How to win a star on GitHub

Michel Kraaijeveld
Delft University of Technology
Delft, the Netherlands
Email: j.c.m.kraaijeveld@student.tudelft.nl

Tom den Braber
Delft University of Technology
Delft, the Netherlands
Email: t.d.denbraber@student.tudelft.nl

Abstract—

I. Introduction

More and more people make use of GitHub to host their Open Source Software (OSS). GitHub is not only a place were people share their software, it has also become a 'social network' for software developers because of it's social features. One of those social features is 'starring' a project. Users can show their appreciation for a project by giving it a star. The amount of stars for a project thus gives an indication of how many people appreciate that project.

However, there might be several factors that influence the amount of stars of a project. One can think of the functionality that the project offers, but what about projects that offer more or less the same functionality but have a greatly different amount of stars?

This paper will be looking at the stars given to OSS projects and whether certain factors can have influence on the amount of stars a project can possibly get, and if there are key points to focus on when aiming at having the most stars as possible.

II. Problem Description

Although the amount of stars for a project are far from the only indicator of a project's success, they give an indication of how many people actually like the project. For developers, it might be important to know how they can gain the appreciation of other GitHub users: the more people that use or collaborate to a project, the faster it will evolve.

We will now further concretize the subject of this research. The main research question is as follows: 'Can we predict the amount of stars a project on GitHub will get when looking at its characteristics?' This high-level question can be split up in several other questions:

- Which factors influence the popularity of a project?
- How can we explain the influence of those factors?
- To what extent can we accurately predict the amount of stars on a project?

In order to answer these questions, a dataset which contains a lot of data concerning the projects on GitHub is needed. This dataset is available: GHTorrent [1] is a dataset which is constantly mining the GitHub application for more data on projects, and contains all the data that we need for this research.

III. Methodology

Now that the research questions have been discussed, the approach for answering those questions can be explained. We will start by detailing our sampling method. Thereafter, we introduce the features which we think do have influence on the amount of stars that a repository has. Lastly, we discuss the methods for actually constructing a learning model that we can use to predict the amount of stars for a given project.

A. Sampling Method

When we did a first exploration of our data, we found that the distribution of stars per project is extremely right skewed: there are a lot of projects with one or zero stars, while there are much less projects which have a lot of stars, as can be seen in Figure 2. Because of this skewness, the decision was made to use so-called stratified sampling. With stratified sampling, we first have to divide the population into homegenous strata. Thereafter, we sample an equal amount of projects from each of these strata; all these samples together form the sample that is being used in the rest of the project.

Each stratum has as characteristic that the amount of stars of the project in that stratum fall within a certain range. The ranges that will be used in this project are: [0, 10], (10, 100], (100, 1000], (1000, +]. Because the dataset is very large, we can also create a quite large sample: out of each stratum, we will randomly select 500 projects for each stratum. The total size of our sample is thus 2000.

Further, we split up our sample in a training set and a testing set. Out of our sample, we randomly pick 1000 projects which will form our training set. The remainder of the sample will be used as a testing set.

Although we mentioned that the selecting of projects is completely random, it actually is not. In order to reduce bias in our sample, we decided to filter out any project by Google, Microsoft or Apple. The way in which they use GitHub is different from most other projects, but they do have a lot of stars, partly due to brand awareness; therefore, they are not included in our sample.

B. Features

In order to be able to predict the amount of stars an OSS project will get, a model need to be designed. Therefore a number of features is discussed first, which could influence the amount of stars on a project. Each of these features fall into one of the high-level features that are specified:

general project characteristics, popularity of developers or organisation, and activity. Next to these high-level features, the domain is also considered as that is another aspect that can influence the amount of stars. After the features are defined, we can use them in combination with a machine learning algorithm to create the actual prediction model.

1) General project characteristics: The first high-level feature contains general features belonging to a project. A feature is considered a general project feature if it cannot be categorized into the other high-level features. In this section the different features and reasons behind choosing them will be explained.

Number of commits This is the total amount of commits since project creation. A commit can either be an addition from one of the project developers, or a merge from an accepted pull request. When the number of commits is larger, this might indicate that a project attracts more users.

Project country of origin The country of origin refers to the country in which the project was intially created. This can either be based on the developer that created the project, or on the majority of developers in the organisation

Number of commits per developer This is the total amount of commits that each of the developers in the project have. These commits do not include merge commits, so it gives an indication of the activity of each of the developers.

Number of forks The number of forks is the total amount of forks that the project got since its intial creation. Since forking a repository means that someone is interested and possibly want to make use of the project as a resource for their own developments, it can be a good indication of the amount of stars a project might possibly get.

Number of pull requests This numerical variable contains information about how involved contributors are to the project: do they make a lot of pull requests or not?

Ratio: accepted/total pull requests This numerical variable contains the number of accepted pull requests divided by the total number of pull requests. It is an indication of how 'open' the project is.

Total amount of contributers Someone is considered a contributer when he or she can directly commit to the repository (developer) or when a pull request is merged into the project (external-developer). When a project has more contributers, it might give a prediction on the amount of stars the project will possibly get.

Main programming language The language that is used most throughout the project is considered the main programming language. Since some languages are easier to learn and understand, projects that use that language might be able to attract more users and therefore stars.

How long does the project exists Newer project might have fewer stars, since they are not discovered by the majority of the GitHub users. Therefore the time a project exists, might have influence on the amount of stars the project currently has.

2) Popularity of developers or organisation: When a developer or organisation is already popular on GitHub, a new project created by them can get attention from other users more easily. This high-level feature therefore contains features that can be used to measure the popularity of a developer or organisation in order to be included in the prediction model.

Average number of followers per developer This is the average number of followers between all developers that contribute to the project.

Maximum number of followers per developer This number indicates the maximum amount of followers the developers that contribute to the project have.

Number of developers in organisation If the project is created by an organisation, this number indicates the amount of developers that belong to the organisation.

Average number of followers per developer in organisation Again, if the project is created by an organisation, this number indicates the average amount of followers the developers in the organisation have.

Maximum number of followers per developer in organisation This feature is like the previous one, except it now shows the maximum amount of followers a developer in the organisation has.

3) Activity: Another thing that we think can have an impact on the amount of stars a project gets, is the activity on that project: is it actively maintained or not?

Number of commits per day This numerical variable measures the average number of commits per day.

Number of releases Although not all projects use the releases feature of GitHub, it is interesting to see if this numerical variable does have influence.

Time between releases This numerical variable is another measure of activity: it is the average time measured in days between each release for a given project.

4) Domain: Since some project domains are more popular, it is easier to get stars for projects in that domain, simply because more people are interested in that domain. Therefore the domain is also considered in the model, to be able to make better predictions as features might weight different across domains. Unfortunately, the domain of a project cannot be automatically deduced from the dataset. Therefore, each project in our sample was given a domain manually. Domain is obviously a categorical variable and can take the following values: decide on possible domains.

C. Finding relations

Now that the features have been listed, we can try to infer relations between those features. More specifically, we have an independent variable - the amount of stars for a project - and we will try to correlate it with the independent variables - the features listed in the previous section.

The machine learning algorithm that first comes to mind is linear regression. However, some of our features are categorical, e.g. the programming language of a project, which makes it impossible to use linear regression. Luckily, there is a

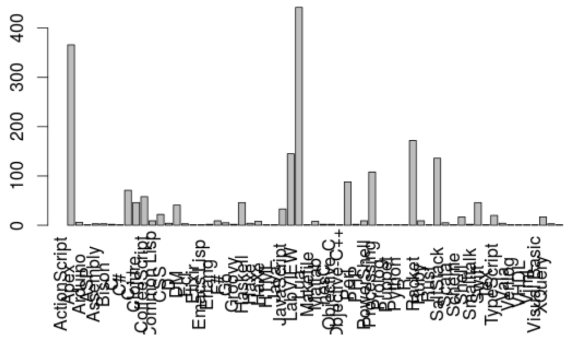

Fig. 1. A histogram of the languages used in the projects

variant available which is able to handle categorical variables; this algorithm is called ‘multiple linear regression’. Initially, this algorithm will be used. Depending on the results, we will try out other algorithms like random forest and naive bayes.

On top of the initial analysis using multiple linear regression, an attempt will be made to finetune our model so that the prediction is as accurate as possible. The step-wise regression algorithm will be used for this task.

IV. Results

A. Initial Data Exploration

During the initial data exploration, some of the features that were discussed earlier were gathered. Due to the fact that some features are quite complex to retrieve, the initial retrieval did not cover them all. During the later stages of this project, we would like to make all our features visual in some way or another; we found that this gives valuable insights regarding the structuring of the data we are dealing with.

To actually make the data that was gathered more useful, some plots were made. Figures 1, 2, 3 and 4 are generated from the data that was gathered from the sample. An important observation is that almost all the data that was gathered is strongly right skewed. This implies that we might need to normalize it in the future, but that decision is yet to be made.

B. Testing the model

Will be added later on, same for threats to validity

V. Related Work

Previous work conducted by Chen et al. [2] aimed at predicting the amount of forks for a given repository on GitHub. Our approach was similar to their’s, as they had divided their data into different strata in order to get a good sample space. Furthermore, they also had multiple features to create a model to predict the amount of forks. The results of their research indicated that they could accurately predict the amount of forks for repositories on GitHub and provided valuable insights on the potentials for predicting properties of a GitHub project.

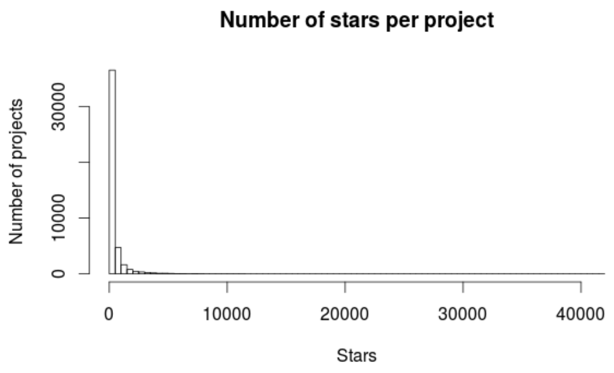

Fig. 2. A histogram of the number of stars per project

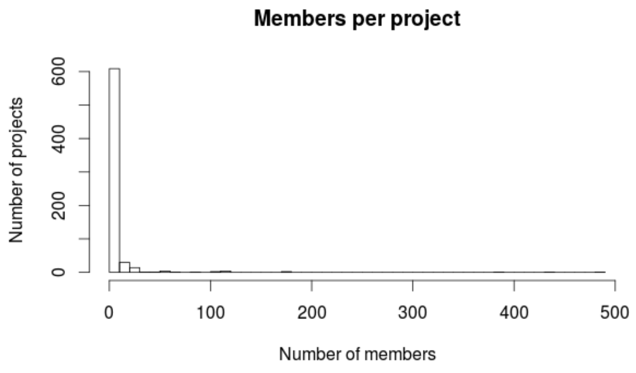

Fig. 3. A histogram of the number of members per project

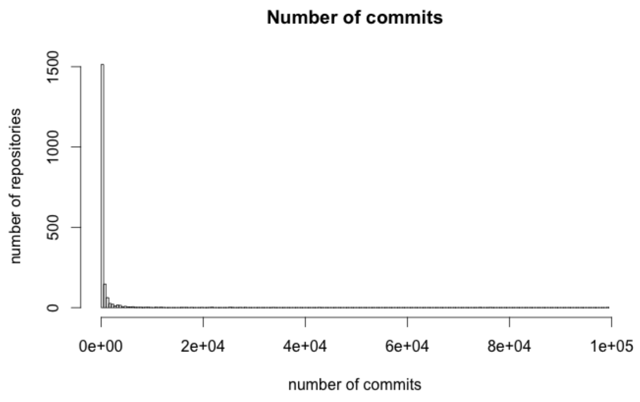

Fig. 4. A histogram of the number of commits per project

Recent research by Blincoe et al. [3] on popularity of GitHub users, reveals that GitHub users can be influenced by each other to join new projects. They carried out this research in two ways: by conducting a survey to find out motivations for people to follow others on GitHub and by repository analysis to see the actual influences. Their findings showed that the main reason for following a user on GitHub, is the benefit of having updates and keeping up-to-date with the activities the user participates in. These activities also include joining or discussing new projects, which popular GitHub user attract their followers to.

Research [4] into reported issues for projects on GitHub also provided useful insights. They looked at 100.000 GitHub repositories to see which factors influence the amount of issues that are filed for a project. Their results showed that there is a correlation between the amount of issues and the popularity of the project, where they based popularity on stars - which resulted in a high correlation - or forks - which resulted in a very high correlation.

VI. Future Work

Something that could be of influence on the amount of stars that is not considered in this paper is the documentation of a project. When a project is documented well, or not documented at all, this might have an influence on the appreciation users show for that project. The reason this was not considered in this paper, is because it is hard to measure documentations throughout different projects as some have extensive documentation on their own website, while they are not using the GitHub features for documenting the projected. Furthermore, at the moment of writing it is not possible to check if a project has created a readme file or a wiki page in their repository, as the GHTorrent dataset does not include the files itself. For future research it might therefore be interesting to investigate this feature and include it in a prediction model.

VII. Conclusion

References

[1] G. Gousios, "The ghtorrent dataset and tool suite," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, (Piscataway, NJ, USA), pp. 233–236, IEEE Press, 2013.

[2] F. Chen, L. Li, J. Jiang, and L. Zhang, "Predicting the number of forks for open source software project," in *Proceedings of the 2014 3rd International Workshop on Evidential Assessment of Software Technologies*, EAST 2014, (New York, NY, USA), pp. 40–47, ACM, 2014.

[3] K. Blincoe, J. Sheoran, S. Goggins, E. Petakovic, and D. Damian, "Understanding the popular users: Following, affiliation influence and leadership on github," *Information and Software Technology*, vol. 70, pp. 30 – 39, 2016.

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