IJICTR
International Journal of Information & Communication Technology Research

Volume 12- Number 2 – 2020 (54 -62)

Structural Analysis of GitLab's Users and Projects Networks

Hadi Safari, Nazanin Sabri, Faraz Shahsavan, Behnam Bahrak*
School of Electrical and Computer Engineering
University of Tehran
Tehran, Iran
{hadi.safari, nazanin.sabri, faraz.shahsavan, bahrak}@ut.ac.ir

Received: 13 August 2019 - Accepted: 28 January 2020

Abstract—**GitHub has long been perceived as the exclusive provider of hosting for software development in the minds of many programmers. However, it is far from the only available service out there. GitLab is one of the many platforms offering similar capabilities, which has experienced rapid growth in recent years. GitLab currently holds the second-largest collection of repositories among its competitors. Despite its rapid growth, little attention has been paid to this website by academia. This lack of information with regard to users and projects on the platform, as well as the fast increase in the number of GitLab users, motivated us to conduct the current study. In this paper, we perform social network analysis on the data we have collected from the public users and repositories of GitLab. We observe that GitLab is similar to other code-hosting services with regard to its network structure. We also find that the most influential users and projects on the website, are associated with the founding team of GitLab. We further analyze the collaboration and membership networks and, among other things, find that both graphs display high values of assortativity with regard to node degree. The relations between various attributes of projects have also been analyzed.**

Keywords-GitLab; social network analysis; hosting services for software development.

I. INTRODUCTION

Many factors have accelerated the growth of distributed version control systems (VCSs). For one, remote collaboration has become common among programmers, especially in the open-source community [1], thus increasing the need for such services. Besides, it has become customary for teams to keep track of changes made to the code base during the development process to ease debugging and to have a log of events for performance assessment.

Nowadays, developers are not the only target audience of these DevOps lifecycle tools, graphic designers, content strategists, and researchers are amongst other professionals using such systems. The ability to create static webpages on some online hosting services (referred to as "Pages" on these systems) has also caused bloggers to gravitate towards these websites [2].

GitLab[1] was established in 2011 by Sijbrandij and Zaporozhets. The website is primarily known as a software hosting platform using git as its version control system. Still, like many of its competitors, it offers several other services such as code review, continuous integration and deployment (CI/CD), and creation of wikis. In recent years, GitLab has seen considerable growth in popularity and is now ranked second among online hosting services, following GitHub. After the acquisition of GitHub by Microsoft in October 2019, the surge of users to GitLab made the

* Corresponding Author
[1] https://gitlab.com

news [3]. The rapid increase in repository counts was also visible in the data collection process of this study.

The importance of the analysis of such platforms stems from the significant role they play in the advances made in the computer science community. Although most notable projects consist of a core team dedicated to their development, these code hosting platforms are providing them with a lot of help and additional human resources along the way. Thus, understanding how users interact with one another on these code hosting platforms, as well as getting some insights into how they collaborate could help us in the understanding of the developments made in the industry today.

In this paper, we perform social network analysis on GitLab's networks of users and projects, investigating structural properties of those networks as well as finding key users and projects, and their characteristics, to better understand network attributes of GitLab. We start by creating the bi-partite project-user graphs with edges indicating membership and collaborations. We then create mono-partite projections of the networks. Analyzing the resulting graphs, we find out that both membership and contribution graphs are assortative with respect to degree. We further find that most members tend to make no contributions to the projects they are part of. Degree distributions, community and clique structures are also analyzed. Our contribution consists of extracting and analyzing the attributes of Gitlab's networks and comparing these attributes with those of some of its competitors, such as GitHub and SourceForge, based on some metrics which are common in literature. Furthermore, we publish a dataset of the aforementioned networks and find some bugs in GitLab, as by-products of this research. To the best of our knowledge, this dataset is the first public dataset of Gitlab's users and repositories, encompassing a huge portion of the data of the platform at the time of collection. A previous version of this paper appeared at the 10th International Symposium on Telecommunications (IST'2020) [4].

The rest of this paper is structured as follows: we first offer a brief explanation of related work in Section II. In Section III, we elaborate on the data collection and analysis methods used. We then present the results in Section IV, and conclude the paper with a summary of the study and possible ways of continuing this research in Section V.

II. Related Work

GitHub, GitLab, Bitbucket, Source-Forge, and Launchpad are the five most popular online code hosting platforms [5]. Google Code, before discontinuing its services in 2016, was also a well-known service [6].

Considerable research has been done on code hosting platforms. The area of focus in these papers includes analysis of the code published on these services, analysis of the version control systems from a software development perspective, or looking at these platforms from a social network perspective and analyzing user relationships and network properties. Older studies were mainly focused on Source-Forge, while newer research is mostly done on GitHub.

Thus far the primary focus of research on GitLab has been centered on the academic use cases of the website [7, 8, 9, 10], while GitHub has been analyzed from multiple perspectives. The analysis of GitHub includes the analysis of codes uploaded to this website [11, 12], investigation of the social network of users and repositories [13, 14, 15, 16], studying the platform for software development purposes [17, 18] and natural language processing [19], as well as examining its connections to other websites [20, 21]. There are also studies concentrated on the effect of factors such as gender, nationality, language, and time on how people behave on the website [22, 23]. Another aspect of GitHub which has been scrutinized over is "pull requests" and "issues" created on repositories [24, 25, 26, 27]. There has also been several work on the academic and governmental use of the platform [28, 29]. [30] introduces a tool for profiling developers and assessing their abilities using information collected from their GitLab profiles. Using this method, which analyzes users' information such as code quality and project participation, the developer's expertise can be extracted.

There are several work on code hosting platforms from a social network perspective. Surian et al. investigated the six degrees of separation theory on Source-Forge's developer network [31]. The maximal connected components of the network were also studied and they found out that only 1.5% of programmers worked alone and that a giant component with 54.07% of users was present in the network. The network's degree distribution was also shown to not fit a power law distribution. They further observed that a large proportion of users only work with a maximum of six other users and that the triadic closure property is present in this network, i.e. there is a high probability that friends of a user, are friends with each other.

Allaho and Lee [13] created user interaction networks based on follower-following relationships on GitHub and Kudo giving patterns on Open Hub (formerly Ohloh). They reported the diameter, the average length of shortest paths, and the average degrees for GitHub and Open Hub networks, thus showing the presence of the small-world and six degrees of separation property in these networks [32]. The authors additionally noted that both networks were scale-free with their degree distributions following a power law distribution. Hubs in these networks tended to have links to low degree nodes thus deriving the conclusion of the existence of relationships between professionals and newcomers.

Influential GitHub projects and users were extracted from their corresponding networks by Thung et al. [14]. From the two projections (for users and projects) of a bi-partite graph of 100,000 GitHub repositories and the random selection of 30,000 of their developers, weighted networks were created and their node degrees, average path lengths, PageRank, and diameter were examined. The obtained values for the network diameter and the average shortest paths were far less than the expected values and that of similar values reported for Source-Forge [31], which was attributed to the more social nature of git and GitHub compared to Source-Forge and subversion (SVN) repositories. Another article that studies influence is [33], where a

new approach is introduced in which multiple indicators and centrality algorithms are used to model different perspectives.

As previously pointed out, to the best of our knowledge, no work has been conducted on the social network properties of GitLab, thus encouraging this study.

III. Data and Methodology

In this section, we begin by introducing the terminology used in online code hosting services, including GitLab. Next, a detailed description of our data collection methods and tools is provided. We then go over the definitions of our analysis methods and metrics.

A. Terminology

Online code hosting platforms offer various features with different terminology. To better understand description of each feature in GitLab, we will go over the terms they use in Table 1.

B. Data Collection

The data used in this study was collected using both the official API of GitLab and a web crawler which has been made available on GitLab[2]. The collected data, saved in a MySQL database can also be accessed in the same repository under the "dbdump" branch. Our data includes project details, including the users who worked on each project, some user properties, and the relationships in the networks of users and projects, consisting of forks, memberships, and contributions. The collection process was completed on August 11, 2019 and includes the information concerning 667,686 projects (the total number of public projects at the time of data collection) and 30,020 users. The created graphs included 47,748 forks, 251,277 memberships, and 90,150 contributions links.

Python and the NetworkX [36] library were our principal tools for analyzing the data. The results were then visualized using Gephi [37], matplotlib [38], and Graphviz [39]. The analysis code is publicly available through the GitLab Analyzer repository[3].

As stated earlier, we had collected users who were members of and those who had contributed to projects. This data allowed us to create two bi-partite user-project networks, with links in one network representing memberships and in the other network representing contributions. Mono-partite projections of both networks were then used to analyze community of users as well as most-worked-on projects.

Another relationship collected was that of forks between projects. To analyze forks, we created a directed network of forks with nodes representing repositories and links connecting repository A to repository B if and only if B was once forked from A. The most forked, and thus to some extent the most influential projects, were then extracted by calculating centrality measures on the nodes.

TABLE I. GitLab Terminology

Star	Users can star topics and repositories to see news related to those topics or projects in their feeds.
Fork	Creating a copy of the target repository which is added to the repositories owned by the forker. This feature is usually used in order to further develop the available code.
Watch	A user can choose to watch a repository to be informed of every single change and announcement made to that project.
Issue	Problems, questions, feature requests, and other issues related to a project can be reported to the owners of the project through this mechanism. An issue can be assigned to a specific user.
Merge Request	Users who are not part of the founding team of a project can develop the code on their own copies of the repository (created via forking that project) and then request their changes to be added to the original project by submitting merge requests. The owners of the project can accept or deny these requests or ask for minor or major revisions before the changes are made.
Commit	This term refers to the act of submitting one or several changes in the codebase to the git log. It is customary for each commit to be followed by a descriptive message listing the changes made.
Group	Groups, in GitLab, are used for grouping projects together. Members can be added to groups which will consequently add them to all projects within that group.
LFS	A Git extension that improves how large files are handled. It replaces them with tiny text pointers that are stored on a remote server instead of in their repository, speeding up operations like cloning and fetching [34, 35].

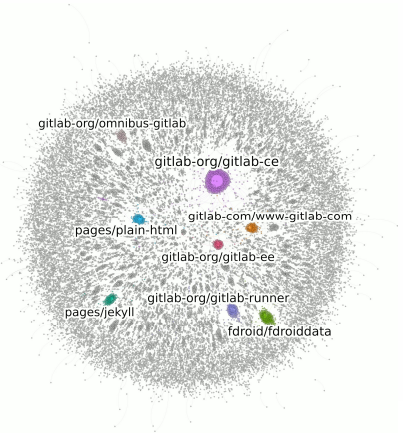

Figure 1. Fork forest. A directed network where nodes are projects and two nodes are connected if one is a fork of the other. Node size corresponds with the number of times the project was directly forked.

[2] https://github.com/hadisfr/gitlab_crawler

[3] https://github.com/hadisfr/gitlab_analyzer

TABLE II. METRIC DEFENITIONS

Scale-free Network	A network whose degree distribution follows a power law distribution. These networks have hubs with degrees much larger than the average degree in the network.
Centrality Measures	The centrality of a node refers to its power and prestige in that network. These metrics are used to offer a quantitative measure of influence in the networks.
Degree Centrality	This metric uses node degrees as the measure of importance. In directed networks, nodes with high outdegrees have a high influence on the network, while nodes with high indegrees are the most popular users.
Closeness Centrality	This measure shows how quickly a node can reach other nodes in the network.
Betweenness Centrality	A measure of the influence of one node on the connection of other individuals in the network.
Eigenvector Centrality	This centrality measure works on this premise that connections with high degree nodes can cause a node to have more power in a network.
Katz Centrality	A special case of eigenvector centrality in which the problem of propagation of zero centrality has been solved.

During data collection, we encountered some bugs and inconsistencies in GitLab API endpoints, which are reported to GitLab team.

C. Analysis methods and terminology

Table 2 shows the definition of the metrics used in the rest of this study.

IV. RESULTS AND DISCUSSION

A. Graph Analysis

As stated in Section III, a graph connecting projects that had been forked of one another was created. This network has been illustrated in Fig. 1. The forks forest is made up of 61,728 nodes, 47,748 edges and 13,980 trees. The longest chain of forks is 8 and the largest tree has 4,100 nodes.

The number of direct and indirect forks of projects follows a power law distribution with law's exponent of $\alpha = 2.32$ and a standard error of 0.06. As a result, the network has a scale-free structure ($2 < \alpha < 3$). To find key projects, in other words those that were forked the most, we calculated centrality measures such as degree and Katz centrality. The results indicated that the projects related to GitLab core, either projects on GitLab's official group of projects[4] or on other groups related to the functionalities offered by the website such as Pages, were the most prominent. These projects have been highlighted in Fig. 1. Additionally, the figure shows how most projects are rarely forked.

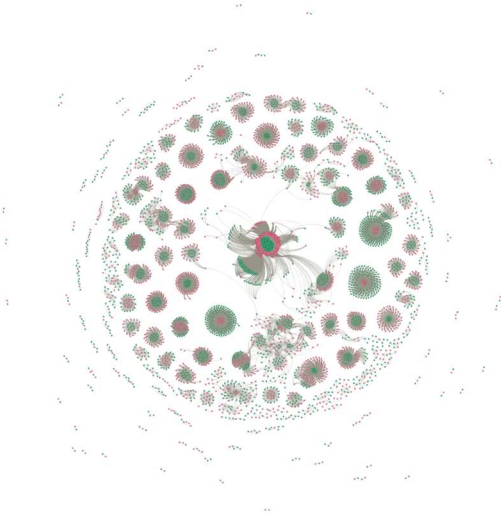

Figure 2. Bi-partite project-user membership graph. The nodes colored in pink are users and those colored in green are projects.

As previously stated as part of our dataset description, two types of relationships could be defined between users and projects, namely membership and contribution. Using these relationships two bi-partite networks were created. Fig. 2 displays the membership connections between users and projects, showing how project membership segregates users into different components.

The contribution graph is a bi-partite graph made up of 18,010 users, 68,849 projects and 90,150 edges among the two sets of nodes. There exist 66,999 maximal complete bi-partite subgraphs in this graph. Meanwhile the membership bi-partite graph includes 30,020 users, 68,491 projects, and 251,277 edges between the two sets of nodes, making a total of 22,240 maximal complete bi-partite subgraphs.

We observe that the membership graph includes 1.6 times more users than the contribution graph, while including fewer projects. The bi-partite membership graph has 0.0122% of all potential edges, meanwhile, the contribution graph includes only 0.007%. In other words, the membership graph is much denser than the contribution graph.

It is interesting to note that if projects with only one member are set aside, on average only 20% of all project members contribute to the project. Consequently, the contribution graph is a better indicator of user activities on the website, while the membership graph can be a better pointer to groups and colonies of people.

We now take a closer look at projects. Using the bi-partite membership graph we find out that the average number of members of a project is 59.23 while the average number of contributors is 9.5, showing that not all members of the teams contribute to all projects of their teams. This fact can be explained by the use of groups (defined in Table 1) since members of groups will automatically be considered members of all

[4] https://gitlab.com/gitlab-org

projects defined as part of that group, which consequently results in a larger number of members than contributors. The low number of average contributors also indicates that, on this platform, most projects are small with only few people working on them, rather than being popular open-source projects that attract many programmers and contributors. The distributions of these parameters are illustrated in Fig. 3. Building the mono-partite projection graph of projects constructed using the contribution graph which is visualized in Fig. 4, we find out that it is made up of 68,849 nodes and 3,162,879 edges. The degree distribution can be modeled using a power law distribution with $\alpha = 1.66$ and a standard error of 0.01. The component size distributions can also be estimated using a power law distribution with $\alpha = 2.40$ and a standard error of 0.06. Both degree and component size distributions are shown in Fig. 5.

In order to better understand user behavior and activities, we create the mono-partite users-

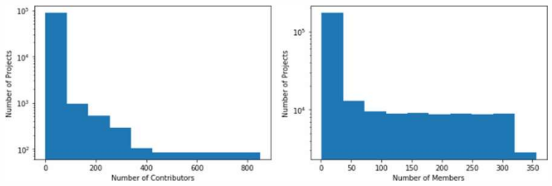

Figure 3. Distribution of number of contributors (left) and number of members (right) of projects.

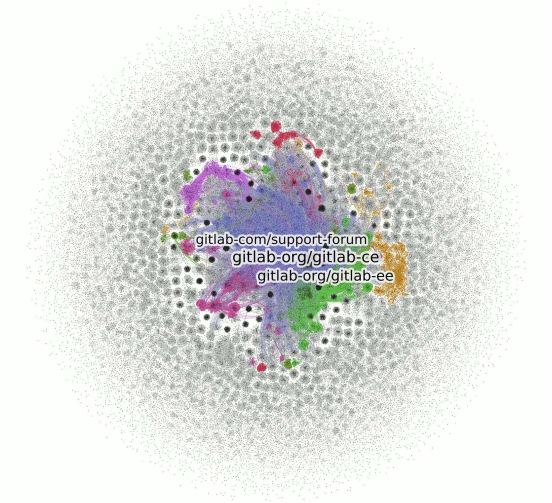

Figure 4. Project graph. Created by mono-partite projection of the user-project contribution bi-partite graph. Colors represent different modularity classes of the nodes.

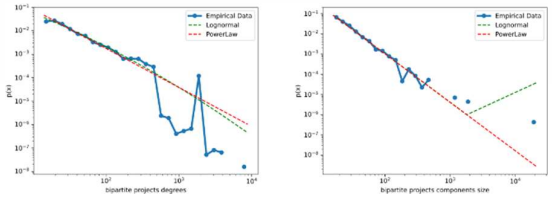

Figure 5. Degree (left) and component size (right) distributions of project graph.

membership graph by creating a projection of users in the bi-partite membership graph described above. By calculating degree assortativity, which has a value of 0.95, we find that most nodes in the graph are linked to nodes with similar degrees. Since degree of a node (user) in this graph demonstrates the number of users this node has worked with in shared projects or has been part of groups with, if we assume that the more people one has been teamed up with, the more experience he/she has, then the high assortativity value shows that developers of different experience levels tend to work with people with similar levels of experience. In [13], it is reported that the assortativity of Github's follower-following network (a feature that is not available on GitLab) is negative, which indicates that newcomers tend to follow experts, but the co-working relationship between the users is not analyzed.

By finding maximal connected components, we discover that 7% of users work alone and that the largest component of the graph encapsulates only 10% of all users. It should be noted that the largest component of this graph is much smaller than the giant component of a typical social network which often includes more than 90% of the nodes. We further find that clustering coefficient is 0.8 in this network, meaning that users form closely knit groups and teams. By calculating average shortest path equal to 9.89 and a diameter[5] of 19 we observe that the six degrees of separation theory is not valid in this network.

We now move on to creating and analyzing the users graph made based on projecting the bi-partite user-contribution graph on its user partite, resulting in a graph with 18,010 nodes and 507,217 connections. Fig. 6 provides a visualization of this network. The degree distribution can be estimated using a power law distribution (Fig. 7). As expected the graph includes a giant component and many small-components. With 7,767 components making up the entirety of the graph. The component sizes can be estimated using a power law distribution. The average shortest path in the network is 4.56 (compared to 6.55 in SourceForge [31] and 2.47 in GitHub [14]) and the graph's diameter is 15 (compared to 19 in SourceForge [31] and 5 in GitHub [14]), thus meaning that the theory of six degrees of separation holds in this graph. Clustering coefficient is 0.46 in this network (compared to 0.85-0.95 in SourceForge [41] and 0.395 in GitHub [42]), showing that contributors have less solid team structure compared to members of projects. Degree assortativity has the value of 0.47 in this graph, which does stipulate that contributors with similar degrees have a higher tendency to work with one another on shared projects (compared to negative degree assortativity of -0.0386 in GitHub [42]).

Additionally, cliques and communities of users are calculated for both contribution and membership projections. Communities are detected using Louvain community detection algorithm [43]. We can observe in Fig. 8 and Fig. 9 that the component sizes follow similar distributions in both projected graphs, with

[5] To calculate distances, we use the SNAP Python package [40] which offers methods to approximate these values using sampling methods and GetBfsFullDiam GetBfsEffDiam.

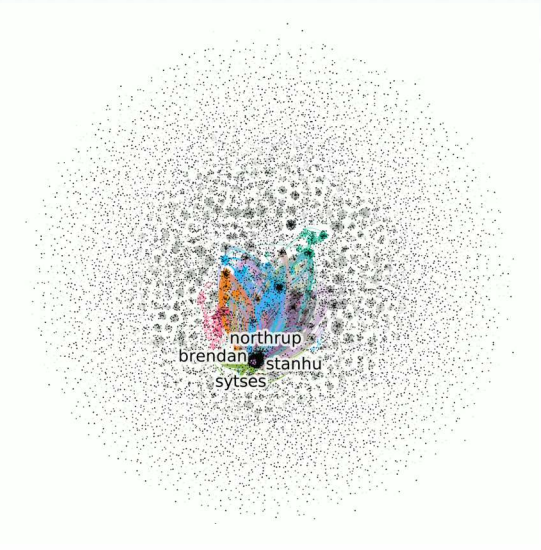

Figure 6. User graph. Created by mono-partite projection of the user-project contribution bi-partite graph. Colors in the graph indicate different modularity classes.

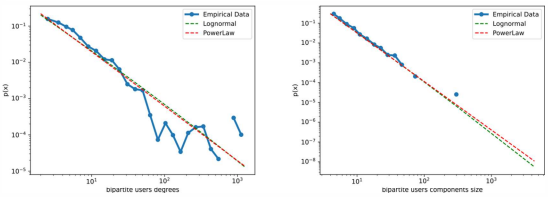

Figure 8. Degree (left) and component size (right) distributions of user graph.

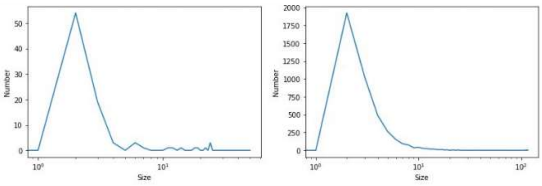

Figure 7. Distribution of users clique sizes in projected contributions graph (left) and projected members graph (right).

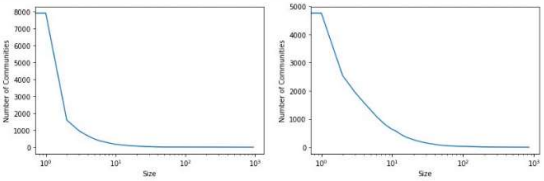

Figure 9. Distribution of users community sizes in projected contributions graph (left) and projected members graph (right).

many communities/cliques having few members and a few being made up of large numbers of users.

Table 3 summarizes the attributes of the mono-partite graphs resulted from the projection of contribution and membership graphs. Using the names given to the graphs in the table, we can see that the number of components of $G_{C\to P}$ equals that of $G_{C\to U}$, and the same holds for $G_{M\to P}$ and $G_{M\to U}$. This is expected, since both of the projection graphs of each pair (i.e. the pair of $\langle G_{C\to P}, G_{C\to U}\rangle$ and $\langle G_{M\to P}, G_{M\to U}\rangle$) are derived from the same bi-partite graphs (i.e. Contribution and Membership graph respectively), and hence, projection on either of projects or users yields the same number of components.

B. *Analysis of Attributes of Projects*

In addition to network analysis of projects and users of GitLab, we analyzed some attributes of GitLab projects. Fig. 10 and Table 4 show the Pearson correlation matrix of attributes of projects.

We observe a strong correlation (0.897) between the number of stars and forks of projects, which is as expected, since both are representatives of a project's popularity.

There is a moderate correlation between storage size and the number of forks and stars of projects (0.502 and 0.463, respectively). Often the majority of a project's storage in contemporary integrated version control systems belongs not to the project's codebase, but to the artifacts of CI/CD builds. As a result, these correlations display a relationship between the size of the CI/CD build artifacts, and the project's popularity.

As previously mentioned, the number of likes and forks of a project is a metric of the project's popularity. Let's now consider the number of commits and the repository size as indicators of the codebase size. We can see a weak correlation between the popularity of a GitLab project and its codebase size.

V. Conclusion & Future Work

In this study, we carefully analyzed the GitLab projects, as well as membership and contribution networks. It is worth mentioning that the collected data only includes public projects, which does introduce a bias towards these projects. Our results indicate that, on average, there are significantly more members than contributors to projects. Additionally, we investigated different network metrics that explain users' behavior on the platform.

We also found that the website is used mostly for small projects and personal usage, and very few notable open-source projects exist on the platform. This fact, however, might change due to the migration of a significant number of users from GitHub to GitLab after the acquisition of the former by Microsoft. We additionally present degree, component size and community size structure of the graphs. As well as showing the assortative nature of both membership and contribution graphs.

The dataset introduced in this study, allows the study of numerous aspects of GitLab in future work. For instance, all the calculations we have done can be repeated across multiple timestamps to monitor the evolution of GitLab's network over time. Additionally, deeper insight into the GitLab's network can be obtained by analyzing language-specific features of projects and developers. Correlation of programming languages used in projects with other attributes such as popularity, can also offer a better understanding of the platform and its users. These results can further be compared with similar analyses on GitHub [44, 45, 46].

TABLE III. ATTRIBUTES OF PROJECTED GRAPHS

Projected on Projected from	Projects Contribution	 Membership	Users Contribution	 Membership
Graph name	$G_{C\to P}$	$G_{M\to P}$	$G_{C\to U}$	$G_{M\to U}$
# nodes	68,849	68,491	18,010	30,020
# edges	3,162,879	3,989,353	507,217	395,444
# components	7,767	4,715	7,767	4,715
# components with size=1	3,117	981	6,307	2,223
largest component size	21,372	9,267	4,368	3,094
diameter	16	19	15	19
clustering coefficient	0.88	0.89	0.46	0.80
degree assortativity	0.81	0.51	0.47	0.95

TABLE IV. CORRELATION MATRIX OF ATTRIBUTES OF PROJECTS

	stars	**forks**	**commits**	**storage size**	**Repository size**	**git LFS size**
stars	1.000	0.897	0.009	0.463	0.021	0.000
forks	0.897	1.000	0.009	0.502	0.021	0.000
commits	0.009	0.009	1.000	0.017	0.210	−0.000
storage size	0.463	0.502	0.017	1.000	0.061	0.022
repository size	0.021	0.021	0.210	0.061	1.000	0.004
git LFS size	0.000	0.000	−0.000	0.022	0.004	1.000

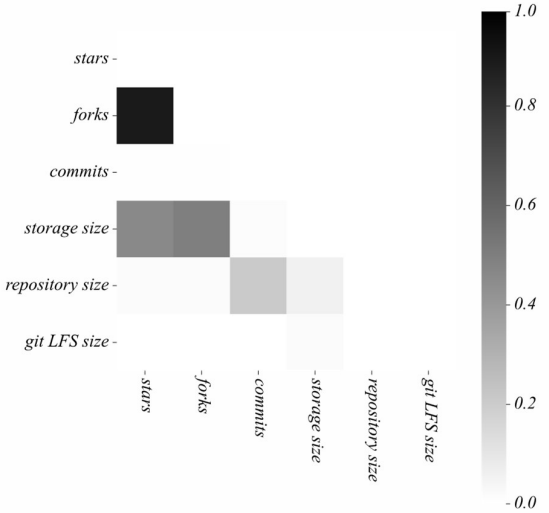

Figure 10. Correlation matrix of attributes of projects using Pearson correlation.

REFERENCES

[1] O. Jarczyk, B. Gruszka, S. Jaroszewicz, L. Bukowski, and A. Wierzbicki, *GitHub Projects. Quality Analysis of Open-Source Software*. Cham: Springer International Publishing, 2014, pp. 80–94.

[2] D.-C. Yan, Z.-W. Wei, X.-P. Han, and B.-H. Wang, "Empirical analysis on the human dynamics of blogging behavior on GitHub," *Physica A: Statistical Mechanics and its Applications*, vol. 465, pp. 775–781, 2017.

[3] D. Oberhaus, "13,000 projects ditched GitHub for GitLab monday morning," *Motherboard*, 2018, accessed 15 February 2020. [Online]. Available: https://www.vice.com/en_us/article/ywen8x/13000-projects-ditched-GitHub-for-GitLab-monday-morning

[4] H. Safari, N. Sabri, F. Shahsavan, and B. Bahrak, "An analysis of GitLab's users and projects networks," in *2020 10th International Symposium on Telecommunications (IST)*. newIEEE, 2020, pp. 194–200.

[5] Wikipedia contributors, "Comparison of source code hosting facilities – Wikipedia, the free encyclopedia," 2018, accessed 17 August 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Comparison_of_source_code_hosting_facilities\&oldid=856012346

[6] L. Yu, A. Mishra, and D. Mishra, "An empirical study of the dynamics of GitHub repository and its impact on distributed software development," *in On the Move to Meaningful Internet Systems: OTM 2014 Workshops*, R. Meersman, H. Panetto, A. Mishra, R. Valencia-Garc\'ia, A. L. Soares, I. Ciuciu, F. Ferri, G. Weichhart, T. Moser, M. Bezzi, and H. Chan, Eds. newBerlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 457–466.

[7] M. Novák, M. Biňas, M. Michalko, and F. Jakab, "Student's progress tracking on programming assignments," *in 2012 IEEE 10th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, November 2012, pp. 279–282.

[8] M. Biňas, "Version control system in CS1 course: Practical experience," in *2013 IEEE 11th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, October 2013, pp. 23–28.

[9] A. Villarrubia and H. Kim, "Building a community system to teach collaborative software development," in *2015 10th International Conference on Computer Science Education (ICCSE)*, July 2015, pp. 829–833.

[10] G. C. Diniz, M. A. G. Silva, M. A. Gerosa, and I. Steinmacher, "Using gamification to orient and motivate students to contribute to oss projects," in *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, May 2017, pp. 36–42.

[11] S. Horschig, T. Mattis, and R. Hirschfeld, "Do Java programmers write better Python? studying off-language code quality on GitHub," in *Conference Companion of the 2nd International Conference on Art, Science, and Engineering of Programming*, ser. Programming'18 Companion. newNew York, NY, USA: Association for Computing Machinery, 2018, pp. 127–134.

[12] P. S. Kochhar and D. Lo, "Revisiting assert use in GitHub projects," in *The 21st International Conference on Evaluation and Assessment in Software Engineering, ser*. EASE'17. New York, NY, USA: ACM, 2017, pp. 298–307.

[13] M. Y. Allaho and W.-C. Lee, "Analyzing the social ties and structure of contributors in open source software community," in *The 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ser. ASONAM '13. newNew York, NY, USA: ACM, 2013, pp. 56–60.

[14] F. Thung, T. F. Bissyandé, D. Lo, and L. Jiang, "Network structure of social coding in GitHub," in *2013 17th European Conference on Software Maintenance and Reengineering*, March 2013, pp. 323–326.

[15] J. Jiang, L. Zhang, and L. Li, "Understanding project dissemination on a social coding site," in *2013 20th Working Conference on Reverse Engineering (WCRE)*, October 2013, pp. 132–141.

[16] C. Hunger, L. Vilanova, C. Papamanthou, Y. Etsion, and M. Tiwari, "DATS - data containers for web applications," in *The Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '18. New York, NY, USA: ACM, 2018, pp. 722–736.

[17] R. L. Q. Portugal, J. C. S. do Prado Leite, and E. Almentero, "Time-constrained requirements elicitation: reusing GitHub content," in *2015 IEEE Workshop on Just-In-Time Requirements Engineering (JITRE)*, August 2015, pp. 5–8.

[18] M. Menichinelli, "A data-driven approach for understanding open design. mapping social interactions in collaborative processes on GitHub," *The Design Journal*, vol. 20, no. sup1, pp. S3643–S3658, 2017.

[19] V. J. Hellendoorn, P. T. Devanbu, and A. Bacchelli, "Will they like this?: Evaluating code contributions with language models," in *The 12th Working Conference on Mining Software Repositories*, ser. MSR '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 157–167.

[20] B. Vasilescu, V. Filkov, and A. Serebrenik, "StackOverflow and GitHub: Associations between software development and crowdsourced knowledge," in *2013 International Conference on Social Computing*, September 2013, pp. 188–195.

[21] A. S. Badashian, A. Esteki, A. Gholipour, A. Hindle, and E. Stroulia, "Involvement, contribution and influence in GitHub and Stack Overflow," in *24th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON '14. Riverton, NJ, USA: IBM Corp., 2014, pp. 19–33.

[22] B. Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik, P. Devanbu, and V. Filkov, "Gender and tenure diversity in GitHub teams," in *The 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI '15. newNew York, NY, USA: ACM, 2015, pp. 3789–3798.

[23] J. Aué, M. Haisma, K. F. Tómasdóttir, and A. Bacchelli, "Social diversity and growth levels of open source software projects on GitHub," in *The 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '16. New York, NY, USA: ACM, 2016, pp. 41:1–41:6.

[24] M. M. Rahman and C. K. Roy, "An insight into the pull requests of GitHub," in *The 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. newNew York, NY, USA: ACM, 2014, pp. 364–367.

[25] S. Yu, L. Xu, Y. Zhang, J. Wu, Z. Liao, and Y. Li, "NBSL: A supervised classification model of pull request in GitHub," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[26] T. F. Bissyandé, D. Lo, L. Jiang, L. Réveillère, J. Klein, and Y. L. Traon, "Got issues? who cares about it? a large scale investigation of issue trackers from GitHub," in *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*, November 2013, pp. 188–197.

[27] T. Zhang, J. Chen, X. Luo, and T. Li, "Bug reports for desktop software and mobile apps in GitHub: What's the difference?" *IEEE Software*, vol. 36, no. 1, pp. 63–71, 2019.

[28] F. Arcelli Fontana and C. Raibulet, "Students' feedback in using GitHub in a project development for a software engineering course," in *The 2017 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '17. newNew York, NY, USA: ACM, 2017, pp. 380–380.

[29] J. Longo and T. M. Kelley, "GitHub use in public administration in canada: Early experience with a new collaboration tool," *Canadian Public Administration*, vol. 59, no. 4, pp. 598–623, 2016.

[30] J. Wang, X. Meng, H. Wang, and H. Sun, "An online developer profiling tool based on analysis of GitLab repositories," in *CCF Conference on Computer Supported Cooperative Work and Social Computing*. Springer, 2019, pp. 408–417.

[31] D. Surian, D. Lo, and E. Lim, "Mining collaboration patterns from a large developer network," in *2010 17th Working Conference on Reverse Engineering*, October 2010, pp. 269–273.

[32] J. Travers and S. Milgram, "An experimental study of the small world problem," *Sociometry*, vol. 32, no. 4, pp. 425–443, 1969.

[33] Y. Hu, S. Wang, Y. Ren, and K.-K. R. Choo, "User influence analysis for github developer social networks," *Expert Systems with Applications*, vol. 108, pp. 108–118, 2018.

[34] G. Docs, "Git large file storage (lfs)," accessed 2 March 2020. [Online]. Available: https://docs.gitlab.com/ee/topics/git/lfs/

[35] B. Support, "Git large file storage," accessed 2 March 2020. [Online]. Available: https://confluence.atlassian.com/bitbucketserver/git-large-file-storage-794364846.html

[36] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in The *7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11–15.

[37] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *International AAAI Conference on Weblogs and Social Media*, 2009, pp. 361–362.

[38] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[39] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphviz–- open source graph drawing tools," in *Graph Drawing*, P. Mutzel, M. Jünger, and S. Leipert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 483–484.

[40] J. Leskovec and R. Sosič, "SNAP: A general-purpose network analysis and graph-mining library," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 1, Jul. 2016.

[41] Y. Gao and G. Madey, "Network analysis of the sourceforge.net community," in *Open Source Development, Adoption and Innovation*, J. Feller, B. Fitzgerald, W. Scacchi, and A. Sillitti, Eds. Boston, MA: Springer US, 2007, pp. 187–200.

[42] A. Lima, L. Rossi, and M. Musolesi, "Coding together at scale: Github as a collaborative social network," *CoRR*, vol. abs/1407.2535, 2014. [Online]. Available: http://arxiv.org/abs/1407.2535

[43] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, Oct 2008.

[44] H. Borges, A. C. Hora, and M. T. Valente, "Understanding the factors that impact the popularity of github repositories," *CoRR*, vol. abs/1606.04984, 2016. [Online]. Available: http://arxiv.org/abs/1606.04984

[45] F. Tomassetti and M. Torchiano, "An empirical assessment of polyglot-ism in github," in *The 18th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '14. New York, NY, USA: ACM, 2014, pp. 17:1–17:4. [Online]. Available: http://doi.acm.org/10.1145/2601248.2601269

[46] C. Casalnuovo, B. Vasilescu, P. Devanbu, and V. Filkov, "Developer onboarding in github: The role of prior social links and language experience," in *The 2015 10th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2015. newNew York, NY, USA: ACM, 2015, pp. 817–828. [Online]. Available: http://doi.acm.org/10.1145/2786805.2786854

Hadi Safari has received his Bachelor's degree in Software Engineering from University of Tehran in 2019. Currently, he is a Master's student in Software Engineering at University of Tehran. His research interests are Network Science, Social Network Analysis, Scientometrics, and Model-Based Software Testing.

Nazanin Sabri is currently pursuing a Master's degree in Artificial Intelligence and Robotics from the University of Tehran, Tehran, Iran. She received her Bachelor's degree in Information Technology from The University of Tehran in 2019.

Faraz Shahsavan is currently a Research Associate at EPFL of Lausanne, University of Tehran, and formerly, IPM Institute for Research in Fundamental Sciences of Tehran, and an undergraduate student in Computer Engineering at the University of Tehran. Graduated from Allameh Helli high school, affiliated with Iran's National Organization for Development of Exceptional Talents (NODET). He is intrigued by the field of Computer Networks and Distributed Systems in general. Some of his other keen research interests are IoT, SDNs, Blockchain, Network Security, and the use of Data Science in Computer Networks.

Behnam Bahrak received his Bachelor's and Master's degrees, both in Electrical Engineering, from Sharif University of Technology, Tehran, Iran, in 2006 and 2008, respectively. He received the Ph.D. degree from the Bradley Department of Electrical and Computer Engineering at Virginia Tech in 2013. He is currently an Assistant Professor of Electrical and Computer Engineering at University of Tehran.