Analysis of Collaboration Network: GitHub

1. Introduction

Github is a leading online open source platform for collaboration across projects. Users can perform actions such as - create repository, contribute to an existing repository, follow users etc there. In this project, we take github network and different sets of problems revolving around it, use network science techniques to solve them and present the results obtained it.

In github lots of users collaborate on different projects. They have different numbers of followers. But we are not sure if having large number of followers make significant impact in terms of number of contributions.There is a chance that the users who are in the same circle or group, who worked on the same repository are not following each other. This means that, following each other in github is not relevant or important.

Many people collaborate across different projects in different domains. These domains form community. These community can be growing or declining. It is important to find these communities so that we can know recent advancements in technologies. As a company or a recruiter, I want to recruit developers on a project. How do I find the best set of applicants from a pool of applicants based on their GitHub profiles? As a developer, I want to find influential projects that I would like to work on. How do I find that? In this project we intend to find answers to these questions using network analysis tools and methods.

2. Works Achieved So Far

2.1 Data Acquisition and Network Development

Github provides all its network information(users, repositories, followers, project collaborators, tags, etc) in API in json format. We developed a parser to parse github api and then load it into the networkx graph data structure. We built the network with 400k nodes and around 600k edges between them.

Following figure is the basic graphical visualization of network done manually using word tool:

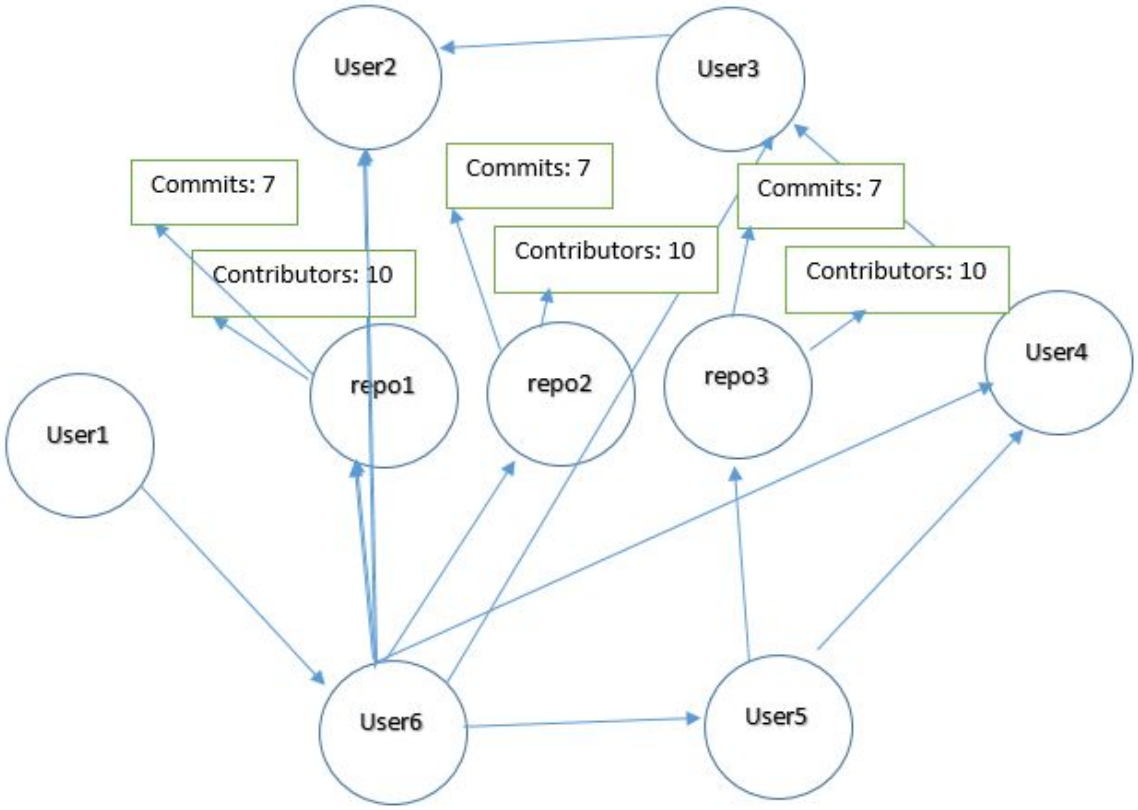


Fig: Small Github repository network Sample

2.2 Basic Network information Collection and Visualization

We started with basic visualization of the network. First we created a graph model in networkx and then converted it to format for Gephi. The visualization done in Gephi is as below:

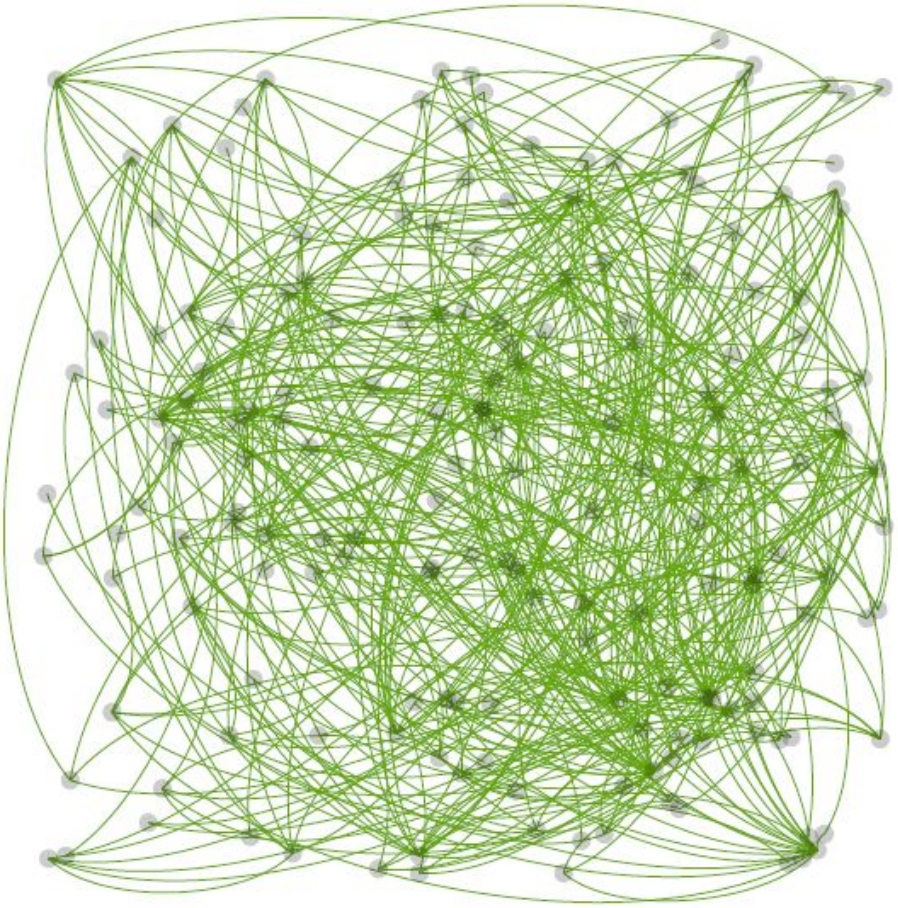


Fig: Github Collaboration Network Visualization (degree > 100)

Here the graph is a network of collaborations across github. We can see grey dots that represent users or projects. Each user collaborates in one or more projects. One user often follows other users and watches projects. Edges here represent either a follow user or a collaboration in a network.

We then continue with other basic analysis of the network. Our network is characterized by the following key features:

2.3 Degree Distribution

In our network, there are 403,378 nodes, whereas there are 578,637 edges. Average Degree of the graph is 1.434. Maximum degree of a graph is 5728 and minimum degree of a graph is 1. Figures below represent the degree distribution across the network.

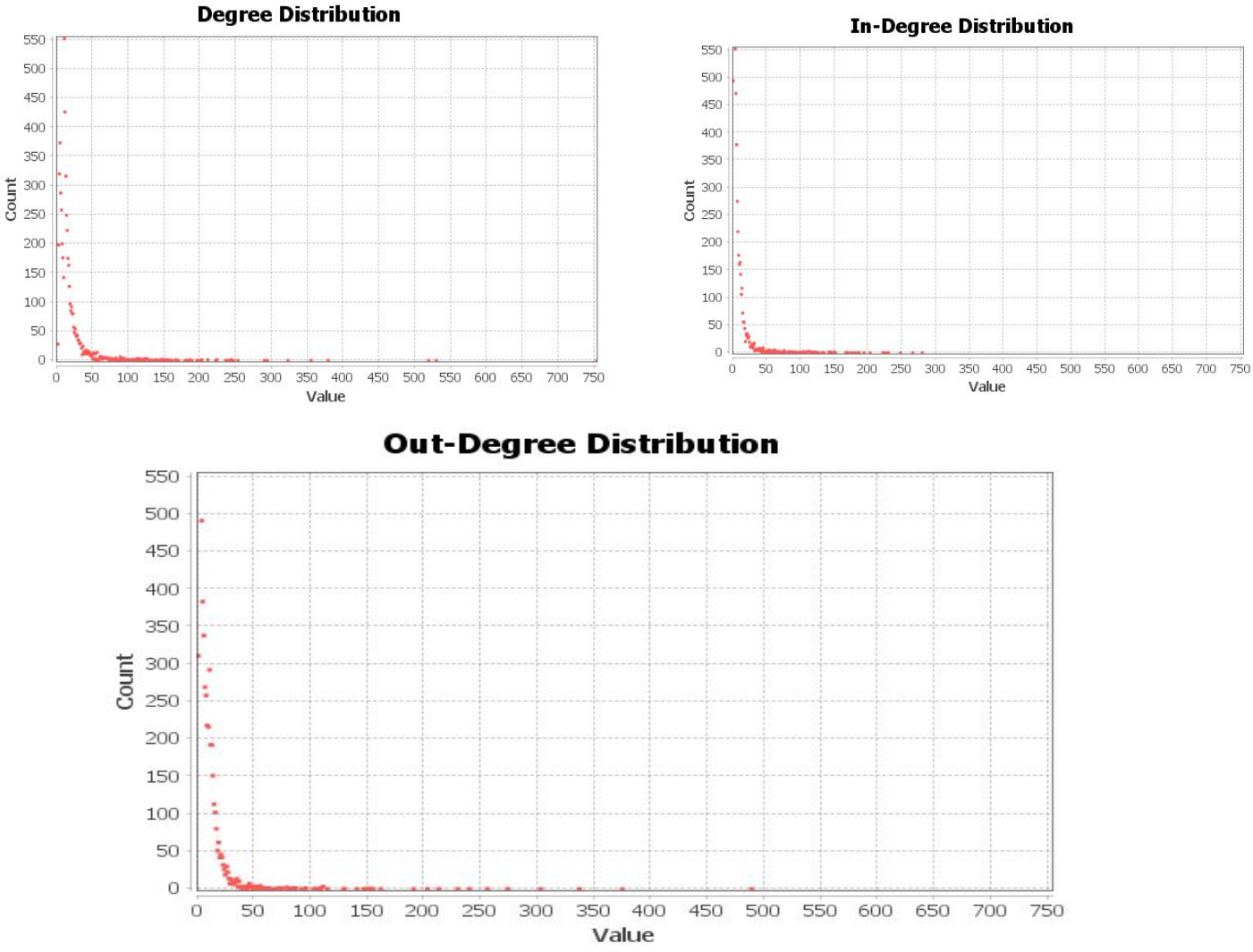
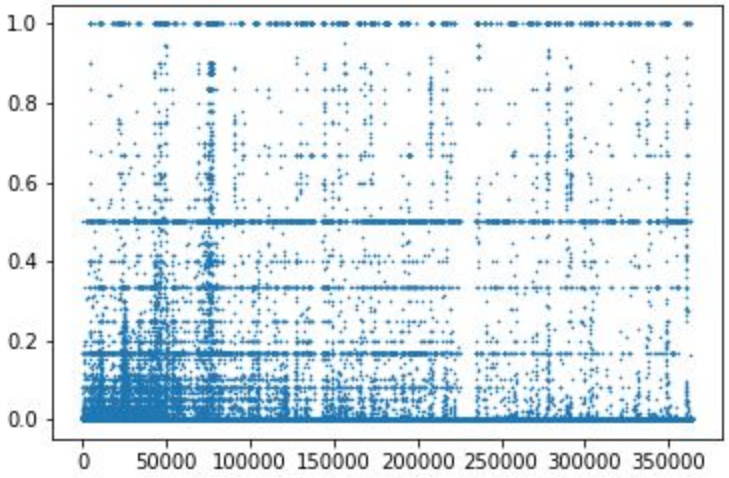


Fig: Degree Distribution Graphs

From the graphs we can see that the degree distribution for github collaboration network follows power law.

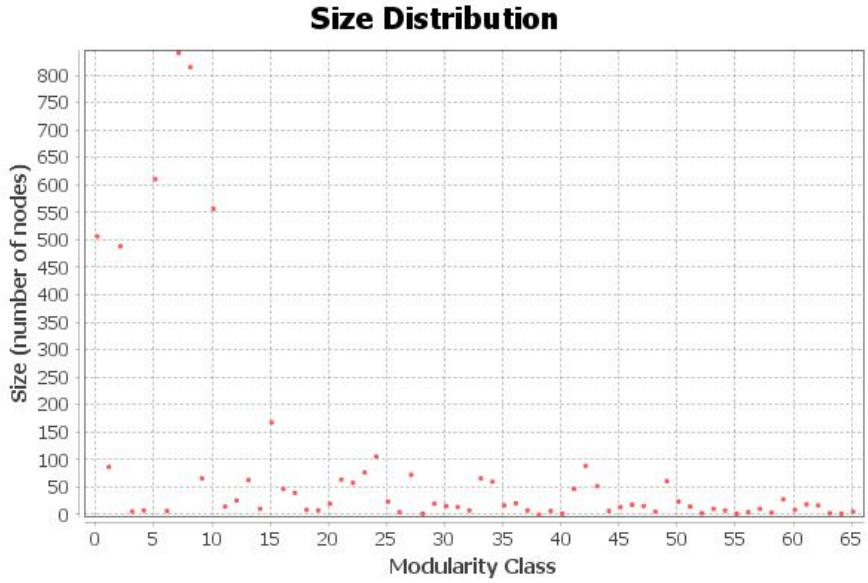
2.4 Clustering Coefficient Distribution



We know that Clustering coefficient of a vertex (node) in a graph quantifies how close its neighbours are to being a clique (complete graph). Figure above shows the distribution of clustering coefficient. From the above picture we can clearly see that the nodes are clearly separated in 4 ranges i.e from 0 - 0.2, 0.2-0.4 and 0.4-0.6 and 0.6+.

2.5 Modularity and Connectedness of the Graph

Modularity is one a concept that provides information about how the communities are formed within social networks. In our analysis of the github network we found over 100 communities. The graph of modularity distribution is shown below.



2.6 Determining popular language by commits counts

Language	Commit Count
c	12899808
python	8731214
javascript	7253045
c++	7100913
java	5308569
ruby	4020526
go	3523440
html	3456242
php	2420438
typescript	2054626

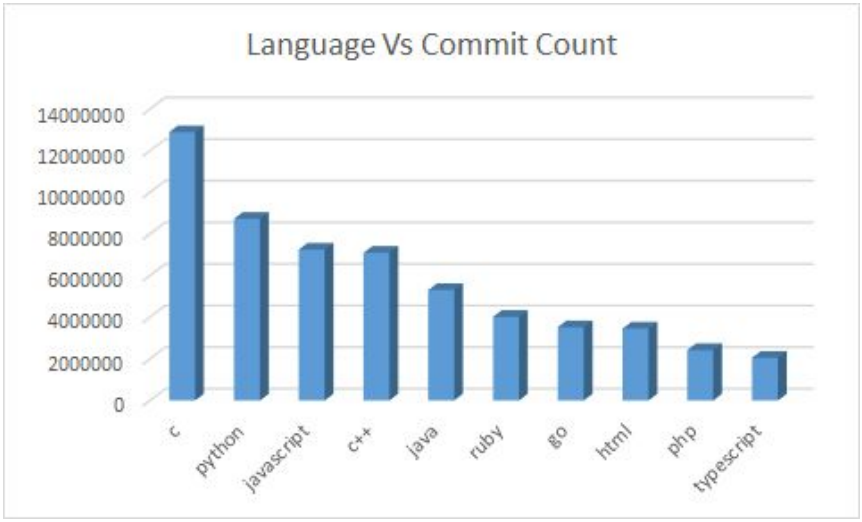


Fig: Commit counts for different languages for different projects

We crawled the network and determined total commits across different projects for different unique language tags. There were around 195 tags in our network, of which we took top 10 language tags based on the total commits made for them across the network. In our observation we found, C was the most used language programming language in our network. Following it was python and javascript.

2.7 Determining popular language by number of projects

Language	Project Count
javascript	22809
python	12457
java	11859
jupyter notebook	9117
html	8005
ruby	6283
c++	5768
c	3919
css	3337
c#	2270

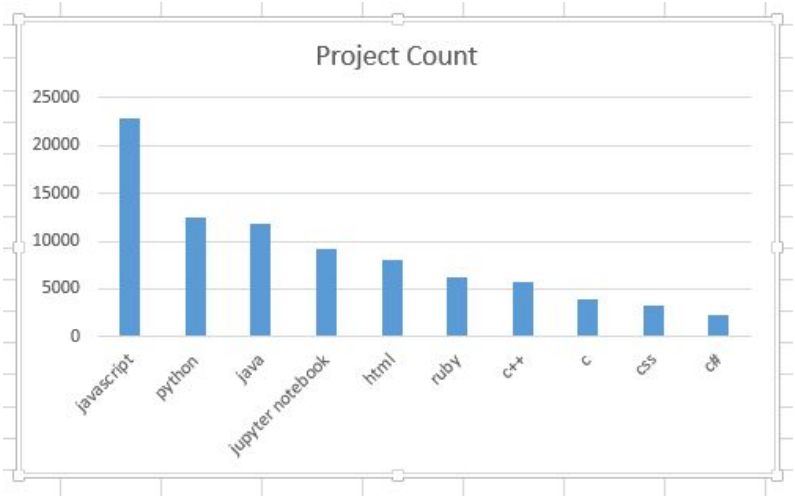


Fig: Number of Projects for different language

From the above table and the picture, we can observe that project javascript is the most popular language with over 20,000 projects in the network, similarly, python stands second in the row with over 10,000 projects.

3. Current Work in Progress

With the beginning of the analysis across network, we are also working on the development of the contribution Index and Similarity Index Calculation. We are working on developing and calculating values for the following 2 indices.

3.1 Contribution Index

As we observe different people contribute across different projects, they actually make commits. These commits can be counted as a measure of their contribution to the project and using these contributions we can actually calculate the contribution Index. We define contribution index to be following value:

Contribution across project = Number of commit made by the user / Total Number of Commits made in the project

Contribution Index for User = $\sum$ (*Number of commit made by the user / Total Number of Commits made in the project) for all projects the user is contributing to*

We completed defining the index. We however are left with calculation of it for all nodes and analyzing the values observed.

3.2 Similarity Index

Users can be related to each other in many ways. Measuring similarity between them can be done by finding similarity across many factors such as: degree similarity, contribution similarity, project tag similarity, common followers similarity, common following similarity and many more. Using these factors we can sum and find Similarity Index between users. This index can be used for prediction. We define the index to be:

Similarity Index = Degree Similarity + Following Similarity + Followers Similarity + Project Tag Similarity + Contribution Similarity

We can also normalize this by taking average of all the number of factors we count for calculation.

4. Next Targets

After we complete calculation of Similarity Index and Contribution Index and perform analysis using these tools developed, we intend to work further towards developing predicting tools for employers and contributors.

5. How to run the code

The main report file is Intermediate Report.pdf. It contains all the relevant information and analysis of the network. Data is contained in github.txt file - this file actually is a python pickle file, which is a dump file of networkx Graph. This can directly be unpickled into a networkx graph.

5.1 Data Generation

To generate data you need to run the following command:

```
>> python parse_github.py
```

This file crawls across github and generates networkx graph which will be used for analysis later. It dumps the data into github.txt file as described previously.

5.2 Basic Network Analysis

Basic graph characteristics can be studied using following python code:

>> python git_analysis.py

This command runs different analysis functions such as clustering coefficient calculation, modularity calculation, minimum degree, maximum degree calculation etc and prints in the terminal.

5.3 Other analysis

For other analysis you need to run the following command:

>> python analysis_github.py

This actually generates intermediate files which are used later for analysis using network tools such as gephi and other graph visualization tools.

7. Tools and Technologies Used

1. Python
2. NetworkX library
3. Gephi
4. Basic Visualizations using, matplotlib, excel etc
5. Crawling using requests library

6. References

1. SAGH: A Social Analysis tool for GitHub, Akash Das Sarma, Ashish Gupta, Jaeho Shin Computer Science Department, Stanford University
2. Coding Together at Scale: GitHub as a Collaborative Social Network, Antonio Lima, Luca Rossi and Mirco Musolesi School of Computer Science University of Birmingham, UK
3. Analyzing the Social Ties and Structure of Contributors in Open Source Software Community, Mohammad Y. Allaho and Wang-Chien Lee, Department of Computer Science and Engineering, The Pennsylvania State University University Park, Pennsylvania 16802

4. Predicting stock market movements using network science: An information theoretic approach, Minjun Kim and Hiroki Sayama Department of Systems Science and Industrial Engineering, Center for Collective Dynamics of Complex Systems, Binghamton University, State University of New York, Binghamton, NY 13902-6000
5. Github API