Diving into Popularity of GitHub Repositories

Md Rayhanul Masud
mmasu012@ucr.edu

Sai Vivek Pidaparthi
spida002@ucr.edu

Sai Teja Pasupulety
spasu009@ucr.edu

Michalis Faloutsos
michalis@cs.ucr.edu

***Abstract*—What are the features that can best define the popularity of GitHub repositories? Since GitHub is being widely used as the most popular open source platform, thorough investigation of the correlation of different features of a repository will greatly benefit the research communities to predict the evolution of popular repositories. The challenging factor is to identify the driving features based on which a repository can be tagged as popular. This study represents an exploratory data analysis on a set of 7.5k malware repositories to view the popularity from various angles. We analyze both author-centric and repository-centric features, and find strong correlation between them. We notice that authors often provide contact information in their profile. Most of these authors have handsome amount of followers, and are influential in their network as well. The repositories they publish receive greater interactions from the community being rewarded more stars. Our analysis suggests that author-centric features alone can better approximate repository-centric features. As a result, instead of manipulating all of the features, author-centric information should be exploited properly to predict the popularity of GitHub repositories. We use python programming language and its several packages for crawling data related to authors and repositories, and data analysis. The project consists of around 600 lines of code.**

***Index Terms*—GitHub, Popularity, HITS, Social.**

I. Introduction

Understanding the emergence of popular GitHub repositories has significant importance to explain the vibrant networking phenomena across the coding platforms. According to https://en.wikipedia.org, there are approximately 70 million developers in GitHub community with more than 28 million public repositories. It is the hub of collaboration among students, professionals, researchers, hackers and so many to mention. In consequence, these repositories differ in topics, structures, contents and interactions.

How to specify the features to be considered to term a GitHub repository as popular? Every repository is entailed with repository-centric and author-centric features. Repository-centric features include contents, metadata and measure of interactions among the repository and the developers: $stars$, $forks$, $comments$. Author-centric features bag the available individual information of the authors and the measure of their participation in the community: $follower\ count$, $following\ count$. Given this plethora of information, it is challenging to identify the appropriate features and their relationships that can identify the popular repositories. The input here is the pool of repository-centric and author-centric features. The desired output is: the minimal features that can approximate the popularity of a GitHub repository. .

Though there have been a very few studies on the prediction of popularity of Github repositories, they lack a holistic approach to express the underlying scenarios. [1], [2] and [3] consider various properties: $stars$, $forks$, $branches$, $open\ issues$, $contributors$, $folder\ structures$, $programming\ languages$ and so on; for popularity prediction tasks. [4] and [5] focus on the evolution of a repository, and use $stars$ as the metric of measuring popularity. These works manipulate a good number of features from author and repository point of view, but none of them analyze the information available in the author profile which has greater correlation with the repository-centric features.

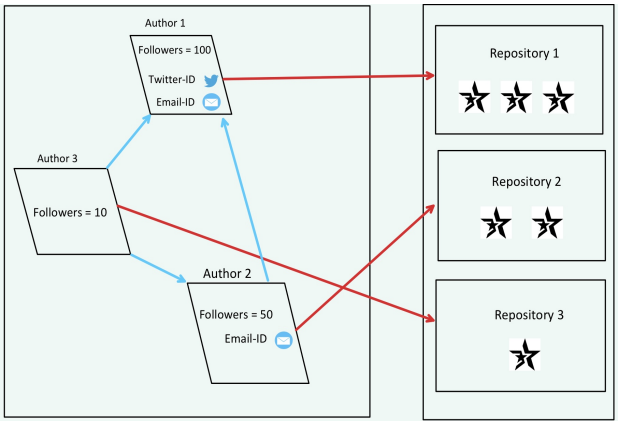

Figure 1: Influential authors providing contact information have more followers and stars for their repositories

As our key contribution, we propose an exploratory data analysis to identify the relationship between repository and author-centric features. We discuss those features concisely, and show that author-centric features have the significant impact on different repository-centric features. Our analysis is performed on a set of malware repositories [6] and their authors. First, an author-author [7] network graph is constructed based on the $follower - followee$ relationship. Second, we run HITS algorithm on the network graph to measure the influence of the authors in the defined network. Third, an author profile is created for each of the authors using the author and repository-centric information. We get $authority\ score$, $hub\ score$ from the output of HITS algorithm [8], and $social\ score$ based on the redundant contact information available in author's GitHub profile. Finally, we analyze these scores and author and repository-centric features to answer the following research questions.

Q.1: Do most of the influential authors provide social information?

Q.2: How does the influence of an author in a subgroup of the GitHub community correlate with the count of *followers*?

Q.3: Do authors providing contact information receive more stars for their repositories than who do not provide?

Q.4: Is there any correlation between the count of *stars*, *forks* and *subscribers*?

The results of our experiment can be summarized as follows:

1) Authors often share their contact information such as email id, twitter handle, link to their blogs/websites.
2) Most of these authors have large follower base and highly starred repositories.
3) Author-centric features have strong correlation with repository-centric features.
4) The more influential the authors are, the more contact information they provide in their GitHub profile, and the more larger their follower base is. The more follower they have, the more stars/forks/subscribers their repositories get.

II. Background

In this section, we provide a basic understanding of different features that a GitHub Repository can have, and describe several approaches that have been addressed to predict the popularity of GitHub repositories.

A. GitHub and its features. Every Github repository has two types of features: (a) author-centric and (b) repository-centric. An author of a repository can provide contact information such as $email$, $twitter\,handle$, $personal\,blog$ and $location$. They can follow other authors and, can be followed by others. Thus they have $follower\,count$ and $followee\,count$ that indicate the degree of networking and collaboration an author demonstrates inside the community. These features belong to the author-centric features of a repository. On the other hand, GitHub users can thumbs up a repository they like, create a copy of the repository to manage it by themselves, and watch the repository to get notifications about any updates to it. Counting these interactions are defined as $stars$, $forks$ [9], and $subscribers$ [10] respectively which belong to the second category, repository-centric.

B. Previous approaches. Previous approaches mostly consider the the number of stars or forks fo the repository for the prediction of popularity. [1] applied multi-variate linear regressions to predict the number of stars based on historical data. [11] introduced an approach that measures the influence of the authors who are starring a repository for the prediction of its popularity. [3] listed 35 features; both author-centric and repository-centric; to be used for the identification of popular repositories. They claimed that degree of collaboration of developers of a repository has the most significant impact for the repositories for being popular. In addition, there are some other approaches which analyze the structural information of the repository. [2] used folder structure and the metadata of the repository to correlate the repository-centric features as the popularity measurement metrics.

III. Problem Definition

How to find the optimal subset of features that can best correlate the popularity of GitHub repositories? A GitHub repository consists of codes, metadata, auxiliary files along with several indicators (i.e. $stars$, $forks$, $subscribers$) of collaboration among the developer with the repository. In addition, the author of that repository has its own profile information and several indicators (i.e. $follower\,count$, $following\,count$) of its influence in the community. The problem is to identify the most effectual features that can predict the popularity of a GitHub repository.

IV. Solution

A. Proposed approach. A GitHub repository does not get popular all at once. By $popular$, we assume the degree of interactions with repository by the community. When a new author gets registered in GitHub, the author has no repositories and no $followers$. So, when the author publishes a repository, it is very unlikely for the repository to be reached out to so many developers. But if the author has good number of collaborations already, the interaction with the repository may speed up by the dint of earlier collaborations. Besides, the interactions among the developers can be affected by the individual information available in their profile. On the other hand, after being published, the repository will be starred, forked, subscribed day by day. The more starred/forked/subscribed events will happen, the more followers the author may have, and vice versa. As a result, the authors themselves can play greater role to make their repositories more reachable to the community resulting more interactions with the repositories to turn them to be popular ones.

$$\begin{aligned} activation\,score &= f_a(author\,centric\,features) \\ mobilization\,score &= f_r(repo\,cetric\,features) \\ popularity\,score &= activation\,score + mobilization\,score \end{aligned} \tag{1}$$

At any instant, thus, the popularity of a repository is viewed as the sum of $activation\,score$ and $mobilization\,score$. $activation\,score$ is calculated from the set of author-centric features, and $mobilization\,score$ from repository-centric features. The more $popularity\,score$ is, the more the repository is popular.

We analyze these author and repository-centric features and find correlations between them. Based on the correlations found, we identify the driving features that can impact the popularity of GitHub repositories.

B. Dataset. We do our experimentation on a dataset of 7.5k malware GitHub repositories. To understand the influence of GitHub authors in the community, we discard the authors who have no followers. We extract 5045 malware authors who have at least one follower accordingly. Furthermore, we attempt to find whether these malware authors are following each other because of the similarity of their contents, and we locate a group of 2123 authors. Among the group, every

author is followed by at least one other author from the same group. To perform our analysis, we crawl repository and author centric features for each of the malware repositories the group members have.

C. Experimentation. We create an Author-Author directed network graph [7], $G(V, E)$ to understand the network-wide influence of the malware authors, where V is set of vertices corresponding to each of the member authors detected in the group, and E is the set of edges. Each (u, v) edge denotes that author v is followed by author u. Then we apply HITS algorithm [8] on the constructed author-author graph to measure the influence of the authors in the network. The influence is calculated in two parts, (a) $authority\,score$ and (b) $hub\,score$.

From the contact information available in the author profile, we also calculate $social\,score$ for each of the authors in the group. We consider email id, twitter handle and blog url as the contact information for $social\,score$ calculation. These scores have been defined as follows:

1) ***authority score*:** is the measurement for an author t0 be followed by the other influential author in the network. The more score an author has, the more he/she is influential.
2) ***hub score*:** is the measurement for an author to follow other influential author in the network. The more score an author has, the more he/she follows other influential authors in the network. Table 1 shows the statistics of contact information provided by the group of malware authors that we consider.
3) ***social score*:** is determined by how many ways an author can be communicated using the contact information provided in the GitHub account. If an author shares email id, twitter handle and blog url, the score will be 3.

Contact Information	Frequency
Twitter	426
Email	822
Blog	1202
Twitter, Email	221
Twitter, Blog	348
Blog, Email	570
Twitter, Blog, Email	196

Table I: The frequency of authors who provide contact information among 2123 malware authors

V. Results

In this section, we present answers to five research questions.

Q.1: Do most of the influential authors provide social information? We get the influence score for each of the malware authors in the group through applying HITS algorithm in the author-author network graph. We calculate the $social\,score$ from the contact information available in the author profile as well. Figure 2 shows that most of the authors provide at least one contact information, where the number of authors who share all of the considering contact information is also significant (Table 1).Figure 3 illustrates that most of the influential authors who have higher influence scores, provide contact information. For every range of influence scores, the frequency in case of $social\,score > 0$ is quite high.

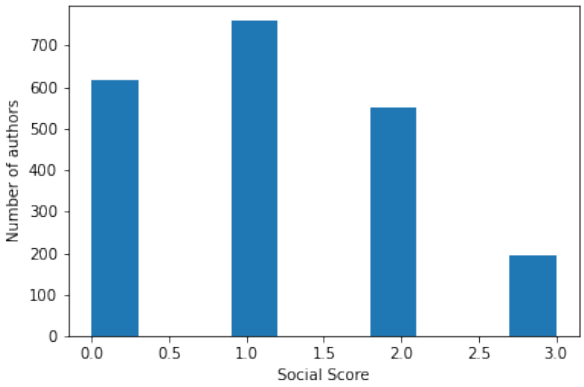

Figure 2: Frequency of social scores for the authors in the group

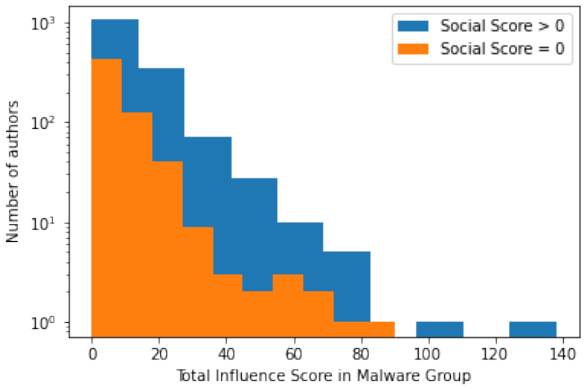

Figure 3: Comparison of frequency of influence scores between the category of authors who provide contact information and others who do not

Q.2: How does the influence of an author in a subgroup of the GitHub community correlate with the count of *followers*? Good $authority\,score$ of an author stipulates that the author has a good number of followers in the considering network, the same as for $hub\,score$ to follow a good number of other others in the network. Figure 4 shows how $authority\,score$ is related with the follower count of an author.

We notice that after a threshold point, the more authoritative an author is inside the small group of malware authors, the more follower count he/she has in the whole community. Figure 5 explains the same for the $hub\,score$ and following count.

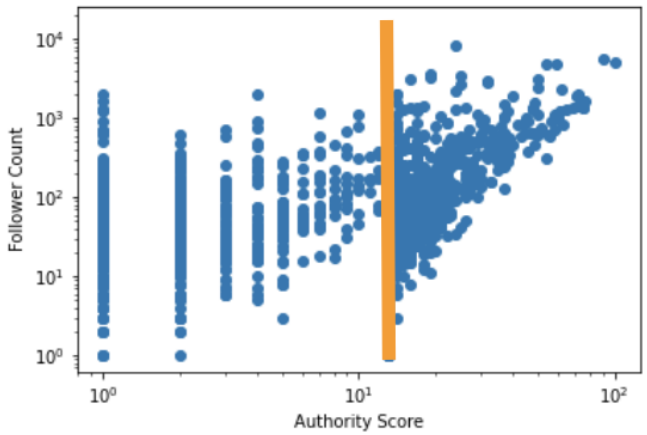

Figure 4: Higher Authority Score belongs to higher follower count

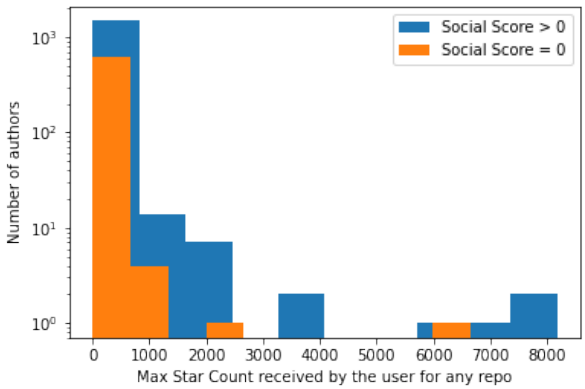

Figure 6: Comparison of frequency of maximum star count that an author has for any of the repositories between the category of authors who provide contact information and others who do not

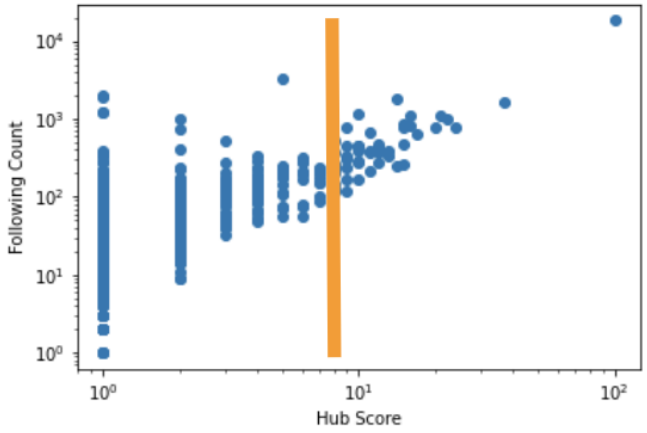

Figure 5: Higher Hub Score belongs to higher following count

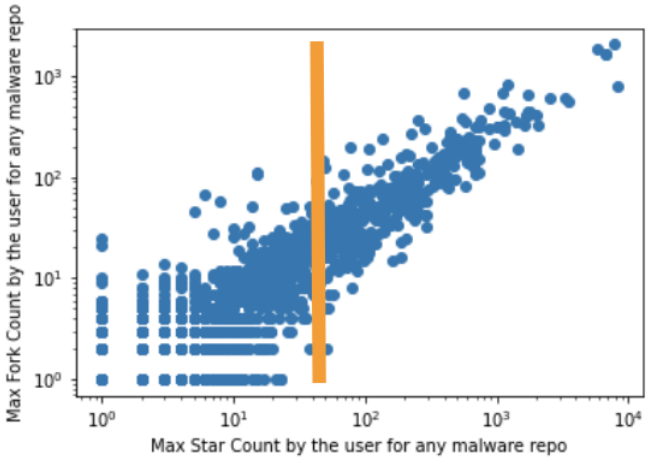

Figure 7: Forks count is largely strongly correlated with Stars count

Q.3: Do authors providing contact information receive more stars for their repositories than who do not provide? Figure 6 displays the frequency of $stars$ that a user can have for any of his/her repositories. We notice that the authors who provide social contact information have more stars than the others who do not provide. The statistics also present that there are very few repositories having large count of $stars$.

Q.4: Is there any correlation between the count of *stars*, *forks* and *subscribers*? In figure 7 and 8, we compare the maximum $stars$ that an author gets for any of the repositories with the maximum $forks$ and maximum $subscribers$ that he/she gets for any of the repositories respectively. In both cases, after some threshold, $forks$ and $subscribers$ counts increase with the increase in $stars$ count.

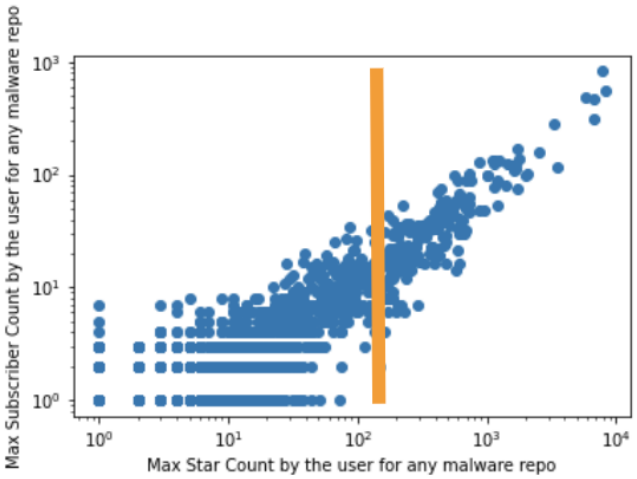

Figure 8: Subscribers count is largely strongly correlated with Stars count

VI. Discussion

In this section, we discuss how the extensions of our findings for the research questions can help identify the features that are correlated with the popularity of GitHub repositories. Results for Q1 provides a clear indication that the influential authors in a small group of GitHub community most often make their contact information available which attract other users to get connected with them. Thus, this behavior can affect the $follower\,count$ of the author. Results for Q2 shows that if the author has substantial number of followers, they are highly influential in their sub-group with higher $authority\,score$. On the other hand, the $star$ count of a GitHub repository is correlated with the $follower\,count$ [1]. Q3 denotes that the authors providing contact information having higher $social\,score$ have higher probability of having stars for their repositories. It is obvious that the social nature of authors has greater impact on their repository for gaining higher attention from the developers. These higher collaboration feeds more stars to their repositories. Results for Q4 derives the strong correlation among stars, forks and subscribers. Thus the influence of an author in its maintaining group, the contact information available in the account profile and $follower\,count$ are correlated with each other. In addition, we show that these author-centric information is also correlated with $stars$, $forks$ and $subscribers$ count.

VII. Conclusions

We present a statistical analysis on a set of malware repositories from GitHub platform to understand the correlation between author and repository centric features of a repository. The key result of our analysis is that features of these two categories are strongly correlated, and this correlation has greater significance to make the repositories popular. The redundant contact information available in the author profile is also found as one of the driving features to be considered for the prediction of popularity. We conclude that author-centric features themselves can better explain the popularity in the GitHub space.

References

[1] H. Borges, A. Hora, and M. T. Valente, "Predicting the popularity of github repositories," in *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, 2016, pp. 1–10.

[2] J. Zhu, M. Zhou, and A. Mockus, "Patterns of folder use and project popularity: A case study of github repositories," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2014, pp. 1–4.

[3] J. Han, S. Deng, X. Xia, D. Wang, and J. Yin, "Characterization and prediction of popular projects on github," in *2019 IEEE 43rd annual computer software and applications conference (COMPSAC)*, vol. 1. IEEE, 2019, pp. 21–26.

[4] Q. Xie, J. Wang, G. Kim, S. Lee, and M. Song, "A sensitivity analysis of factors influential to the popularity of shared data in data repositories," *Journal of Informetrics*, vol. 15, no. 3, p. 101142, 2021.

[5] H. Zhou, H. Ravi, C. M. Muniz, V. Azizi, L. Ness, G. de Melo, and M. Kapadia, "Gitevolve: Predicting the evolution of github repositories," *arXiv preprint arXiv:2010.04366*, 2020.

[6] SourceFinder, "Finding malware source-code from publicly available repositories in github," http://hackerchatter.org/sourcefinder/, [Online; accessed 12-January-2021].

[7] R. Islam, M. O. F. Rokon, A. Darki, and M. Faloutsos, "Hackerscope: The dynamics of a massive hacker online ecosystem," *Social Network Analysis and Mining*, vol. 11, no. 1, pp. 1–12, 2021.

[8] M. Agosti and L. Pretto, "A theoretical study of a generalized version of kleinberg's hits algorithm," *Information Retrieval*, vol. 8, no. 2, pp. 219–243, 2005.

[9] J. Jiang, D. Lo, J. He, X. Xia, P. S. Kochhar, and L. Zhang, "Why and how developers fork what from whom in github," *Empirical Software Engineering*, vol. 22, no. 1, pp. 547–578, 2017.

[10] J. Sheoran, K. Blincoe, E. Kalliamvakou, D. Damian, and J. Ell, "Understanding" watchers" on github," in *Proceedings of the 11th working conference on mining software repositories*, 2014, pp. 336–339.

[11] L. Ren, S. Shan, X. Xu, and Y. Liu, "Starin: An approach to predict the popularity of github repository," in *International Conference of Pioneering Computer Scientists, Engineers and Educators*. Springer, 2020, pp. 258–273.