Predicting Pull Request Acceptance on GitHub from Social Factors

Matthew Heston

Abstract

GitHub is a social website used to host open source software projects. In this study, we collect data from the website attempt to measure a user's activity in a repository, his reputation on the website, and the attention his first contribution to a repository receives. We posit hypotheses about the relationship between these variables and the likelihood that a user's first code contribution is accepted to the project. We find that these variables lack predictive power of pull request acceptance, but present some discussion on how the data relates to previous studies on how members become involved in open source software communities.

1 Introduction

GitHub is a social website that open source software developers use to host their software projects and to browse other developers' projects. It includes many features that are present on social networking sites, such as the ability to follow other users and leave comments on projects. In the study of computer supported collaborative work, GitHub provides a wealth of data, as it is a centralized location where many different tasks take place. For example, users can create bug reports, submit fixes, and engage in discussions about new features all on one website. In this study, we use statistical and machine learning methods on data from GitHub repositories to explore how different social factors may affect the acceptance of code changes from first time contributors.

1.1 Related Work

GitHub itself has not been extensively studied. [2] use data from the website to examine what they call *herding behavior* of developers in open source projects. [9] provide a qualitative exploratory study on how developers on the website measure success of projects. Of importance to our study is their findings that developers believe the GitHub interface has changed the way developers are able to participate in the community. While they describe how features of the website are used by developers to measure community involvement and activity, we are concerned with how these features can provide measures that provide insight into how contributions by new community members are accepted by core members.

While there are not many studies on GitHub itself, there are many studies that explore different open source communities. These include theoretical perspectives on knowledge building and success in open source projects [4] [5] and the motivations of open source developers [6][8].

Our study seeks to identify the relationship between social factors and the acceptance of code contributions of first time contributors. We draw on findings from previous studies that describe the joining process of new developers [7][12]. We also draw from work that describes joining processes not only in open source software, but also from other computer mediated collaborative contexts, such as Wikipedia [1]. Our goal is to provide an empirical understanding of community acceptance on a relatively new social platform.

Outline The remainder of this article is organized as follows. Section 2 begins with a description of some of the terminology specific to GitHub, since many of the features in our models rely on an understanding of this terminology. We then describe how we measured various social factors and present our hypotheses. In Section 3 we describe our experiments. We find that the variables we chose lack predictive power of pull request acceptance. Section 4 describes why our models may have failed and presents some observations from the data. Finally, Section 5 presents our conclusions and ideas for future work.

2 Data

Terminology A software project on the website is referred to as a *repository*. Any user on GitHub can *star* a repository. Users star repositories to be able to easily navigate to it and to receieve updates on activity from the repository. If a developer wants to contribute to another one of developer's repositories, he can *fork* the repository, which creates a copy of the project for him to work on. As the developer makes changes to this code, he *commits* his changes. A *commit* is a snapshot of the code at a certain point in time. When the developer is finished, he can submit a *pull request* to the owner of the project. All pull requests for a project are viewable on GitHub, and any user of the site can comment on them. A pull request can have a status of open or closed. A status of open indicates that that owner of the repository has not made a decision about whether or not to include the changes. If the owner of a repository wants to incorporate the changes the developer made, he can *merge* them into the repository. A pull request can be closed without being merged, which means that the changes the developer made were not accepted.

Data was collected using the GitHub API.[1] We collected pull requests from 45 different repositories. The repositories selected came from the top 100 most starred repositories on GitHub. We chose popular repositories with the assumption that they would be maintained by an active community. We only consider pull requests with a status of closed. This resulted in approximately 44,400 pull requests. We then filter this data set to include only the first pull request a user submitted, which results in 13,383 pull requests. Of these, 4,352, or 32.5% of the pull requests were merged. For our intital experiments, we consider three features of the pull requests.

2.1 User Participation

In their study on members of Wikipedia, [1] note that members initially become involved through peripheral activities. These are simple and low risk activities members can take part in to learn more about the community before trying to become major contributors. Similarly, [12] from observing open source communities generate the construct of a *joining script*, where each project has a set of tasks for new developers to go through before being

[1]http://developer.github.com/

accepted into the community. In the case of GitHub, we consider the act of commenting on previous pull requests to be the main peripheral activity a user can participate in before submitting a pull request of their own. For each pull request in our data set, we then count the number of previous pull request discussions the user participated in before submitting their own pull request, and present our first hypothesis.

Hypothesis 1 *Community members who have been active in previous discussions are more likely to have their code changes accepted.*

2.2 Reputation of User

In addition to participation within the community, we also suspect that a developer's reputation can affect whether or not his pull request is accepted. In particular, we hypothesize that a user who has popular projects of his own is more likely have his pull requests accepted. To measure this property, we use the total number of stars a user's repositories has received. This seems to be a good proxy for measuring the popularity of projects. Unfortunately, we cannot retrieve historical data for stars from the GitHub API. It is therefore not possible to know how many stars a user's projects had at the time he submitted his pull request. We do, however, know the when repositories were created. Therefore, to measure a user's reputation, we look at repositories that were created at the time his pull request was submitted, and sum the current number of stars those projects have at the time our data was collected. This is not a perfect approximation, as some of these projects may have gained many more stars long after the pull request was submitted. However, since we cannot access historical data, we rely on this as an approximation for user reputation.

Hypothesis 2 *Community members who have other popular projects are more likely to have their code changes accepted.*

2.3 Attention Pull Request Recieves

Lastly, we consider the attention the pull request receives, which we measure as the total number of comments on the pull request. Our intuition is that a high number of comments indicates the pull request is generating interest within the community and is therefore more likely to be merged.

Hypothesis 3 *Pull requests with a high number of comments are more likely to be accepted.*

3 Methods

To test the relationship between the variables identified in Section 2, we apply a logistic regression model using these variables as indepedent variables and the binary outcome of merged or not as the dependent variable. The results from the regression are shown in Table 1. There is a positive correlation between user participation and user reputation. Unlike our hypothesis though, the relationship between the number of comments on a pull request and whether or not it is accepted is negative.

Examining the decrease in deviance from the null deviance from our parameters indicates our model significantly ourperforms a null model. However, the χ^2 value for the residual deviance is approiximately equal to 1, indicating that the logit model overall is a poor fit.

In addition to this initial logistic regression model, we train several classifiers using these features. The feature set is first normalized by removing the mean and scaling to unit variance. Class weights for the classifiers are also adjusted given the skew of the dataset towards not merged. The results are shown in Table 2. Although the decision tree and support vector machine classifiers are able to achieve close to 70% accuracy, they also have low recall, indicating that they are doing a poor job of correctly distinguishing merged pull requests.

Given the poor performance in accurately discriminating the two classes, we also attempt to train these classifiers again adding two new features to these existing features. These two new features are the number of commits and the number of changes associated with the pull request. As mentioned previously, a commit is a snapshot of the code at a certain point in time. Different developers might have different commit habits, but in general, larger changes will probably include more commits. The number of changes is calculated simply as the number of lines added and the number of lines deleted. The results are shown in Table 3. The purpose in choosing these features is to look outside of strictly social factors and include features related to the code itself. The addition of these features do not seem to improve our results.

Table 1: Logistic Regression results.

	Dependent variable:
	merged
User participation	0.022^{***}
	(0.003)
User reputation	0.0001^{**}
	(0.00003)
Attention pull request receives	-0.093^{***}
	(0.006)
Constant	-0.496^{***}
	(0.024)
Observations	13,383
Log Likelihood	−8,284.418
Akaike Inf. Crit.	16,576.840
Null Deviance	16882 on 13382 degrees of freedom
Residual Deviance	16569 on 13379 degrees of freedom
Note:	*p<0.1; **p<0.05; ***p<0.01

Table 2: Classifier Results using only our social variables. Testing was run using 10 fold cross validation.

	Accuracy	Precision	Recall
Logistic Regression	59.7%	42.3%	66.0%
Decision Tree	68.7%	52.6%	37.9%
SVM	69.9%	55.0%	41.4%

Table 3: Classifier Results including commits and changes as features. Testing was run using 10 fold cross validation.

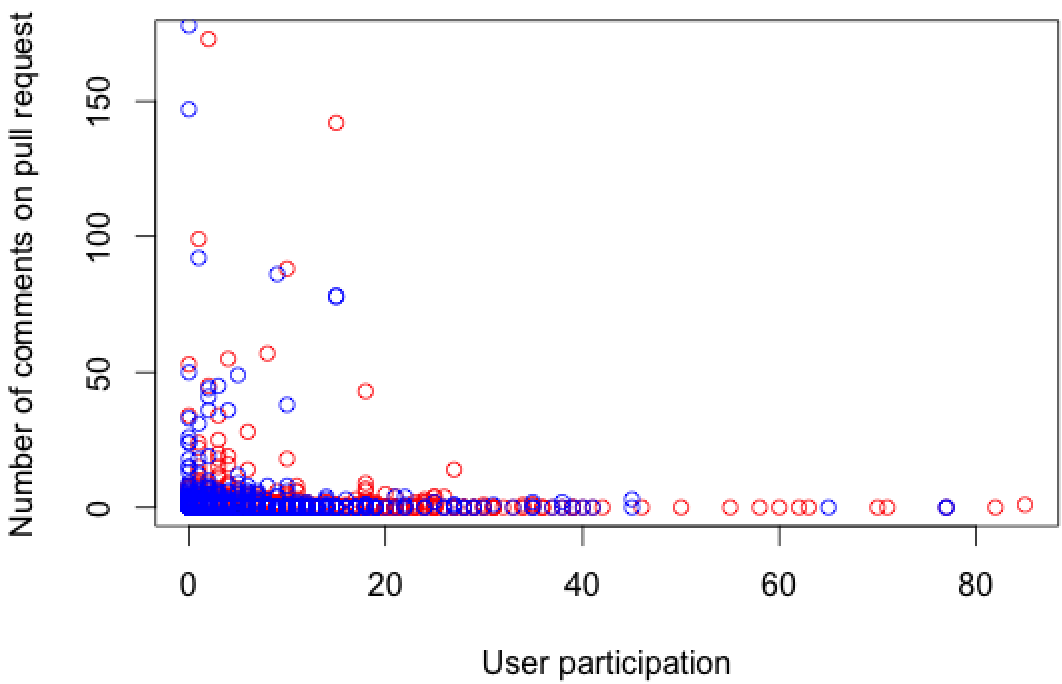
	Accuracy	Precision	Recall
Logistic Regression	61.1%	43.7%	67.7%
Decision Tree	64.8%	45.7%	40.9%
SVM	70.3%	55.3%	45.3%

4 Discussion

The results of the experiments indicate that these social factors are not useful to distinguish between merged and not merged pull requests. Visualizing these variables as we do in Figure 1, this is not surprising. In both cases, most of the values for these features are zero. This in itself is an interesting observation. In evaluating the types of activity which take place in developer joining scripts, [12] note that the most common type of activity is to join an ongoing technical discussion, rather than suggest a technical solution. In a discussion of successful developers on the Python project, [3] describe the first steps of their trajectory as including peripheral monitoring of development activity and reporting of bugs. In both cases, developers first participate in peripheral activities. Both of these studies examine CVS repositories and project mailing lists. The GitHub interface makes it relatively easy for a user to fork a repository, make changes, and submit the changes for consideration. Previous studies on GitHub have shown that the number of contributions did increase for some projects that moved from other hosting options to GitHub [9]. It is possible this interface lowers the barrier of entry for a developer who wants to contribute to a project.

Another interesting observation is the negative relationship between the number of comments on a pull request and the likelihood of it being merged. Manual inspection of some of these pull requests suggests there are several causes for this. For example, it may be the case that the owner of a repository is not interested in merging a change, but other community members think the change is valuable, so they leave comments on the pull request to encourage the repository owner to merge it. In other cases, a discussion might develop between only the person who submitted the pull request and the repository owner where the two of them debate the value of the submitted change. It would be interesting to see if inclusion of things like number of people who commented on a pull request and analysis of linguistic features

Figure 1: Visualization of number of comments on a pull request and user participation variables. Merged pull requests are blue. Not merged are red.



along with the number of comments on the pull request help make it more predictive.

5 Conclusion and Future Work

Although the results of the logistic regression indicate the positive correlation posited in our first two hypotheses, the predictive power of that model, and all our other models are weak, suggesting that these variables are not useful to predict whether or not a pull request is merged or not. Future work should find better features to improve the accuracy of these predictive models.

In this paper, we examined the first pull request users submitted to repositories. It might be useful to further discriminate the types of users who submit pull requests. [10] describe eight different roles members of an open source community assume, including bug reporter, bug fixer, and peripheral developer. In future work, we might continue to examine users' first pull requests, but only those users who go on to become active, stable members, rather than just one off bug fixers. It may be the case that this fixes the problem of our independent variables in both cases clustering around 0, if a majority of those cases come from those one time contributors.

Finally, future work should work to identify how the way both contribution by members outside the community and acceptance of their changes by those in the community change over time. [11] notes that for new members, control and fit are important in deciding whether or not to contribute. It may be that as projects grow, more rigid control structures are put in place, and contribution patterns change over that time.

In all these cases, GitHub as a social platform enables new forms of collaboration and a wealth of data about how that collaboration is taking place, and much more work can be done to explore the social nature of the collaborative efforts of open source developers.

References

[1] Susan L. Bryant, Andrea Forte, and Amy Bruckman. Becoming wikipedian: Transformation of participation in a collaborative online encyclopedia. In *Proceedings of the 2005 International ACM SIGGROUP*

Conference on Supporting Group Work, GROUP '05, page 110, New York, NY, USA, 2005. ACM.

[2] Joohee Choi, Junghong Choi, Jae Yun Moon, Jungpil Hahn, and Jinwoo Kim. Herding in open source software development: an exploratory study. In *Proceedings of the 2013 conference on Computer supported cooperative work companion*, CSCW '13, page 129134, New York, NY, USA, 2013. ACM.

[3] Nicolas Ducheneaut. Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work (CSCW)*, 14(4):323–368, August 2005.

[4] Andrea Hemetsberger and Christian Reinhardt. Learning and knowledge-building in open-source communities a social-experiential approach. *Management Learning*, 37(2):187–214, June 2006.

[5] Andrea Hemetsberger and Christian Reinhardt. Collective development in open-source communities: An activity theoretical perspective on successful online collaboration. *Organization Studies*, 30(9):987–1008, September 2009.

[6] Guido Hertel, Sven Niedner, and Stefanie Herrmann. Motivation of software developers in open source projects: an internet-based survey of contributors to the linux kernel. *Research Policy*, 32(7):1159–1177, July 2003.

[7] Shih-Kun Huang and Kang-min Liu. Mining version histories to verify the learning process of legitimate peripheral participants. In *Proceedings of the 2005 international workshop on Mining software repositories*, MSR '05, page 15, New York, NY, USA, 2005. ACM.

[8] Karim Lakhani and Robert G. Wolf. Why hackers do what they do: Understanding motivation and effort in Free/Open source software projects. SSRN Scholarly Paper ID 443040, Social Science Research Network, Rochester, NY, September 2003.

[9] Nora McDonald and Sean Goggins. Performance and participation in open source software on GitHub. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, page 139144, New York, NY, USA, 2013. ACM.

[10] Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, and Yunwen Ye. Evolution patterns of open-source software systems and communities. In *Proceedings of the International Workshop on Principles of Software Evolution*, IWPSE '02, page 7685, New York, NY, USA, 2002. ACM.

[11] Sonali K. Shah. Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science*, 52(7):1000–1014, July 2006.

[12] Georg von Krogh, Sebastian Spaeth, and Karim R Lakhani. Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, 32(7):1217–1241, July 2003.