A Large-Scale Dataset of Popular Open Source Projects

Muna Altherwi, Andy Gravell*
School of Electronics and Computer Science, University of Southampton, Southampton, United Kingdom.

* Corresponding author. Email: m.altherwi@soton.ac.uk
Manuscript submitted January 30, 2019; accepted March 13, 2019.
doi: 10.17706/jcp.14.4.240-246

Abstract: Online open source software repositories offer a wealth of information related to software artifacts and the development process, making them a valuable source for research data. Mining software repositories and retrieving project data from them provide an opportunity to build large-scale datasets of selected, high quality, real project data. Such datasets could be used to empirically validate assumptions, test hypotheses, and verify anecdotal claims about software development processes and the resulting artifacts. Moreover, publishing them would make replicability and verification of studies possible that, in turn, can enhance research quality. Thus, in this work, we publish a large-scale dataset, of 4349 projects in 11 general-purpose programming languages gathered from Github repositories, where a primary language can be identified. The usage of such a dataset can vary from empirically validating claims in the software engineering field, to machine learning training and test sets.

Key words: Dataset, mining software repositories, open source software.

1. Introduction

Github is a widely used, online host for software projects. It provides a centralized storage along with project management and change tracking facilities based on git, the version- control system. Github offers an interactive collaborative platform for developers, with additional social features [1]. Also, it hosts some prominent open source projects, such as Linux kernel, Ruby on rails, and JQuery. In addition to the availability of source codes, it also offers a wealth of data related to the software artifact and the development process, making it a valuable source for researchers. These data can be used to increase our understanding about software development in open source projects.

Mining software repositories (MSR) to uncover patterns and discover findings about the artifact and the delivery process [2] has gained recognizable importance as a research area during the last decade. In 2004, a specialized conference on mining software repositories evolved from the premium international conference on software engineering (ICSE) [3], in recognition of the importance and potential of this field. This active research area has utilized the availability of projects data along with data mining tools and techniques to analyze and understand software projects. Moreover, it provides an opportunity to build large scale datasets of selected, high quality, real project data for research purposes.

Nevertheless, mining Github repositories and retrieving large amount of data from Github is a challenging task. Github allows their data to be accessed over HTTPS as JSON, however, it does not provide a schema for its data. Thus, for them to be examined it is necessary to traverse back their data using REST requests and JSON responses. Moreover, it imposes a rate limit on their API of 5000 requests per hour.

Given the huge number of events generated per day, and that every single event would lead to a series of dependent requests, pulling large amounts of data from Github would suffer significant delay.

2. Importance and Objectives

The issue and importance of the reproducibility of empirical software engineering studies have been discussed by different researchers [4]-[8]. The availability of research datasets is critical to validate findings and for the replicability and reproducibility of studies which can, in turn, enhance research in the field. However, a study on 170 MSR research papers [8] found that a very small number of the authors make their datasets publicly available (6 papers only). Another study [9] conducted on MSR paper authors reported that only one third of respondents stated that their datasets are publicly available. This is in line with the findings from [10], that more than two thirds of empirical studies on Github do not publish their dataset. Hence, despite the use of publicly available data, authors do not make their research datasets available, making it difficult to replicate the findings or even compare their outcomes.

Although Github is the most used on-line hosting service for open source projects in the world [11] with more than 31 million users and it has more than 96 million software repositories as of October, 2018 [12], 80% of its repositories have no stars (favoring or liking by registered users) [13]. Moreover, after excluding the non-starred repositories, 95% have 13 stars or less. Thus, this dataset with each repository having at least 500 stars is a good representative of open source popular projects, and can be used to empirically validate claims about open source software (OSS) development and artifacts.

In addition, our dataset is relatively large, made up of real, popular projects reflecting current practices in the software industry, and the data are collected from Github directly, rather than curated mirroring APIs. It is important here to emphasize that the dataset includes only projects where a main language can be identified. The main language is the one defined as comprising at least 95% of the total project code. This further restriction makes this dataset a good choice for comparative language studies.

In summary, in this work we have generated and published a dataset that meets the following objectives:

1. The volume of the dataset is relatively large, with 4349 projects in 11 languages.
2. The projects are the most popular ones on Github, with at least 500 stars given by registered users.
3. Up-to-date project data reflects current, modern practices in the software industry.

3. Approach and Tools

The approach we follow to retrieve the dataset is based on mining software repositories. We have checked about 15,000 repositories on GitHub, and retrieved the data of 4349 selected projects. The method and tools used to retrieve and store the dataset are summarized in the following steps:

Data retrieval:

1. retrieval of the initial list of popular repositories from GitHub: repositories that have at least 500 stars rating,
2. automated identification of primary language: the language that makes up at least 95% of the project's total code,
3. retrieval of the projects' data and source code from those repositories.

The initial retrieving phase has been implemented through GitHub Archive mirroring API on Google BigQuery. After that, the resulting JSON file was parsed, and repositories were checked for having a main language. This checking process was conducted using GitHub REST API (V3). If the repository had a primary language, we pulled the repository data (such as project ID, owner, commits, contributors, etc.) along with the source code files directly from Github, instead of the mirroring APIs. This was to ensure data freshness, instead of retrieving curated ones from the archiving services. It is also worth noting that Github

uses a specialized tool, called *linguist*[1], to determine the repository's main language. This tool detects the main language by aggregating all the languages in a repository and names the top one as the main language. Accordingly, the main language can sometimes make up just a small proportion of the total code, thus, we have implemented our definition to determine the main language the one which makes up at least 95% of the project's code.

Data storing: the projects' data are stored locally in JSON files as well as in a relational MySQL database. The source code files are also stored locally to be inspected for their content. The approach and tools used here are illustrated in Fig. 1.

After examining the top starred repositories on GitHub, we found that the projects where a main language can be identified comprise 48% of the total inspected population, whereas Repositories that did not have a main language (i.e. the codebase is shared between multiple languages) constitute 44.5%, and repositories dedicated for documenting purposes only (tutorials and books) make up about 7.5% of the total population.

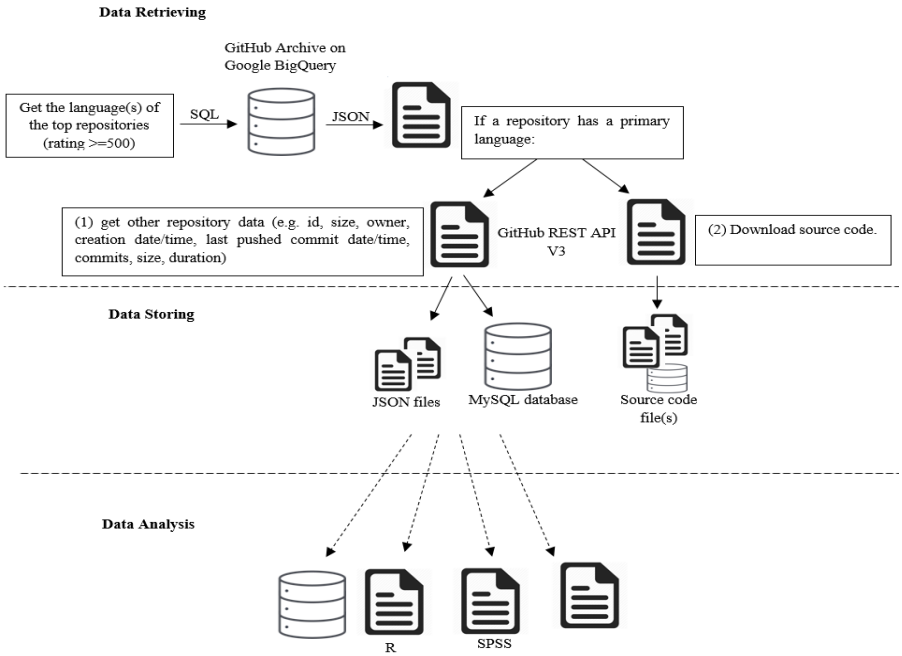


Fig. 1. Tools used in data extraction and retrieval.

4. The Dataset

For this dataset, we examined the most popular 15,000 repositories on GitHub, as indicated by ratings from January 2012 to December 2017. The selected projects have the following characteristics:

1. The project should have a main language, which makes up 95% of the total project's code.
2. The main language should be a high-level, and general-purpose one.
3. The repository should not have a reportedly infinite number of contributors and/or commits by Github, or missing key project attributes.
4. There should be at least 60 projects per language for it to be considered for the dataset.

The collected data per project include, but are not limited to languages (including the main one), source code, size in bytes, creation time/data, last pushed commit time/date, contributors, and commits. After excluding projects with special purpose languages (such as Shell and HTML) and the ones with missing key data (for the purposes of dataset consistency and completeness), cleaning and removing any duplications,

[1] https://github.com/github/linguist

we got 4349 projects, which comprise this dataset. The top 11 main languages here are: JavaScript, Java, Python, Go, Objective-C, Swift, PHP, Ruby, C#, C++, and C. Table 1 shows the programming languages along with the number of the selected projects per language in descending order.

This dataset is not a random sample, it is rather systematic and complete for the selected criteria and only covers Github repositories. In addition, it only includes projects where a main language can be identified.

Table 1. Programming Languages and the Corresponding Number of Projects in Descending Order

Programming Language	Number of Projects
JavaScript	1271
Java	978
Python	564
Go	347
Objective-C	316
Swift	226
PHP	194
Ruby	166
C#	129
C++	94
C	64
TOTAL	4349

4.1. Data Summary

The dataset projects are varied in type, characteristics, and complexity. The majority of them (51.71%) are written in JavaScript (29.23%) and Java (22.49%), followed by Python (10.54%), and Go (7.98%). C++ and C programming languages come last with 94 and 64 projects, making up 2.16% and 1.47% of the dataset size.

When projects are sized according to lines-of-code, using a suggested scale from a social forum discussion on programming projects scale [14] , the majority of projects in the sample are small (35%) and medium (27%), as can be seen in Fig. 2. Specifically, the majority of JavaScript (31.9%), Java (43.1%), Objective-C (46.2%), Swift (53.5%), PHP (38.1%), and Ruby (45.2%) projects are small. The small and medium projects in Python and Go form almost the same proportion at 27% and 28.2% respectively, that is the majority of projects in these two languages. When it comes to C# and C++, the majority of projects are of a very large size, at 51.9% and 43.6% respectively. These figures are based on the means of project sizes and are detailed in Table 2.

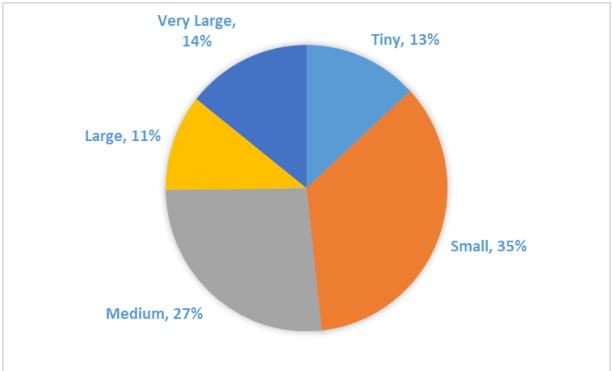


Fig. 2. Distribution of projects' sizes (in lines-of-code).

Table 2. Distribution of Projects' Sizes (in Lines-of-Code) per Language

	JavaScript	Java	Python	Go	Objective-C	Swift	PHP	Ruby	C#	C++	C
Tiny (<1000 SLOC)	15.42	10.22	17.73	7.49	18.04	14.16	18.04	8.43	1.55	3.19	10.94
Small (1,000 - 5,000 SLOC)	31.94	43.15	26.95	28.24	46.20	53.54	38.14	45.18	6.20	17.02	15.63
Medium (5,001 - 20,000 SLOC)	29.58	23.93	27.48	28.24	23.73	22.57	25.26	30.72	19.38	19.15	32.81
Large (20,001 - 50,000 SLOC)	12.12	9.92	12.59	10.37	7.91	4.42	9.28	7.23	20.93	17.02	12.50
V.large (>50,000 SLOC)	10.94	12.78	15.25	25.65	4.11	5.31	9.28	8.43	51.94	43.62	28.13

Nonetheless, when project sizes are classified based on the number of contributors using a suggested scale from industry [15], the majority of JavaScript (58%), Python (54%), Go (56%), PHP (81%), Ruby (84%), C# (75%), C++ (56%), and C (47%) projects have a large number of contributors (>20), whereas, this is medium (5-20) for Objective-C (42%) and Swift (47%) projects. Java is the only language with a majority (43%) of a small number of contributors, i.e. <5. These figures are shown in Fig. 3.

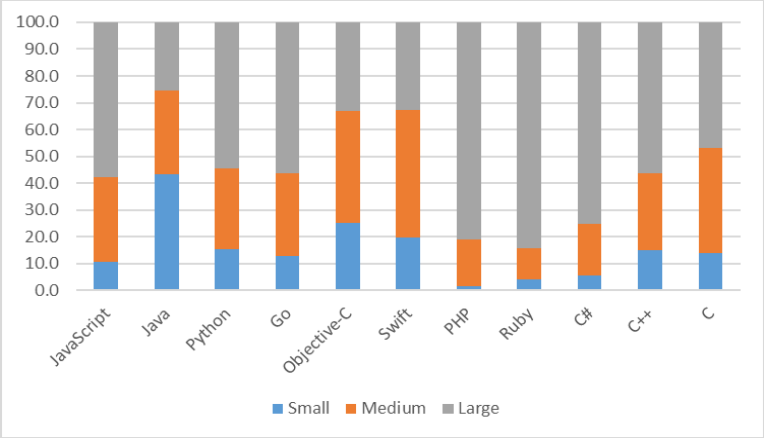


Fig. 3. Distribution of projects' sizes (in number of contributors) per language.

4.2. Descriptive Statistics

Included projects vary in duration from a minimum of 1 month up to 121 months, with a mean size of 61567 LOC and 20961 KB. The mean number of commits is 1020, with a mean of 58 contributors per project. Table 3 shows descriptive statistics of projects' attributes in the dataset.

Table 3. Descriptive Statistics of Selected Projects' Data

	Size (KB)	Size (SLOC)	Commits	Contributors	Duration
MEAN	20961	61567	1020	58	40
MEDIAN	2448	5376	269	19	36
MIN	7	3	2	1	1
MAX	1818501	11885756	70069	21111	121
STD	89360	427516	3183	406	24

5. Dataset Availability

The dataset associated with this paper is published under CC BY-NC-SA 4.0 license and is available at https://www.kaggle.com/muname/github-repos-mainlang.

6. Conclusions

The availability of research datasets is important in order to empirically validate claims about software development processes and the resulting artifacts. However, it has been found that more than two thirds of empirical studies on Github do not publish their dataset [10] making it difficult to replicate findings or even compare outcomes. Thus, in this work we publish a reasonably large, systematic and complete for the selected criteria dataset of 4349 projects in 11 general-purpose, programming languages for research purposes. It is made up of real projects reflecting current practices in the software industry, and the data are collected from Github directly, rather than curated mirroring APIs. This dataset can be used to empirically validate claims about software development, in data mining, and machine learning training and test sets.

References

[1] Lima, A., Rossi, L., & Musolesi, M. (2014). *Coding Together at Scale: GitHub as A Collaborative Social Network,* 295-304.

[2] Mining Software Repositories. Retrieved from http://www.msrconf.org/

[3] International Conference on Software Engineering (ICSE). Retrieved from http://www.icse-conferences.org/

[4] De Leeuw, J., Udina, F., & Greenacre, M. (2001). *Reproducible Research: The Bottom Line.*

[5] Shull, F. (2004). Knowledge-sharing issues in experimental software engineering. *Empir. Softw. Eng.*, *9(1/2),* 111-137.

[6] Vegas, S., Juristo, N., Moreno, A., Solari, M., & Letelier, P. (2006). Analysis of the influence of communication between researchers on experiment replication. *Proceedings of the 2006 ACM/IEEE International Symposium on International Symposium on Empirical Software Engineering - ISESE '06*.

[7] Shull, F. J., Carver, J. C., Vegas, S., & Juristo, N. (2008). The role of replications in empirical software engineering. *Empir. Softw. Eng.*, *13(2),* 211-218.

[8] Robles, G. (2010). Replicating MSR: A study of the potential replicability of papers published in the mining software repositories proceedings. *Proceedings of 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)* (pp. 171-180).

[9] Sureka, A., Tripathi, A., & Dabral, S. (2015). Survey results on threats to external validity, generalizability concerns, data sharing and university-industry collaboration in mining software repository (MSR) research. *arXiv Prepr. arXiv1506.01499*.

[10] Cosentino, V., Luis, J., & Cabot, J. (2016). Findings from GitHub. *Proceedings of the 13th International Workshop on Mining Software Repositories - MSR '16* (pp. 137-141).

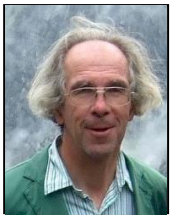
[11] Gousios, G., Vasilescu, B., Serebrenik, A., & Zaidman, A. (2014). Lean GHTorrent: GitHub data on demand. *Proceedings of the 11th Working Conference on Mining Software Repositories* (pp. 384-387).

[12] Github. Retrieved from https://github.com/about

[13] Sanatinia, A., & Noubir, G. (2016). On GitHub's programming languages. *arXiv Prepr. arXiv1603.00431*.

[14] What is the scale for a programming project's size? Retrieved from https://www.quora.com/What-is-the-scale-for-a-programming-projects-size

[15] Haste makes waste when you over-staff to achieve schedule compression | QSM SLIM-estimate. Retrieved from http://www.qsm.com/risk_02.html



Andrew Gravell was born in 1956 in Exeter, UK. He studied mathematics and computer science at Cambridge, and was awarded a Ph.D by the University of Southampton in 1995. He held the position of software developer for 7 years, most of which was spent at IBM's Hursley laboratory working on a range of projects and writing software for

microprocessor-based systems. In 1987 he joined ECS, Southampton, as a lecturer in computer science. He was promoted to senior lecturer/associate professor in 2002. Andrew also held the position of associate dean for education and the student experience, Faculty of Physical and Applied Sciences from 2010 to 2015. He has supervised a number of Ph.D students and published on a range of approaches to software development, empirical approaches to software development, agile methods, and lean information technology.

Muna Altherwi is a Ph.D student in software engineering at the University of Southampton, UK. Her research interests include mining software repositories, empirical software engineering, and programming languages. Muna has worked as a demonstrator (teaching assistant) in the Faculty of Computing and Information Technology, King AbdulAziz University, Saudi Arabia, prior to starting her Ph.D studies.