Community Detection and Influencer Analysis in GitHub Social Networks

Anushka Chaubal - A20511568 and Rewa Deshpande - A20492328

Illinois Institute of Technology

Abstract :

Exploring GitHub's open-source software development reveals an intricate network of collaborations and technical interactions. In this study, we analyze the social structures within GitHub, using network analysis to determine how developers form communities and identify key influencers. We gathered data from the GitHub API to study the connections between top users and their followers, focusing on shared programming languages and geographical locations to map the network. Our methods included sophisticated graph theory algorithms to pinpoint closely-knit groups and central figures within these groups. The project faced challenges with sparse data and the complexities of handling large datasets, which we overcame by implementing batch processing and thorough data preparation. The findings of this study provide insight into how developers with different levels of proficiency in various programming languages collaborate within the GitHub community. This collaboration fosters a diverse learning environment where knowledge and skills are shared across different expertise levels, enhancing the overall innovative capacity of the community.

1 Introduction

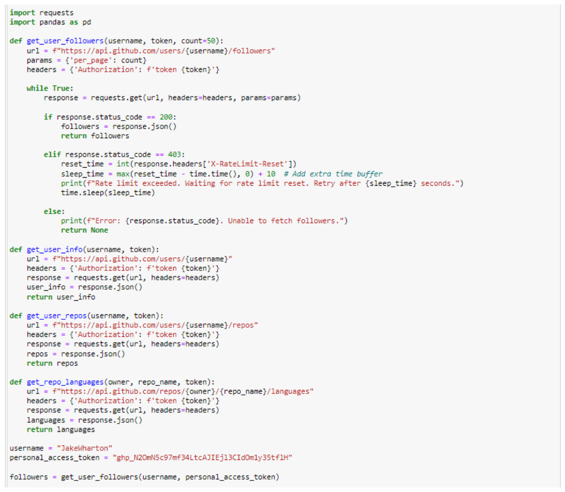
GitHub is a major platform where developers from around the world come together to work on projects, share their code, and connect with others. This platform is not just a place to store projects but also a network where people interact in complex ways. Understanding how people connect and form groups on GitHub can tell us a lot about how they work together and what trends are popular in software development. This project uses GitHub's API to gather and analyze data from some of the most influential users on GitHub and their followers. We look at where these users are from and what programming languages they use to try and figure out how communities form on GitHub. The research tackles the problems of collecting this data and figuring out how these groups come together when the information is scattered and varied. Through this study, we aim to shed light on how people form connections and communities in digital spaces, especially focusing on the leaders of these groups and how common interests and technologies bring people together.

2 Project Process and Methodology

2.1 Data Collection

Our methodology involves collecting data for the top 20 GitHub users, including information about their followers, repositories, and programming languages used. To manage the scale of the dataset and facilitate clear visualization, we employ data sampling techniques to select 250 nodes from the collected data. We then utilize community detection algorithms to partition the sampled data into cohesive communities and identify influential users within each community. By focusing on a subset of the GitHub network consisting of top users and sampled nodes, we aim to provide insights into the underlying community structure and influential actors within the GitHub ecosystem.

2.1.1 Code: Our code collects data from GitHub for a given user's followers. It retrieves information such as the location and programming languages used by each follower. Here's a breakdown of how the code works: **get_user_followers(username, token, count=50):** This function retrieves the followers of a given GitHub user. It sends a GET request to the GitHub API endpoint for retrieving followers. It handles rate limiting by waiting for the rate limit reset if the API rate limit is exceeded. **get_user_info(username, token):** This function retrieves information about a GitHub user, such as their profile details. It sends a GET request to the GitHub API endpoint for retrieving user information. **get_user_repos(username, token):** This function retrieves the repositories of a GitHub user. It sends a GET request to the GitHub API endpoint for retrieving user repositories. **get_repo_languages(owner, repo_name, token):** This function retrieves the programming languages used in a GitHub repository. It sends a GET request to the GitHub API endpoint for retrieving repository languages.



```
import requests
import pandas as pd

def get_user_followers(username, token, count=50):
    url = f"https://api.github.com/users/{username}/followers"
    params = {'per_page': count}
    headers = {'Authorization': f'token {token}'}

    while True:
        response = requests.get(url, headers=headers, params=params)

        if response.status_code == 200:
            followers = response.json()
            return followers

        elif response.status_code == 403:
            reset_time = int(response.headers['X-RateLimit-Reset'])
            sleep_time = max(reset_time - time.time(), 0) + 10  # Add extra time buffer
            print(f"Rate limit exceeded. Waiting for rate limit reset. Retry after {sleep_time} seconds.")
            time.sleep(sleep_time)

        else:
            print(f"Error: {response.status_code}. Unable to fetch followers.")
            return None

def get_user_info(username, token):
    url = f"https://api.github.com/users/{username}"
    headers = {'Authorization': f'token {token}'}
    response = requests.get(url, headers=headers)
    user_info = response.json()
    return user_info

def get_user_repos(username, token):
    url = f"https://api.github.com/users/{username}/repos"
    headers = {'Authorization': f'token {token}'}
    response = requests.get(url, headers=headers)
    repos = response.json()
    return repos

def get_repo_languages(owner, repo_name, token):
    url = f"https://api.github.com/repos/{owner}/{repo_name}/languages"
    headers = {'Authorization': f'token {token}'}
    response = requests.get(url, headers=headers)
    languages = response.json()
    return languages

username = "JakeWharton"
personal_access_token = "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"

followers = get_user_followers(username, personal_access_token)
```

Figure 1. Data Collection code

2.1.2 Compilance With Github Policies: In this research, we made sure to strictly follow GitHub's Privacy Policy to handle user data ethically. GitHub's policy has clear rules about using data from its API, including the need for user consent and rules against using data for unauthorized purposes. We followed these rules by only using publicly available data accessed through the official API and kept our analysis anonymous to protect user privacy. This careful approach helped maintain the integrity of our research and highlighted the importance of ethical data use.

2.2 *Data Analysis and Algorithms*

Algorithms: There are two algorithms which can be used to form the communities- Greedy Modularity Algorithm and Girvan-Newman. The Greedy Modularity algorithm is a heuristic algorithm that iteratively combines communities to maximize the modularity of the network partitioning. It identifies communities by evaluating the increase in modularity resulting from merging pairs of communities and selecting the pair that yields the highest increase in modularity. The Girvan-Newman algorithm is a hierarchical divisive algorithm that iteratively removes edges with the highest betweenness centrality, leading to the gradual disassembly of the network into smaller components. It identifies communities by iteratively removing edges that are likely to be between communities, based on high betweenness centrality, until the network is partitioned into distinct communities.

2.2.1 Choice of Algorithm We chose Greedy Modularity algorithm over Girvan Newman algorithm because of the following reasons: The Greedy Modularity algorithm directly optimizes modularity, a measure that quantifies the quality of community structure. By maximizing modularity, the algorithm aims to identify communities that are densely connected internally while having fewer connections between communities. The Greedy Modularity algorithm is computationally efficient for medium-sized networks compared to the Girvan-Newman algorithm, especially when the network size is not extremely large. Its heuristic approach to maximizing modularity results in faster runtime for networks of moderate size. The Girvan-Newman algorithm relies on betweenness centrality to identify edges for removal, which may not always effectively capture the underlying community structure, especially in noisy networks. In contrast, the Greedy Modularity algorithm focuses on optimizing modularity directly, which may make it more robust to noise and irrelevant edges.

2.2.2 Greedy Modularity Algorithm Initially, each user (represented as a node) forms its own community. Each user's community is based on the languages they use. The algorithm evaluates the increase in modularity resulting from merging pairs of communities (users). For each pair of communities, it calculates the modularity increase based on the shared languages between users in the two communities. This increase is determined by considering the edges (connections) formed between users who use similar languages. The algorithm selects the pair of communities that yields the highest increase in modularity and merges them into a single community. This process continues iteratively, with communities being combined based on the modularity increase until no further improvement in modularity is possible. Modularity quantifies the extent to which users within communities share languages compared to what would be expected by chance. The modularity calculation takes into account the number of edges (connections) formed between users who share languages within communities and compares it to the expected number of such connections in a random network. By maximizing modularity, the algorithm aims to identify a partitioning of users into communities that maximizes the similarity of language usage within communities while minimizing similarity between communities. The algorithm terminates when no further increase in modularity can be achieved by combining communities. This indicates that the current partitioning of users into communities optimally captures the similarities in language usage patterns. At this point, the algorithm outputs the final partitioning of users into communities, where each community consists of users who share similar language usage patterns and are less similar to users in other communities.

2.3 *Data Visualization*

3 Results

3.1 *Community Detection Analysis:*

We decided to use niche languages to detect communities so that users which use these rare languages can collaborate with eachother in the future. The languages which we used for community detection are: Vue, Clojure, Gherkin, Nginx, Mustache, Lex, ApacheConf, Kotlin, ProcFile, Puppet, Scala, haskell, CoffeeScript.

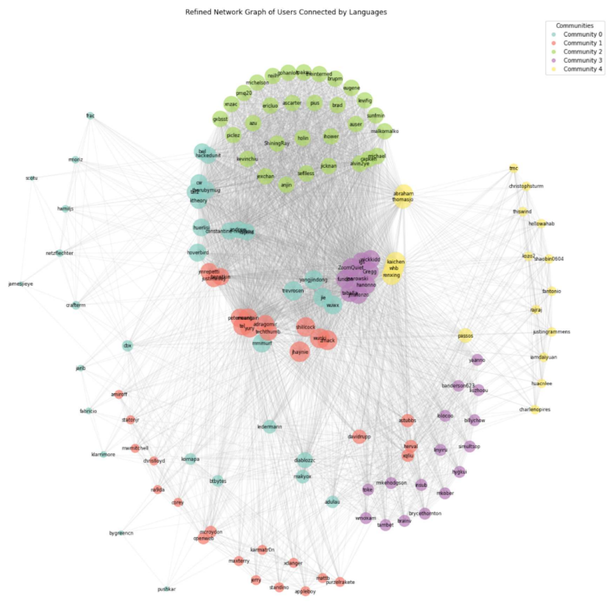


Figure 2. Communities

3.1.1 Communities We got 5 communities. **Community 1:** crafterm, adulau, taf2, huerlisi, btbytes, cbx, frac, ktheory, pushkar, constantine-nikolaou, andrew, hoverbird, makyox, trevrosen, klarrimore, bygreencn, fabricio, hamiljs, yangjindong, diablozzc, komapa, scotu, bwl, jarib, wuwx, jamesjieye, netzflechter, ledermann, rapind, therubymug, jie, hackedunit, cw, mmmurf, rmoriz. **Community 2:** xdanger, chrislloyd, karmatr0n, jerry, herval, xqliu, appleboy, standino, benatkin, mountain, purzelrakete, justinwiley, yury, jmrepetti, davidrupp, openweb, adragomir, mcroydon, mwmitchell, techthumb, jhaynie, shillcock, zmack, peterwang, tel, statonjr, astubbs, mattb, wunki, na9da, amiroff, maxterry, corey. **Community 3:** gxbsst, anjin, levifig, piclez, eugene, brupm, auser, pius, pmq20, brad, seflless, sunfmin, tpaksu, ascarter, malkomalko, capken, holin, gohanlon, theinterned, neilh, alvin2ye, jexchan, ihower, michael, kevinchiu, michelson, xnzac, ericluo, ShiningRay, jicknan, azu. **Community 4:** Gregg, mikehodgson, yaanno, ZoomQuiet, fundon, mkober, jmalonzo, insub, tambet, lolocoo, liuzhoou, lgs, banderson623, hanonno, wmoxam, jnarowski, linyiru, wickkidd, toke, taballa, billychow, hygkui, brainv, brycethornton, simultsop. **Community 5:** christophsturm, justingrammens, whb, charlenopires, thomasjo, rajraj, abraham, fantonio, kozo2, thiswind, huacnlee, passos, kaichen, iamdaiyuan, renxing, hellowahab, tmc, shaobin0604.

3.2 *Influencer Detection Analysis:*

We used Degree Centrality to find out the influencers of each community. The node/users having the highest degree centrality were chosen as the influencers of their respective community. We are displaying the top 5 influencers of each community.

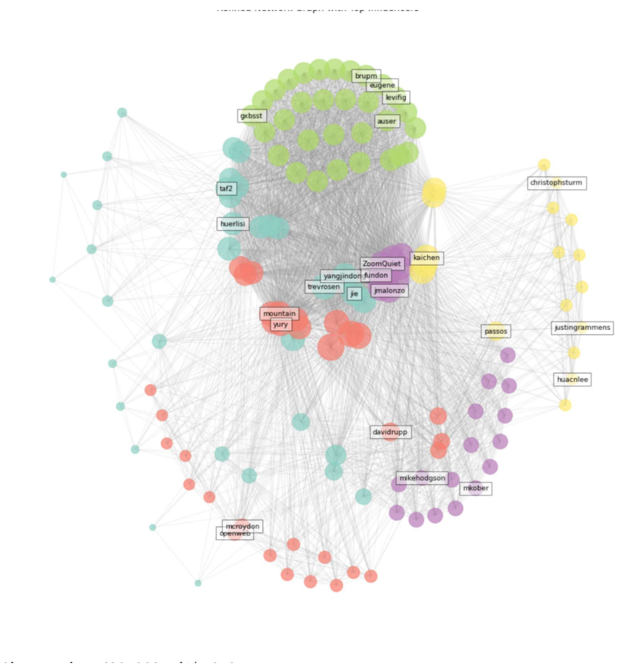


Figure 3. Influencers graph

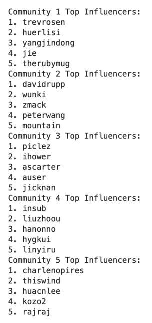

```
Community 1 Top Influencers:
1. trevrosen
2. huerlisi
3. yangjindong
4. jie
5. therubymug
Community 2 Top Influencers:
1. davidrupp
2. wunki
3. zmack
4. peterwang
5. mountain
Community 3 Top Influencers:
1. piclez
2. ihower
3. ascarter
4. auser
5. jicknan
Community 4 Top Influencers:
1. insub
2. liuzhoou
3. hanonno
4. hygkui
5. linyiru
Community 5 Top Influencers:
1. charlenopires
2. thiswind
3. huacnlee
4. kozo2
5. rajraj
```

Figure 4. Influencers list

The Influencers are those user whose degree centrality is the highest among all other users of the communities.

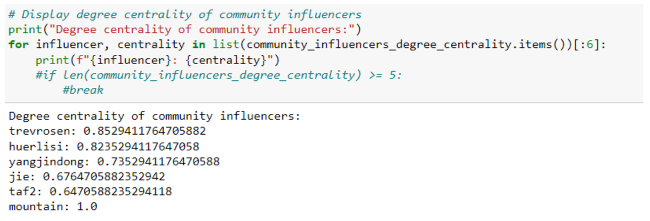

```python
# Display degree centrality of community influencers
print("Degree centrality of community influencers:")
for influencer, centrality in list(community_influencers_degree_centrality.items())[:6]:
    print(f"{influencer}: {centrality}")
    #if len(community_influencers_degree_centrality) >= 5:
        #break
```
```
Degree centrality of community influencers:
trevrosen: 0.8529411764705882
huerlisi: 0.8235294117647058
yangjindong: 0.7352941176470588
jie: 0.6764705882352942
taf2: 0.6470588235294118
mountain: 1.0
```

Figure 5. Influencers - Degree Centrality

3.3 *Insights:*

Through our community analysis of the GitHub user network, several key insights have emerged, guiding our approach to further analysis and interpretation of the data. We observed that users with high degree centrality, indicating a large number of connections, tend to play influential roles within their respective communities. Leveraging this insight, we identified these highly connected users as influencers within the network, recognizing their potential to disseminate information and foster collaboration among community members.

Furthermore, visualization can serve as a catalyst for community building within specific programming language ecosystems. Users can utilize visual representations to identify key influencers and thought leaders within their programming language communities, enabling them to reach out for mentorship, guidance, and collaboration opportunities. Additionally, visualization tools can be integrated into online platforms and social media channels dedicated to programming languages, providing users with interactive visualizations that facilitate networking and community engagement.

Overall, visualization serves as a bridge for connecting individuals with similar interests in programming languages, fostering collaboration, and facilitating knowledge exchange within the GitHub community. By leveraging visual representations of the network, users can unlock new opportunities for collaboration, innovation, and growth in their respective programming language communities.

3.4 *Comparison with Expected Results:*

When we looked at the results we got from our project and compared them to what we thought we'd find, we noticed a few things. At first, we thought we would see clear groups of GitHub users who shared the same programming interests and lived in similar areas. But it turned out to be a bit more complicated than that. While we did find some groups that fit our expectations, others were more mixed. Some groups had a lot in common, like using the same programming languages and interacting a lot, just as we thought. But other groups were more diverse, with members having different interests and not interacting much. We also expected to face challenges like not having enough data and limits on how much data we could get at once, and we did. Even though we faced these challenges, comparing what we expected with what we actually found gave us a better understanding of how social networks like GitHub work. This will help us in future research and improve how we analyze networks like these.

3.5 Reflection on Project Goals:

Reflecting on the goals of our project, we aimed to explore the structure and dynamics of the GitHub user network, with a focus on community detection and analysis. Our primary objective was to uncover meaningful insights into how GitHub users interact and form communities based on shared interests and geographical proximity. Throughout the project, we strived to apply network analysis techniques to identify and visualize these communities, with the ultimate aim of gaining a deeper understanding of the underlying patterns and connections within the network. While we encountered challenges and limitations along the way, such as data sparsity and rate limiting, we remained committed to our goals and adapted our approach to address these challenges. In doing so, we gained valuable insights into the complexities of online social networks and the factors that influence community formation. Looking back on our project goals, we are pleased with the progress made and the insights gained, and we recognize the opportunities for further exploration and refinement in future research endeavors.

4 Discussion

4.1 Challenges Encountered

4.1.1 Data Collection Time Constraints One of the significant challenges encountered during the project was the extended duration required to fetch data from the GitHub API. The inherent limitations on the number of API calls that could be made per hour posed a significant bottleneck, slowing the data collection phase considerably. To address this, we implemented a batch processing system that optimized the timing of API requests, allowing us to gather data efficiently within the constraints without triggering rate limits.

4.1.2 Community Formation Based on Programming Languages Another challenge arose when attempting to form communities based on the programming languages extracted from user repositories. Initially, including all languages led to the formation of an overly large and undifferentiated community, which was not useful for detailed analysis. To refine our approach, we shifted our focus to unique, less commonly used programming languages. This strategy aimed to identify niche communities where collaboration might be centered around specific technologies or languages, providing insights into how diverse programming preferences could influence community structures.

4.1.3 Visualization Challenges The visualization of network data presented its own set of challenges, particularly when displaying user names within communities. Overlapping labels in the initial graphs made it difficult to distinguish between individual nodes, detracting from the usability and clarity of the visual representations. To overcome this, we experimented with different visualization techniques, adjusting node placements, and experimenting with label positioning and size. This trial-and-error process was crucial in achieving clearer, more informative visualizations that effectively communicated the structure and composition of the communities.

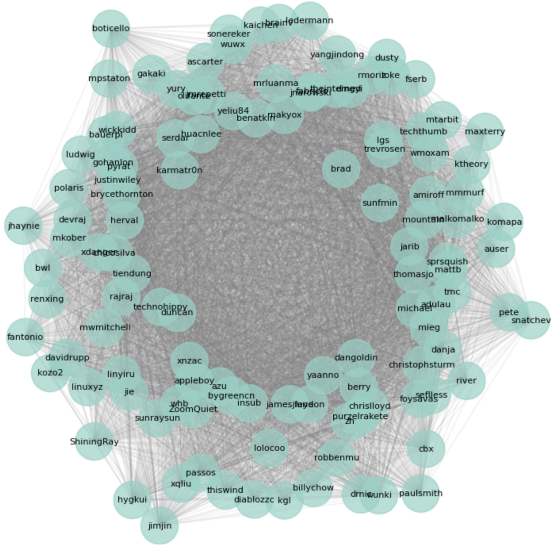


Figure 6. The network graph depicts users connected by the languages Dockerfile, JavaScript, and Python. Notably, JavaScript emerges as the most influential language, forming dense connections among users. This centrality of JavaScript demonstrates its widespread adoption across the network. However, the dominance of JavaScript also limits the formation of distinct communities, as it is shared among a significant proportion of users. Consequently, the prevalence of a common language such as JavaScript can hinder the delineation of distinct communities within the network, emphasizing the role of language diversity in community formation."

4.2 Exploring Different Approaches

In response to these challenges, the project explored several innovative approaches. For data collection, the batch processing approach was refined over time to maximize efficiency, adapting the timing and frequency of API requests to balance speed with compliance to GitHub's usage policies.

For community detection, the focus on unique programming languages provided a fresh perspective on how less mainstream technologies could serve as focal points for collaboration. This approach not only helped in forming more distinct community clusters but also offered a unique lens through which to view collaboration patterns within the GitHub network.

In terms of visualization, the adjustments made to improve the clarity of network graphs involved not only technical tweaks in software settings but also a deeper understanding of graphical data presentation principles. These adjustments ensured that final outputs were not only visually appealing but also rich in information.

4.3 Lessons Learned

This project underscored several crucial lessons about handling large datasets and the intricacies of network analysis. It highlighted the importance of being adaptable in research methodologies—especially when faced with technical constraints such as API limitations. The exploration of community formation based on unique identifiers like uncommon programming languages also shed light on the diverse ways in which digital communities can organize and interact.

Furthermore, the visualization challenges emphasized the critical role of effective data presentation in network analysis. Clear visualizations are essential for conveying complex information in an accessible way, underscoring the need for precision and innovation in graphical representation techniques.

These experiences have not only enriched our understanding of network dynamics on GitHub but also provided valuable insights into effective strategies for data analysis and presentation in complex digital ecosystems.

5 Conclusion

In conclusion, our project represents a significant endeavor to explore the structure and dynamics of the GitHub user network through the lens of network analysis and community detection. Through our efforts, we have successfully uncovered meaningful insights into the organization and connectivity of the network, shedding light on how GitHub users interact and form communities based on shared interests and geographical proximity. Despite encountering challenges such as data sparsity and rate limiting, we have demonstrated resilience and adaptability in overcoming these obstacles and advancing towards our goals.

Our project has contributed to the broader understanding of online social networks and their complexities, providing valuable insights into the factors that influence community formation and network dynamics. By leveraging network analysis techniques and community detection algorithms, we have gained a deeper understanding of the underlying patterns and connections within the GitHub user network, paving the way for future research and exploration in this field.

Moving forward, there are several avenues for further investigation and refinement. Future research endeavors may focus on refining community detection algorithms, addressing data quality issues, and exploring additional factors that influence community formation. Additionally, the insights gained from our project can inform the development of tools and strategies to enhance collaboration and engagement within online communities.

In summary, our project represents a significant step towards understanding and analyzing online social networks, with implications for a wide range of applications, including community building, social media analytics, and digital marketing. We are excited about the possibilities for future research and innovation in this area and look forward to contributing to the ongoing dialogue on online social network analysis.

6 Future Work

To enhance the efficiency of data collection from the GitHub API, various strategies can be explored, including optimizing API requests, implementing parallel processing techniques, and utilizing caching mechanisms to minimize redundant requests. Additionally, there is a scope for improving community detection methods by investigating alternative algorithms or extensions to the Greedy Modularity approach, such as Louvain Modularity and Label Propagation, to enhance accuracy and scalability. Furthermore, integrating additional features from user profiles and repositories, such as user activity metrics and project descriptions, can enrich the community detection and influencer identification process, providing a more comprehensive understanding of collaboration networks on GitHub. With additional time and resources, future research endeavors could extend to dynamic analysis, tracking how communities evolve over time. This could involve implementing time-series analysis techniques to monitor changes in community structures and member interactions. Furthermore, the integration of machine learning methodologies could facilitate the prediction of future changes in community compositions and the identification of emerging influential users. Additionally, exploring the impact of specific events, such as GitHub's annual Game Off or Hacktoberfest, on community behavior and dynamics could offer valuable insights into the role of external factors in shaping collaboration patterns and fostering community engagement. By incorporating these aspects into the research framework, a more comprehensive understanding of community dynamics and their evolution within the GitHub ecosystem can be achieved.

References

[1] GeeksforGeeks. Directed Graphs, Multigraphs, and Visualization in NetworkX. Available online: `https://www.geeksforgeeks.org/directed-graphs-multigraphs-and-visualization-in-networkx/`

[2] Author Name. Title of the article. Journal Name. Year;Volume:Pages. Available online: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1482622/`

[3] NetworkX Development Team. NetworkX Documentation. Available online: `https://pydocs.github.io/p/networkx/2.8.2/api/networkx.algorithms.community.modularity_max.greedy_modularity_communities.html`

[4] Thomas Aynaud. Python-Louvain Documentation. Available online: `https://python-louvain.readthedocs.io/en/latest/api.html`

[5] NetworkX Development Team. NetworkX Reference: Community Detection. Available online: `https://networkx.org/documentation/stable/reference/algorithms/community.html`

[6] Towards Data Science. Identifying Influencers on Social Media: A Guide to Social Network Analysis Using Python. Available online: `https://towardsdatascience.com/identifying-influencers-on-social-` media-a-guide-to-social-network-analysis-using-python-e05f4da151b8