CSP 579–Online Social Network Analysis
Project 2
Final Report

Title:

GitHub Analysis

TEAM MEMBERS

Naman Rajendra Jangid
njangid@hawk.iit.edu
A20527772

Pranjali Deshmukh
pdeshmukh3@hawk.iit.edu
A20527773

GitHub Analysis

Pranjali Deshmukh
Illinois Institute of Technology
Chicago, Illinois
pdeshmukh3@hawk.iit.edu

Naman Rajendra Jangid
Illinois Institute of Technology
Chicago, Illinois
njangid@hawk.iit.edu

Abstract—***GitHub is a prominent platform for collaboration among developers, hosting millions of repositories and facilitating interactions through various events such as commits, pull requests, and issues. Understanding the dynamics of this social network provides valuable insights into collaboration patterns, popular projects, and influential users. Our project aimed to analyze the GitHub social network to uncover user interactions, community structures, and popular languages & projects. We aimed to identify influential users, visualize network dynamics, and gain insights into collaboration and coding trends within the GitHub ecosystem.***

Keywords—*Network Model, Community Detection, Python, NetworkX.*

I. INTRODUCTION

Analyzing the GitHub social network holds great importance for improving the GitHub community and its applications. GitHub evolves into a central hub for collaboration among developers worldwide, hosting millions of repositories and fostering connections through various events such as commits, pull requests, and issues. Understanding the dynamics of this network can lead to valuable insights into collaboration patterns, coding trends, and the technological landscape of software development.

Our project aims to explore the GitHub social network to uncover user interactions, community structures, and popular languages and projects. By doing so, we aim to achieve several objectives. Firstly, we seek to identify influential users within the GitHub community by analyzing follower relationships and popular topics (programming languages). These users play a pivotal role in shaping discussions, driving collaborations, and promoting projects. Secondly, we aim to visualize network dynamics by constructing a network model representing GitHub users and their connections. This visualization allows us to explore the structure of the GitHub community, including clusters of users with similar interests and collaborative relationships.

Furthermore, our project aims to discover popular programming languages and projects within different communities on GitHub. By analyzing repository data, including descriptions, user's bio, and repository details on topics, we aim to identify emerging trends and technologies in software development. Understanding these trends provides valuable insights for developers making decisions about their own projects and contributions. Lastly, our project aims to gain insights into collaboration and coding trends within the GitHub ecosystem. By studying user interactions, project contributions, and community structures, we aim to uncover patterns that could inform future development and collaboration efforts.

In addition to achieving our project goals, this project provides valuable learning opportunities to understand the course CSP 579–Online Social Network Analysis. By studying GitHub Network Analysis, we have the chance to explore network modeling, social network analysis, community detection, and data mining. Overall, this project presents a unique opportunity to gain insights into the GitHub community while expanding our knowledge in network modeling and social network analysis.

In further sections, we will learn about I Data, II Project Methodology (Project Process), III Results, IV Discussions and V Future Scope.

II. Data And Data Collection

The GitHub dataset used in our analysis consists of user information, follower data, and repository details obtained through the GitHub API. We collected data own our using these APIs. For each user, we collected data such as user bio, number of public repositories, number of followers, location, programming languages, login username, etc. To fetch user data, we utilized the `fetch_user_data` function, which takes a GitHub username and access token as input parameters. This function constructs the API URL for the specified user and sends a HTTP GET request to the GitHub API endpoints and parsing the JSON responses to extract relevant data.

```python
def fetch_user_data(username, access_token):
    # Fetch user data from GitHub API
    url = f"https://api.github.com/users/{username}"
    headers = {"Authorization": f"token {access_token}"}
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Failed to fetch data for {username}. Status code: {response.status_code}")
        return None
```

Fig 1: Data Extraction API

Additionally, we fetched data for their followers and repositories, limiting the followers' data to 1500 nodes to manage network size. We sampled data to 1500 as it helped cover all relevant data without missing out any valuable insights and also created a readable and self-explanatory graph. We also fetched repositories data using the `fetch_repositories`

function, which retrieves information about the repositories owned by a user.

There was no change in our approach from what was submitted for the data description deliverable. We maintained consistency in fetching, cleaning, and sampling the data to ensure the integrity of our analysis. By following this process, we obtained a comprehensive dataset that enabled us to conduct meaningful analysis of the GitHub social network and its dynamics.

III. Project Methodology

Our project methodology was structured around a series of key steps, each designed to uncover different aspects of the GitHub network.

A. Data Collection:

- We fetched user data, follower lists, and repository information from the GitHub API for a selected group of users. This involved making HTTP requests to the API endpoints and parsing JSON responses to extract pertinent details.[1] The data collection process was driven by the goal of obtaining comprehensive insights into the GitHub network's structure and dynamics. We have described more details in above section II.

Challenges: One of the main challenges encountered during data collection was the GitHub API's rate limits and data access restrictions. The rate limits-imposed constraints on the number of API requests we could make within a given time frame, leading to delays in data retrieval. Additionally, handling large volumes of data required careful planning and optimization to ensure efficient data collection without exceeding rate limits. We made sure we collect data that is most relevant and consider all data privacy constrains mentioned in pervious project data deliverable.

B. Data Cleaning

With the collected data in hand, we performed data cleaning to ensure its quality and consistency. This involved addressing missing values, standardizing data formats, and resolving any discrepancies encountered during the collection process. The python library was instrumental in facilitating efficient data cleaning operations, enabling us to prepare the dataset for further analysis.[2]

```
def clean_data(user_data):
    # Clean data by replacing missing values
    cleaned_data = {key: user_data.get(key, "Not Available") for key in ["bio", "public_repos", "followers", "login"]}
    return cleaned_data
```

Fig 2: Data Cleaning

Challenges: Data cleaning posed challenges in dealing with missing values, inconsistencies, and discrepancies in the collected data. Standardizing data formats and resolving inconsistencies required manual inspection and decision-making, leading to increased processing time and effort. Addressing missing values in large dataset, challenged us to use the most efficient way to execute it and consider the runtime limit of the code.

C. Network Construction

Using the NetworkX library in Python, we constructed a network representation of the GitHub ecosystem. Each GitHub user was treated as a node within the network, with different relationships (projects and programming languages) represented as edges connecting the nodes. [3,4] This network representation facilitated the visualization and analysis of the GitHub network's structure, enabling us to identify communities and influential individuals.

```
# Community detection
# Example using NetworkX
G = nx.Graph()
for username in usernames:
    followers_data = fetch_followers(username, access_token)
    for follower in followers_data:
        G.add_edge(username, follower['login'])

communities = list(nx.algorithms.community.label_propagation.label_propagation_communities(G))
```

Fig 3: Network Construction

Challenges: Constructing the GitHub network posed challenges in managing memory usage and computational complexity, particularly when dealing with large datasets. Building a network representation of the GitHub ecosystem required efficient data structures and algorithms to handle the diverse and interconnected nature of user relationships. Optimizing network construction algorithms and data storage mechanisms helped mitigate performance bottlenecks and improve overall efficiency.

D. Community Detection

- We applied community detection algorithms to partition the GitHub network into cohesive communities of users with similar collaboration patterns. Leveraging the label propagation algorithm, we identified distinct communities within the network, each characterized by shared interests or collaborative behaviors.[5,6] This analysis provided insights into the diverse and interconnected nature of the GitHub community.

```
# Plotting communities
plt.figure(figsize=(15, 10))
pos = nx.spring_layout(G)

for i, community in enumerate(communities):
    nx.draw_networkx_nodes(G, pos, nodelist=list(community), node_color=community_colors[i], node_size=100, alpha=0.7)

nx.draw_networkx_edges(G, pos, alpha=0.5)
nx.draw_networkx_labels(G, pos, font_size=10)

# Legend
legend_handles = [plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=community_colors[i], markersize=10, label=f"Community {i+1}") for i in range(len(communities))]
plt.legend(handles=legend_handles, loc="best")

plt.title("GitHub Network - Communities")
plt.axis('off')
plt.show()
```

Fig 4: Community Detection

Challenges: Community detection presented challenges in selecting appropriate algorithms and parameter settings to accurately delineate community structures within the GitHub network. Evaluating the performance of different community detection methods required thorough experimentation and validation against ground truth data. Tuning algorithm parameters and interpreting community detection results necessitated domain knowledge and expertise in network analysis techniques.

E. Influencer Identification

- In addition to community detection, we identified influential users within each community based on their degree centrality scores. Degree centrality measures the number of connections a node possesses within the network, serving as a proxy for user influence. [7,8] By identifying users with high degree centrality scores, we pinpointed influential individuals who play pivotal roles in facilitating collaboration and information dissemination within their communities.

```
# Influencer identification for each community
for i, community in enumerate(communities):
    subgraph = G.subgraph(list(community))
    degree_centrality = nx.degree_centrality(subgraph)
    influencers = sorted(degree_centrality, key=degree_centrality.get, reverse=True)[:5]
    print(f"\nTop 5 Influencers in Community {i+1}:")
    for j, influencer in enumerate(influencers):
        print(f"{j+1}. {influencer} - Degree Centrality: {degree_centrality[influencer]}")

# Plotting influencers for each community
for i, community in enumerate(communities):
    subgraph = G.subgraph(list(community))
    degree_centrality = nx.degree_centrality(subgraph)
    influencers = sorted(degree_centrality, key=degree_centrality.get, reverse=True)[:5]
    plt.figure(figsize=(10, 10))
    plt.bar(range(5), [degree_centrality[influencer] for influencer in influencers], color='skyblue')
    plt.xlabel('Influencers')
    plt.ylabel('Degree Centrality')
    plt.title(f'Top 5 Influencers in Community {i+1}')
    plt.xticks(range(5), influencers, rotation=45)
    plt.tight_layout()
    plt.show()
```

Fig 5: Influencer Identification for Each Community

Challenges: Identifying influential users within the GitHub network posed challenges in determining suitable centrality

metrics and threshold values for influencer identification. Choosing the right balance between sensitivity and specificity in identifying influential individuals required careful consideration of network characteristics and user behaviors. Addressing biases and limitations inherent in centrality measures was crucial for obtaining reliable influencer rankings.

F. Data Visualization

Throughout our analysis, we employed various data visualization techniques to visualize the GitHub network's structure, community formations, and programming language preferences.[9] These visualizations helped us gain insights into complex network relationships and communicate our findings effectively, which are further discussed in part IV Project Results.

IV Project Results

The analysis of the GitHub network yielded several noteworthy results, including:

A. Community Structure

Through community detection techniques, we successfully identified cohesive communities within the GitHub network. These communities exhibited diverse sizes and degrees of interconnectedness, reflecting the multifaceted nature of collaboration and shared interests on the platform.

Results Discussion: We attempted to create communities based on the projects users were involved in, as well as communities based on the programming languages used in their projects. Both approaches resulted in community graphs, but we noticed some differences. Communities based on projects appeared disconnected, with users scattered across various ranges. However, communities formed around programming languages were evenly distributed, and users often worked with multiple languages, creating links between these communities. This is illustrated in the figure below. Taking these factors into account, we decided to focus our further analysis on the communities based on programming languages.

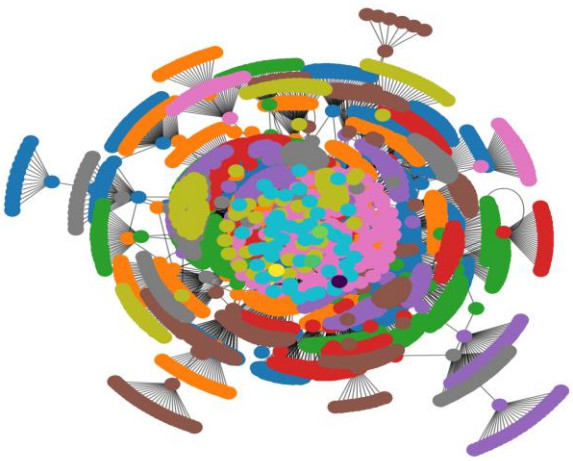

Fig 6: Community Based Projects

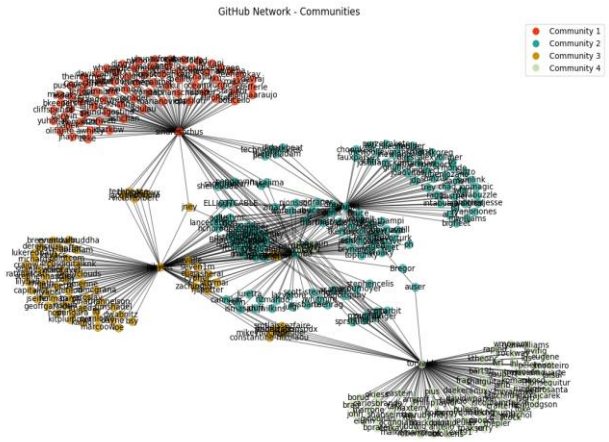

Fig 7: Community Based Programming Languages

B. Influential Users

Our analysis uncovered influential users who wield significant influence within their respective communities. These users, characterized by their high degree centrality scores, played pivotal roles in facilitating collaboration and information dissemination within their communities.

Results Discussion: We tried creating influencers over entire dataset, and picked top 5 influencers whose degree centrality values are given below. The graph also showcases the same in form of graph. (Fig 8 & 9). Also, we created influencers for individual community. We have consider the users with highest number of dense connection formed the networked based of the programming languages and we calculated their degree centrality, as shown in Fig (10 & 11).We have seen that, the degree centrality differs with respect to community and dataset consideration as the node density consideration is changed.

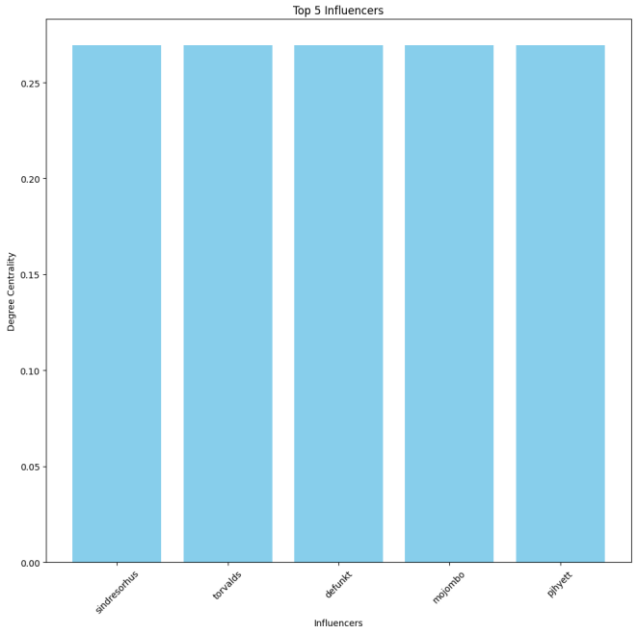

Fig 8 : Graph of Top 5 Influencers of Dataset

```
Top 5 Influencers:
1. sindresorhus - Degree Centrality: 0.2695417789757413
2. torvalds - Degree Centrality: 0.2695417789757413
3. defunkt - Degree Centrality: 0.2695417789757413
4. mojombo - Degree Centrality: 0.2695417789757413
5. pjhyett - Degree Centrality: 0.2695417789757413
```

Fig 9: Degree Centrality for Top 5 Influencers of Dataset.

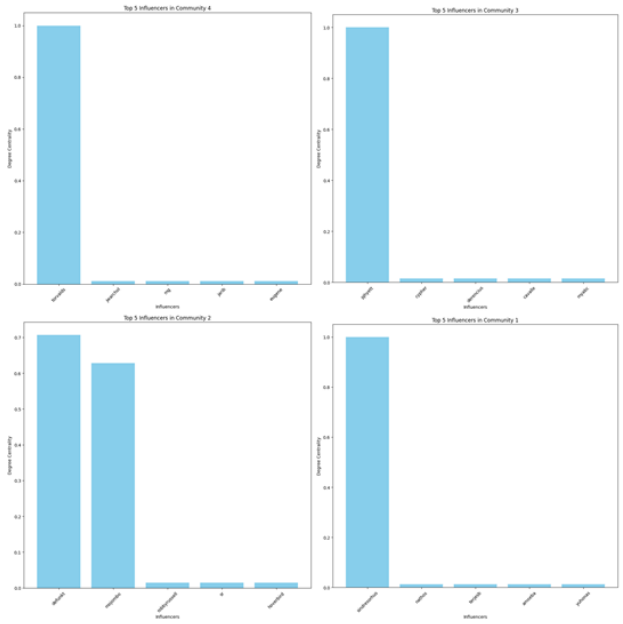

Fig 10: Top Influencers in Individual Communities

```
Top 5 Influencers in Community 1:
1. sindresorhus - Degree Centrality: 1.0
2. nathos - Degree Centrality: 0.013333333333333334
3. terjesb - Degree Centrality: 0.013333333333333334
4. amoeba - Degree Centrality: 0.013333333333333334
5. yuhonas - Degree Centrality: 0.013333333333333334

Top 5 Influencers in Community 2:
1. defunkt - Degree Centrality: 0.7071428571428571
2. mojombo - Degree Centrality: 0.6285714285714286
3. robbyrussell - Degree Centrality: 0.014285714285714285
4. sr - Degree Centrality: 0.014285714285714285
5. hoverbird - Degree Centrality: 0.014285714285714285

Top 5 Influencers in Community 3:
1. pjhyett - Degree Centrality: 1.0
2. cypher - Degree Centrality: 0.014492753623188406
3. derencius - Degree Centrality: 0.014492753623188406
4. cavalle - Degree Centrality: 0.014492753623188406
5. myabc - Degree Centrality: 0.014492753623188406

Top 5 Influencers in Community 4:
1. torvalds - Degree Centrality: 1.0
2. jwarchol - Degree Centrality: 0.011904761904761904
3. mg - Degree Centrality: 0.011904761904761904
4. jarib - Degree Centrality: 0.011904761904761904
5. eugene - Degree Centrality: 0.011904761904761904
```

Fig 11 : Degree Centrality of Top Influencers in Individual Communities

C. Network Dynamics

We observed dynamic interactions within the GitHub network, characterized by the formation, evolution, and dissolution of communities over time. The network exhibited a self-organizing behavior, with users forming clusters around common interests or projects, and new communities emerging as collaborations evolve and diversify. Like the below figure shows that, javascripts and Ruby have been used by most of the users in the community.[10]

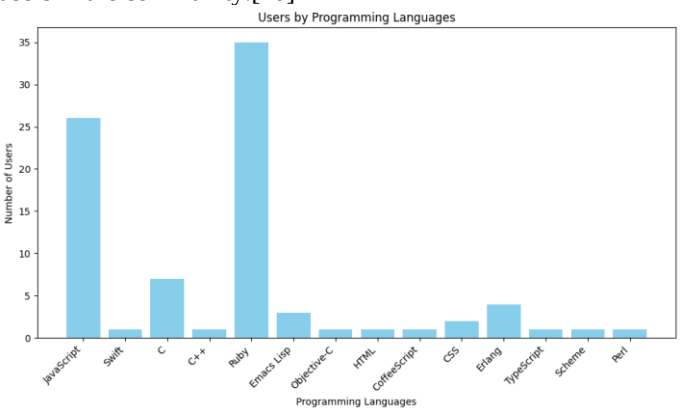

Fig 12: Populare Programming Language in Communities

IV Discussions & Conculsion

Reflecting on the journey of our project, we believe it has been a fulfilling and a good learning experience. Despite encountering challenges along the way, the project provided valuable insights into the GitHub network's structure, collaboration patterns, and influential users. Overall, we feel that the project went well, and we have achieved our primary objectives of conducting a comprehensive analysis of the GitHub ecosystem.

In terms of what we would do differently next time, there are a few key areas we would focus on. Firstly, we would refine our data collection and cleaning processes to address issues more efficiently and ensure the robustness of the dataset. This could involve implementing automated data validation techniques and exploring alternative data sources to enhance data quality and coverage. Additionally, we would invest more time in fine-tuning our community detection and influencer identification algorithms to improve accuracy and reliability. Experimenting with different algorithm parameters and validation techniques could help us achieve more precise results.

From this project, we have learned valuable lessons about the complexities of network analysis, data visualization, and community detection in the ecosystems. We gained insights into the challenges of working with large-scale datasets, navigating API rate limits, and interpreting network structures. Furthermore, the project enhanced our understanding of community detection algorithms, centrality metrics, and their applications in identifying key players within networks. Overall, the project served as a rich learning experience, equipping us with valuable skills and insights that will inform our future endeavors in data analysis and network science.

IV Future Work

Building upon the foundation laid by our project, there are several avenues for future exploration and research within the realm of GitHub network analysis:

A. Temporal Dynamics: Investigating the temporal evolution of collaboration patterns and community structures within the GitHub network could provide valuable insights into trends, seasonality, and emergent behaviors over time. Analyzing how collaboration networks evolve in response to external events, software releases, and community-driven initiatives could offer a deeper understanding of the GitHub ecosystem's dynamics.

B. Geospatial Analysis: Exploring the geographical distribution of GitHub users and repositories could uncover regional variations in collaboration patterns, programming language preferences, and community dynamics. Geospatial analysis techniques could be employed to visualize the geographic spread of collaboration networks, identify regional hubs of activity, and examine the impact of location on collaboration behaviors.

C. Social Network Analysis: Expanding the analysis to incorporate social network features such as user interactions, sentiment analysis, and community engagement metrics could provide a richer understanding of social dynamics within the GitHub ecosystem. Analyzing social networks within GitHub communities could reveal patterns of influence, information flow, and community sentiment, offering insights into social capital and community cohesion.

By embarking on these future research directions, we aim to deepen our understanding of collaborative ecosystems, advance the state of the art in network analysis, and contribute to the ongoing evolution of the GitHub platform as a hub for open collaboration and innovation.

REFERENCES

[1] Gousios, G. (2013). The GHTorrent dataset and tool suite. In Proceedings of the 10th Working Conference on Mining Software Repositories, 233-236.

[2] Vasilescu, B., Capiluppi, A., & Serebrenik, A. (2014). Gender, representation and online participation: A quantitative study of StackOverflow. In Proceedings of the 11th Working Conference on Mining Software Repositories, 202-211.

[3] Newman, M. (2010). Networks: An introduction. Oxford University Press.

[4] Barabási, A. L. (2016). Network science. Cambridge University Press.

[5] Fortunato, S. (2010). Community detection in graphs. Physics Reports, 486(3-5), 75-174.

[6] Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008(10), P10008.

[7] Freeman, L. C. (1979). Centrality in social networks: Conceptual clarification. Social Networks, 1(3), 215-239.

[8] Borgatti, S. P. (2005). Centrality and network flow. Social Networks, 27(1), 55-71.

[9] Heer, J., & Shneiderman, B. (2012). Interactive dynamics for visual analysis. Queue, 10(2), 30-59.

[10] Tufte, E. R. (2001). The visual display of quantitative information. Graphics Press.