Social Network Analysis Of Github Users

Jerelyn Preeja Premjit
School of Computer Science and Engineering
Vellore Institute of Technology
Vellore, India
jerelyn.preeja2018@vitstudent.ac.in

Soumya Vishwakarma
School of Computer Science and Engineering
Vellore Institute of Technology
Vellore, India
soumya.vishwakarma2018@vitstudent.ac.in

Lydia Jane G.
Assistant Professor (Senior)
School of Computer Science and Engineering
Vellore Institute of Technology
lydiajane.g@vit.ac.in

***Abstract*—In this paper we will understand the software development process by bridging the gap between source codes and social interactions. In the software development process there are six steps- Planning, analysis, design, implementation, testing and maintenance. Our main focus is on the design and implementation part of this project as these steps require collaboration. There will be a three step analysis. The first step is to analyse an event for their participants and the influential actors in the network. In the second step, we will study the stargazer and forked network generated by a popular open source project and the third part of this paper, we will construct and analyse a follower's network which will further help us identify communities.**

Keywords—github, soc*ial network, starg*azers, followers, communities.

INTRODUCTION

There are many code hosting services available. Names like SVN, Google Cloud Source Repositories, Bitbucket etc are not as popular as GitHub. Being the largest of code hosting services with many open source projects , we can study the links and realise how these are developed at such a large scale since GitHub also provides us with many collaborative tools.
In this project we aim to analyse the patterns of collaboration and work to realise the important aspects of vast collaborations. We will further be comparing the collaborative networks with that of multiple single influential github users to determine who would play the central actor among the two categories. (i) To find the central actor in any large coding event. (ii) To find the core development group in an open source project by the measure of stargazers and follower community. (iii) To study the depth and width of the trees corresponding to forked repositories. (iv) To find the reason why some repositories/projects are popular and why others are not. (v) To compare and contrast the vast network generated by a single influential user, the contributors of an event and that of an open source project to determine which of these would make a more significant contribution(be a central actor) to the network.

A RELATED WORK

[1] This paper describes the unstructured data that is stored on GitHub. It segregates data into 8 event types- CreateEvent, WatchEvent, IssuesEvent, IssueCommentEvent, PullRequestEvent, ForkEvent, PullRequestReviewCommentEvent, CommitCommentEvent. Taking this into consideration, we can conclude that there are 4 types of users- the owners who Create the repositories, the followers who Watch, the takers who Fork, and the contributors. In this paper- the communities and the centrality degrees were detected. Two types of analysis were used for two different regions- the Marseille and the Paris Dataset Analysis to extract the communities based on the follower network and the programming languages used. These two findings were compared.
In the Marseilles Dataset Analysis, the in degree showed the 10 most important repositories in the network and all were about web application; and the out degree shows the most important users.
Modularity optimization used to detect subcommuties detected 41 subcommunities. When they assessed a new dataset (programming language) and added it to the network, they obtained a direct and bipartite network. They detected 11 communities. With the help of the programming languages, it was easier to detect communities. Their biggest community was found to be the JavaScript Community with Sanchez as the most important actor.
In the Paris Network, the most important projects were also about web development. For a bipartite network, there were 278 communities detected. However, when the network was made tripartite, it was reduced to 21 communities, JavaScript being the largest, followed by PHP, C++, C etc.
The top 10 popular languages obtained are almost identical. However, the similarities end there. The Marseilles developers did not know each other and worked on their own repositories and forks were very rare. Contrastingly, the Paris developers were interconnected and worked together and made contributions to worldwide projects. Here, they create more repositories than they fork while the Marseilles fork more than they create.

[2] At first, a thorough and brief introduction was given which is quite similar to the previous paper. Functions like fork, pull and the create are described in detail. In the paper GitHub is represented as a social network and a collaborative platform. It was analysed for a period of 18 months. Events were measured based on the number the users have authored and how the geographical characterization influences the collaboration.
Basic structure analysis and show the distribution of the number of contributors and stargazers per user. Social ties and repository mediated collaboration patterns were analysed. The depth and width of trees regarding the forked repositories. They also investigate the correlation between the activity of users and their popularity in the network and they have observed that active users do not necessarily have a large number of followers.
The degree, in degree and out degree of the social graph exhibit a power law scale with different exponents. The users are assigned a star to a repository as a bipartite graph called the stargazers graph. The network can be generated by the information obtained from the watch information. The contributors are analysed by every push event which also includes the authorship information of the pushed commits.
This project has analysed GitHub data for 18 months from 11th March 2012 to September 11 2013. From the vast information sought from 2.19 million users and 5.68 million repositories. There are 4 networks derived from the dataset- a bipartite network describing the collaboration of users on repositories, a bipartite network describing the contributions of users on repositories and a directed social network describing the following relation.
The distributions of the number of collaborators/project, stargazers/project and user followers show a power law like shape. A very low reciprocity of the social ties which is very different from the other social media networks. Also, the collaborations between users happen on a very small scale and active users do not have a large number of followers. Finally, the impact of geography on the collaboration. Users had the tendency to interact with those who they were geographically close to as it is easier to manage. The paper provides novel insights into the complex data of collaboration on a planetary scale.

[3] This paper presents Archnalyzer, a method and toolset to support the work of architects and software developers in the search and selection of components meeting the best possible functional and nonfunctional requirements using the component-based software engineering (CBSE) strategy. The proposal is based on a repository mining technique.
According to the keyword set under the dataset obtained, we notice that there is a somewhat even distribution of the repositories in the selected three

languages (Java, Python, PHP. We also presented a comparison of the number of commits, stars and forks for each technology grouped by search factor. From our data seems to point out the next two statements: (i) the more followers a repository have, the more likely is that some additional work is done based on that particular repository, and (ii) the number of commits doesn't necessarily correlate with the number of followers and the number of fork for a particular open source project.
The tool has been designed to scale in both number of technologies and number of keyword terms, and has shown to be feasible and effective when supporting the process of search and analysis of software components.
Although the current work was limited to three technologies (java, python and php) our solution was designed to scale to other technologies. Similarly, the analysis was limited to 16 search terms that were believed to be the most representative in current web-based systems.

[4] The paper proposes a novel recommendation system, RepoPal to detect similar repositories on GitHub. A number of prior approaches have been previously proposed to identify similar applications, unfortunately they are not optimal or proper for GitHub. One approach relies only on similarity in API method invocations, while another relies on tags which are not present in GitHub. This system is based on three new heuristics leveraging two data sources (i.e. GitHub star and readme files) which are not considered in prior works.
It evaluated RepoPal and CLAN on a dataset of 1,000 Java repositories and showed that our technique outperforms CLAN by substantial margins in terms of success rate, confidence, and precision. The system is designed on the bases of 3 heuristics :
Heuristic 1: Repositories whose readme files contain similar contents are likely to be similar with one another. Heuristic 2: Repositories starred by users of similar interests are likely to be similar. Heuristic 3: Repositories starred together within a short period of time by the same user are likely to be similar.
Finding similar repositories on GitHub can be helpful for software engineers as it can help them reuse source code, build prototypes, identify alternative implementations, explore related projects, find projects to contribute to, and discover code theft and plagiarism.

B. MINING GITHUB DATA

We collected information about GitHub users/owners of these projects, via GitHub API.[5] Github already has a statistics page for each repo, but that is a bit limited and cannot respond to all of our requirements. So we began exploring all possibilities for retrieving these statistical APIs.

Although the prior work has made significant progress in the identification of similar projects, there are a number of challenges in applying them to detect similar repositories on GitHub. First, GitHub contains millions of repositories that get updated frequently over time and statically analyzing them periodically to retrieve API calls is a task with too much cost. Second, GitHub does not allow users to tag repositories. In addition to the above two main challenges, prior works do not leverage additional data sources specific to GitHub that can provide new insights. For example, GitHub allows users to the star repositories to keep track of the repositories that they find interesting. Starting a repository is a public activity that can be viewed and tracked by others. Moreover, repositories often contain readme files that describe the high-level features of the projects developed in the repositories.

C. EXPERIMENTAL ANALYSIS

PART I - EVENT NETWORK ANALYSIS

A number of prior approaches have been proposed to identify similar repositories, unfortunately they are not optimal or proper for GitHub. One approach relies only on similarity in API method invocations [10], while another relies on tags which are not present in GitHub [11] So, in our case, we have selected our target repository and the owner of the event repository by their GitHub username. During an event, GitHub repositories are observed with a high degree of activity. We can further analyse the network to find the central, most popular participant of that particular event. The 2 events we have chosen to demonstrate the same are : Hacktoberfest and Gsoc.

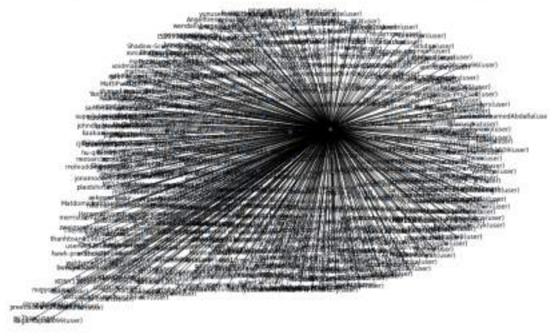

Fig 1 : Stargazer network from the event Hacktoberfest(2019)

Events on github vary in size by 2 factors, popularity and irregular naming standards of repos. The forked network and stargazer's network helps us determine the popularity with respect to participants and level of impact. For the events that do follow a naming standard we will be able to determine the most popular actor. This is done by taking into account users with the maximum number of followers in the graph as centrality measures here wouldn't apply as the graph generated is a star graph. These repo's are only active for a short period of time and don't show a lot of growth in the network.

PART II - OPEN SOURCE PROJECT NETWORK ANALYSIS

A higher degree of activity is also observed in most open source project repositories This paves way for us to construct stargazers networks and forks networks of the chosen open source project repository [14],[15]. We will further analyse the network to identify the core development group of the project repository by the number of commits.

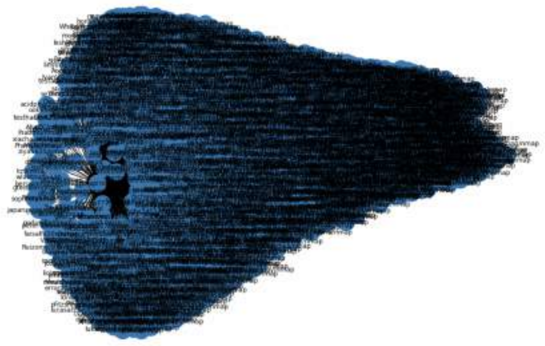

Fig 2 : Forked network for nmap

Detecting similar repositories can be useful for code reuse, rapid prototyping, identifying alternative implementations, exploring related projects, finding projects to contribute to, and discovering code theft and plagiarism (when they are reused inappropriately) [7], [8], [9]. So we have chosen 3 Open Source Projects : SE-Toolkit, nmap, wireshark. Among all the Open Source Projects we can come across, the security oriented projects are the most active as they are regularly updated.
Hence the forked and stargazer graphs constantly change and grow over time and if the project becomes inactive, the respective graph will saturate. By

applying proper engineering techniques, we were able to identify the Core developing group by analysing commits.

PART III - SINGLE USER NETWORK ANALYSIS

We will at last choose a single user and determine the follower network of that particular user. This is done by determining the users followers and then extending the graph is to add the followers of the followers of the user into the same network and then we will subjectively find the central actors using centrality measures.
We will also determine the hub score and the authority score based on the follower network generated. The communities formed here are also discovered.

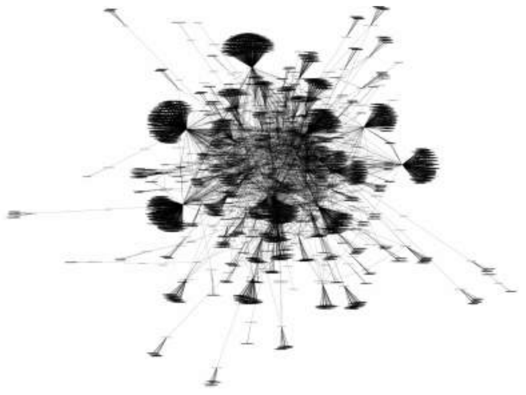

Fig 3 : Communities formed by generating followers of followers network

We infer this to be the most dynamic network of them all as each active user will show a slow and gradual growth over an extended period of time. Single user analysis involves observing a followers network that can help one determine their closest central actor and identify the various communities around them.

It isn't possible to finalize on a central actor when dealing with multiple users as it is determined subjectively, but this will help a person determine the most popular user close to them.

C. ALGORITHM

PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measuring the importance of website pages. PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites. The above centrality measure is not implemented for multi-graphs.
A hub score is the amount of the scaled power upsides of the pages it focuses on to update every hub's center point score to be equivalent to the amount of the position scores of every hub that it focuses on. That is, a hub is given a high center point score by connecting to hubs that are viewed as experts regarding the matter.
In graph theory and network analysis, markers of centrality distinguish the main vertices inside a diagram. Applications incorporate distinguishing the most powerful individual in an informal community, key framework hubs in the Internet or metropolitan organizations, super-spreaders of sickness, and cerebrum organizations.

D. CONCLUSION

This paper proposes a factor of importance in following a common naming strategy for repo's during an event, for doing so will help generate more accurate networks to work and analyse to help deduce more accurate results, say for who the popular actor can be, and the various applications and purposes this information can direct or cater to. The paper highlights the significance of where centrality measures come into play and where it would be necessary to work around the same by taking other factors like followers count into consideration.
This paper also determines why centrality measures for a single follower network is subjective rather than objective as the central actor is based around a single user and is completely dependent upon the single user's connections. This must be taken into account when analysing various communities to make comparative inferences.
.

REFERENCES

[1] Anicet HOUNKPE, "Make Github Community Great Again : Network Analysis With Graph Algorithms" 2018

[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]Analysis of Intercrossed Open-Source Software Repositories Data in GitHub, Gabriel Farah, Departamento de Sistemas y ComputaciónUniversidad de Los AndesBogotá, Colombiag.farah38@uniandes.edu.co, Dario Correal

[4] Detecting Similar Repositories on GitHub Yun Zhang1 , David Lo2 , Pavneet Singh Kochhar2 , Xin Xia1‡ , Quanlai Li3 , and Jianling Sun1 1College of Computer Science and Technology, Zhejiang University, Hangzhou, China 2School of Information Systems, Singapore Management University, Singapore 3 University of California, Berkeley, USA

[5] F. Chatziasimidis and I. Stamelos, "Data collection and analysis of GitHub repositories and users," 2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA), 2015, pp. 1-6, doi: 10.1109/IISA.2015.7388026.

[6] on-line text by Robert A. Hanneman (Department of Sociology, University of California, Riverside) and Mark Riddle (Department of Sociology, University of Northern Colorado), " Introduction to Social Network"

[7] C. Liu, C. Chen, J. Han, and P. S. Yu, "GPLAG: Detection of software plagiarism by program dependence graph analysis," in KDD, pp. 872– 881, 2006.

[8] T. Sager, A. Bernstein, M. Pinzger, and C. Kiefer, "Detecting similar java classes using tree algorithms," in MSR, pp. 65–71, 2006.

[9] C. McMillan, N. Hariri, D. Poshyvanyk, J. Cleland-Huang, and B. Mobasher, "Recommending source code for use in rapid software prototypes," in ICSE, pp. 848–858, 2012.

[10] C. McMillan, M. Grechanik, and D. Poshyvanyk, "Detecting similar software applications," in ICSE, pp. 364–374, 2012.

[11] S. Kawaguchi, P. Garg, M. Matsushita, and K. Inoue, "Mudablue: an automatic categorization system for open source repositories," in APSEC, pp. 184–193, 2004. [7] F. Thung, D. Lo, and L. Jiang, "Detecting similar applications with collaborative tagging," in ICSM, pp. 600–603, 2012.

[12] Mining Software Repositories for Traceability Links (June 2007), pp. 145-154, doi:10.1109/icpc.2007.28 , by Huzefa Kagdi, Jonathan I. Maletic, Bonita Sharif

[13] Nico Zazworka and Christopher Ackermann. 2010. CodeVizard: a tool to aid the analysis of software evolution. In Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '10). ACM, New York, NY,USA

[14] Bird, C.; Pattison, D.; D'Souza, R.; Filkov, V.; and Devanbu, P. 2008. Latent social structure in open source projects. In Proceedings of FSE'08, 24–35. ACM.

[15] Dabbish, L.; Stuart, C.; Tsay, J.; and Herbsleb, J. 2012. Social Coding in GitHub: transparency and collaboration in an open software repository. In Proceedings of CSCW 12, 1277–1286. ACM.