of content. The center concentration which mixes all game communities is obvious in the most popular streamers. However, this characteristic does not hold for less popular streamers who belong in entirely different game communities. This leads to the conclusion that users that immerse themselves with one game tend to know and follow less popular streamers which do not share common followers with streamers from other game categories. The second one aims to investigate how connected are the communities of gamers presented before as categories. However, in this question instead of creating the edge between two streamers based on the number of common followers, the edge was created if the streamer A follows streamer B etc. This constitutes an unweighted directed graph structure. Their findings from both research questions suggest that the largest subculture is the Casual Gamers, and the smallest is the Competitive Gamers. They interpret this result based on the difficulty of the way the games are played in each of these three categories. Therefore, it is obvious that the Competitive Gamers play in a more strategic and difficult way to understand which requires more focus from the audience compared to the Casual Players who play for fun and with the aim to entertain their audience (Churchill & Xu, 2016).

The second paper takes a more social and behavioral approach on Twitch by utilizing the Uses and Gratification Theory (UG) which underlines the reasons why users prefer to watch a certain type of streamer. The outcomes of a behavioral analysis could be useful for Twitch to suggest to the users better related content or identify streamers central to the network to whom they could provide financial incentives. Beginning to understand why users spend time on Twitch, Twitch is characterized as an online "Third Place" in which people with common interests engage with each other and form social interactions. However, this interaction

between streamers and users during a live session is reportedly impossible for streamers for more than 150 users. For that number, it becomes impossible to keep up with the comments and generally interacting with their audience. On total, 175 streams were selected whose edges were the number of common followers two streamers shared. The network showed an interactivity between smaller streamers compared to more popular streamers who were on the outskirt of the network. In the conclusion, he stresses the need for well-defined communities which could yield better analysis result. He suggests that Twitch could provide to streamers or to users a filtering system according to their needs. The limitations of the study include the fact that even though streamers are coded as belonging in a certain community they could actually be part of multiple communities at the same time as well as the game they are playing. Lastly, coding the genre the game belongs at is difficult to categorize since each game might have different elements such as adventure or strategic elements (Dux, 2018).

Concept Understanding and Leads for research

To better understand this concept and the analysis that could be conducted on Twitch, I decided to contact and interview the author of the second paper, James Dux. My questions were mainly focused on the data that he used and how I could fetch the necessary data to acquire my dataset but also on the social side of streaming. In the paper presented, he had suggested some further research which could be conducted on the social side of streaming and the definition of communities, but I was interested to hear his opinion now, after two years of the publish year of his paper that Twitch has changed and introduced new categories of streaming content such as “Music” and “Just chatting”. He suggested first that I could include simple characteristics such as the game the streamer played, the genre the game belongs at or the gender of the streamer. Regarding the social side of Twitch, he suggested that I consider considering how Twitch is both an interactive social tool and rich media platform. Unfortunately, Twitch introduced a new API in 2020 so the resources he provided me were mostly working on the previous one, but the social side of Twitch is very interesting and could be investigated further starting from this university assignment.

Identification of characteristics

Two relevant websites were identified for identifications of possible characteristics for further analysis. The first website can be used to identify if a streamer is a “Speed Gamer” or not and the second one to identify the genre and the theme of each game and also identify the mode of each game. Lastly, with the tag ids from each stream we can identify the type of stream tags they used in their stream.

Speed run provides leaderboards, resources, forums and more, for speed running (streamers who play as fast as they can a game). The streamers there can be filtered by game, platform or series. This site can be used to identify streamers in the network who are “Speed Gamers”.

Stream kick which allows to find streamers based on the filtering for “gaming” and “creative” as an initial separation. Regarding “gaming” the following filtering options are available:

1. Game/ Streamer:
    - Search by Game

    - Search by Streamer Name
    - Search by Stream title
2. Viewership:
    - Minimum Viewers
    - Maximum Viewers
3. Language
4. Platform:
    - Linux
    - PC (Microsoft Windows)
    - Mac
    - PlayStation 4
    - Xbox One
    - Nintendo Switch
    - PlayStation 3
    - Xbox 360
5. Genre:
    - Adventure
    - Arcade
    - Fighting
    - Hack and slash/ Beat em up
    - Indie
    - Music
    - Pinball
    - Platform
    - Point-and-click
    - Puzzle
    - Quiz/ Trivia
    - Racing
    - Real Time Strategy (RTS)
    - Role-playing (RPG)
    - Shooter
    - Simulator
    - Sport
    - Strategy
    - Tactical
    - Turn-based strategy (TBS)
6. Theme:
    - 4X (explore, expand, exploit, and exterminate)
    - Action
    - Business
    - Comedy
    - Drama
    - Educational
    - Erotic
    - Fantasy

- Historical
- Horror
- Kids
- Mystery
- Non-fiction
- Open world
- Party
- Sandbox
- Science fiction
- Stealth
- Survival
- Thriller
- Warfare

7. Game Rating:
   ESRB game ratings (ranking for parents to identify if it is suitable for their children for example):
   - T
   - M
   - AO
   - RP
   - EC
   - E
   - E10

   PEGI game ratings (system classification of games according to ages):
   - Three
   - Seven
   - Twelve
   - Sixteen
   - Eighteen
8. Mode:
   - Co-operative
   - Massively Multiplayer Online (MMO)
   - Multiplayer
   - Single Player
   - Split Screen
9. Stream Tags:
   - Affiliate
   - Background Music
   - Charity
   - FaceCam
   - Fundraising

Limitations

Apart from the important analysis and findings of this study there are of course some limitations to this study which should be pointed out. The most important limitation is that this study was based on limited and random data. Even though the randomness helps to acquire valuable data due to follower constraints the data for each streamer were based on their ranking of live followers at the time of fetching of the data. It was chosen to fetch data for 200 random streamers in Twitch who were in 1000th position to 1500th position in the ranking of live followers. The reasoning is simple; a streamer who has many live followers usually has many followers as well. As the code snippet was run in a 16GB server over a period of 24 hours to fetch the results used for this study, it was impossible to handle streamers with millions of followers and even hundreds of thousands. This random ranking selection guaranteed to yield results in the time window specified (one day).

Methodology

Nodes

Each streamer is represented by a node in the network. Each node has certain attributes to help group them. The dataset consists of 61 streamers randomly selected from the streamers who were live at the time the data was fetched. It is ensured that the language of streaming is English and that these streamers are not along the streamers with the highest number of viewers which consequently the streamers who have the highest number of followers. For each streamer it has been identified: their name, the game they were playing and the tags they included in their live streaming. The gender of the streamer wasn't included since some profiles were automated streaming of gaming and it wasn't possible to identify each streamer's gender.

Edges

An edge between two nodes represents the number of common followers the two streamers share. If two streamers have only one common follower, their relation in their followers is not so clear, and these edges are removed from the network.

Programs/ Resources

- Twitch.tv: application in which users stream games they are playing,
- Gephi: network visualization tool,
- Python: programming language used to fetch data and manipulate them into the desired form,
- python-twitch-client: library which provides functionalities for
- speedrun.com,
- streamkick.com,
- dev.twitch.tv,
- twitchapps.com.

Dataset Generation

Twitch App Registration

The first step in the process was to create a Twitch account and authorize this account to also become a Twitch developer account by joining this website. Then, I had to register my application in this website by adding a name, an Organization Authentication redirect URL (OAuth redirect URL) and the category of the application for which I chose "Analytics Tool". Then the system generates a unique Client ID and a Client secret token. The app after it has been created it looks like this:

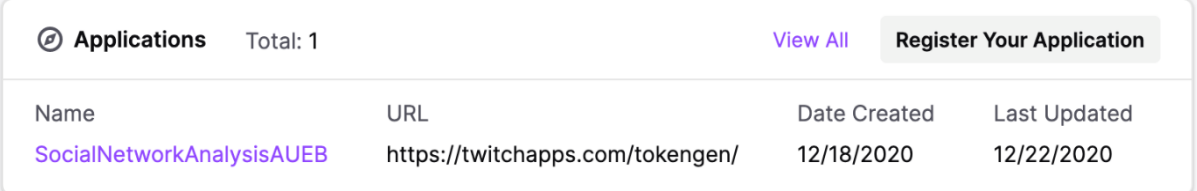


Figure 5 Twitch Developer Dashboard

Identification of python APIs

The next step was to identify a python library with which I could fetch all the relevant data from Twitch and perform my analysis. I found two:

1. Python Twitch Client : an easy-to-use library for accessing the Twitch API which supports the new API introduced in 2020, the new Helix API.
2. TwitchIO: an Async Bot/API wrapper for Twitch in Python.

In my implementation I used the first one, which fetched faster all the results I needed but I also created a bot in the second one, which fetches comments from live streams and is presented at the end of this project as an extra implementation, which could be useful for a social network analysis on the content of the comments for each streamer and how similar it is with one another, or other findings such as important keywords used, which could be presented in a network.

In order to use the Python Twitch Client API, I had to generate some tokens first which were the client ID available in my application and OAuth token which could be generated in the token generator website. In that I needed to insert a Client ID, insert this redirect URL: https://twitchapps.com/tokengen/ in the redirect URLs of my application in the Twitch Developer Console and define the scopes of my connection. The available scopes are available in the authentication website linked here, out of which I could specify any scope name I wanted based on my needs but since I wasn't sure I left the gap blank. With these data, I was ready to start developing my network.

Fetching Twitch Streamers

The following code was used to fetch certain streamers with specific characteristics from Twitch. Firstly, I needed to import the numpy library for the random number generation and the helix API which is the new API from Twitch implemented in Python. In line 4 I create a list of 200 random streamers in the range 1000 -1500. This happens because the way I decided

to fetch the streamers was by getting the live viewer list from Twitch which is returned according to the number of live viewers watching a specific stream in decreasing order. Due to computational complexity the first 1000 streamers who had many followers (hundred to thousands to even millions) it was impossible for the server I used to run my calculations to run them in a timely manner, so I decided to fetch less streamers with less viewers in a random way. That's what line 7 achieves. In the loop in lines 13-20, I iterate through the streamers list returned from the server and for every streamer who is included in the random list of numbers I fetched, before adding the streamer in my final streamer list, I check if the game ID is not blank and if the language of the streaming in English. It turns out that out of the 200 initial nodes I selected only, 61 were returned from this code, which made sense since 23% of the streams are from the US and it is logical to expect about the same percentage of streamers randomly selected to stream in English. In this random case, the percentage of streamers from the initial ones was around 30%.

```
1  import numpy as np
2  from twitch.helix.api import TwitchHelix
3
4  random_ids = np.random.randint(1000, 1500, 200)
5  client = TwitchHelix(client_id='g39od75kqbxwu2lf988t52ttfqd96p', oauth_token='iuqqudf4nxvh07nqbbzsekutd5erh9')
6
7  streams = client.get_streams() ### returns according to viewer count let's say I keed the first 100
8  streams_with_most_live_viewers = list()
9
10 # Users with many live viewers will be disregarded since they require enormous computational power
11 sum=0
12
13 for i in streams:
14     if sum in random_ids:
15         if i['language'] == 'en' and i['game_id'] != '':
16             print(i)
17             streams_with_most_live_viewers.append(i)
18     sum += 1
19     if sum > 1500:
20         break
```

Figure 6 TwitchHelix.py

Parsing of the streamers

Disregarding the initial imports necessary for accessing the Twitch API and saving the results in a CSV file, this is the code file used to create the dataset. Before moving to the main part, the necessary functions which were implemented will be explained. The function user_follows(user_id), returns the users who follow a specific streamer. Instead of returning the dictionary of these relationships for each follower, instead returns an array with the follower ids of all the followers of a user to reduce the computational effort. The function list_to_csv(streamer_channels_common), writes in a csv file the streamers who have common followers (namely the Source and Target columns) and the number of common followers (the Weight of the network). The last function presented, find_common_followers(streamerA, streamerB), returns the number of common followers between two streamers by translating the lists to sets which makes it computationally easier to calculate. For implementing the last two functions, I found useful the article "How to Find Your Mutual Connections on Medium" (Chugh, 2020).

```
6   def user_follows(user_id):
7       followers = list()
8       client = TwitchHelix(client_id='g39od75kqbxwu2lf988t52ttfqd96p', oauth_token='iuqqudf4nxvh07nqbbzsekutd5erh9')
9       user_follows_iterator = client.get_user_follows(to_id=user_id)
10      print("Total: {}".format(user_follows_iterator.total))
11      for user_follow in islice(user_follows_iterator, 0, user_follows_iterator.total):
12          followers.append(user_follow['from_id'])
13      return followers, user_follows_iterator.total
14
15
16  def list_to_csv(streamer_channels_common):
17      with open('edges_data.csv', 'a') as file:
18          writer = csv.writer(file)
19          writer.writerow(('Source', 'Target', 'Weight'))
20
21          for channel in streamer_channels_common:
22              writer.writerow([channel['Source'], channel['Target'], channel['Weight']])
23
24
25  def find_common_followers(streamerA, streamerB):
26      a_set = set(streamerA)
27      b_set = set(streamerB)
28      common = []
29      if a_set & b_set:
30          print(a_set & b_set)
31          common = a_set & b_set
32      else:
33          print("No common elements")
34      print(len(common))
35      return len(common)
36
37  streamers = [{...},
```

Figure 7 Main File 1

Moving on the main part of the code, after having fetched the streamers, for every streamer in line 401, the code returns the list of the streamer's followers and the number of total followers. Then, I created a list with entries for every streamer the list of the streamer's followers. Also, I added a new key-value pair in the dictionary of each streamer, storing the number of total followers. In lines 407-412, the number of common followers between all pairs of streamers is calculated by ensuring at the same time, in line 409, that the same pair of streamers is not added twice. In lines 410-412, for each pair, a dictionary is created which stores the "Source" streamer, the "Target" streamer and the "Weight" as the number of common followers they share. Lastly, in line 413 this list of common followers is written in a csv file.

```
streamers_followers = list()

for idx, val in enumerate(streamers):
    follow, total = user_follows(val['user_id'])
    streamers_followers.append(follow)
    streamers[idx]['total_number'] = total

common_followers = list()

for idx, valx in enumerate(streamers_followers):
    for idy, valy in enumerate(streamers_followers):
        if idx < idy:  # ensure that we don't compare the same streamers twice
            common_followers.append({'Source': streamers[idx]['user_id'],
                                     'Target': streamers[idy]['user_id'],
                                     'Weight': find_common_followers(valx, valy)})
list_to_csv(common_followers)
```

Figure 8 Main File 2

Gephi

The csv file created with the code presented in the previous section was imported in Gephi by importing it as an edges table. However, the characteristics of the streamers were not imported with this way. Therefore, manually, the name of the streamer was added in the "Label" column, the game id in the "game_id" column, the game name in the "game_name" column, a column "isGame" if the streamer is playing a game or not and the tags in the column with String set "tag_ids". All edges with weight of one were removed since only one node is not enough to justify a clear relationship between two nodes. This approach was also used in the literature overview to reduce the number of non-important edges.

Social Network Analysis

Graphical representation of the network

An initial graphical representation of the network was created by partitioning the modularity class in which the size of each node depends on their degree. When we explore the degree measures, we will explain which is the interpretation of this network. The layout used was "Fruchterman Reingold":

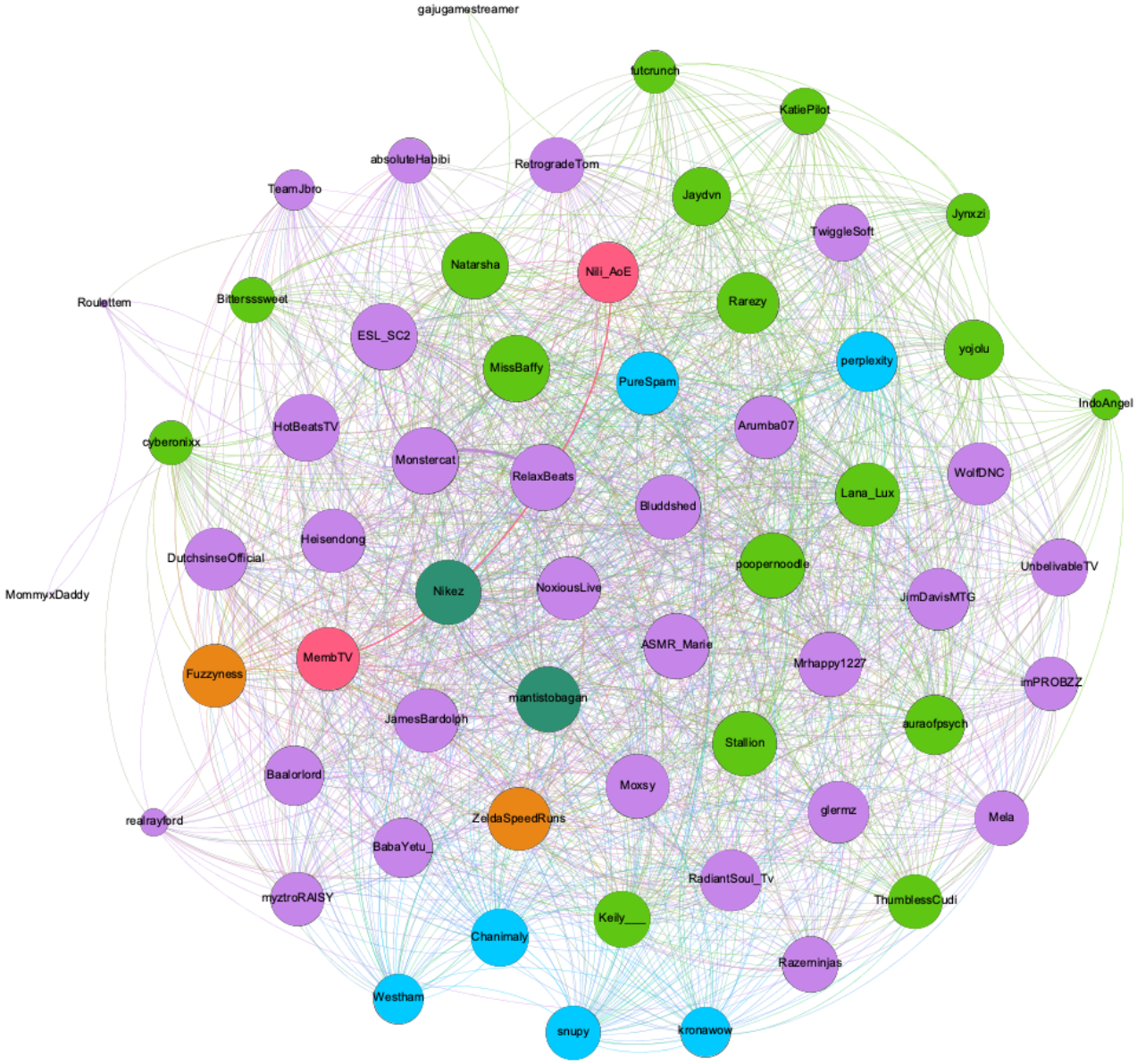


Figure 9 Visualization partitioned by modularity class

Basic topological properties

The network type in undirected, since the link between two nodes is the common number of followers they share, and this relationship cannot be expressed with an exclusive direction. The number of Nodes in the Network is 61 and the number of Edges in the Network is 1476.

The network diameter is the longest shortest path in the network (Newman, 2018). In this network, the network diameter is 3. Since the network diameter in the network is only 3, this means that the random network generated with the methodologies presented in the dataset generation section is quite tight. This means that, the streamers selected have probably some universal characteristics that introduce similarity in the network, hence it's very easy to reach any node in the network.

The average path length is the average shortest path in the network (Newman, 2018). In this case it's equal to 1.194535519125683. This number is very low which was expected since the diameter is small. These indicators suggest that Twitch is a tight knit community in which users engage with many different streams of different genres. This observation will become clearer with the following deep-level analysis.

Component Measures

Components are well-connected regions of nodes in the network. When a network component's size grows to N, which is the number of nodes, the component is instead called a giant component (Newman, 2018). In this network the number of Weakly Connected Components is 1 and the number of connected components is also 1.

As expected, the algorithm yielded one giant component in the network. This giant component occupies significant fraction of the network while there seem to be no smaller components in the network.

To better visualize this giant component and its existence the topological filter "Giant component" was applied. Here's the network:

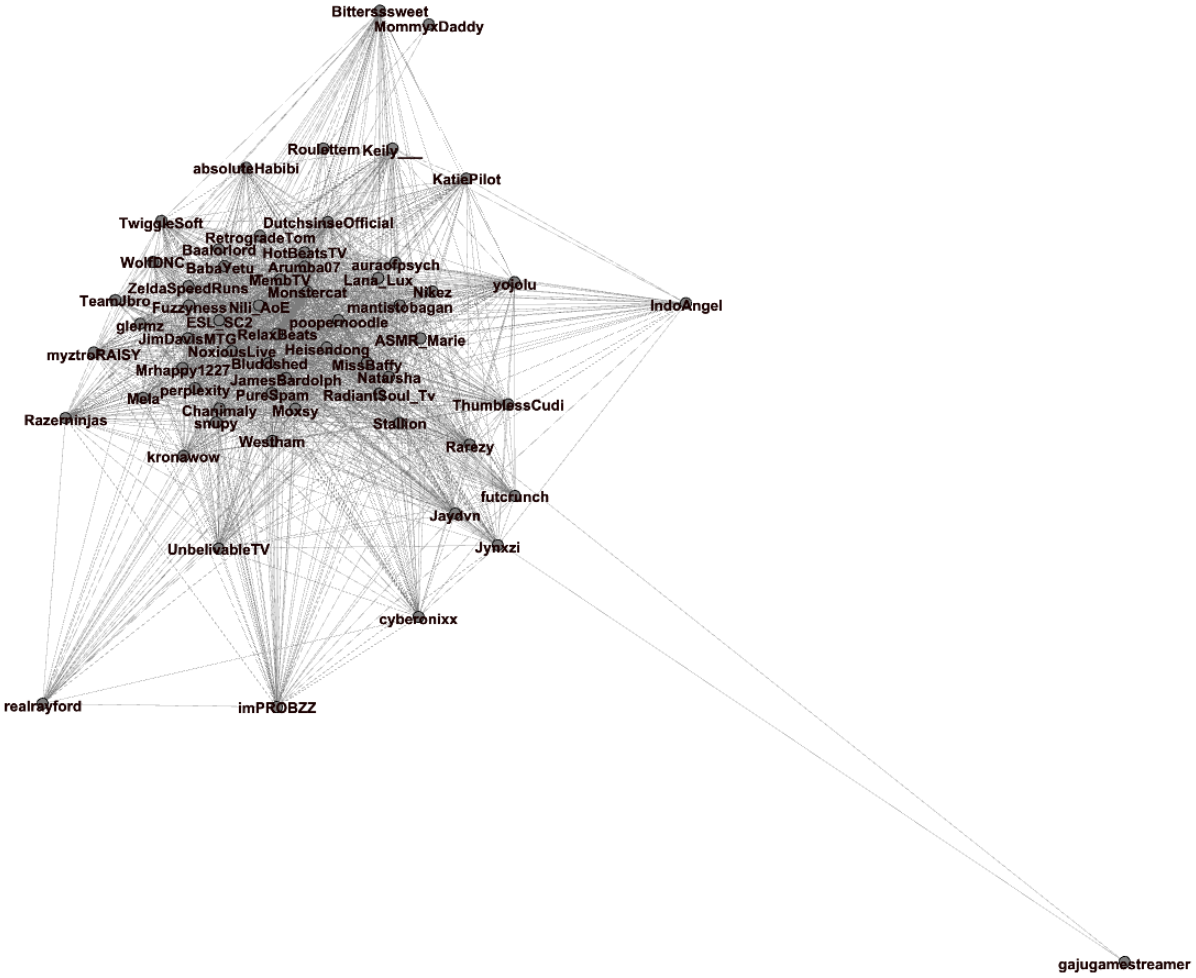


Figure 10 Giant Component

Component Size Distribution:

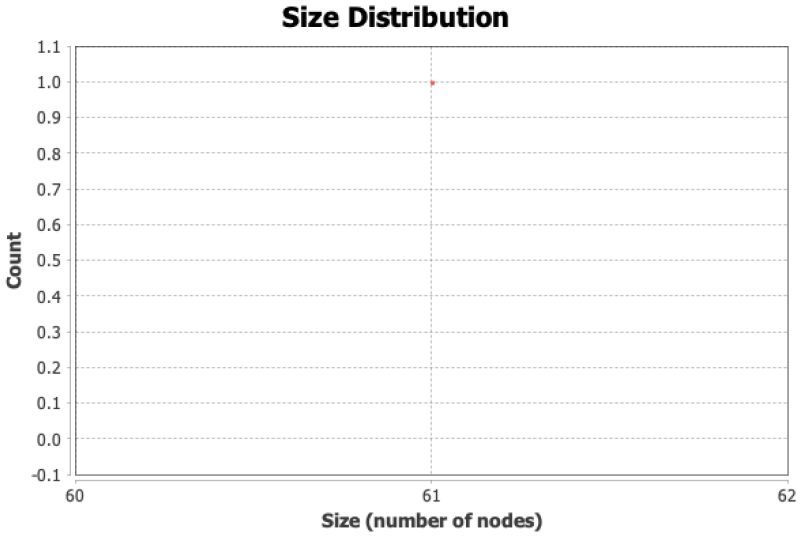


Figure 11 Component Size Distribution

The component size distribution clearly visualizes this giant component. The number of nodes is 61 and the count for this number is 1, meaning that all nodes are part of this network.

Degree Measures

The degree in an undirected network denotes the total number of links of one node to other nodes (Newman, 2018). Since the graph is weighted in some cases, it will be more meaningful to look at the weighted average degree but in this project for reasons of completeness, we will present the average degree as well.

The average degree in an undirected network is the average number of total links one node has with other nodes (Newman, 2018). The average degree is useful for the average number of streamers with whom one streamer shares common followers with. The average degree is 48.393 with a maximum node degree of 59 and a minimum node degree of 2. This average degree is quite high which is also visible in the degree distribution presented below. In this distribution all nodes tend to have higher than 35 streamers with whom they share common followers with. The distribution resembles an increasing function, in which when the node degree increases the count increases as well. Even though this network was created with a random process this increasing connection seems to hold, which would suggest if we had more nodes in the network, they would be more connected with each other. This measure shows the tightness of the network as well of Twitch's streamers. It is clear that this degree distribution provides the probability that a randomly selected node in the network has degree k if we divide the count in the y-axis by the total number of nodes in the network.

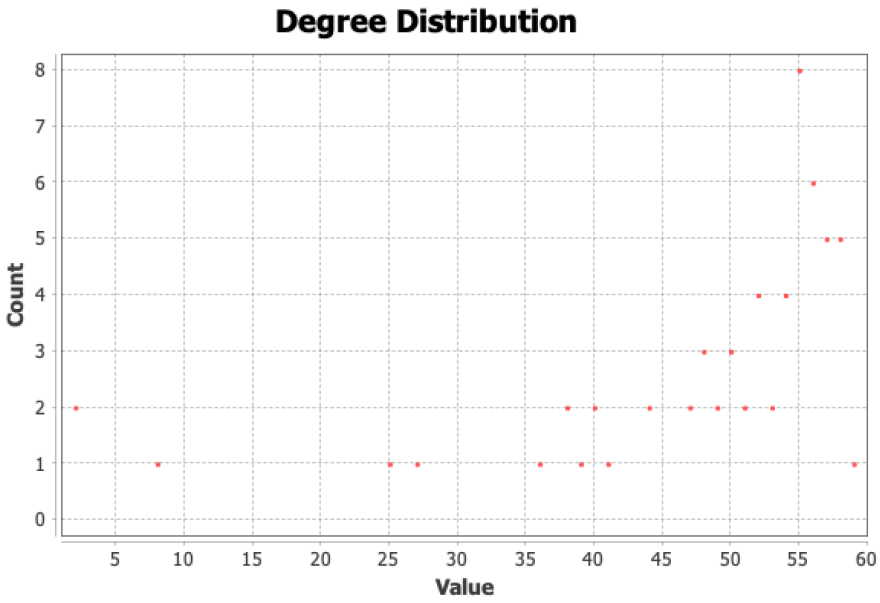


Figure 12 Degree Distribution

To better understand which is the situation in the network we can create the visualization of the network in which the size of the nodes is determined by the degree of the node and the colors of the nodes are based on their modularity class using the layout "Fruchterman Reingold":

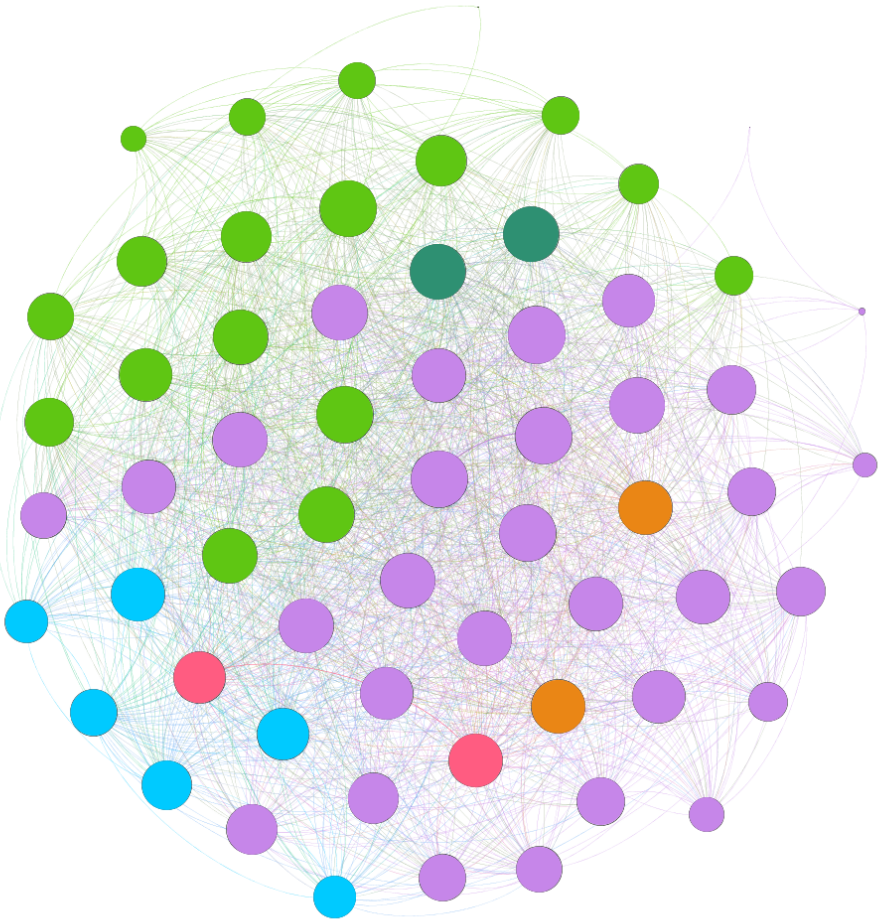


Figure 13 Network size: degree, color: modularity class

This visualization shows that most of the nodes in the network have similar degree, meaning that the number of streamers they share common followers with is on average almost the same. This was expected since the lower bound for a streamer to be connected with another streamer is to have at least 2 common followers. Therefore, it might be better to look at the weighted degree.

The streamers who had the highest degree were:

Streamer name	Degree
HotBeatsTv	59
Monstercat	58
ESL_SC2	58
MissBaffy	58
RelaxBeats	58
Natarsha	58

The average weighted degree is the average of sum of weights of the edges of nodes (Newman, 2018). In this network it is 15,855.213 common followers on average. The maximum weighted degree is 172,513 and the minimum weighted degree is 5. It is obvious that there are some outliers in the network who increase the average since the maximum weighted degree is 172,513. Most nodes seem to have less than 20,000 common followers.

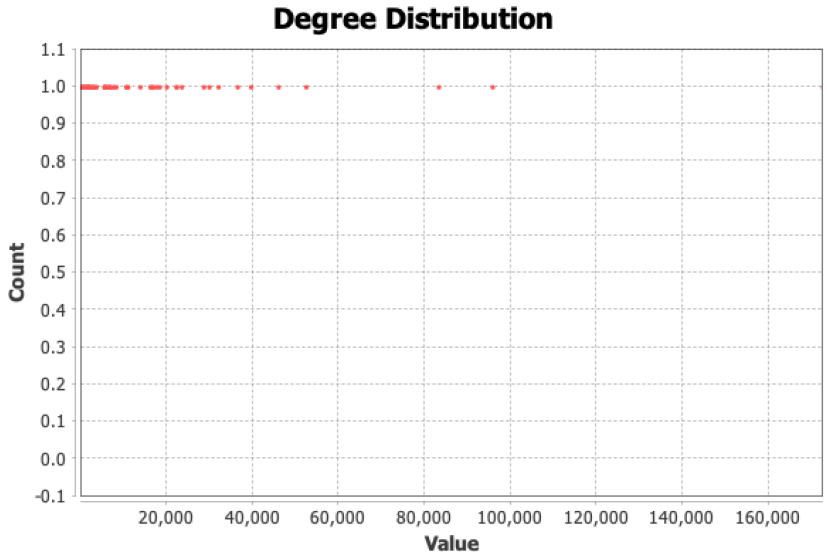


Figure 14 Weighted Degree Distribution

To better understand which is the situation in the network we can create the visualization of the network in which the size of the nodes is determined by the weighted degree of the node and the colors of the nodes are based on their modularity class using the layout "Fruchterman Reingold":

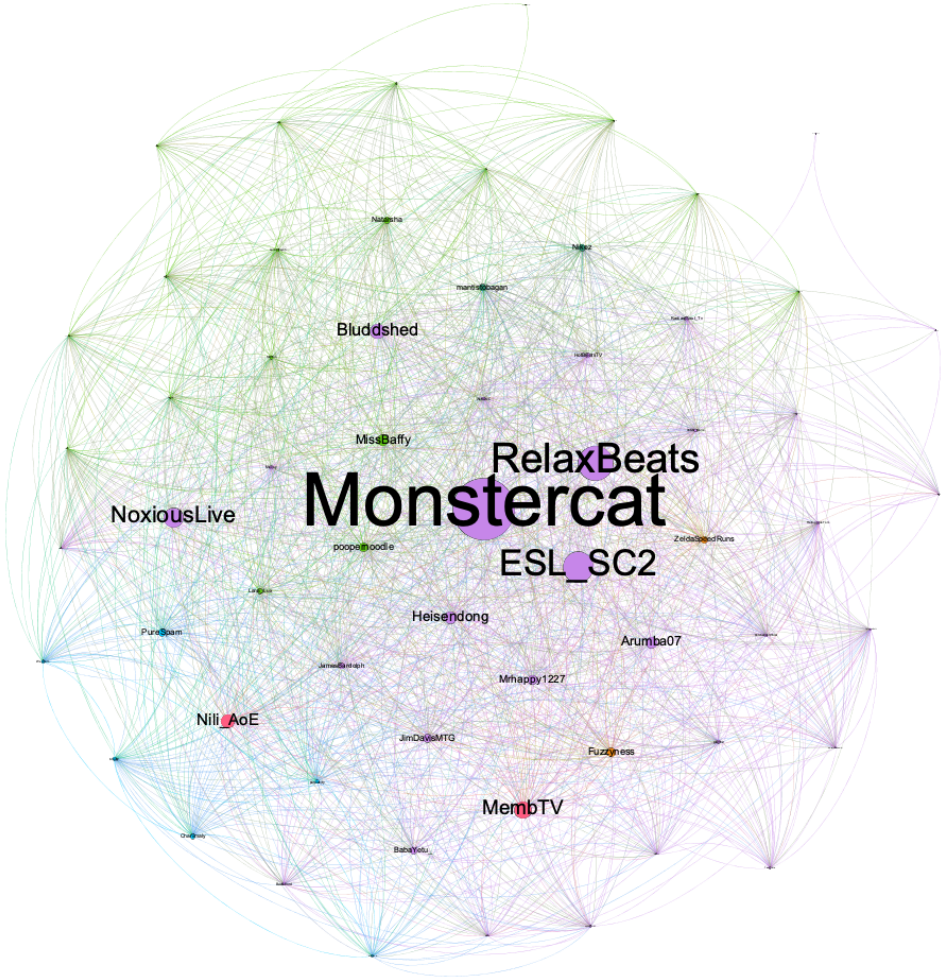


Figure 15 Network size: weighted degree, colour: modularity class

This map shows that the streamer with the highest number of common followers on average is Monstercat. The second streamer with most followers is RelaxBeats. It is important to note out that both Monstercat and RelaxBeats share content about Music and not gaming. Probably these users have enormous crowd compared to other streamers who play specific games but also have common followers with many nodes in the network hence their central position in the network. We will explore some specifics about the network in the section "Network specific analysis".

The streamers who had the highest weighted degree were:

Streamer name	Weighted Degree
Monstercat	172513
RelaxBeats	95648
ESL_SC2	83143
NoxiousLive	52297
MembTV	45833
Bluddshed	39443

Centrality Measures

Betweenness Centrality

In Network theory, betweenness centrality counts how many times a node is a bridge in the shortest path between any two nodes. As the score increases, so does the power and control of the node in the network (Newman, 2018). In the network examined, we can interpret this measure as the streamers with the highest betweenness centrality are the ones who can increase the followers of other streamers. For example, if a user watches the content from one streamer, probably Twitch could offer suggestions to the user for other streamers with which the main streamer is part of the shortest path to reach them. This is useful for the streamers as well, if they want to increase their audience in a specific side of the network, for example for a specific game, and they need to approach streamer A to do so, they might need to collaborate with streamer B first who has high betweenness centrality and then it will be easier for them to connect with Streamer A in order to reach this specific audience.

In the following plot we can see the betweenness centrality distribution of the network:

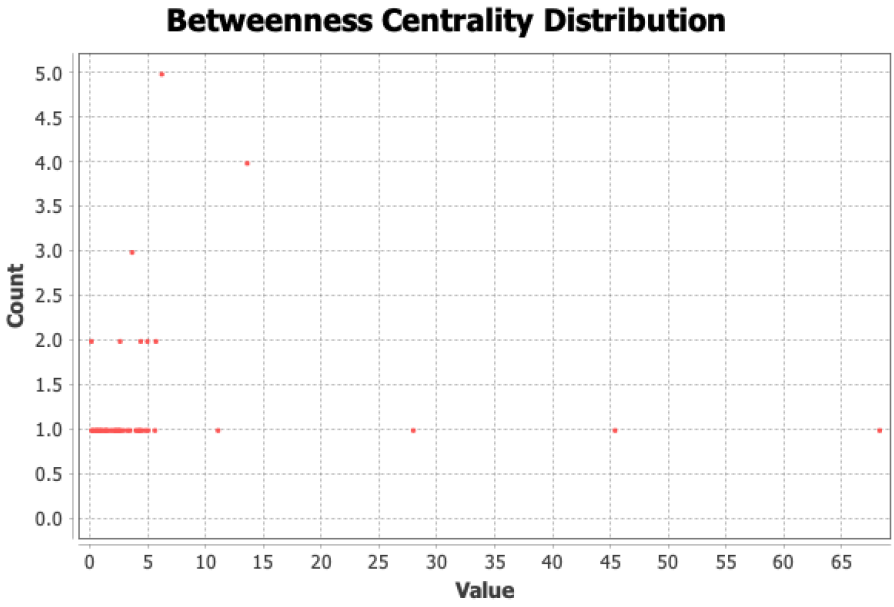


Figure 16 Betweenness Centrality Distribution

Visualizing the network when the size of the nodes depends on their betweenness centrality:

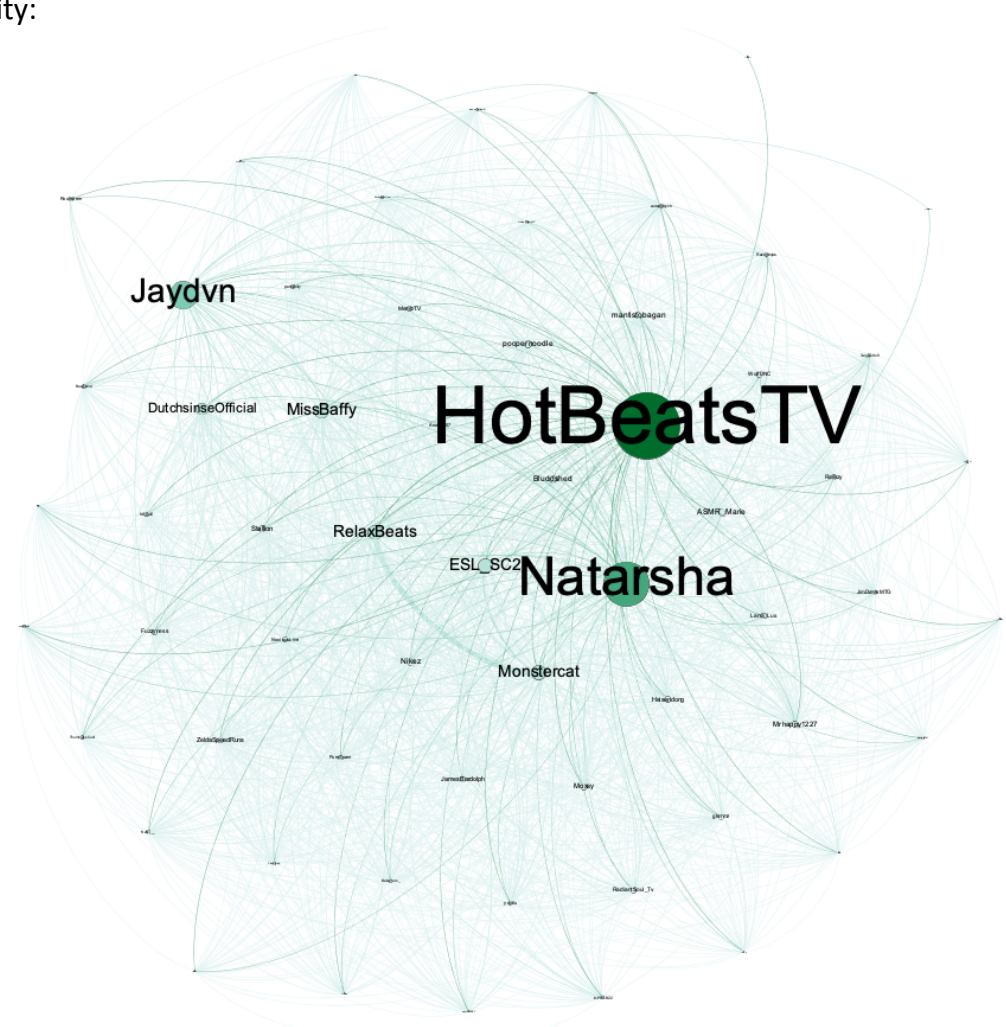


Figure 17 Network ranked/ coloured by betweenness centrality

This graph helped us identify some interesting trends. While Monstercat had the highest weighted degree in the betweenness centrality this is not the case. Here, we see that different streamers have more control in the network. These are HotBeatsTV and Natarsha. HotBeatsTV shares music and not gaming and also Natarsha's streaming belongs in the category "Just Chatting". This leads to the conclusion that streamers who share more relaxing content compared to streamers who play games are more powerful in the network, both in terms of betweenness centrality and weighted degree.

The streamers who had the highest betweenness centrality were:

Streamer name	Betweenness centrality
HotBeatsTV	68.130696
Natarsha	45.267028
Jaydvn	27.823369
Monstercat	13.464029
ESL_SC2	13.464029
MissBaffy	13.464029

Harmonic Closeness Centrality

Closeness centrality is the inverse of the mean shortest distance from node i to any other node in the network. If the node is very central to the network, then it has high closeness centrality. If the network wasn't connected in a giant component, then we would investigate the network's harmonic centrality (Newman, 2018).

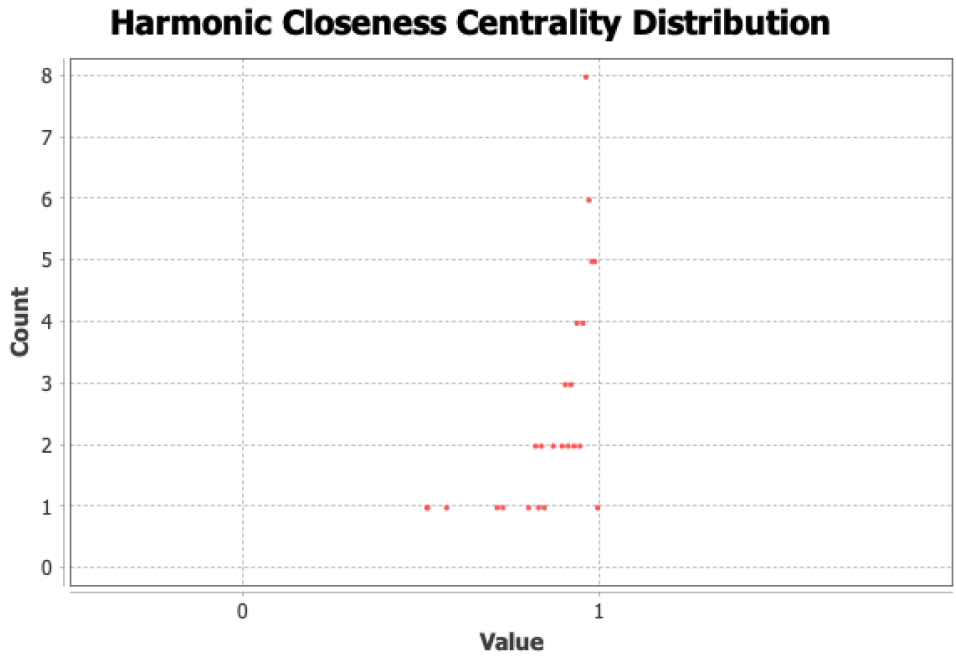


Figure 18 Harmonic Closeness Centrality Distribution

If we visualize the network in which the size of the nodes depends on their closeness centrality, we observe that the majority of the nodes has high closeness centrality. This was expected since the length of the shortest paths is very low.

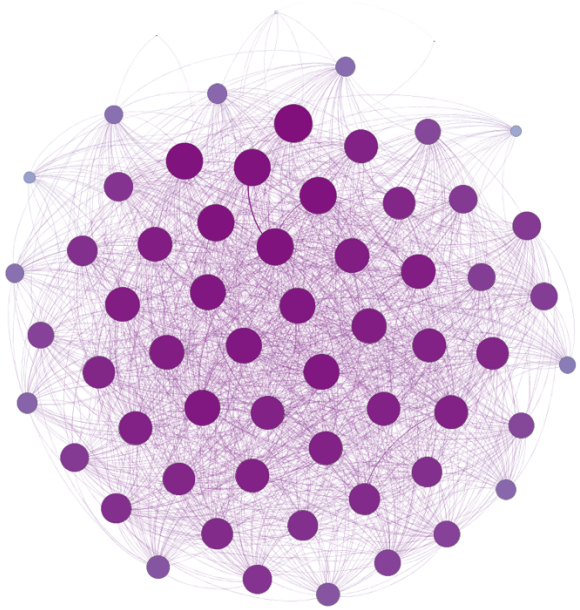


Figure 19 Network colour/ size: closeness centrality

The streamers with the highest closeness centrality were:

Streamer name	Closeness centrality
HotBeatsTV	0.983607
Natarsha	0.967742
Monstercat	0.967742
ESL_SC2	0.967742
MissBaffy	0.967742
RelaxBeats	0.967742

Eigenvector Centrality

Eigenvector centrality measures the influence that some nodes have in the network to other nodes. The eigenvector centrality of a node i is defined to be proportional to the sum of the centralities of the neighbors of node i, meaning that it takes into consideration the node's neighborhood (Newman, 2018).

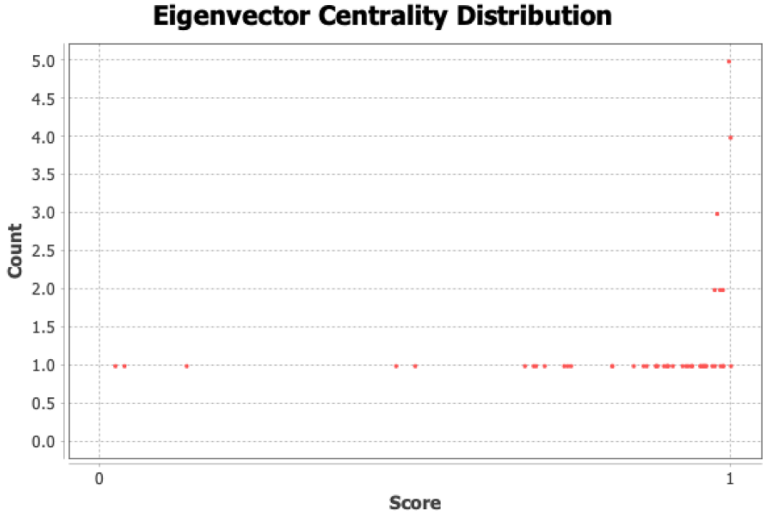


Figure 20 Eigenvector Centrality Distribution

Visualizing the network once again, this time the size of the nodes depending on their eigenvector centrality the situation is more or less the same with before; most of the nodes have an eigenvector centrality close to 1:

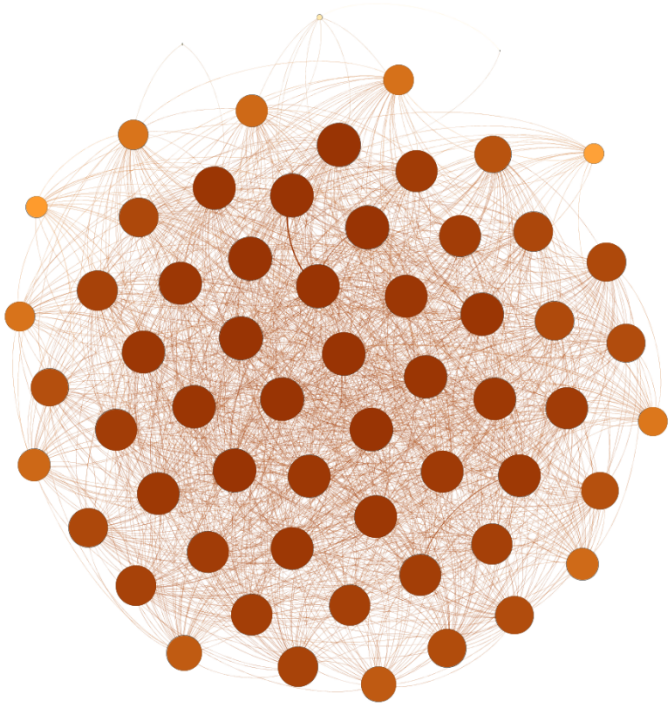


Figure 21 Network color/ size: eigenvector centrality

The streamers who had the highest eigenvector centrality were:

Streamer name	Eigenvector centrality
HotBeatsTV	1
Monstercat	0.999392
ESL_SC2	0.999392
MissBaffy	0.999392
RelaxBeats	0.999392
Bluddshed	0.999392

Clustering effects in the network

Triangles and Triadic Closure

A triangle means three nodes, each connected by edges to both of the others. Triadic closure refers to the phenomenon when there exists an “open” triad of nodes. A triad occurs when one node is linked to the other two, but the third possible edge is missing. This “open” triad can become “closed” if the last edge is added, forming a triangle (Newman, 2018).

In order to identify clustering effects in the network the plug in “Triangle Clustering Coefficient” was used. With this plugin the value for the average clustering coefficient was 0.9107 while with the predefined method for average clustering coefficient the value was 0.920, therefore the changes are neglectable. The total number of triangles which was found was 22682. For the triadic closure with the plugin “Triangle Clustering coefficient” we identified the number of paths with length 2 which were 74713 paths. This proves once again that the network is very connected. This means that it’s very probable if we take a look at the same network again after a short amount of time (for example some months) that the streamers which weren’t connected to be connected with at least two common followers. If we raise the bar above the lower bound of two followers, the connection between two streamers with this condition might take more time or never even happen.

Clustering Coefficient

The distribution of the clustering coefficient is presented below:

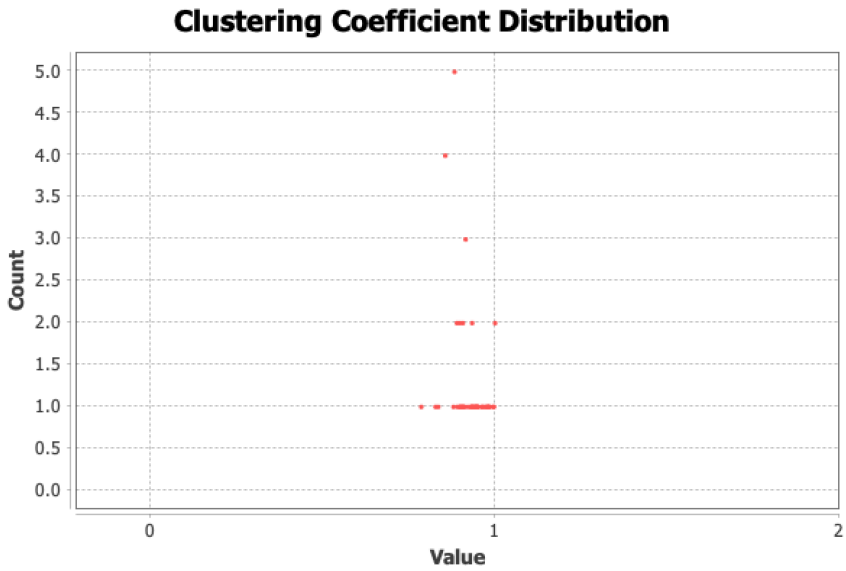


Figure 22 Clustering Coefficient Distribution

Using the layout “Yifan Hu Proportional” and for adjusting the nodes the layouts “Neverlap” and “Label adjust” we can see that the communities identified partitioned with their Clustering coefficient were 9. One category out of them was very general and just accounts for the nodes which didn’t belong to anywhere else. The visualization is presented below:

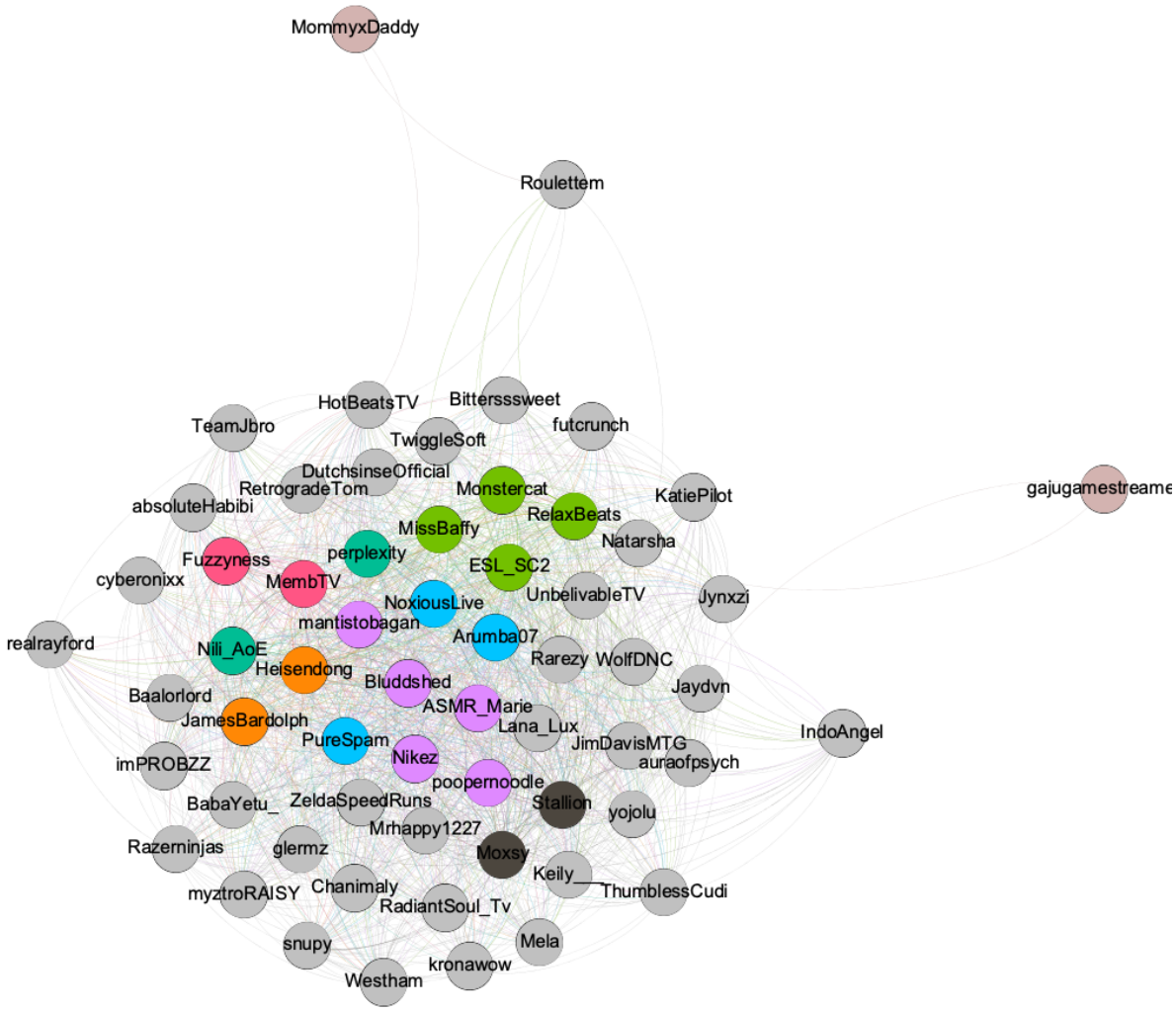


Figure 23 Visualization partitioned with Clustering Coefficient

We can now assess this clustering and identify if the algorithm categorized based on the game these streamers were playing or the other content they were sharing. We will work with the streamers in the first 4 of these clusters apart from the grey one to get an idea how well the clustering preformed:

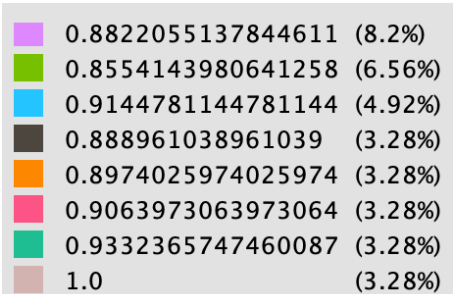


Figure 24 Clustering Coefficient Classes

Cluster 1: Two of the streamers were playing “Grand theft auto V”, one “Diablo immortal”, one “My time at Portia” and the last one was practicing “ASMR”. Overall, it seems like the clustering was based on action games.

Cluster 2: Two of the streamers in this cluster were sharing music, one “Starcraft” which is a strategy game and one League of Legends. These games can be team played and an idea is that this clustering focused more on the social activities and on the social side of gaming/ team player games.

Cluster 3: “Mini Healer” and “Old School RuneScape” which have been described as role playing games and “Hearthstone” which is a strategy game. Another categorization of these games can be done by looking further into their contents. “Mini Healer” also belongs in the category “Casual Gaming” and “HearthStone” is a virtual card game so it can be considered casual. In any case, a cluster identity is not clear in this cluster.

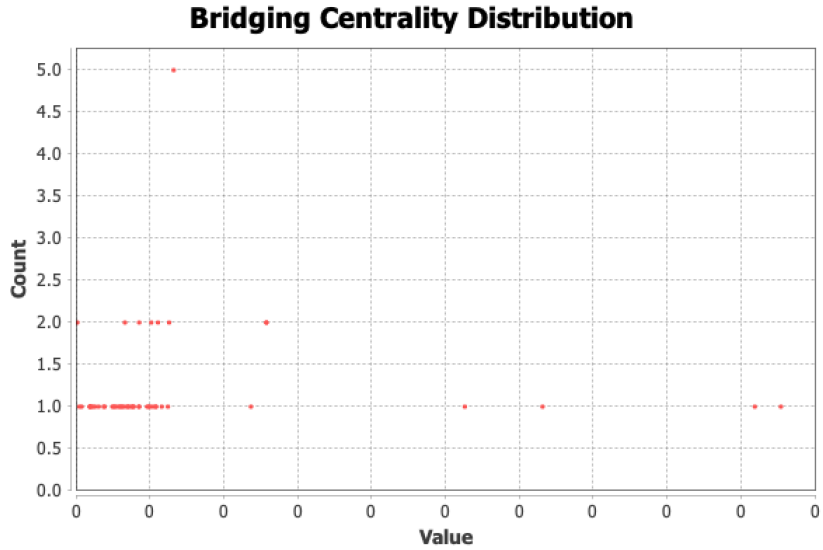
Cluster 4: One streamer was streaming “Call of Duty: Black Ops Cold War” and another one “Borderland 3”. This is a very good clustering since both these games belong in the category “First-person shooter video game”.

It turns out as we saw that this clustering worked very well on our nodes. However, the clusters are either very limited or very large which does not lead to equal size clusters on average and the clusters are not balanced.

Bridges and local bridges

Bridges are strongly connected with betweenness centrality we examined in previous section. A bridge unites a group with a different group with an edge which connects the two nodes. A group is defined as some nodes which are heavily connected with one another and one of them, for example node 1, is connected with node 2, which is part of another strongly connected group of nodes (Newman, 2018). In order to identify the bridges, the Plugin "Bridging centrality" was used with the setting normalized [0,1] since the network is undirected.

The distribution of bridging centrality is presented below but it is clear that the values of the bridging centrality are very low and close to 0:



The values of the bridging centrality lie between [0, 0.0004). The nodes with the highest bridging centrality were:

Streamer name	Bridging centrality
Roulettem	0.00037..
HotBeatsTV	0.00036..
Natarsha	0.00025..
Jadyvn	0.00020..
MissBaffy	0.00010..
RelaxBeats	0.00010..

The visualization of the network based on its bridging centrality is presented below:

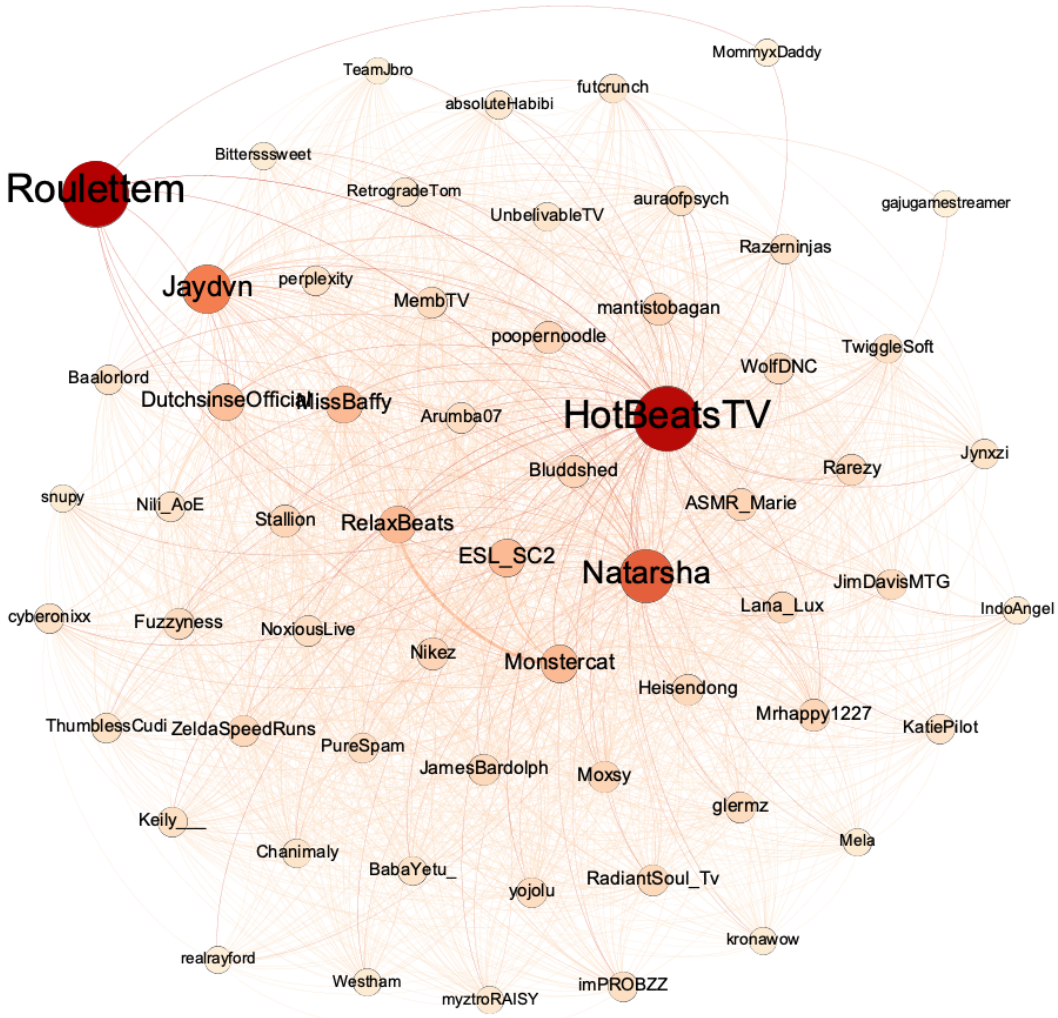


Figure 25 Visualization ranked with Bridging Centrality

This metric helped us identify a new streamer with power in the network. Roulettem didn't appear in the top 6 nodes with the highest scores in the previous metrics but this streamer appeared in our bridge analysis. If we look at the game name, this account played slots. If we take a look on this twitch account which is related to Roulette master strategy while as we discussed HotBeatsTV shares music. Roulettem is the link to other nodes with Music and Relaxing content and also to nodes to the popular game "World of Warcraft".

Gender and homophily

Homophily is the common tendency of people to associate with others who are similar to themselves. This phenomenon is also referred to "assortative mixing". This means that similar nodes have higher probability to connect with each other than the nodes which are not similar (Newman, 2018).

Radial Axis layout was used for assessing the homophily of the network. Homophily based on modularity class. Nodes are grouped by modularity class and ranked by their degree:

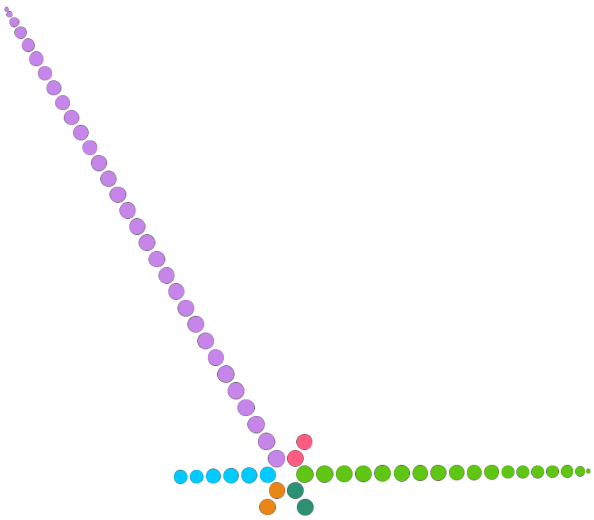


Figure 26 Homophily grouped by modularity class and ranked by degree

However, it's not easy to assess the homophily of the network if we use the degree, since most of the nodes are very similar. It's better to visualize the network in a different way, namely grouped by modularity class and ranked by weighted degree, and we can clearly see a different picture:

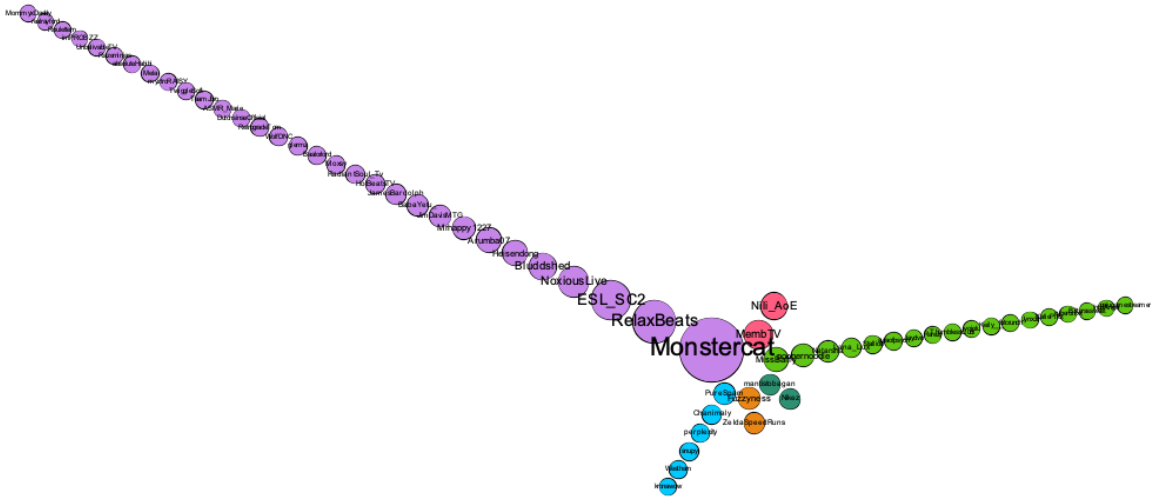


Figure 27 Homophily grouped by modularity ranked by weighted degree

The nodes that are closest to the center are among the most important ones. This visualization yielded some interesting results. The streamers with the highest average degree from every group are in the center and it is clear that Monstercat has a central position in its class but also in the network in the aspect of weighted number of common followers they share with the other streamers. It's interesting to note that in the other analysis we conducted HotBeatsTV stands out as the node with the highest number of connections in the network and also stands out in terms of betweenness centrality and bridging centrality.

Graph Density

Density is often referred to as connectivity of the network and is the fraction of the edges of the network compared to the number of edges it would have been if the network was fully connected. It can be thought of as the probability that a pair of nodes selected at random from the whole network, is connected by an edge (Newman, 2018). In this network the density is 0.807. This means than any two random streamers have a high probability of having one common follower. This makes sense since the lower bound as a number of followers is very low.

However, Gephi allows us to perform dynamic visualizations with our network and we can explore what would happen if we increase this lower bound in the weight between two streamers. In the visualization presented below when we color the nodes based on their weighted degree, we can see that by increasing the lower bound in the edge weight from 2 to 1000, we can see that the network changes a lot since the number of edges has decreased significantly and now less streamers are connected with each other:

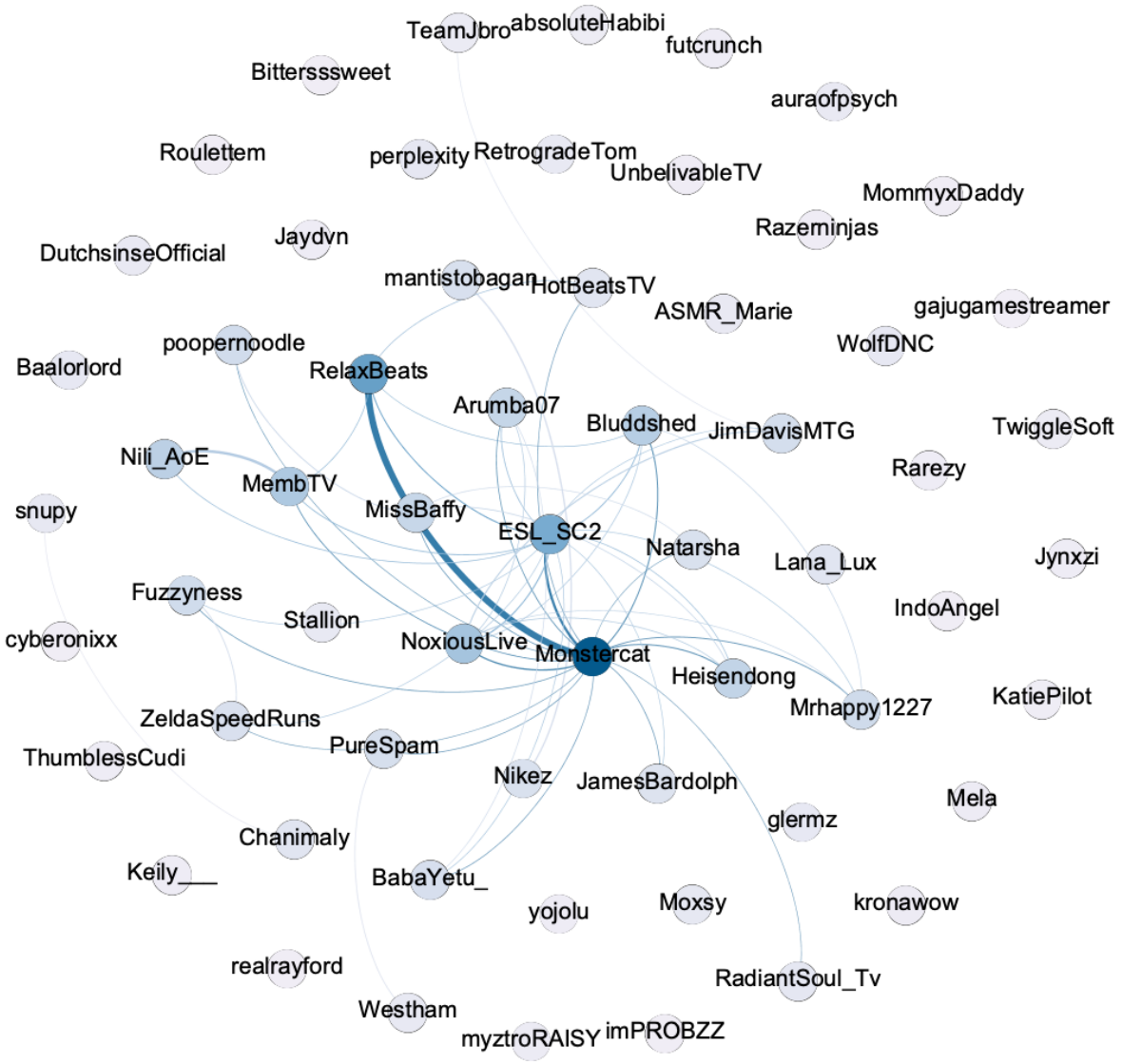


Figure 28 >1000 common followers

Community Structure (modularity)

In order to understand the network better, modularity analysis was implemented. This analysis is conducted when the network should be divided in communities. Since the optimal solution is not guaranteed, algorithms are used to find the best division of the network in clusters. Usually, it fails to recognize small communities, but this wasn't the case in this analysis since there's only one giant component.
Statistics from modularity analysis:
Modularity/ Modularity with resolution: 0.199
Number of Communities: 6

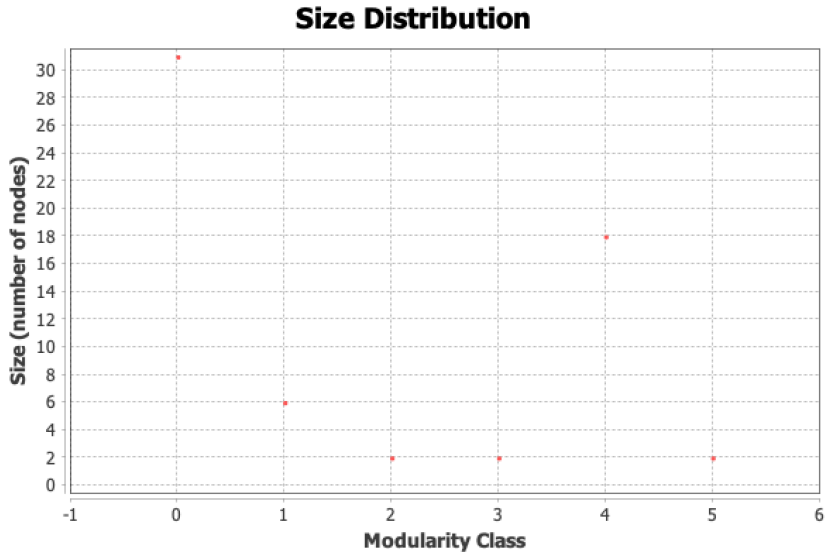


Figure 29 Modularity Class Distribution

Visualization of network in which the color of the nodes depends on the community they belong to:

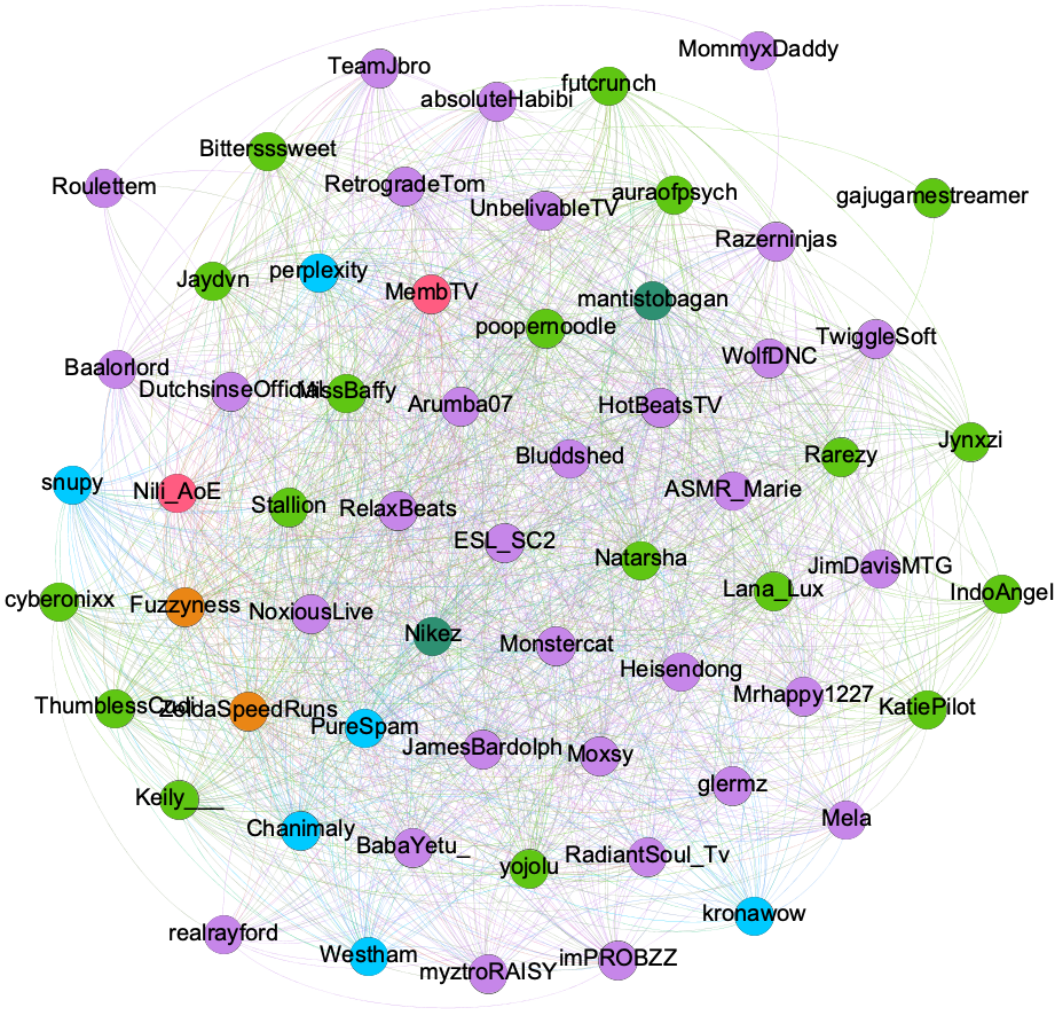


Figure 30 Network partitioned by modularity class

We observe that the majority of the nodes belong to the 0-modularity class then another 30% belongs to the modularity class 4 and another 10% is assigned to modularity class 1. However, what stands out is the remaining 3 clusters. We can take a look which games they include.

0	(50.82%)
4	(29.51%)
1	(9.84%)
2	(3.28%)
3	(3.28%)
5	(3.28%)

Figure 31 Modularity classes

Modularity class 2:

- Super smash Bros Melee
- The Legend of Zelda: Ocarina of Time

By searching on google we find that both these games were published by Nintendo, the first one in 2001 (Wikipedia, n.d.) and the second one in 1998 (Wikipedia, n.d.). Even though the cluster is limited it seems to be a separation based on real characteristics of these games.

Modularity class 3:

- Grand theft auto V

Both streamers streamed the same game. It seems to be a good separation as well.
Modularity class 5:

- Counter- Strike: Global Offensive
- Age of Empires II

Even though the time each one of these games was published isn't connected (Age of empires in 1999 and Counter strike 2012) there seems to be a connection with the platform that can be used to play the game since both of them include the platform MacOS and Microsoft (Steam, 2012), (Empires, n.d.). However, this seems to be almost an irrelevant connection.

In the smallest clusters the algorithm seems to have performed well but in the largest ones, namely 0 and 4, it didn't. This happens because the number of nodes belonging to these communities is enormous and they include different content, for example apart from games also music etc. Therefore, we will implement a different clustering algorithm "Newman-Girvan Clustering" which can be used by downloading the plugin with the same name. The following settings were specified for the use of this algorithm:

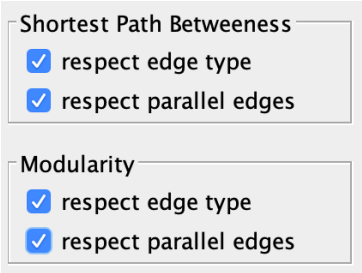


Figure 32 Settings Newman-Girvan Clustering

This algorithm yielded 4 communities. The modularity report returned from the algorithm was the following:

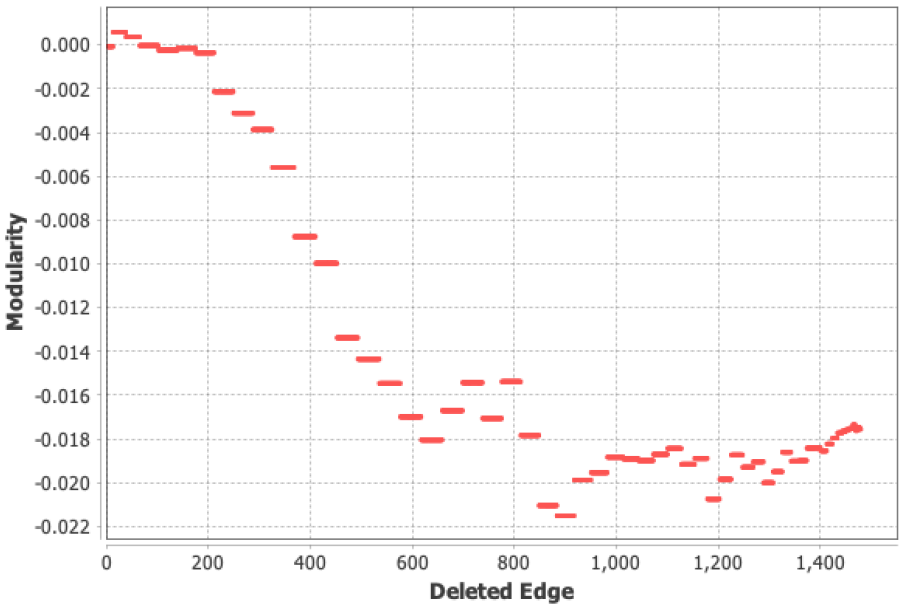


Figure 33 Newman-Girvan Algorithm

Unfortunately, this algorithm performed even worse than the previous one since it identified every node as one community. A different tuning of settings might help to cluster the network more efficiently. The visualization is displayed below:

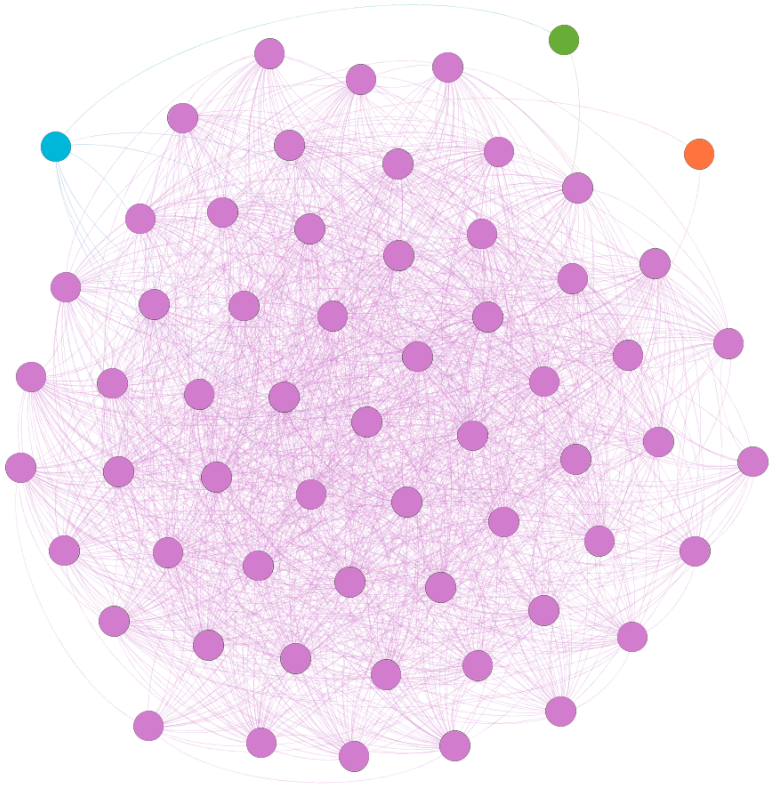


Figure 34 Network Newman - Girvan Clustering Algorithm

PageRank Algorithm

Pagerank is an algorithm whose name was given by Google. It's a centrality measure and it is used by Google to rank web searches, by estimating the importance of each one of the web pages. When a webpage has links from other important websites it's probable that the website is important too (Newman, 2018). In the context of this assignment, PageRank algorithm helps us identify which streamers become important because they have mutual connections with other important streamers. In our network this can be interpreted as: "*the streamers who have connections with other important streamers have higher PageRank scores compared to the streamers who have many connections with other streamers*". The tuning for the algorithm is presented below. PageRank can be used also for undirected networks even if it was designed for directed networks. This holds true also for HITS algorithm (Kleinberg, et al., 1999).
Pagerank Parameters:
Epsilon = 0.001
Probability = 0.85

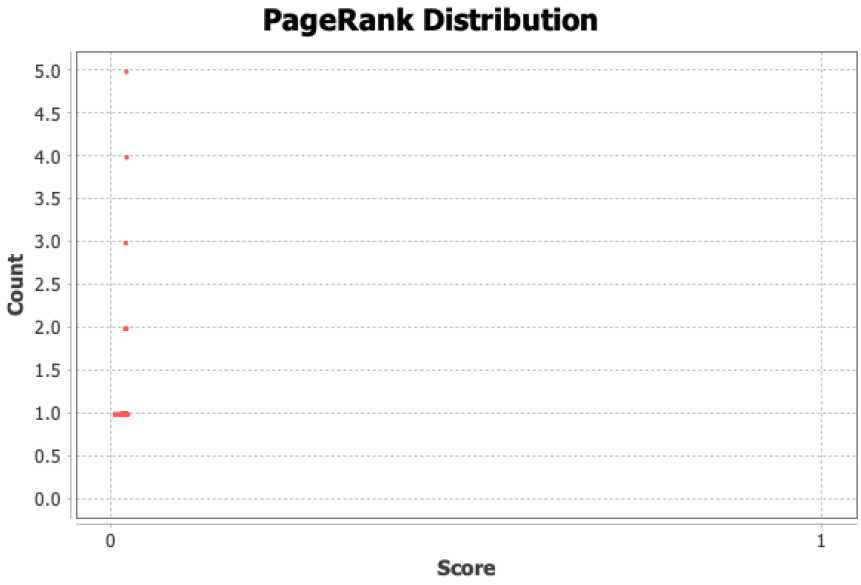


Figure 35 PageRank Distribution

In order to visualize the network using the PageRank results, the plugin "Dual Circle Layout" was used, and the nodes were ordered by their PageRank scores:

Streamer name	PageRank Score
HotBeatsTV	0.02086
Natarsha	0.02041
Monstercat	0.01942
ESL_SC2	0.01942
MissBaffy	0.01942
RelaxBeats	0.01942

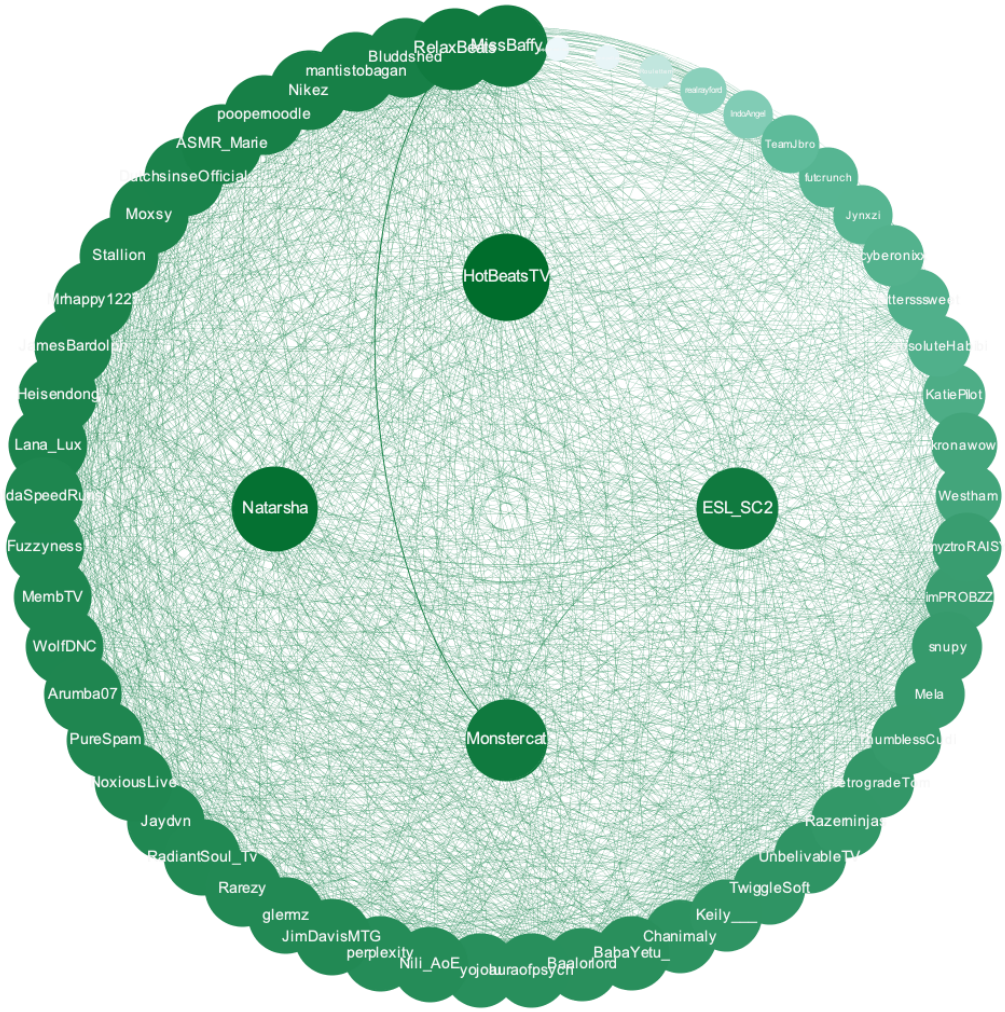


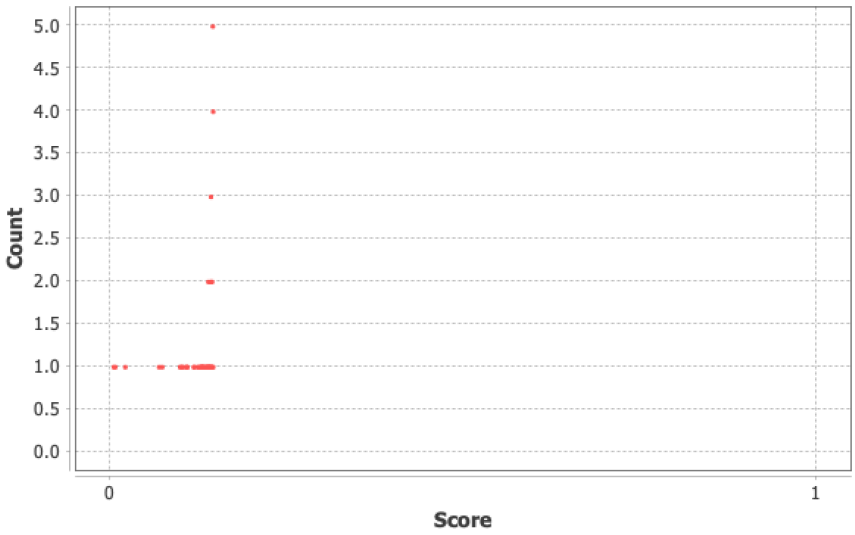
Figure 36 PageRank Algorithm

This algorithm verifies what we already knew; HotBeatsTV is the most important streamer in the network. Even though this streamer doesn't have the highest weighted degree, it's connected with the node with the highest weighted degree which is Monstercat. We saw that HotBeatsTV has the highest betweenness and bridging centrality in the network and has the power to "control" the information flow in the network which will be discussed further in the next section.

HITS Metrics

The initials for the HITS algorithms stand for hyperlink-induced topic search (HITS). As discussed before, HITS also works for the case of undirected network. The case is that both the authorities and the hubs get the same score. The scores for hubs and authorities are different in the case of a directed network. HITS takes into account both in and out going links, whereas PageRank is only using incoming links (Kleinberg, et al., 1999). A node with high authority centrality is pointed to by many nodes with high hub centrality. The characteristic of a node with high hub centrality is that it points to many nodes with high authority centrality. Hubs help us assess the access to information. For example, a significant scientific paper (high authority score) is expected to be sited by significant reviews (high hub score) (Newman, 2018).

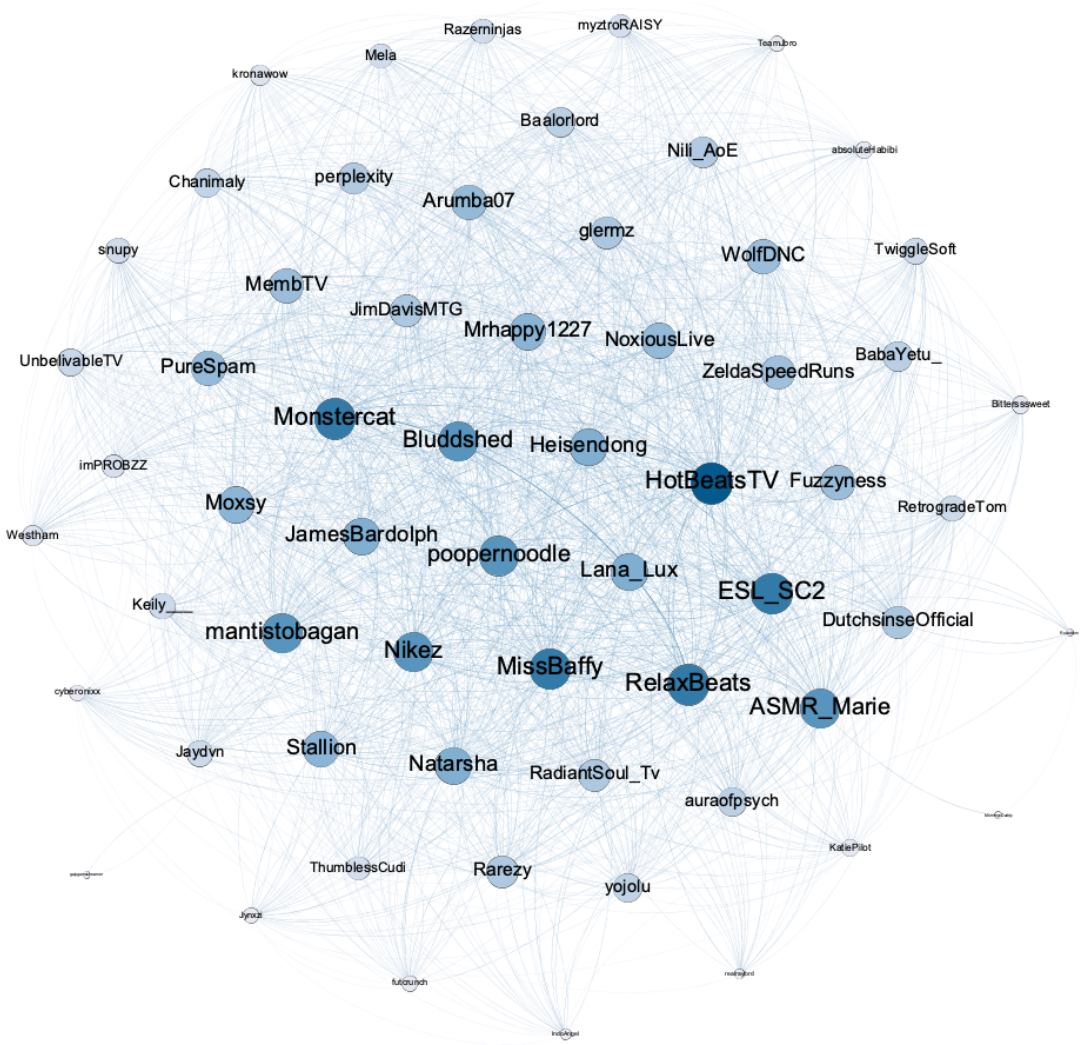
Hubs / Authority Distribution:



The highest authority / hub score of each streamer is given:

Streamer name	Authority/ Hub Score
HotBeatsTV	0.1441
Monstercat	0.1440
ESL_SC2	0.1440
MissBaffy	0.1440
RelaxBeats	0.1440
Bluddshed	0.1437

The visualization presents the network ranked with the streamers' authority:



The authority / hub score is almost the same for the streamers with the highest scores. Since the network is undirected this means that the streamers with the highest authority scores are the ones that get the most information from other streamers. They get more followers from them and at the same time they have high hub's score since they link with many other streamers. The results of HITS algorithm in our network mean that streamers who have a common edge both benefit from the connection; they get access to nodes who have more information while they share their information. Information in this case is the followers of the streamers in the sense that if two streamers start having common followers they may start "exchanging" their followers through Twitch's streamers suggestions for their users.

Network Specific Analysis

By partitioning the network with the game name, we can see that the nodes with the highest weighted degree belong in the category “Music”:

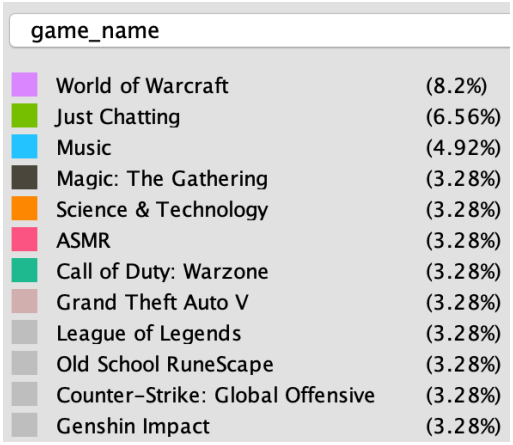


Figure 37 Game Name

Even though most streamers were playing the game “World of Warcraft”, these streamers didn’t have the highest weighted degree in the network.

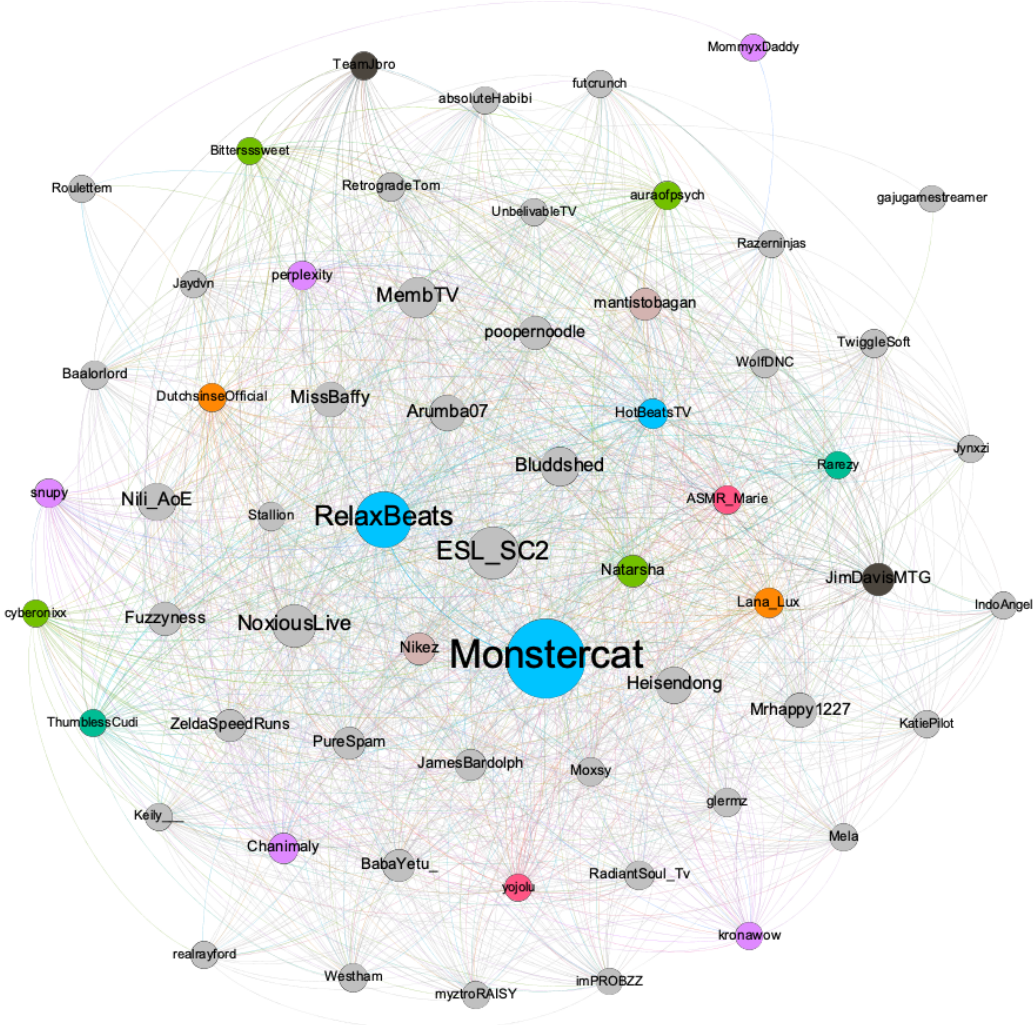
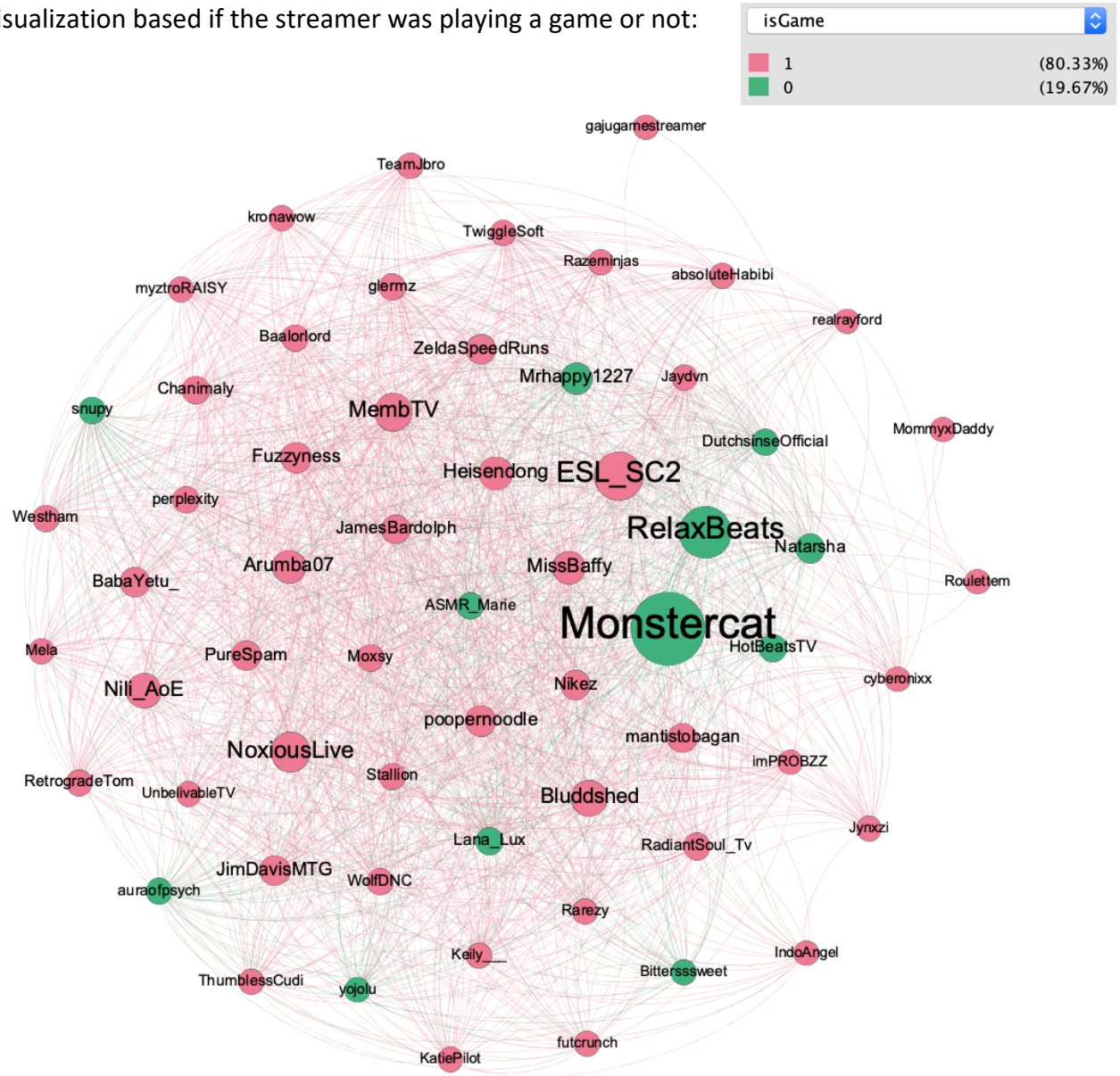


Figure 38 Network partitioned with game name

Visualization based if the streamer was playing a game or not:



Conclusion

To conclude, this social network analysis yielded some crucial findings. While Twitch.tv started as an online gaming streaming platform, this is not the case anymore. People not only participate in online communities to learn to play or watch gaming, but they have shifted to seeking more meaningful connections. This is proved by the fact that streamers in the “Music” and “Just Chatting” have either the most connections with other streamers or the most power in the network. Even though more than 80% of the streamers were playing some game, still the power in the network does not belong to these nodes. This can be interesting for Twitch itself, since they were always competing with YouTube in terms of their platform YouTube Gaming and not about the regular content which is posted on YouTube. This can be an opportunity for Twitch to try to focus in a new and emerging customer segment of their platform which enjoys music and wants to be part of a personalized online community.

Extras

An Async bot for twitch

The code presented below creates a bot which communicates and fetches the comments from the live streaming of one or multiple users. These channels are specified in line 8 in the variable “initial channels”. The IRC token is needed this time which can be generated in this website, for testing the API token can be set to “test” and the nickname is the name of my channel.

```
1  from twitchio.ext import commands
2
3  bot = commands.Bot(
4      irc_token='oauth:677hgdmf1jw8pdktptf2txxann84wr',
5      api_token='test',
6      nick='effieko',
7      prefix='!',
8      initial_channels=['MacieJay']
9  )
10
11 # Register an event with the bot
12 @bot.event
13 async def event_ready():
14     print(f'Ready | {bot.nick}')
15
16 @bot.event
17 async def event_message(message):
18     print(message.content)
19
20     # If you override event_message you will need to handle_commands for commands to work.
21     await bot.handle_commands(message)
22
23 # Register a command with the bot
24 @bot.command(name='test', aliases=['t'])
25 async def test_command(ctx):
26     await ctx.send(f'Hello {ctx.author.name}')
27 bot.run()
```

The expected result of this code is a continuous flow of asynchronous live comments fetched directly from the Twitch server:

```
Ready | effieko
Is the Pog emote fr gone??
doggo is. very chill
twitch removed the pog champ emote
How old is she?
```

An interesting analysis of these comments from one or multiple streamers for future network analysis could be to perform a sentiment analysis or a keyword analysis of which terms appear more frequently and how often are different keywords discussed together in the continuous chat flow.

Bibliography

1. DiPietro, M., 2011. *Business Wire.* [Online]
   Available at:
   https://www.businesswire.com/news/home/20110606005437/en/Justin.tv-Launches-TwitchTV-World%E2%80%99s-Largest-Competitive-Video
   [Accessed 07 01 2021].
2. Twitch.tv, n.d. *About Twitch.tv.* [Online]
   Available at: https://www.twitch.tv/p/el-gr/about/
   [Accessed 07 01 2021].
3. Geeter, D., 2019. *CNBC.* [Online]
   Available at: https://www.cnbc.com/2019/02/26/history-of-twitch-gaming-livestreaming-and-youtube.html
   [Accessed 07 01 2021].
4. Soper, T., 2014. *Geek Wire.* [Online]
   Available at: https://www.geekwire.com/2014/amazon-acquires-twitch-970m/
   [Accessed 07 01 2021].
5. Iqbal, M., 2020. *Business of Apps.* [Online]
   Available at: https://www.businessofapps.com/data/twitch-statistics/
   [Accessed 07 01 2021].
6. Churchill, B. C. B. & Xu, W., 2016. The Modern Nation: A First Study on Twitch.TV Social Structure and Player/Game Relationships. *IEEE International Conferences on Big Data and Cloud Computing, Social Computing and Networking, Sustainable Computing and Communications,* pp. 223-228.
7. Dux, J., 2018. *SOCIAL LIVE-STREAMING: TWITCH.TV AND USES AND GRATIFICATION THEORY SOCIAL NETWORK ANALYSIS.* s.l.:s.n.
8. Chugh, A., 2020. *Medium.* [Online]
   Available at: https://medium.com/better-programming/how-to-find-your-mutual-connections-on-medium-d0c586c04e6f#id_token=eyJhbGciOiJSUzI1NiIsImtpZCI6IjZhZGMxMDFjYzc0OThjMDljMDEwZGMzZDUxNzZmYTk3Yzk2MjdlY2IiLCJ0eXAiOiJKV1QifQ.eyJpc3MiOiJodHRwczovL2FjY291bnRzLmdvb2dsZS5jb2
   [Accessed 27 12 2020].
9. Wikipedia, n.d. *Wikipedia.* [Online]
   Available at: https://en.wikipedia.org/wiki/The_Legend_of_Zelda:_Ocarina_of_Time
   [Accessed 24 02 2021].
10. Wikipedia, n.d. *Wikipedia.* [Online]
    Available at: https://en.wikipedia.org/wiki/Super_Smash_Bros._Melee
    [Accessed 24 02 2021].
11. Steam, 2012. *Steam.* [Online]
    Available at:
    https://store.steampowered.com/app/730/CounterStrike_Global_Offensive/
    [Accessed 24 02 2021].
12. Empires, A. o., n.d. *Age of Empires.* [Online]
    Available at: https://www.ageofempires.com/games/aoeiide/
    [Accessed 24 02 2021].
13. Kleinberg, J. M. et al., 1999. The Web as a graph: measurements, models, and methods. *Computing and Combinatorics,* Volume 1627, pp. 1-17.

14. Newman, M., 2018. *Networks.* Oxford: University of Michigan.

Figures