OssAssist

A system to assist open source organizations plan and forecast projects!

Nitin Kodial, Rudra Purohit, Biswajyoti Pal, Deepthi Venkitaramanan

Introduction

Open Source Software (OSS) organizations like Mozilla and Apache are responsible for maintaining plenty of mission critical software just like commercial software providers. However, unlike commercial organizations, they do not have engineers on payroll and depend on geographically distributed open source contributors. Hence their planning and forecasting needs are different and subject to greater volatility. Our project enables OSS organizations to predict the number of contributors and visualize key contribution indicators of their projects.

Problem Description

Presently, organizations like Mozilla track the number of contributors in various projects and run simple forecasting for the next time period. Also, only a small number of performance indicators, related to issues, are tracked, analyzed and visualized, on spreadsheets. Mozilla had expressed an early interest on systems running advanced analysis on their projects.

We developed a system which gathers different attributes of the various projects run by OSS organizations like Mozilla from online project hosting platforms like Github. The system utilizes this data for two purposes. Firstly, the system predicts the number of contributors to be available for the project in the next cycle using attributes of the projects and their repositories, extracting features and applying machine learning algorithms. Secondly, the system presents visualizations of certain key contribution indicators of the projects which can additionally help in planning and analysis of these projects. These visualizations are presented in a semi-realtime web based dashboard for quicker and easier analysis.

Literature Review

OSS projects and their contributors have been studied in great detail. Mockus et. al. [4] and Comino et. al. [5] present that as the number of contributors grow, the quality of the OSS projects improve along with the chances of project progress, but the progress of the projects begin to degrade slightly and shows signs of coordination problems. Studies utilizing surveys and archived contributor communications suggest that one of the primary motivation for OSS contributors to contribute to the bigger OSS projects is identification with the larger OSS community [1] and there is high correlation between this motivation and performance of the contribution [3]. While there is an increasing community effect among the contributors, growth of a project is more related to the share of corporate contributions than individual contributors [2]. This is because of the fact that corporate contributions tend to be less volatile and more manageable. Hence increasing the efficiency of planning and management of individual contributions in OSS projects is an important need for every OSS organization.

Understanding various aspects of individual contribution is also valuable in predicting number of future contributors. For every set of significant OSS projects, there exists a collaboration graph between the contributors, amongst each other and across projects [7]. Certain contributors come out to be as more influential. This socio-technical network impacts future contributions [8] and project outcomes [6].

Github is a fertile ground for data mining software repositories. Ray et. al. [9] extract features like programing languages used and their types, project domains (from project description), bugs categories (from issue text) to find interesting correlations between them. Borges et. al. [12] utilize features from metadata of repositories like owner, date and frequency of release along with features similar to Ray et. al. to predict the popularity of projects. Gousios et. al. [11], on the other hand, also dive deeper into the metadata of code commits along with repository and contributor characteristics to explore and analyze pull requests. Thus, while features extracted from such metadata should have strong predicting power with respect to number of contributors, it remains to be validated through analysis.

Ben et. al. [10] utilize visualization techniques to analyze developer contribution temporally across their period of contribution and also geographically. They do not attempt to predict the number of contributors. Moreover, we shall also attempt to use

visualizations for repository indicators rather than developer indicators. Robles et. al. [13] present an empirical methodology to estimate development effort in OSS projects. We shall attempt to predict the number of contributors, a factor influencing development effort and planning, using machine learning techniques.

Proposed Methodology

Intuition: Simple forecasting techniques utilize the number of contributors for a repository as a time-series data and use just the historical values of the variable to forecast the future value. We wanted to improve the prediction using a variety of other features obtained from the attributes of the repositories and by using a machine learning approach. Moreover, we distinguish our visualizations by presenting the ability to drill down from overview to details of the contribution on a combined current and future view.

Data Collection: We have utilized the GitHub API to obtain data corresponding to 'Mozilla', as we shall be building and validating our system w.r.t 'Mozilla'. However, our system can be configured to collect data and build models for any organization. The data collection module obtains information regarding repositories, the users interested in them, their releases, activities and statistics. In addition to these JSON data files, we also gathered 'diff' and the 'patch' files for the corresponding commits and pull requests. In total, we generated a data corpus with a size of 15G with the Commits data alone containing 7GB worth of data.

Data Cleaning and Integration: Given the variety of schema of the different types of data, it was decided 'MongoDB' was a better fit as our project database. Thus the different data files were integrated into a 'MongoDB' database. The data files were cleaned using a data cleaning module, developed in Python, to transform the data into a format recognized by 'MongoDB' and were imported into the database.

Feature Engineering: We started off with a seeding set of features for the prediction model -

1. Number of pull requests on the repository
2. Number of releases of the repository
3. Number of branches of the repository
4. Number of tags of the repository
5. Number of issues in the repository
6. Number of events in the activity stream of the repository and the issues
7. Number of comments on the repository, the issues and the reviews.

These attributes were extracted for a time period T where T is the date when the system will be used to predict the number of contributors at $T + \Delta T$ and plan for ΔT period. An additional feature of number of commits to the repository was added to the above list.

Subsequently, experiments utilized these attributes till T from the creation of the repository at every ΔT of time. One of the major addition in these sets of experiments was the addition of the number of contributors from the creation of the repository at every ΔT of time. Thus our features incorporated the various aspects of the repository as well as the historical trend of the number of contributors. This helped us combine the best of both worlds - forecasting and classification.

Supervised Learning: There were two sets of tasks to be performed on the above features - regression and classification. The independent variables was the features at either T or every ΔT of time for as discussed above, the dependant variable was the number of contributors at $T + \Delta T$.

The dependant variable was predicted in two ways - continuously and categorically. For predicting the number of contributors continuously, we used linear regression models. For predicting the number of contributors categorically, we utilized Jenks natural breaks optimization to break the number of contributors into few categories. We ran experiments with various types of nonlinear classification algorithms.

Validation: The latest $T + 2\Delta T$ for which number of contributors is available is considered. ΔT period was experimented with. The features at T and at every ΔT in the past and number of contributors at $T + \Delta T$ are extracted and were used to build the classifiers. We ran two types of validation for the classifiers. The classifiers were internally validated by using 80% of the repositories as the training set and 20% of the repositories

as the testing set. The classifiers were then externally validated by using features at $T + \Delta T$ or at every ΔT in the past on number of contributors at $T + 2\Delta T$.

System Integration: The system derived using the above methods was built as a predictive tool.

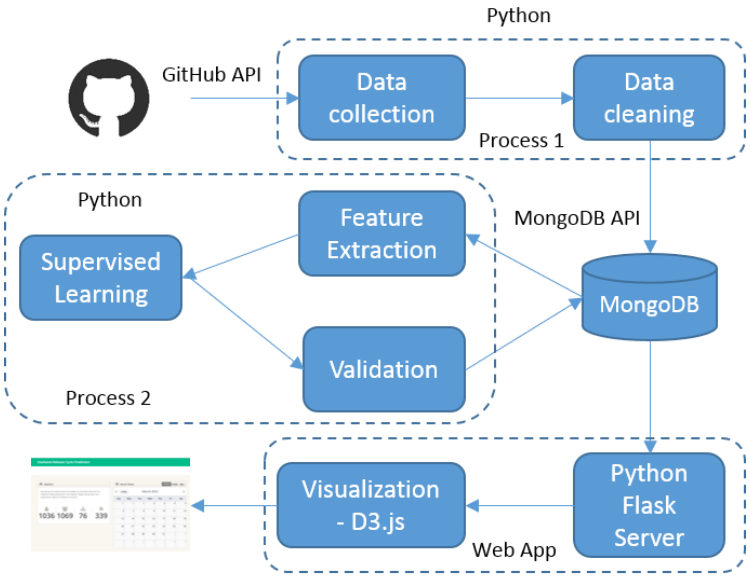

Figure 1: System Summary

The system runs two offline processes which can be set up as scheduled jobs. The first process runs the Python script for data collection, cleaning and integration. The data obtained is stored in a MongoDB database. The second process runs the feature extraction and supervised learning methods using another Python script. The required inputs are obtained from the MongoDB database and the outputs are stored back into the same database. The visualization system obtains the required data from the database and presents it using a web application with a Python Flask based server and D3.js based frontend.

Visualization: We identified certain contribution indicators to be presented to enable further project analysis and planning.

In accordance with the Shneiderman's Visual Information-Seeking Mantra, we have two overviews in our visualization - the prediction overview and the analysis overview.

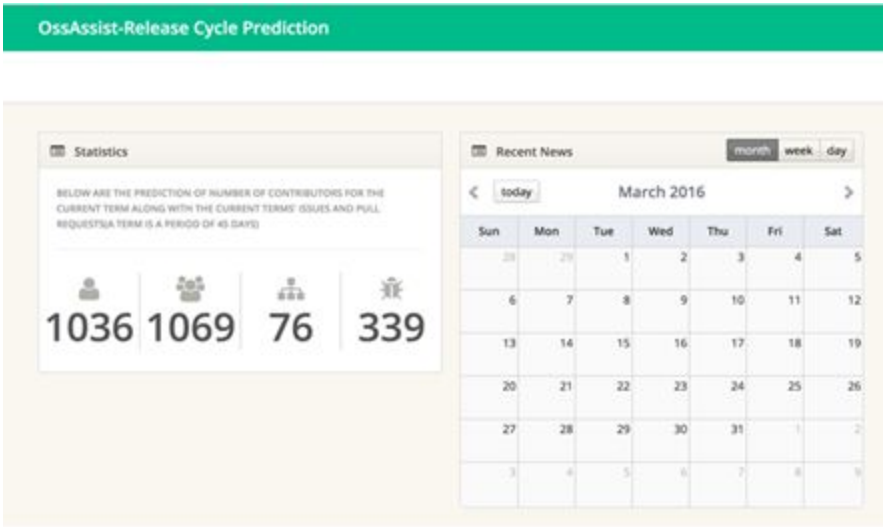

Figure 2: Prediction Overview

The prediction overview presents the prediction of the number of contributors for the entire organization along with the number of issues and pull requests currently open on the current date. The calendar format enables the user to navigate into a future date to see the prediction or a past date for analysis and checking how the predictions did.

Figure 2: Analysis Overview – Repositories

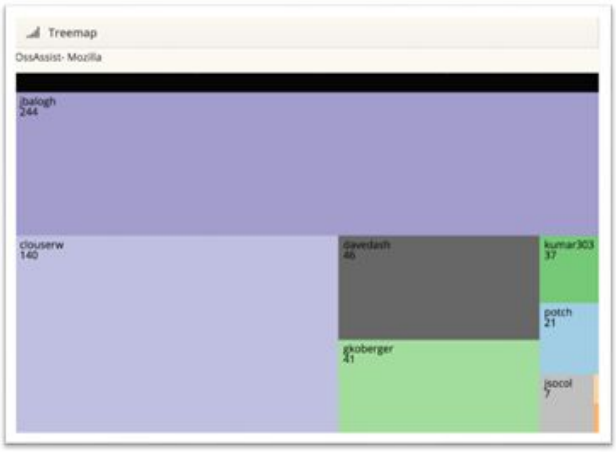

Figure 3: Analysis Detail - Contributors

The analysis overview essentially is a 'treemap'. As the overview, it presents current and the predicted number of contributors for the entire organization. A user can then drill down to the various repositories for the same information. From the repositories, the user can drill down to the users who actually contributed to the repositories and their contributions.

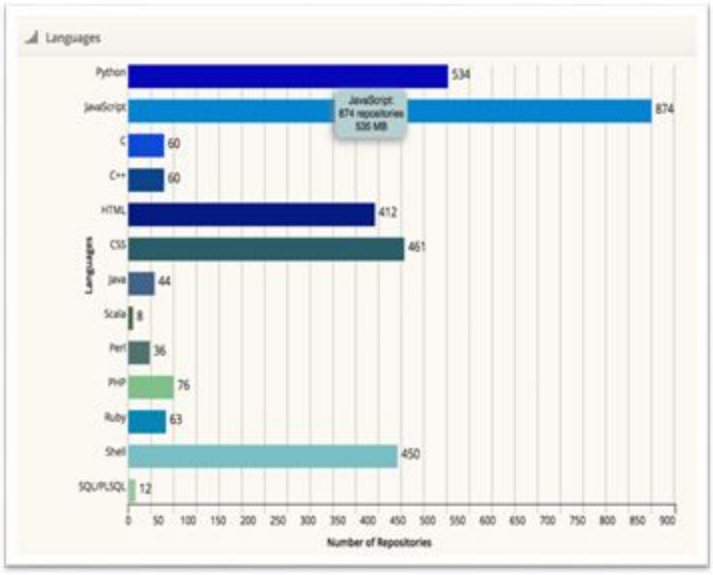

Figure 4: Analysis Overview – Languages

We have a secondary analysis overview of the programming languages used across the organization.

These visualizations have been specially identified from the data source with respect to their utility in terms of planning OSS projects.

Some of the analytical questions which these visualizations help explore are -

- How is the organization doing in terms of open source activities?
- Which repositories deserve more attention while planning?
- Who are the bigger contributors?
- Which programming languages should have more focus while planning and conducting contribution drives?

Experiments

As mentioned above, we ran various experiments with different feature extraction models and supervised learning models, using different validation techniques and parameter ΔT.

Linear Models with internal validation: To test the validity of our approach, we ran an early experiment with the seeding set of features, ΔT = 45 days considering only features at T. A linear models using Multiple Linear Regression with L1 prior as regularizer (Lasso) and with combined L1 and L2 priors as regularizer (Elastic Net) fitted the attributes the number of contributors with R2 value at 0.6046 when internally verified. This presented the preliminary predictive power of these features.

Nonlinear Models with internal validation: We conducted the above initial experiment using Nonlinear models. Using Extremely Randomized Trees, the classifier is able to predict number of contributors 18% of the time when internally verified. The continuous number of contributors were considered categorical to fit into the multi-class classification model. It was concluded that this was the cause for the low accuracy and hence prompted our use of Jenks natural breaks optimization to break the number of contributors into few categories.

Linear Models with additional features with external validations: Internal validation seemed like an interesting initial method for validating the models, but for our use case a more rigorous analysis with external validations was needed. This is because, for the use case of this predictive tool, validation using backfitting past data was more important. The below table presents the R2 scores and the mean squared errors for linear models ran at T = 1st of July 2016, with ΔT = 45 days, for 12 ΔT and additional features of number of commits and past contributor for the entire organization.

Model	R2 (features at T)	R2 (features at 12 ΔTs)	MSE (features at T)	MSE (features at 12 ΔTs)
ElasticNet	0.5912	0.6524	34.44	39.13
Lasso	0.5921	0.6522	34.37	39.16
Linear Regression	0.5824	0.6475	35.18	39.68
Decision Tree Regression	0.3104	0.5171	58.10	54.36
ExtraTreesRegressor	0.5211	0.5158	40.35	54.51
KNNRegressor	0.4885	0.3964	43.09	67.95

Table 1: Performance of Linear Models for ΔT = 45 days

The additional features and the different feature models enabled us to obtain a better R2 score - 0.65 compared to 0.60. The mean squared error is manageable considering that the number of contributors predicted is nearly 1000. Also, linear regression models performed better compared to Nonlinear regressors.

Nonlinear Models with additional features with external validations: The above experiment was repeated using Nonlinear classification models after breaking the the number of contributors into 20 categories using Jenks natural breaks optimization.

Model	Accuracy (features at T)	Accuracy (features at 12 ΔTs)
DecisionTreeClassifier	0.2443	0.2802
AdaBoost Classifier	0.2452	0.2780
RandomForest Classifier	0.2479	0.2672
SVM Classifier	0.2590	0.2435
Naive Bayes Classifier	0.1139	0.1336
MLP Classifier	0.2314	0.1336

Table 2: Performance of Nonlinear Models for ΔT = 45 days

While the changes in the methods did result in increase in the accuracy from 18% to 28% (for the best classifier), supervised classification still is not very useful in our case.

Nonlinear Models with additional features with external validations with different ΔT: In order to understand the effects of ΔT, we conducted the same experiment at ΔT = 30 days and ΔT = 60 days. While increasing the ΔT does improve the accuracy, these models still considerably lag behind the linear models.

Model	Accuracy using features at T (ΔT = 30)	Accuracy using features at 12 ΔTs (ΔT = 30)	Accuracy using features at T (ΔT = 60)	Accuracy using features at 12 ΔTs (ΔT = 60)
SVM Classifier	0.2410	0.2463	0.3139	0.3149
RandomForest Classifier	0.2303	0.2488	0.3007	0.2941
DecisionTree Classifier	0.2459	0.2377	0.3194	0.2872
MLP Classifier	0.2438	0.1556	0.2907	0.2180
AdaBoost Classifier	0.2353	0.2096	0.3051	0.1246
Naive Bayes Classifier	0.1921	0.1434	0.0352	0.0830

Table 3: Performance of Nonlinear Models for ΔT = 30, 60 days

Nonlinear Models with additional features with external validations with different ΔT: The earlier experiments concluded that linear models were a better fit for our dataset. Thus we conducted experiments on linear models to understand the effect of ΔT, by varying ΔT to 30 and 60 days. While increasing ΔT from 45 to 60 days did not result in a significant changes in R2 scores (from 0.65 to 0.66), the error increased from 39 to 54. When ΔT was decreased from 45 to 30 days, there was a significant improvement in both R2 score (from 0.65 to 0.74) and error (from 39 to 17). All cases of ΔT, presented the feature model of having every past features from T to be a better performer. Multiple Linear Regression with regularizer seemed to be the regression model standing out.

Model	R2 at T (ΔT = 30)	R2 at 12 ΔTs (ΔT = 30)	MSE at T (ΔT = 30)	MSE at 12 ΔTs (ΔT = 30)	R2 at T (ΔT = 60)	R2 at 12 ΔTs (ΔT = 60)	MSE at T (ΔT = 60)	MSE at 12 ΔTs (ΔT = 60)
Lasso	0.6380	0.7380	19.67	17.53	0.6400	0.6604	34.82	54.22
ElasticNet	0.6380	0.7369	19.67	17.60	0.6376	0.6603	35.06	54.24
Linear Regression	0.6264	0.7245	20.31	18.43	0.6262	0.6529	36.15	55.42
ExtraTrees Regressor	0.5841	0.6418	22.54	21.09	0.5107	0.3514	47.32	103.56
Decision Tree Regression	0.1865	0.4643	42.90	37.66	0.2640	0.4872	71.19	81.88
KNN Regressor	0.5973	0.4033	21.88	39.93	0.5096	0.4085	47.43	94.45

Table 4: Performance of Linear Models for $\Delta T = 30, 60$ days

Conclusions and Discussion

We successfully present a predictive tool to enable OSS Organizations predict the number of contributors and visualize certain contribution indices for more effective project planning and analysis. Such a tool can effectively collect data from Github using their APIs, clean and store the data. The data contains sufficient features to predict the number of contributors, both current and historical. The features interact linearly especially over shorter durations of time. Hence, Linear Regression with regularizers can be used for the predicting around a month into the future. A web based system can effectively the prediction as well as the current state of contribution to the planner.

Distribution of team member effort

All team member contributed similar amount of effort.

References

[1] Hertel, G., Niedner, S., & Herrmann, S. (2003). Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. Research policy, 32(7), 1159-1177.

[2] Lerner, J., Pathak, P. A., & Tirole, J. (2006). The dynamics of open-source contributors. The American Economic Review, 96(2), 114-118.

[3] Roberts, J. A., Hann, I. H., & Slaughter, S. A. (2006). Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the Apache projects. Management science, 52(7), 984-999.

[4] Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology (TOSEM), 11(3), 309-346.

[5] Comino, S., Manenti, F. M., & Parisi, M. L. (2007). From planning to mature: On the success of open source projects. Research Policy, 36(10), 1575-1586.

[6] Surian, D., Tian, Y., Lo, D., Cheng, H., & Lim, E. P. (2013, March). Predicting project outcome leveraging socio-technical network patterns. In Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on (pp. 47-56). IEEE.

[7] Thung, F., Bissyande, T. F., Lo, D., & Jiang, L. (2013, March). Network structure of social coding in github. In Software maintenance and reengineering (csmr), 2013 17th european conference on (pp. 323-326). IEEE.

[8] Tsay, J., Dabbish, L., & Herbsleb, J. (2014, May). 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.

[9] Ray, B., Posnett, D., Filkov, V., & Devanbu, P. (2014, November). 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.

[10] Ben, X., Beijun, S., & Weicheng, Y. (2013, January). Mining developer contribution in open source software using visualization techniques. In Intelligent System Design and

Engineering Applications (ISDEA), 2013 Third International Conference on (pp. 934-937). IEEE.

[11] Gousios, G., Pinzger, M., & Deursen, A. V. (2014, May). An exploratory study of the pull-based software development model. In Proceedings of the 36th International Conference on Software Engineering (pp. 345-355). ACM.

[12] Borges, H., Hora, A., & Valente, M. T. (2016, September). Predicting the Popularity of GitHub Repositories. In Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering (p. 9). ACM.

[13] Robles, G., González-Barahona, J. M., Cervigón, C., Capiluppi, A., & Izquierdo-Cortázar, D. (2014, May). Estimating development effort in free/open source software projects by mining software repositories: a case study of openstack. In Proceedings of the 11th Working Conference on Mining Software Repositories (pp. 222-231). ACM.