An Analysis of top GitHub Repositories, their Contributors, and the Languages Used

Justin Gray (A00426753)
justin.gray1@smu.ca

CSCI 6878: Complex Networks
Dr. Somayeh Kafaie

December 7, 2022

An Analysis of top GitHub Repositories, their Contributors, and the Languages Used

By Justin Gray

Abstract

This experiment samples GitHub to create a bipartite graph between contributors and repositories. The report verifies some results from a previous study [1], like how the number of contributors or stars of a repository follows the power-law distribution. Using Cytoscape for network visualization, we observe that open source superstars are most likely to contribute to a single repository. And when users contribute to multiple repositories they are more likely to work on the same programming language or area of expertise.

Contents

1 Introduction

GitHub is a collaborative tool that helps programmers from around the globe work together on software, coding projects, and computer files of all kinds. Git already has an excellent implementation of maintaining previous versions and edit history of text files, but GitHub extends these concepts with in-depth analytics and social features. Each user can create multiple *repositories*, each containing an organized collection of files. If another user finds the repository particularly interesting, they may choose to *star* (bookmark or favourite) it for later. One significant benefit of open source software is allowing other developers to *contribute* to the software; any improvements made by an individual can benefit every user of the software.

One notable study [1] collected and analyzed GitHub's logs over a fixed period of time to create various complex networks. Among other findings, they concluded that the number of contributors per project followed a power-law distribution. A figure from that paper is included as Figure 1 to show that the number of contributors per repository follows a power-law distribution.

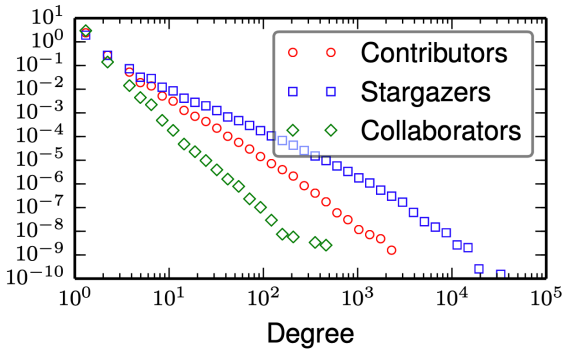


Figure 1: From [1], the number of contributors & stargazers per repository follows a power-law distribution.

This study had significant breadth, and it made conclusions about social reciprocity, the number of contributors and watchers of repositories, how users utilized the forking feature, geographic location, and the num-

ber of followers of different users. This study may have been limited by how the data was collected: instead of directly collecting information about repositories and users, they relied on event logs. There were 18 categories of events to describe everything from "PushEvent" to "WatchEvent" and "ReleaseEvent". Because the logs were only collected over a period of 18 months, any part of the network which was last modified prior to March 11, 2012 would be missing from their analysis. The true topology of the GitHub social & collaborative network is defined by *all* events through the service's lifetime.

This report will focus on repositories, who contributes to them, and which programming languages are used. Firstly, the results from [1] will be confirmed by showing the number of contributors per repository follows the power-law distribution. Next, several questions can be investigated since the topology of the network has been reproduced: what role do open source superstars play, do some repositories share contributors between them, and which programming languages involve the most collaboration? In Section 2 the methodology for this experiment is outlined, in Section 3 the results from the experiments are presented, in Section 4 some insights are given for the results as well as some areas for improvement, and finally in Section 5 the report is summarized.

2 Methodology

The network will be constructed as a bipartite graph with undirected edges between repositories and users, weighted by the number of commits each user has made to the repository. Users are simply identified by their email address used in the commit. Edges need only the number of commits a user made to the repository. And repositories can be identified according to their GitHub URL: owner of the repository, slash, the name of the repository. In addition to owner and name of the repository, each repository also has a star count, and a distribution describing how frequently each programming language is used in the source code.

The first step is to find a list of repositories we want to survey. There are millions of repositories on the platform, so only a small subset of them can be included in our list. A main goal of this research is to find how users collaborate across projects, so this must be taken into account when choosing a sampling method. Random sampling would give the best results, but since most repositories are not updated frequently, or have very few contributors, the sample size would need to be enormous to get interesting results. So instead of random sampling, highly starred repositories are selected from GitStar [2]. GitStar maintains a list of GitHub repositories which have the most stars. Stars are an indication of popularity, interest, and ultimately contributions, and is therefore a suitable index over the repositories. The top 978 most starred repositories were scraped off of GitStar using an automated Python script.

At this point we have the relative ranking of the repository, its owner, name, and number of stars. The next step is to collect the languages used in the source code. Using a personal GitHub API token, the GitHub API is queried once per repository to find the programming languages it uses. The response is saved to a database.

The tricky part is getting the list of contributors for each repository. Unfortunately, the GitHub API will only give a list of the top 100 contributors, but many repositories (especially those indexed by GitStar for having a lot of stars) have thousands of contributors. The chosen solution is to clone the repository onto the local disk, and then run a git log command to retrieve *all* the contributors of a repository. The selected repositories can be arbitrarily large, so the cloned content may take a lot of network and disk resources. A Python script was written to manage and optimize the cloning scripts for the 978 selected repositories. These clones took many hours to complete and used over 500 GB of disk space. A total of 347,255 unique contributors were found through this process, and 466,805 repository-contributor connections of various weights.

A combination of custom Python scripts and Cytoscape [3] visualizations were used to analyze the network. To confirm various aspects of this

network follow the power-law (like in [1]), a cumulative distribution function was written to prepare some dataset for plotting as a cumulative degree distribution. This function works by counting the number of elements in the dataset greater than or equal to each unique element, and then normalizes the distribution into a probability. Finally this function is plotted on a logarithmic scale (x and y axis).

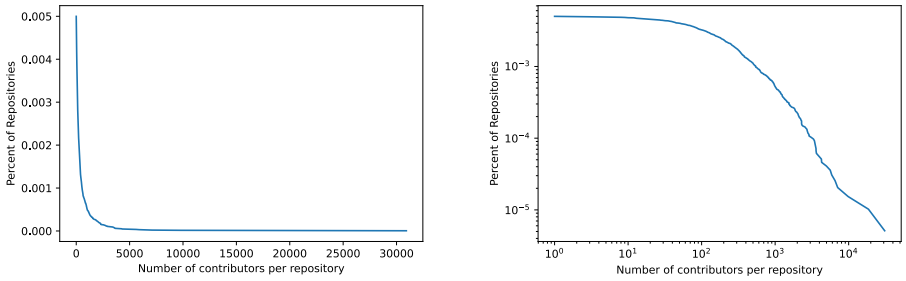
Next, the open source superstars are identified as users who have made at least 1,000 commits to any of our sampled repositories. After throwing out repositories which no longer had any contributors, there were 1,604 contributors and 738 repositories with 5,252 connections. This network was imported into Cytoscape for visualization.

Finally, the bipartite graph was projected onto repositories so that two repositories are connected if they share some common contributors. The number of common contributors is given as the edge weight between two repositories. Edges between repositories were discarded if they had less than 25 common contributors. Each repository is assigned a primary language as the programming language used most frequently in the source code. This network will be imported into Cytoscape, and then a language-based analysis will be performed.

All source code and implementations are openly available at this project's GitHub page: `https://github.com/just1ngray/CSCI6878`. Detailed step-by-step instructions are also given for reproducing the results of the experiment.

3 Results

The surveyed network has 978 repositories, 347,255 contributors, and 466,805 edges representing the number of times a given user has contributed to a repository. Because of its large size, it is impossible to display in Cytoscape without applying some filters. Nevertheless, it is possible to run some network statistics manually.



(a) Linear scale

(b) Logarithmic scale

Figure 2: Cumulative degree distribution for repositories

Figure 2 shows the cumulative degree distribution plot for repositories; equivalently the number of contributors on each repository. On a linear scale as in Figure 2a it appears to follow a power-law distribution, but when plot on the typical logarithmic scale in Figure 2b it is far less consistent. A characteristic power-law distribution should be a straight line in this plot, but there is only a straight line beyond approximately 500. The source of this behaviour will be hypothesized in Section 4.

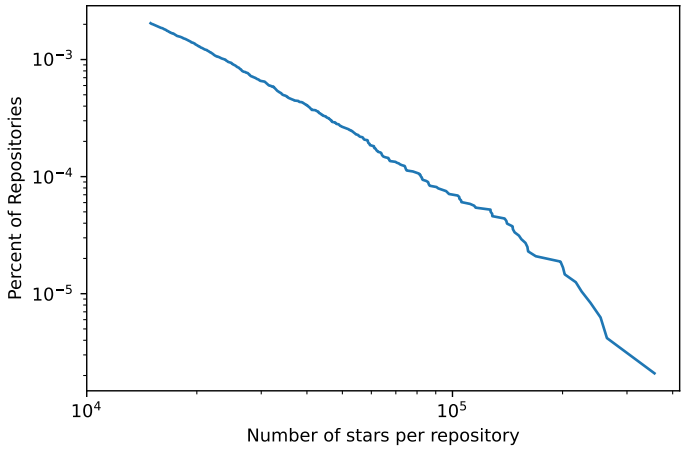


Figure 3: Number of stars by repository

By contrast, the number of stars per repository shown in Figure 3 follows the power-law distribution closely. The reference paper [1] also creates the stargazers graph which represents the number of users who have starred a particular repository. This is included as Figure 1, and it also follows the power-law distribution.

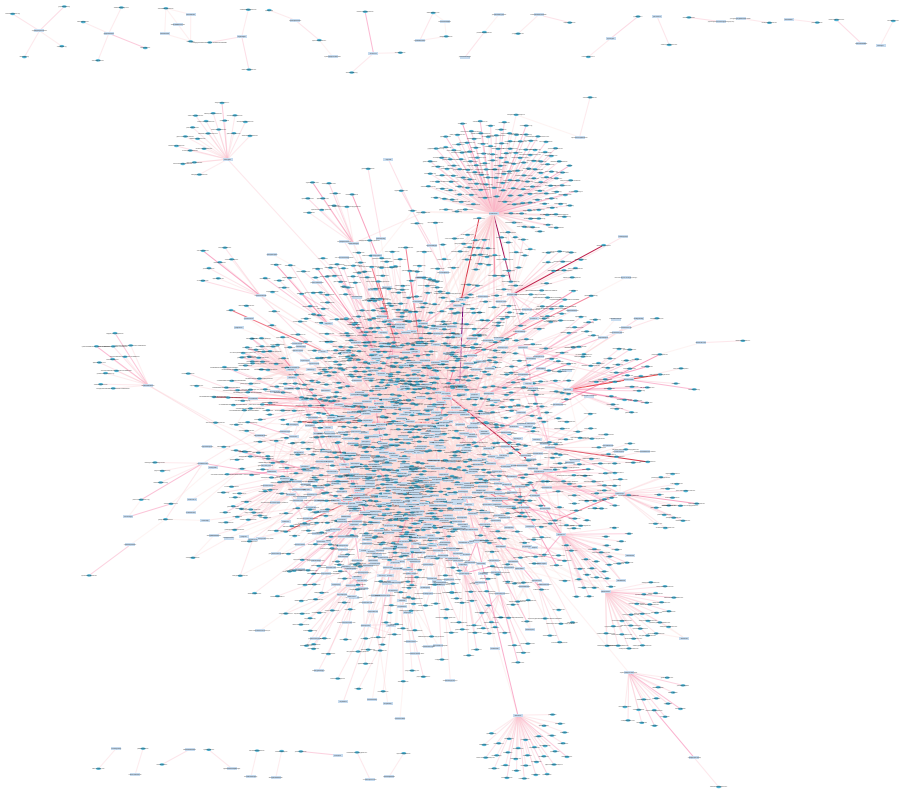


Figure 4: Open source superstars who committed at least 1,000 times to the selected repositories

Next, the open source superstars are identified as any user who has committed a total of at least 1,000 times to any of the selected repositories. After removing repositories without any superstars there are: 738 repositories, 1,604 open source superstars, and 5,252 connections between them.

The edges meeting filtering criteria are left unmodified and still represent the number of commits a particular user has pushed to a given repository. This network is displayed in Figure 4. Users are darker ovals, and repositories are lighter squares. The more commits an edge represents the more significant the line. One observation from this network is it seems to be organized into communities of frequent contributors who are mainly focused on developing one repository. Some of the largest and most well distinguished communities are found on the "outside" of the main component, and include `oracle/graal`, `torvalds/linux`, `JetBrains/kotlin`, `rapid7/metasploit-framework`, and `odoo/odoo`. And through non-scientific analysis of these particular communities, it appears the user's email address is often linked to the repository they commit to – e.g., many superstars of the `JetBrains/kotlin` repository have JetBrains email addresses.

To simplify things, this bipartite graph is projected onto its repositories. This means that edges now represent the number of contributors two repositories have in common. Of course, there are still 978 repositories so there are exactly 978 nodes in this projection. Edges have been filtered out of there are fewer than 25 common contributors between the two repositories. The resulting graph has 978 nodes and 1,170 edges. As shown in Figure 5, this graph still follows the power-law distribution.

Of the 978 repositories, only 358 had at least 25 common contributors with at least one other repository. We focus on the subgraph of 358 nodes and 1,170 edges. The repository nodes are sized according to their degree, and coloured according to their most used programming language. Some languages receive the same colour such as JavaScript and TypeScript, but this is because these languages are essentially the same. As observed in Figure 6 it appears that there is the most collaboration between repositories with the same primary language. JavaScript, Python, and Go appear to be the most prevalent languages in this network.

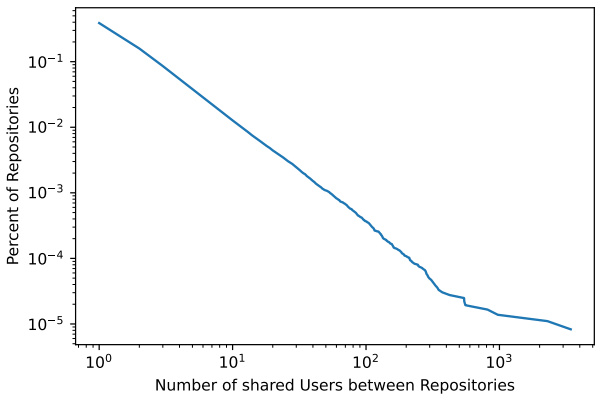


Figure 5: Cumulative degree distribution plot for the number of common contributors between repositories

4 Discussion

Figure 2b represents the number of contributors for each repository in the original network without any filtering. To satisfy the power-law distribution, the cumulative degree distribution must be close to a straight line on a logarithmic scale after some minimum degree. Usually this minimum degree is quite small, but here it appears to be closer to 500. This could be caused by the sampling methodology: repositories were selected according to their number of stars. The number of stars and the number of contributors are positively correlated as shown in Figure 7. The sampled repositories all had a lot of stars, so it is highly likely that they also had a lot of contributors. To fix this issue, either more repositories would need to be sampled, or a different approach all together should be taken.

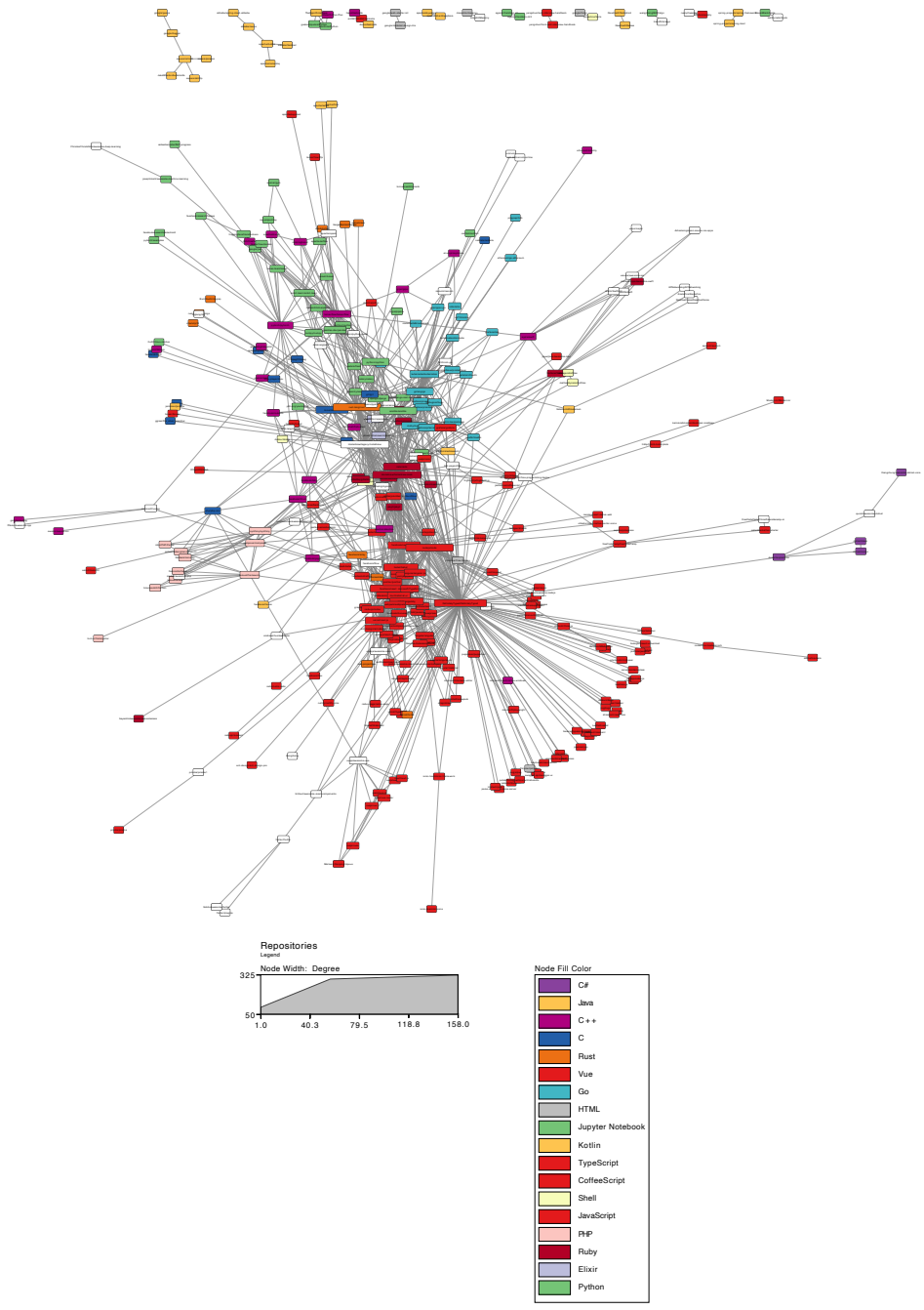


Figure 6: Projection network of repositories with at least 25 common contributors

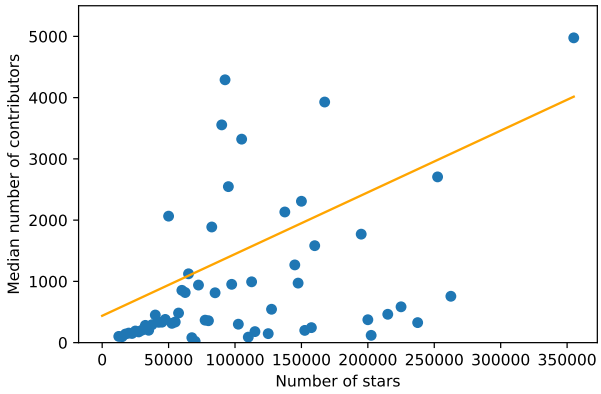


Figure 7: Positive correlation between contributors and stars

In contrast, Figure 3 *firmly* follows the power-law distribution. This is very easy to explain since we have surveyed the *entire* high-starred population of public GitHub repositories. If we continued to survey repositories according to their number of stars, this plot would likely remain much the same shape, but be extended along the x-axis (representing the number of stars per repository).

The repositories found in Figure 6 appear to be loosely clustered together according to their primary programming language. Many of the JavaScript repositories are connected through the `DefinitelyTyped/DefinitelyTyped` repository: this repository allows plain JavaScript repositories to have opt-in TypeScript types. Library developers contribute to this central repository to keep their types up to date with the current state of their library. Many of the Python repositories are connected to `pytorch/pytorch` or `tensorflow/tensorflow`, both are C++ implementations for various machine learning tasks in Python. Using C++ instead of pure Python improves the speed at which expensive models can be executed. Sometimes when working on a project relying so heavily on another library, it is necessary to contribute improvements to the dependency library. Or perhaps developers contribute to multiple open source repositories when they meet

the necessary qualifications and expertise.

JavaScript and Python are the most commonly used languages [4], so it makes sense that there are so many repositories whose primary language is one of these. However, the Go language is somewhat less popular [4], but despite this developers frequently contribute to multiple Go repositories. When compared against language popularity, Go is disproportionally represented in Figure 6.

Additionally, when inspecting any pair of directly connected repositories it is somewhat likely that either the repository owner or name will be the same. Repository owners can be organizations or companies, which would naturally have a community of developers who are likely to contribute to multiple of the organization's repositories. Alternatively, when the repository name is the same, perhaps this represents a fork. Users can fork a repository if they wish to maintain previous project history, but take ownership over their own version. All contributors of the original repository would be maintained in the forked version, so it makes sense that these forked repositories would share many of the same contributors.

In general, it seems that users are likely to contribute to multiple repositories if they are written in the same language, and if there is some common expertise between them. For example, `Homebrew` is a package manager that has multiple repositories written in different languages; the edges between these repositories are heavily weighted since they share many contributors. They may share many contributors because there is some level of expertise involved in maintaining the `Homebrew` suite. Repositories that are developed in the same language are more likely to be used together, and therefore might need parallel development to create new features.

5 Conclusion

A bipartite network between top GitHub repositories and its contributors has been collected. Like in [1], the number of stars per repository follows

a power-law distribution. The number of contributors per repository also followed the power-law distribution, but with one minor contradiction which was discussed in Section 4. It has been observed that open source superstars most frequently commit to a single repository, and this forms clusters around repositories as shown in Figure 4. When developers contribute to multiple projects, they tend to remain within a particular programming language, or within a specific domain of expertise.

Various improvements could be made to this experiment in future studies. A larger selection of repositories could be scraped from GitStar, or perhaps a random sampling of repositories without any indexing at all. The major limitation to overcome is getting the list of *all* contributors of a given repository; the approach taken in this experiment is very slow and resource expensive, and perhaps there is another way to retrieve the list of contributors faster. Finally, instead of only considering each repository's primary language, further research could consider the entire distribution of programming languages, and analyze how cross-language collaboration works on GitHub.

References

[1] Antonio Lima, Luca Rossi, and Mirco Musolesi. Coding together at scale: Github as a collaborative social network. *Proceedings of the International AAAI Conference on Web and Social Media*, 8(1):295–304, May 2014.

[2] Takashi Kokubun. Gitstar. https://gitstar-ranking.com/repositories, December 2014.

[3] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S. Baliga, Jonathan T. Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res*, 13(11), November 2003.

[4] JetBrains. The state of developer ecosystem 2021. https://www.jetbrains.com/lp/devecosystem-2021/, 2021.