StarIn: An Approach to Predict the Popularity of GitHub Repository

Leiming Ren(✉), Shimin Shan, Xiujuan Xu, and Yu Liu

School of Software, Dalian University of Technology,
Dalian 116620, Liaoning, China
martin.rlm@foxmail.com

Abstract. The popularity of repository in GitHub is an important indicator to evaluate its quality. Exploring the trend of popularity is a crucial guideline to study its development potential. Herein, StarIn, a stargazer-influence based approach is proposed to predict the popularity of GitHub repository. Using the followers in GitHub as a basic dataset, stargazer-following based network was established. The indicator, stargazer influence, was measured from three aspects of basic influence, network dynamic influence and network static influence. Experiments was conducted, and the correlation of StarIn was analyzed with the popularity of repository from the perspective of six characteristics. The experimental evaluation provides an interesting approach to predict the popularity of repositories in GitHub from a new perspective. StarIn achieves an excellent performance of predicting the popularity of repositories with a high accurate rate under two different classifiers.

Keywords: Stargazer influence · Popularity prediction · Network relationship · GitHub

1 Introduction

With the development of the Internet and cloud computing technologies, social coding becomes a hot research field [5, 20, 21]. Software developers gradually realized that distributed software coding can fully exploit the talents of each development entity, which enables the development of software projects break boundary in time and region. GitHub, the pioneer warehouse in the field of code hosting, has become the largest open source code social platform for learning, collaboration and communication among scholars and developers worldwide [12, 19]. In order to make it more convenient for users to learn and use, GitHub not only allows users to create, submit, modify, and comment on repositories [11], but also provides users to star, fork, and clone repositories [20]. Users could express their recognition or satisfaction of a repository through the behavior of starring or forking. Therefore, star or fork is regarded as a manifestation of popularity [5, 9, 17].

P. Qin et al. (Eds.): ICPCSEE 2020, CCIS 1258, pp. 258–273, 2020.
https://doi.org/10.1007/978-981-15-7984-4_20

It is challenging for plagues developers to choose the code which are suitable for their task with a good prospect. On the other hand, it is also crucial for enterprises or companies to filter high-popularity code. Hence, an appropriate method or tool for effectively predicting its popularity is urgently needed.

So far, there are only few studies about predicting popularity of GitHub repository. For instance, Borges et al. [4] are the pioneers to focus on the study of this direction. They analyzed the main factors that influence the popularity of repository, including application domain and programming language. Meanwhile, they found and verified several questions about repository popularity. For example, whether the popularity is related to the number of forks, contributors and submissions, how early repository became popular, etc. For further research, they proposed a traditional linear regressions method, which is applied to two models to predict the popularity of GitHub repositories in different situations [3]. They found that the predicted rankings have a strong correlation with the real ones when newcomers are not considered. However, the study remained several defects that cannot be ignored. First, the experiment of estimating the repositories ranks was conducted under a precondition, that is the proposed models were pre-trained with the number of stars received in a period of time, usually about six months. As a result, it works not very well for the newly released repositories. Second, the experiment was conducted on the premise of a hypothesis of newcomers being not considered. Third, it paid more attention to the repositories themselves, ignoring the influence of GitHub users on them. Taken together, a more comprehensive and effective approach is warranted.

As the previous studies show [1, 3, 5, 8], the star is the most intuitive manifestation of popularity. Consequently a stargazer, who gives stars to a repository, plays an essential role in the development of repository. Stargazers may have different degrees of seniority, and the existing work has explained how the stargazers work on the popularity. For example, Blincoe et al. [2] provided a comprehensive analysis about what motivates people to follow others and how the popular users influence on their followers. Experiments indicated that followers are indeed influenced by popular users, and a major way is that popular users usually guide their followers to the projects they newly release or star. He mentioned that a new type of leadership is emerging through the behavior of following and popularity of users can be more influential than their contribution on the followers. Another important behavior in GitHub is unfollowing. Unfollowing is a kind of relationship dissolution actually. Jiang et al. [10] focused on studying the motivation behind unfollowing. They conducted researches and analyzed the potential impact factors that can cause unfollowing behavior. All these studies indicate that in addition to the quality of repository, the following relationship among users plays a very important role in repository popularity.

In this paper, we present a prediction method of repository popularity called StarIn, which is based on the stargazer influence of repository. A high-popularity repository will attract more influential stargazers. Similarly, influential stargazers paying attention to a repository, will bring more widespread attention to the repository and increase its

popularity because they will guide their followers or other users to focus on the repository. Therefore, the aim of this paper is to assess the impact of repository from the perspective of stargazer, and explore the underlying relationship between stargazer influence and popularity in the development process of repository. Specifically, we first establish a stargazer-following based network, and introduce the concept of HFN, which is an evaluation method based on the number of followers of a user. Then we analyze the stargazer influence from three aspects of basic influence, network dynamic influence and network static influence. We carry out an experiment to demonstrate the rationality of our proposed approach. Finally, to evaluate the performance of StarIn, we extract randomly 200 repositories involving different fields and types that the total number of stargazers is over one million. Our experimental results show that the StarIn achieves a good performance of predicting the popularity of repositories with a high accurate rate under two different classification criteria.

Our study provides a number of insights into popularity analysis on GitHub repositories, and compared to previous works, the contribution of this paper is twofold:

- We are the first to evaluate the influence of GitHub repository from the view of stargazer.
- Our proposed StarIn has high prediction performance under different classification criteria.

The remaining of this paper is organized as follows. The workflow and approach we proposed are presented in Sect. 2. In Sect. 3, the rationality of approach is demonstrated. Section 3 also evaluates its performance on popularity prediction, followed by Sect. 4 that discusses our work. Finally, we summarize this paper in Sect. 5.

2 Definition and Methodology

In this section, we will introduce the analysis process firstly, and then present the necessary definitions. Finally, we will describe the approach we proposed to evaluate the stargazer influence. Our workflow is presented in Fig. 1, which consists of the following steps:

1. We collect the GitHub data source, and divide them into the follower, stargazer and repository dataset after preprocessing, and store them in the local database.
2. We establish stargazer-following and stargazer-starring relation based on the datasets.
3. We evaluate the stargazer influence from three aspects of basic influence, network dynamic influence and network static influence, and then demonstrate its rationality and effectiveness.
4. We conduct experiments to analyze the relation between the stargazer influence and popularity from multiple perspectives, and conclude the prediction criteria of repository popularity.

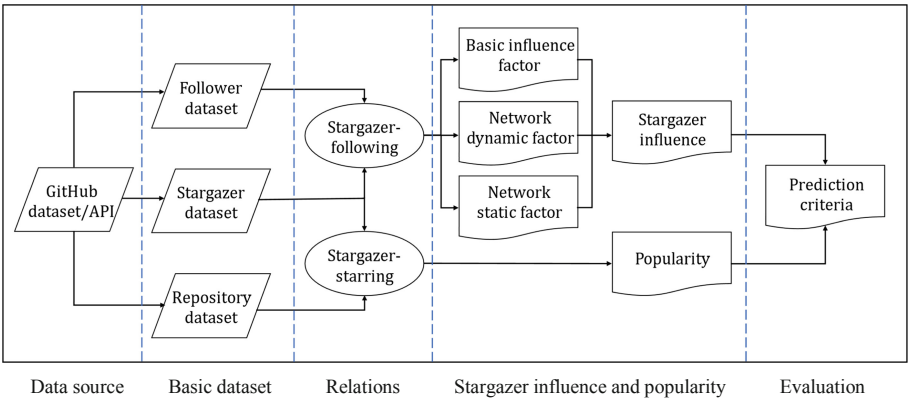


Fig. 1. The workflow of analysis.

2.1 Definition

HFN. In this paper, we define ***HF*** as the followers of a user, and ***HFN*** as the number of *HF*. Give a network of following relationship as shown in Fig. 2. *HFN* is the indegree of node in graph theory.

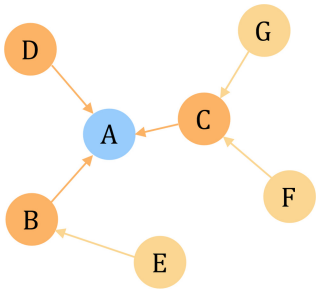


Fig. 2. The network of following relationship. Nodes of the same color are at the same follower level if we choose a target user A.

Each node in Fig. 2 represents a user, and the directed arrow among nodes represents a following relationship, that is, an arrow pointing from f to u represents that u is followed by f. In Fig. 2, suppose A is the target user we analyze, then *HFN* of A is 3, *HFN* of B is 1, and *HFN* of C is 2, etc.

HFN. The network dynamic factor (*NDF*), is an evaluation factor generated by the joint influence of all nodes and link relationships in the network. It has the following characteristics:

1. The weight calculation of the network node is based on the entire network.
2. Adding a new node or relationship affects the weight of all other nodes in the network.
3. Network computing overhead is large.

NSF. The network static factor (*NSF*), is an evaluation factor generated by the targeted nodes and their link relationships in the network. It has the following characteristics:

1. The weight calculation of the network node is based on the network formed by the targeted nodes and finite layer link relationships.
2. Adding a new node or relationship only affects the weight of nodes in the finite layer subnetwork, and nodes outside the finite layer subnetwork are not affected.
3. Network computing overhead is small, but it increases exponentially as the number of layers increases.

2.2 StarIn

The repository will increase a certain number of stargazers per time node (in days). We establish the network of stargazer-following relationship and calculate the influence of each stargazer. It must be pointed out that the stargazer with more followers does not always mean that his/her contribution to promoting the influence of repository is more valuable. This is because there is no direct correlationship between a stargazer's starring behavior and his/her follower number in the network. The behavior itself is the real reflection of high popularity of the repository, and it reflects the basic influence of repository to some degree. Therefore, it is of vital importance to consider the basic influence of repository when evaluating its stargazer influence. Based on this characteristic, our proposed approach to calculating stargazer influence can be formulated as:

$$SI_i = \frac{1-\alpha-\beta}{N} + \alpha * \frac{NDF_i}{\sum NDF} + \beta * \frac{NSF_i}{\sum NSF} \tag{1}$$

where the first term represents the basic influence of each stargazer, NDF_i and NSF_i describe the global and local characteristics of the stargazer in the network respectively. α and β are coefficient, and $\alpha+\beta<1$, N is the total number of stargazers of the repository. We calculate SI_i of each stargazer per day, and take the mean value of SI_i as the day stargazer influence, named as SI_t:

$$SI_t = \frac{\sum SI_i}{n} \tag{2}$$

where t represents the t_{th} day from the date of statistics, and n is the number of stargazers on t_{th} day.

From the date of release, the repository will be in two states: 'popular period' and 'flat period'. No repository will be in 'popular period' all the time. Usually, the repositories have a tendency to become popular right after their first release. After released, the growth rate of the repositories tends to stabilize [4]. When repository pass its 'popular period', it will enter 'flat period', and the maintenance time of 'flat period'

is difficult to estimate. A new topic or an international conference will make the repository active again. Therefore, we need to pay attention to which state the repository is in when evaluating the stargazer influence. If the stargazer influence has been at a low level for a while, in other words, the maximum value of SI_t is below a certain threshold, it indicates the repository enters the 'flat period'. As a result, the influence change caused by the first increase of popularity after the 'flat period' will be discounted, so we introduce the time weight:

$$SI_t = \begin{cases} SI_t * Weight_t, & if\ in\ flat\ period, \\ SI_t, & else \end{cases} \tag{3}$$

where 'flat period' means that the maximum of SI_i is less than a threshold of stargazer influence, and $Weight_t$ is the time weight.

3 Experiment

In this section, we will present the specific calculation factors applied to our proposed approach in the previous section. Then we will demonstrate the rationality and validity of factors. Finally, we will analyze the relationship between stargazer influence and popularity of repository and conclude the prediction criteria of popularity.

3.1 Data Collection

The page[1] released GitHub's datasets of different versions, and we choose follower dataset as basic dataset after preprocessing [7]. The real-time information of repositories or users can also be accessed through GitHub APIs[2].

Follower dataset describes the following relation of users in GitHub. Since not all users follow others or are followed by others, the users in follower dataset are not all users of GitHub. Table 1 displays the overall information of the dataset. Users with followers account for less than 12% of all users.

Table 1. The overall information of follower dataset.

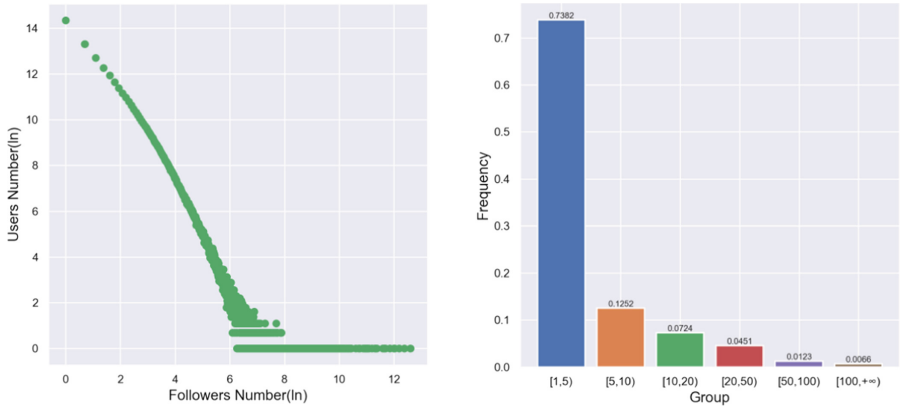
Being followed	Relationships	Following or being followed	Average	Median
3,833,652	29,809,738	5,375,944	7.7	2

We remove users having no followers and only display the users having followers with the scatter and proportion distribution, as shown in Fig. 3.

[1] http://www.ghtorrent.org/downloads.html.

[2] https://api.github.com/repos/torvalds/linux/stargazers for getting stargazer information of a repository, https://api.github.com/users/torvalds/followers for getting follower information of a user.

Figure 3(a) displays the scatter distribution of users with different follower number, where the horizontal axis represents the follower number, and the vertical axis is the user number. We find that when the logarithm of variables is taken, the following relationship is consistent with the power-law distribution in the social network, which can be formulated as $f(x) = cx^{-\alpha}$.



(a) Scatter distribution (b) Proportion distribution

Fig. 3. The user distribution of different follower number.

The proportion of users with different follower number is shown in Fig. 3(b). In Fig. 3(b), the horizontal axis represents different ranges of follower number, and the vertical axis represents the proportion of users of different ranges in all users having followers.

3.2 Selection of Calculation Factors

Based on the characteristics of *NDF* and *NSF*, we choose two methods to calculate the stargazer influence: *PageRank* and *HFN*. *PageRank* is a classic network node link algorithm [15], and *HFN* is an evaluation method based on the follower number. In order to facilitate measurement and analysis, we quantify the influence factors, that is:

$$IF(index), index \in \{PageRank, HFN\} \tag{4}$$

where *IF* denotes the *InfluenceFactor* (hereafter called *IF*), which is the metric to measure the influence of *index*. The *PageRank* algorithm applied to the network of following relationship can be formulated as:

$$PageRank(u) = \frac{1-q}{N} + q \sum_{f\ follows\ u} \frac{PageRank(f)}{Followees(f)} \tag{5}$$

where u is the user to be analyzed the influence, f is a follower of u, $Followees(f)$ denotes the number of users that f follows, N is the total number of users in the network, q is a damping factor, and it is generally set to 0.85. *PageRank* can directly derive the weight of nodes according to the node-link algorithm, and use the weight as the value of Eq. (4), that is:

$$IF(PageRank) = PageRank \tag{6}$$

The most intuitive and convenient way to evaluate whether a user has high impact is to determine the number of the followers for a user. Since *HFN* cannot directly describe the influence of each user through the weight of nodes like *PageRank*, we need to establish a mapping function to calculate the value of Eq. (4).

We first obtain the scatter distribution of *HFN* frequency through the follower dataset, and then use the Least Square methods to fit the scatter distribution to obtain the frequency function (see Eq. (7)). Then we map the frequency to weighted value. Since the user's distribution is a power-law distribution, we choose the function Eq. (8). Finally, considering that the user's frequency changes slightly when *HFN* is high, in order to make the high *HFN* users with differentiation, we perform power calculation on *HFN*, that is Eq. (9):

$$frequency(HFN) = fitting(HFN) \tag{7}$$

$$weight(HFN) = -\log(frequency(HFN)) \tag{8}$$

$$IF(HFN) = weight(HFN) * HFN^{\frac{1}{n}} \tag{9}$$

We group all users in follower dataset by *HFN* to get frequency distribution of different *HFN*, and use linear fitting, cubic fitting, inverse fitting, exponential fitting and exponential-inverse fitting to fit the scatter distribution. Experiments show that the exponential-inverse fitting can make the best fitting performance, and the fitting function is:

$$frequency(HFN) = a * e^{(b/(HFN+0.1))} + c * (1/(HFN+0.1)) + d \tag{10}$$

where a, b, c, and d are the fitting coefficients, which are generated during the fitting process, and different sample data generates different fitting coefficients. For the follower dataset, the coefficients are: $a = -1.61207626e{-}01$, $b = 7.53413079e{-}06$, $c = 2.80668098e{-}01$, $d = 1.55082975e{-}01$. Since the proportion of users with no followers is as high as 90% of all users, we set $weight(HFN)$ of them as 0, and the power exponent of Eq. (9) is set to 1/5.

3.3 Rationality Demonstration

We take the repository[3] as an example (hereafter called Repo_GAN) for demonstrating the rationality of influence factors. Repo_GAN describes a new training methodology for generative adversarial networks, which is currently an area with a high degree of attention in the field of deep learning. Therefore, it is of higher significance and representativeness to study its development.

We calculate the daily average of $IF(HFN)$ of Repo_GAN. As is shown in Fig. 4, the larger the curve slope of *Popularity* is, the higher (i.e. the red and yellow parts) the $Mean(IF)$ is, which suggests that *IF* is consistent with the popularity closely. Then we calculate the stargazer influence of Repo_GAN for a total of 3,730 stargazers, and analyze the relationship between stargazer influence and popularity. Considering the factor of 'popular period' and 'flat period', we introduce the time weight. Figure 5 show the relationship between stargazer influence and popularity after introducing time weight.

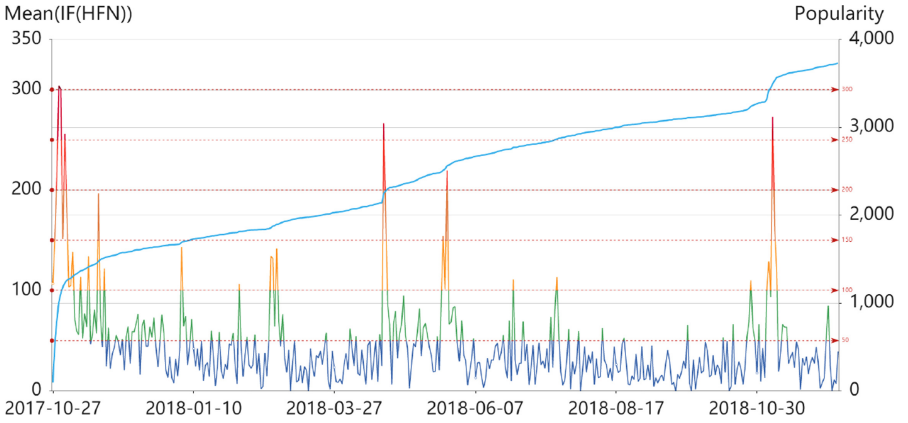


Fig. 4. Mean(IF(HFN)) and Popularity.

We observe that the stargazer influence with time weight achieves a reasonable and effective transition from 'popular period' to 'flat period', which makes the change of stargazer influence on different days more smooth. More interestingly, besides keeping consistent with the popularity on the overall trend, the stargazer influence has a distinct feature. It does not always synchronize with the change of popularity on the same day, but there is a chronological order. From Fig. 5, we can see that the peak of blue curve will be accompanied by red curve. It indicates that the stargazer influence rises earlier than popularity. The time interval is less than 20 days, which means that the popularity of repository will increase significantly in a certain period of time after the rise in stargazer influence.

[3] https://github.com/tkarras/progressive_growing_of_gans.

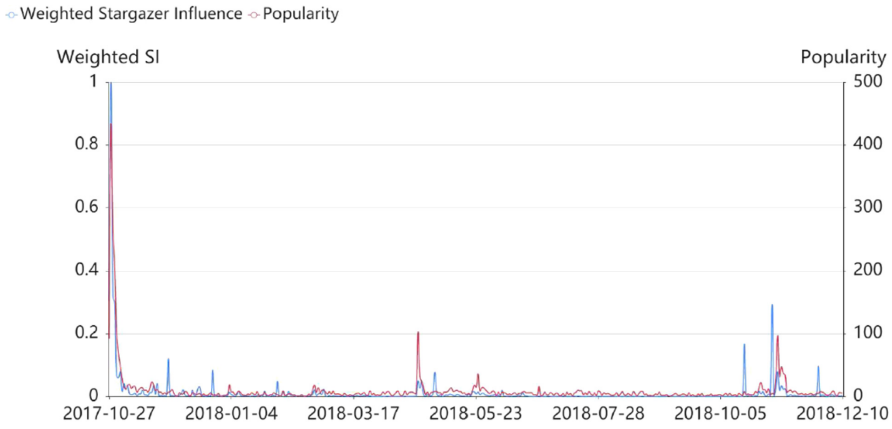


Fig. 5. Weighted stargazer influence and Popularity.

3.4 Evaluation and Results

In order to evaluate the performance of StarIn, a set of 200 repositories are manually selected by 14 software engineers who are working on different directions in software engineering. The 200 repositories involve different fields and types from GitHub. We evaluate the performance of StarIn from the perspective of following six characteristics:

1. The stars increase but the stargazer influence decreases, then the stars will decrease next day.
2. The stars increase but the stargazer influence decreases, then the daily average stars for the next 3 days are lower than the stars on that day.
3. The stars and stargazer influence increase, and the average incremental influence is less than the average influence of all stargazers, then the daily average stars for the next 3 days are lower than the stars on that day.
4. The stars decrease but the stargazer influence increases, then the daily average stars for the next 7 days are higher than the stars on that day.
5. The stars decrease but the stargazer influence increases suddenly, then the stars in the next 20 days increase suddenly (to 30) or keep high (6 stars per day) in consecutive days (5 days).
6. The stars and stargazer influence decrease, and the average reduced influence is greater than the average influence of all stargazers, then the daily average stars for the next 7 days are lower than the stars on that day.

We utilize two classifiers to study the performance of StarIn. One is author-type based classifier, and the other is project-type based classifier.

For the author-type based classifier, we classify the repositories according to whether the author is user or organization [6, 14], and record respectively the prediction results of six characteristics with different influence factor weights in Table 2.

Table 2. The prediction results of six characteristics with different factor weights.

Author type	Characteristic	Best factor weight $(1-\alpha-\beta, \alpha, \beta)$	Prediction Acc
User	Characteristic 1	(0.8, 0.1, 0.1)	0.8066
	Characteristic 2	(0.8, 0.1, 0.1)	0.8275
	Characteristic 3	(0.3, 0.1, 0.6)	0.9378
	Characteristic 4	(0.6, 0.4, 0)	0.7417
	Characteristic 5	(0.7, 0.1, 0.2)	0.8169
	Characteristic 6	(0.6, 0.4, 0)	0.6949
Organization	Characteristic 1	(0.3, 0.1, 0.6)	0.7519
	Characteristic 2	(0.1, 0.1, 0.8)	0.7830
	Characteristic 3	(0.4, 0.1, 0.5)	0.9239
	Characteristic 4	(0.8, 0.1, 0.1)	0.7423
	Characteristic 5	(0.8, 0.1, 0.1)	0.7735
	Characteristic 6	(0.7, 0.1, 0.2)	0.6376

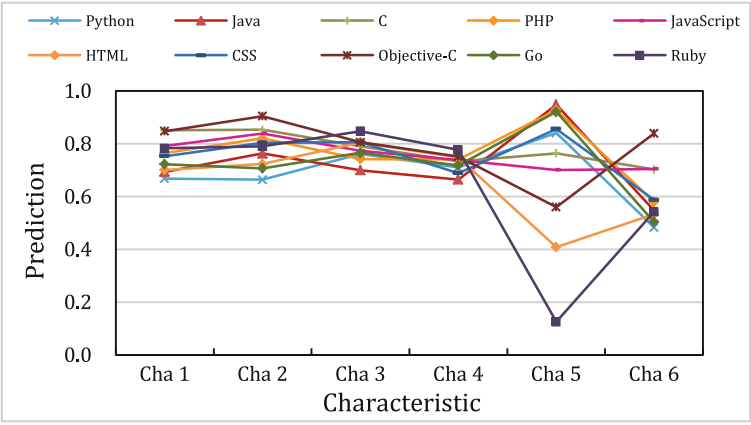
For project-type based classifier, we classify the repositories by languages (Python, Java, C, PHP, JavaScript, HTML, CSS, Objective-C, Go, Ruby) [1, 4, 16] and properties (tool, framework, algorithm, application, document) [13], and calculate the prediction accuracy of the six characteristics separately. The specific meanings of the five properties are as follows:

- Tools usually refer to application interfaces or third-party libraries.
- Frameworks are often the mature and complete components or packages to facilitate the design reusable and extensible.
- Algorithms usually refer to code sets that are used to solve specific problems and have a clear purpose.
- Applications usually refer to software or finished projects.
- Documents usually refer to tutorials for the use of certain tools and software or systematic documents summarized by predecessors on a topic.

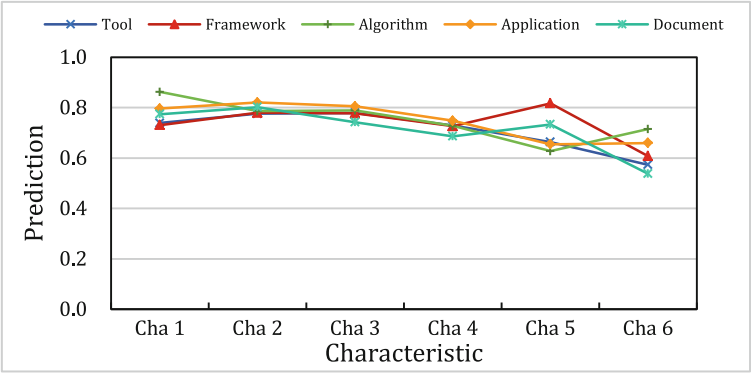
Figure 6(a) and Fig. 6(b) show that the prediction accuracy of six characteristics obtained by classifying the repositories according to languages and properties. It can be concluded the following findings:

A. All languages have high and stable prediction performance on characteristic 1 to 4, with the prediction accuracy reaching 70%–80%.
B. Almost all languages have higher prediction accuracy on characteristic 2 than characteristic 1.
C. Java, PHP and Go have extremely high prediction performance on characteristic 5, and the prediction accuracy can reach over 90%.
D. Characteristic 4 and 5 have obvious symmetry characteristics, that is, the language on characteristic 4 with lower prediction accuracy has higher prediction accuracy on characteristic 5.
E. Except Objective-C, C, and JavaScript, the prediction performance on characteristic 6 is not obvious in other languages.

F. Algorithmic repositories have the highest prediction performance on characteristic 1 and 6, the prediction accuracy can reach 87% and 72%.
G. Except the algorithmic repositories, the prediction performance of repositories of other properties on characteristic 2 is improved relative to the characteristic 1.
H. The repositories of framework have a better prediction performance on characteristic 5 than others.



(a) The prediction performance with languages



(b) The prediction performance with properties

Fig. 6. The prediction performance.

Repositories of different languages and properties have different prediction performance on different characteristics. Most repositories are sensitive to changes in stargazer influence, that is, the popularity of repositories will increase and decrease correspondingly in a short period of time after the stargazer influence increasing or

decreasing. It can be seen from finding D that characteristic 4 and 5 are symmetrical. It indicates that after the stargazer influence rises or even surges, the stars of repository will either rise within a week, or surge or continuously keep high within 20 days. From finding F, G and H, repositories of algorithm and framework have a more positive response to the change of stargazer influence. In contrast, repositories of document, tool and application are insensitive. This is because repositories of algorithm and framework are more technical, and they are likely to be concerned by high-influence stargazers. When the stargazer influence of such repositories rises, it usually means that they begin to enter the public view. For the repositories of document, tool and application, they are usually paid attention to only when the stargazers have requirements. Therefore, the attention of high-influence stargazers to the repositories often does not have a distinct driving effect.

The quality of repositories determines whether it will attract high-influence stargazers. In order to discover the predictive characteristics of different quality repositories, we divide the repositories into 0–5k, 5k–10k, 10k–20k and 20k+ according to the number of stars, and describe the prediction results of six characteristics respectively in Fig. 7.

Figure 7 shows that the repositories have excellent prediction performance on others except characteristic 5 and 6, and prediction accuracy reaches 70%–80%. The number of stars reflects the quality of repository to a certain extent. Therefore, it can be seen from the variation curve of characteristic 5 that the higher the quality of repository, the more significant the prediction performance of characteristic 5. In other words, it is extremely possible that the popularity will become higher in the future after the stargazer influence of high-quality repositories surges. When the number of stars reaches 10 k+, the probability is close to 1. It is not difficult to explain this phenomenon. High-quality repositories will be more likely to be noticed by high-influence stargazers, and their attention will guide more people to notice the repositories. While people who star the low-quality repositories are mostly users with relatively low stargazer influence, the guiding effect they produce is more contingent and accidental, which is difficult to represent the trend of popularity.

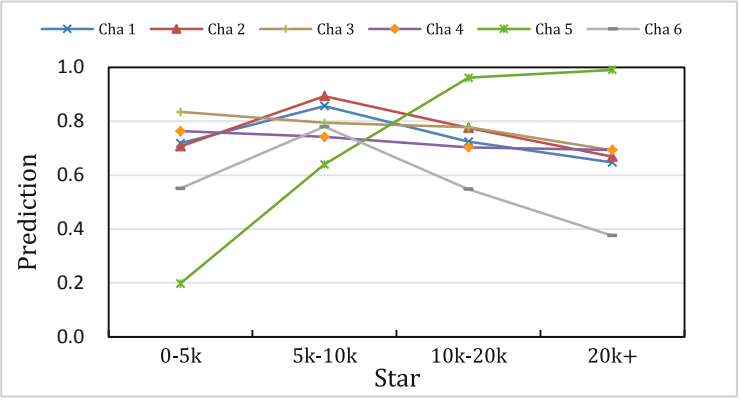


Fig. 7. The prediction performance with stars.

4 Discussion

4.1 Why Does StarIn Work?

StarIn achieves a great success in predicting the popularity of GitHub repositories, and the precision can reach an extremely high level. We conclude several points that can explain the effectiveness of StarIn.

Firstly, StarIn is designed to evaluate the influence of repositories based on the obtained stars, which is the most intuitive manifestation of popularity. It is more convincing than evaluating the influence by measuring the quality of project itself.

Secondly, stargazer is one who gives stars to a repository. Therefore, stagazers play an essential role in the development of repository. A change in the number of stargazers means an increase or decrease of people's attention to repository. In addition, stargazers have different degrees of seniority, the influence of stargazers determines whether the popularity of the project will develop in a good or bad direction.

At last, StarIn is highly universal because it not only considers the impact of stargazers on the repository, but also the basic influence of project itself. The design of StarIn will be suitable for predicting the popularity of repositories with different programming languages, usages and quality.

4.2 Threats to Validity

Our proposed approach is to predict the popularity of GitHub repository through measuring its stargazer influence. There is a threat of the causal relationship between the characteristics of influential stargazers and the construct of popularity. To mitigate this threat, we perform a verification experiment before conducting the evaluation experiment. In the verification experiment, we utilize a representative sample, Repo_GAN, which is one of the 'Top Ten Global Breakthrough Technologies' published by the MIT Science and Technology Review in 2018, to demonstrate that there exists a close relationship between influential stargazers and popularity. Based on this argument, we evaluated the effectiveness of approach on a dataset of 200 repositories.

In our evaluation experiment, we use two classification criteria, namely, author-type based classifier and project-type based classifier. They are widely used classification criteria in previous literature and contain most repository types. We consider that such classification criteria is reasonable. In real-world repositories search, developers usually focus on what the author type or project type of the repository is. In addition, each classification type contains dozens of samples in our experiment. Therefore, StarIn works well even if a new repository comes.

5 Conclusion

In summary, we analyzed the critical factor associated with the popularity of repository, the stargazer influence. We explored a variety of characteristics in the chronological order between the stargazer influence and popularity. We found that the stargazer influence can effectively predicted the popularity of repository. First, we propose the

concept of HFN, and establish influence factor of *PageRank* and *HFN* when evaluating the stargazer influence. Second, we divided the influence factor into network dynamic factor and network static factor according to the characteristics of the factor in the social network, and established an evaluation criteria of stargazer influence. Third, we demonstrated the rationality of influence factor through the relationship between the influence factor and popularity. Finally, we found that the stargazer influence exhibits an excellent prediction performance to the popularity by analyzing the repositories with different languages, properties and stars.

Our results provide an approach to predict the popularity of repositories in GitHub from a new perspective. The approach we proposed is of high extensibility and flexibility because the influence factor is alternative and the parameters is also dynamic to meet different demands. In the future, we propose to develop a method of associating the repositories with papers. Furthermore, we will try to adopt the prediction method to evaluate academic papers, which will contribute to the evaluation of the scientific success with a more comprehensive perspective. Meanwhile, the prediction method will help developers to gain insights into how to choose a high-quality repository for their work [18], and help GitHub to develop a better recommendation scheme, which may promote the development of software or the coding society.

Acknowledgment. This work is supported in part by the Natural Science Foundation of 453 China grant 61502069, 61672128, and by the Fundamental Research Funds 454 for the Central Universities grant DUT18JC39, DUT18GF108.

References

1. Bissyande, T.F., Thung, F., Lo, D., Jiang, L., Reveillere, L.: Popularity, interoperability, and impact of programming languages in 100,000 open source projects. In: 2013 IEEE 37th Annual Computer Software and Applications Conference, pp. 303–312. IEEE (2013)
2. Blincoe, K., Sheoran, J., Goggins, S., Petakovic, E., Damian, D.: Understanding the popular users: following, affiliation influence and leadership on GitHub. Inf. Softw. Technol. **70**(1), 30–39 (2016)
3. Borges, H., Hora, A., Valente, M.T.: Predicting the popularity of GitHub repositories. In: Proceedings of the 12th International Conference on Predictive Models and Data Analytics in Software Engineering, p. 9. ACM (2016)
4. Borges, H., Hora, A., Valente, M.T.: Understanding the factors that impact the popularity of GitHub repositories. In: 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 334–344. IEEE (2016)
5. Borges, H., Valente, M.T.: What's in a GitHub star? Understanding repository starring practices in a social coding platform. J. Syst. Softw. **146**(10), 112–129 (2018)
6. Chatziasimidis, F., Stamelos, I.: Data collection and analysis of GitHub repositories and users. In: 2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA), pp. 1–6. IEEE (2015)
7. Gousios, G.: The ghtorent dataset and tool suite. In: Proceedings of the 10th Working Conference on Mining Software Repositories, pp. 233–236. IEEE Press (2013)

8. Jarczyk, O., Gruszka, B., Jaroszewicz, S., Bukowski, L., Wierzbicki, A.: GitHub projects. quality analysis of open-source software. In: Aiello, L.M., McFarland, D. (eds.) SocInfo 2014. LNCS, vol. 8851, pp. 80–94. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13734-6_6
9. Jiang, J., Lo, D., He, J., Xia, X., Kochhar, P.S., Zhang, L.: Why and how developers fork what from whom in GitHub. Empir. Softw. Eng. **22**(1), 547–578 (2017)
10. Jiang, J., Lo, D., Yang, Y., Li, J., Zhang, L.: A first look at unfollowing behavior on GitHub. Inf. Softw. Technol. **105**(6), 150–160 (2019)
11. Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D.M., Damian, D.: An in-depth study of the promises and perils of mining GitHub. Empir. Softw. Eng. **21**(5), 2035–2071 (2016)
12. Long, Y., Siau, K.: Social network structures in open source software development teams. J. Database Manag. (JDM) **18**(2), 25–40 (2007)
13. Marques, O.: Tools, frameworks and applications for high performance computing: minisymposium abstract. In: Kågström, B., Elmroth, E., Dongarra, J., Waśniewski, J. (eds.) PARA 2006. LNCS, vol. 4699, p. 239. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75755-9_29
14. Munaiah, N., Kroh, S., Cabrey, C., Nagappan, M.: Curating GitHub for engineered software projects. Empir. Softw. Eng. **22**(6), 3219–3253 (2017)
15. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: bringing order to the web. Technical report, Stanford InfoLab (1999)
16. Ray, B., Posnett, D., Filkov, V., Devanbu, P.: A large scale study of programming languages and code quality in GitHub. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 155–165. ACM (2014)
17. Robles, G., González-Barahona, Jesús M.: A comprehensive study of software forks: dates, reasons and outcomes. In: Hammouda, I., Lundell, B., Mikkonen, T., Scacchi, W. (eds.) OSS 2012. IAICT, vol. 378, pp. 1–14. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33442-9_1
18. Sun, X., Xu, W., Xia, X., Chen, X., Li, B.: Personalized project recommendation on GitHub. Sci. China Inf. Sci. **61**(5), 1–14 (2018)
19. Treude, C., Leite, L., Aniche, M.: Unusual events in GitHub repositories. J. Syst. Softw. **142** (1), 237–247 (2018)
20. Tsay, J., Dabbish, L., Herbsleb, J.: Influence of social and technical factors for evaluating contribution in GitHub. In: Proceedings of the 36th international conference on Software engineering, pp. 356–366. ACM (2014)
21. Wang, T., Zhang, W., Ye, C., Wei, J., Zhong, H., Huang, T.: FD4C: automatic fault diagnosis framework for web applications in cloud computing. IEEE Trans. Syst. Man Cybern. Syst. **46**(1), 61–75 (2015)