Exploring Collaboration and Connections within the GitHub Social Network

Dongting Cai
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, CA 92093
docai@ucsd.edu

Zhihan Li
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, CA 92093
zhl023@ucsd.edu

Abstract

GitHub serves as a central hub for open-source collaboration, connecting contributors across millions of repositories. This study explores the structural properties, key contributors, and robustness of GitHub's top 100-starred repositories using network analysis techniques. We construct and analyze a collaborator network, identifying highly influential contributors and community structures. Centrality analysis reveals that automation tools like Dependabot and GitHub Actions play a crucial role in sustaining connectivity, alongside a small number of highly active contributors. [1]

1 Introduction

The open-source movement has fundamentally transformed software development, fostering a global ecosystem of collaboration. GitHub, as one of the most widely used platforms for open-source projects, serves as a central hub where developers, organizations, and automated tools collectively contribute to software evolution. While our prior engagement with GitHub has primarily involved basic functionalities such as version control and code sharing, the platform's underlying social and technical structures remain largely unexplored. Beyond individual repositories, GitHub operates as a complex collaboration network where contributors interact through issues, pull requests, code reviews, and automated tools, shaping the development workflow in unique ways.

1.1 Motivation and hypothesis

Understanding GitHub's collaborative structure is crucial for uncovering patterns of influence, teamwork, and knowledge flow within the open-source community. Our hypothesis is that GitHub collaboration is highly modular, with distinct contributor groups forming specialized communities. A small subset of key contributors and automated tools play an outsized role in connecting and maintaining the network.

1.2 Project goals

This project aims to analyze the structural properties of GitHub's collaboration network by:

1. Identifying key communities through network clustering techniques.
2. Quantifying influence using centrality measures to determine the most critical contributors.

[1] All utilized codes are accessible in our publicly available GitHub repository: https://github.com/claireZHL/DSC291-Final-Project.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

3. Evaluating network robustness by analyzing how the removal of key nodes affects connectivity.

By leveraging network analysis methodologies, we seek to provide actionable insights into how contributions, influence, and knowledge dissemination occur within GitHub's ecosystem.

2 Data source & method

To obtain detailed information, we generated a personal access token and developed a data collection pipeline that retrieves influential repositories and their corresponding collaborator information using the GitHub REST API.

One of the primary challenges we encountered was the sheer size of the dataset. While it is theoretically possible to collect data on all repositories and contributors, practical constraints—such as API rate limits and the computational resources available to us—made analyzing the entire GitHub contributor network infeasible. To address this limitation while still identifying influential contributors, we narrowed our focus to prominent open-source top-starred repositories repos, which are likely the "key players" in the network. Specifically, we leveraged an open-source dataset created by EvanLi (nd) 1 that provides a daily updated ranking of the top 100 GitHub repositories based on the number of stars. Since the number of stars reflects a repository's popularity as determined by GitHub users, we reasoned that contributors to these repositories are likely to be influential figures within the platform's ecosystem. By concentrating on these highly starred repositories, we aim to extract meaningful insights while remaining within practical computational limits.

	rank	item	repo_name	stars	forks	language	repo_url	username	issues	last_commit	description
	1	top-100-stars	freeCodeCamp	410695	39079	TypeScript	https://github.com/freeCodeCamp/freeCodeCamp	freeCodeCamp	185	2025-03-03T20:27:00Z	freeCodeCamp.org's open-source codebase and cu...
	2	top-100-stars	free-programming-books	351574	62907	HTML	https://github.com/EbookFoundation/free-progra...	EbookFoundation	31	2025-02-14T02:09:41Z	:books: Freely available programming books
	3	top-100-stars	awesome	349652	28636	NaN	https://github.com/sindresorhus/awesome	sindresorhus	13	2025-02-28T04:32:36Z	😎 Awesome lists about all kinds of interesting...
	4	top-100-stars	build-your-own-x	346773	32150	Markdown	https://github.com/codecrafters-io/build-your-...	codecrafters-io	192	2024-09-03T14:39:35Z	Master programming by recreating your favorite...
	5	top-100-stars	public-apis	328910	34879	Python	https://github.com/public-apis/public-apis	public-apis	51	2024-10-31T19:50:02Z	A collective list of free APIs

Figure 1: EvanLi's Top 100 Stars GitHub Repository Ranking by EvanLi (nd)

With this narrowed scope, we focused on retrieving all contributors associated with each repository in the Top 100 Stars Repository Ranking dataset. We used the ranking data captured on March 4th of 2025 and used that as the basis for our data collection. For each selected repository, we collected contributor information, specifically their ID and the number of contributions made by each user. Ultimately, this process yielded a dataset containing 27,291 entries, where each row represents a unique contributor and their respective contribution count across the top 100 starred repositories (2).

	freeCodeCamp	free-programming-books	awesome	build-your-own-x	public-apis	coding-interview-university	developer-roadmap	system-design-primer	996.ICU	awesome-python	...	django	app-ideas	bitcoin	ui	sve
contributors																
007gzs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	6.0	0.0	0.0	0.0	
00dani	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
0130w	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
0525hhgus	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
0532	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
zz85	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
zzen	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
zzndb	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
zzzaries	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
zzzgydi	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	

27291 rows × 100 columns

Figure 2: Contributor-Repository Dataset Obtained

3 Network representation & structural analysis

3.1 The repository - contributor bipartite graph

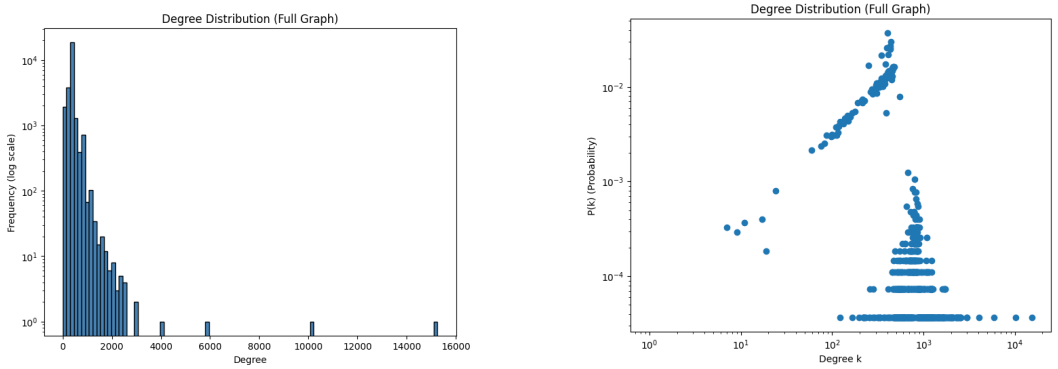
Our initial network was created using the `NetworkX` package in Jupyter Notebook. We first constructed a bipartite graph, where the top 100 starred repositories formed one set of 100 nodes, and the contributors to these repositories formed another set of 27,291 nodes. A link was established if a contributor had contributed to a given repository. Following this rule, a total of 29,715 edges were formed between contributors and repositories.

We observe that the number of edges exceeds the number of contributors, indicating that some contributors have contributed to multiple top 100 starred repositories. This finding suggests the presence of interesting connectivity patterns between contributors, as it highlights the potential for complex collaboration structures within the network. However, due to the large number of nodes and edges, we encountered challenges in effectively visualizing the graph.

3.2 The contributors collaboration network - full data

To better understand the relationships between contributors in the top 100 starred repositories, we projected the Repository-Contributor Bipartite Graph onto a Contributor Collaboration Network, where edges represent collaborations on at least one of these repositories. As a result, the Contributor Collaboration Network consists of 27,291 nodes, each representing a contributor, and 5,402,044 edges, each indicating a collaboration between contributors.

After constructing the network, we conducted a structural analysis. Due to the network's size, computing global metrics like average shortest path length and clustering coefficient was infeasible (`NetworkX` could not compute them in a reasonable time). We, therefore, report only the average degree and degree distribution for the full network and perform a deeper analysis on a filtered subnetwork. Nevertheless, we were able to compute other measures. Using functions from the `NetworkX` package, we found that the average degree of the full network is approximately 396. Additionally, the degree distribution is presented in Figure 3.



(a) Degree Frequency (Full Graph)

(b) Degree Distribution (Full Graph)

Figure 3: Degree Distribution for Full Collaboration Graph

Both figures depicting the degree distribution of the full collaboration graph indicate that the graph is highly connected and does not follow the Poisson distribution typically observed in the Erdős–Rényi (ER) random graph model. This is expected, as the nodes we extracted represent contributors who are assumed to be more actively involved than others on the platform. These contributors work on highly rated projects with a significant number of stars and strong influence.

The full network's degree distribution is heavy-tailed (a few nodes have extremely high degree), which is expected in a collaboration network focusing on top contributors. However, the tail decays faster (exponent ~ 4) than a typical scale-free network, meaning the dominance of the largest hubs is somewhat less extreme than in, say, an ideal Barabási–Albert model.

3.3 The contributors collaboration network - top 10 percent data

Because computing certain metrics (like shortest paths or betweenness centrality) on the full 27k-node network was computationally intensive, we restricted the analysis to the top 10% of highest-degree contributors. This yields a subgraph of 3,104 nodes that presumably contain the most influential actors, allowing more tractable analysis of metrics like path length and clustering. This approach reduces the data size while preserving the most influential contributors, enabling more efficient analysis of their connections on GitHub. Given these benefits, we will use the filtered contributor collaboration data for the remainder of our analysis.

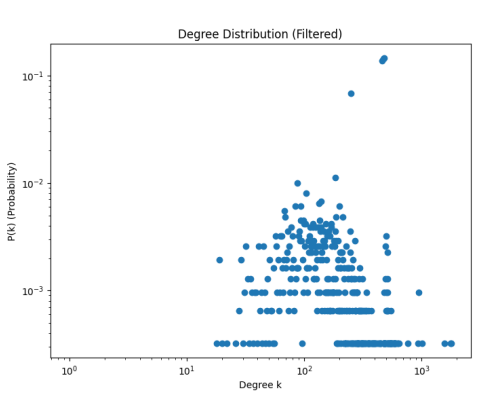
3.4 Structural analysis

The analysis of the filtered contributor collaboration network reveals a highly interconnected structure characterized by a short characteristic path length ($L_t = 2.07$), a high average degree ($< k >= 305.76$) (see Figure 4a), and a significant global clustering coefficient ($< C >= 0.805$), and many nodes have a clustering of 1.0 (complete clique among their neighbors, see Figure 4b) as seen in the distribution, indicating contributors often form fully connected teams on projects.

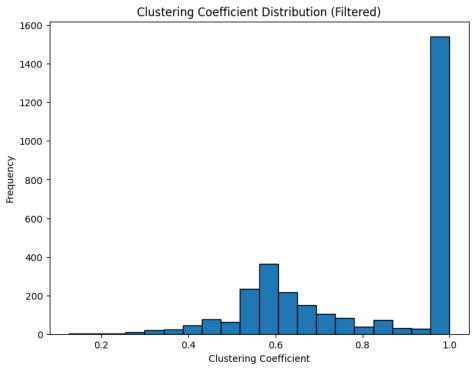
The short average path length indicates that most contributors can be reached within approximately two steps, suggesting a small-world effect, where information or collaboration can spread rapidly across the network. The high average degree implies that contributors are extensively connected, with each node, on average, interacting with over 300 others, which is significantly denser than many real-world collaboration networks.

Additionally, the high clustering coefficient ($< C >= 0.805$) suggests that contributors tend to form tightly knit groups, reflecting localized collaboration patterns within repositories. These properties indicate that the network deviates from a purely scale-free structure, as the steep degree distribution exponent ($\alpha = 4.15$) suggests a rapid decay in high-degree nodes, meaning that while hubs exist, they are not as dominant as in traditional preferential attachment models.

This structure is likely influenced by the presence of automated contributors (e.g. bots) and a dense core of highly active users, leading to a network that is neither fully random (Erdős–Rényi) nor strictly scale-free (Barabási-Albert), but rather, though not perfectly resemble, a dense, small-world collaboration network.



(a) Degree Distribution (Filtered Graph)



(b) Clustering Coefficient Distribution (Filtered Graph)

Figure 4: Structural Analysis for Top 10% Degree Filtered Collaboration Network

3.5 Centrality analysis of key contributors

To better understand the structure of the network and identify influential contributors, we examined four centrality measures: Degree Centrality, Betweenness Centrality, Eigenvector Centrality, and PageRank (Table 1, based on filtered 10% network).

Node (Contributor)	Degree Centrality	Betweenness Centrality	Eigenvector Centrality	PageRank
dependabot[bot]*	0.5798	0.1154	–	0.0040
github-actions[bot]*	0.5659	0.1740	–	0.0019
eltociear	0.5027	0.1246	0.0453	0.0029
bharatr21	0.3261	0.0188	0.0545	–
charliermarsh	0.3029	0.0120	0.0544	–
alexiri*	0.3029	0.0120	0.0544	–
untilhamza	0.3029	–	0.0544	–
salmantec	0.3007	0.0243	–	–
0xflotus	0.2469	–	–	0.0009
kant*	0.2063	0.0252	–	–
suravshrestha	–	0.0144	–	–
lex111	–	0.0128	–	–
ttsugriy	–	–	0.0445	–
mgoin	–	–	0.0443	–
Bearnardd	–	–	0.0443	–
parambharat	–	–	0.0443	–
harupy	–	–	0.0443	–
jsoref	–	–	–	0.0012
SimenB	–	–	–	0.0010
styfle	–	–	–	0.0010
Andarist	–	–	–	0.0009
jamesgeorge007	–	–	–	0.0009
karlhorky	–	–	–	0.0009

Table 1: Top contributors ranked by Degree Centrality, Betweenness Centrality, Eigenvector Centrality, and PageRank (filtered 10% network).
"–" indicates that the contributor does not appear in the top 10 for that centrality measure.
"*" after the contributor name indicates that the contributor is a bot.

According to our calculation, dependabot[bot][2] and github-actions[bot][3] emerged as the most prominent nodes, ranking highly across multiple measures. Dependabot[bot] exhibited the highest Degree Centrality (0.5798), indicating that it directly connects to a significant number of other nodes. Similarly, github-actions[bot] had a comparably high Degree Centrality (0.5659) and ranked first in Betweenness Centrality (0.1740), suggesting it plays a key role in facilitating communication between other nodes.

Beyond automated accounts, eltociear was consistently present across different measures, ranking third in Degree Centrality (0.5027) and second in Betweenness Centrality (0.1246), signifying its strong connectivity and role in network cohesion. Bharatr21, charliermarsh, alexiri, and untilhamza were among the top nodes in Eigenvector Centrality, implying their connections to other well-connected nodes, which enhances their overall influence. Interestingly, salmantec and lex111, while not among the highest in Degree Centrality, appeared in Betweenness Centrality, indicating their importance in bridging different parts of the network.

From a PageRank perspective, dependabot[bot] remained dominant (0.0040), reaffirming its influence in terms of overall importance within the network. Eltociear and github-actions[bot] followed, reinforcing their strong roles in connectivity and influence propagation. Other notable contributors, such as jsoref, SimenB, and styfle, surfaced in PageRank despite not ranking highly in other centrality measures, highlighting their relative importance in network-wide collaboration.

[2]dependabot[bot]: Dependabot is an automated tool integrated into GitHub that helps keep dependencies up to date by generating pull requests for version and security updates.GitHub (2024)

[3]github-actions[bot]: This is GitHub's automation tool that enables continuous integration and continuous deployment (CI/CD) workflows. It automates tasks such as testing, building, and deploying code, enhancing development efficiency.GitHub (2024)

4 Advanced analysis

4.1 Community structure and key contributors in GitHub collaboration network

To analyze the structural organization of collaboration within GitHub, we applied Louvain community detection on a filtered subgraph of the network (Figure 5, identifying five distinct communities. The modularity score of 0.7694 indicates a highly modular structure, meaning contributors tend to interact more frequently within their respective communities than across different groups. This suggests that the network is composed of well-defined teams or specialized contributor groups that primarily collaborate within their subgroups.

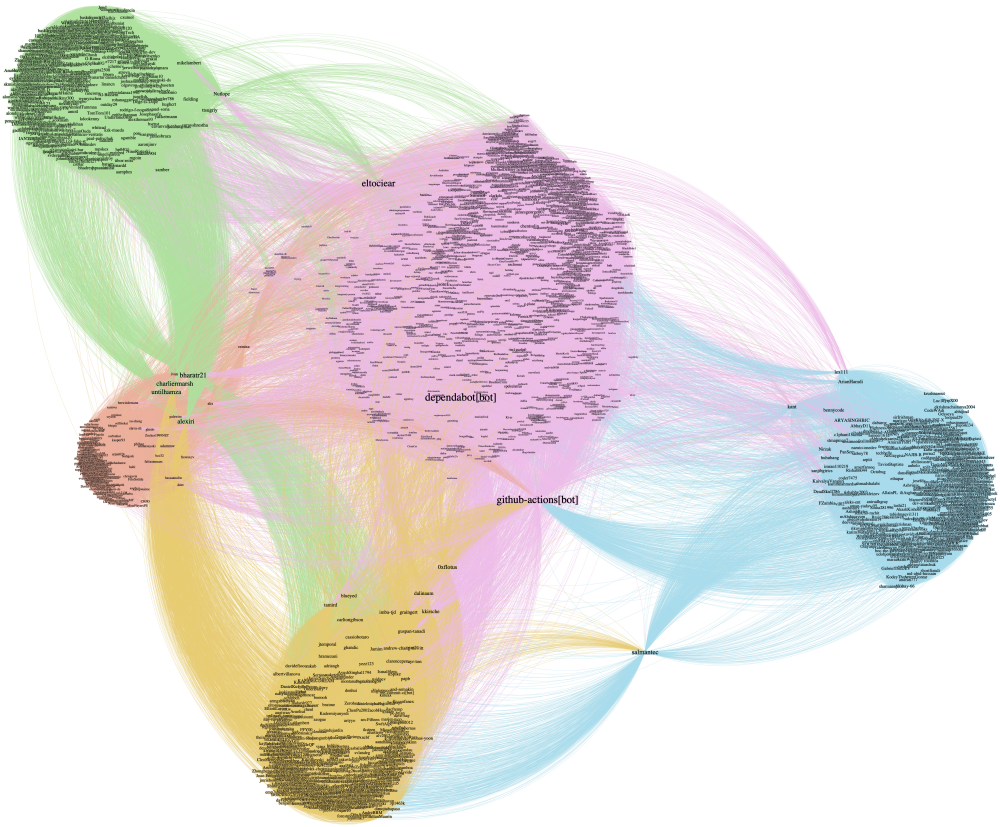


Figure 5: Contributor-Repository Dataset Obtained

Each detected community has a key contributor, identified using degree centrality, which highlights the most well-connected member within that cluster. The most central contributors in each community are salmantec (Community 0), dependabot[bot] (Community 1), bharatr21 (Community 2), 0xflotus (Community 3), and ivan (Community 4). Among them, dependabot[bot] exhibits the highest degree centrality (0.5798), reflecting its critical role in automating dependency management across multiple repositories. The presence of automated tools such as dependabot[bot] as a top contributor suggests that automation plays a crucial role in shaping GitHub collaboration dynamics. Meanwhile, human contributors such as salmantec, bharatr21, and 0xflotus likely serve as core developers or maintainers within their respective communities.

In addition to intra-community influence, we examined betweenness centrality to identify bridging nodes, or contributors who facilitate interactions between different communities. The top cross-community influencers include github-actions[bot] (0.1820), eltociear (0.1269), dependabot[bot] (0.1068), kant (0.0294), and salmantec (0.0278). The high betweenness centrality of github-actions[bot] and dependabot[bot] underscores the role of automation in maintaining inter-community connectivity, ensuring that updates, pull requests, and CI/CD workflows integrate smoothly across

multiple repositories. Additionally, human contributors such as eltociear and salmantec act as important bridges between different teams, suggesting that they may serve as knowledge disseminators or contributors across multiple projects.

Interestingly, kant, a detected bridging node, is not a human contributor but rather a ROS 2 tool for managing PDDL-based knowledge in Python and C++. It is based on several software design patterns, including DTO (Data Transfer Object), DAO (Data Access Object), and Factory patternsULERoboticsGroup (2025). This highlights the increasing role of specialized tools in facilitating inter-community collaboration, particularly in projects that rely on robotics and AI-based planning frameworks.

Based on our findings, we think that the top 100-starred GitHub repository collaboration network is highly modular, where teams or repositories form distinct communities with limited cross-group interaction. The central role of automation in both intra- and inter-community collaboration suggests that tools like dependabot[bot], github-actions[bot], and kant are essential for project maintenance and integration across repositories. Again, we acknowledge such a community behavior might be specific to the dataset we use.

4.2 Network robustness analysis: impact of node removal

To evaluate the robustness of the GitHub collaboration network, we conducted a node removal simulation using three different strategies: degree-based targeted removal, betweenness-based targeted removal, and random node failure. The goal of this analysis was to assess the network's resilience by measuring how the largest connected component (LCC)—the most significant remaining subgraph—shrinks as key nodes are removed. The results of this experiment, visualized in the Figure 6, provide insights into the structural vulnerabilities of the network and the roles of key contributors.

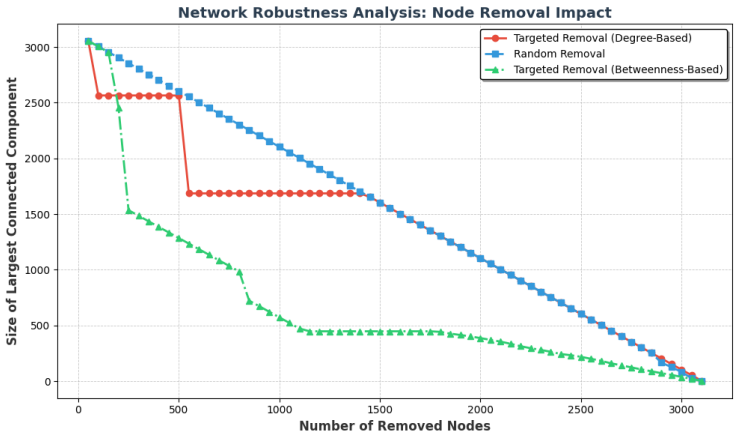


Figure 6: Robustness Analysis Results

The degree-based removal strategy (red line) prioritizes the elimination of the most highly connected nodes, meaning contributors with the greatest number of direct connections. As seen in the graph, this method initially leads to a moderate decline in the LCC size. However, after approximately 500 nodes are removed, the network undergoes a sharp collapse, indicating that it is highly dependent on a small number of influential contributors to maintain its structure. The delayed but abrupt drop suggests that the network has some initial resilience but reaches a critical threshold where the loss of major hubs severely disrupts connectivity.

The betweenness-based removal strategy (green line) follows a different pattern, targeting nodes that serve as bridges between communities rather than those with the highest number of direct connections. This method has an immediate and aggressive impact on fragmentation, as removing these bridging nodes quickly disconnects different sections of the network. Unlike degree-based removal, where the network retains some structure until a tipping point, removing high-betweenness nodes results in gradual but consistent fragmentation from the very beginning. This highlights the importance of key intermediaries in maintaining the overall network structure, even if they are not the most directly connected nodes.

In contrast, random node removal (blue line) exhibits a more gradual decline in network connectivity. Because nodes are removed in an unstructured manner, the LCC size decreases smoothly and slowly compared to the targeted strategies. This suggests that the network is highly resilient to random failures, as many removed nodes are likely to be less influential. This behavior aligns with findings from scale-free network theory, where highly connected nodes (hubs) are relatively rare, and most removals do not significantly disrupt the system unless an extremely large number of nodes are eliminated.

Although the comparison of these three strategies highlights key vulnerabilities in GitHub's collaboration ecosystem, it is important to note that our dataset consists of contributors from top-starred repositories. While the simulation demonstrates that the network is highly dependent on a small subset of highly connected contributors and bridge nodes—making it susceptible to targeted attacks or the departure of key contributors—this finding must be considered within the context of the dataset's scope. Given the tremendous size of the overall GitHub network, the presence of alternative connectivity through other collaborations, and the continuous influx of new contributors, the departure of key contributors, unless occurring at an extreme scale, is unlikely to significantly impact the long-term sustainability and growth of the GitHub community.

5 Findings & conclusion

Our analysis of GitHub's collaboration network, focusing on top-100-starred repositories, reveals several important structural insights about top contributor influence, community structure, and network robustness. One of the most interesting findings is the prominent role of automation in shaping collaboration. Bots such as Dependabot[bot] and GitHub Actions emerge as key contributors, ranking highest in multiple centrality measures. Their widespread presence indicates that automation tools play a fundamental role in maintaining repository management, making them essential components of modern open-source collaboration. Beyond automation, a small number of highly active human contributors, including eltociear and salmantec, hold substantial influence in sustaining connectivity within the network.

Examining the community structure of GitHub's contributor network, we find that collaboration is highly modular, with contributors forming five distinct clusters. These communities interact primarily within their own groups, with only a few bridging nodes facilitating cross-community collaboration. Notably, contributors with high betweenness centrality, such as GitHub Actions, Dependabot, and eltociear, act as crucial links between different communities. This suggests that a limited number of contributors serve as critical knowledge bridges, helping information flow across otherwise isolated groups.

To assess network robustness, we simulated targeted attacks and random failures, analyzing how the network responds to the removal of key contributors. Our results show that the network is vulnerable to targeted attacks on high-degree nodes and high-betweenness nodes. On the other hand, random node removals have minimal impact, with the network degrading gradually rather than collapsing abruptly. These findings roughly align with scale-free network theory, where networks are naturally resistant to random failures but more fragile when key hubs are targeted.

While our findings highlight vulnerabilities in the network of top-starred repositories, they do not necessarily imply that GitHub as a whole is fragile. The broader GitHub ecosystem benefits from its tremendous size, the presence of alternative collaboration pathways, and the continuous influx of new contributors, all of which help mitigate the effects of key player departures.

References

EvanLi (n.d.). Github ranking: Top 100 stars. Accessed: March 04, 2025.

GitHub (2024). *Automating Dependabot with GitHub Actions*. Accessed: March 17, 2025.

ULERoboticsGroup (2025). Kant: A ros 2 tool for managing pddl-based knowledge. Accessed: March 17, 2025.