Public Git Archive: a Big Code dataset for all

Vadim Markovtsev
source{d}
Madrid, Spain
vadim@sourced.tech

Waren Long
source{d}
Madrid, Spain
waren@sourced.tech

ABSTRACT

The number of open source software projects has been growing exponentially. The major online software repository host, GitHub, has accumulated tens of millions of publicly available Git version-controlled repositories. Although the research potential enabled by the available open source code is clearly substantial, no significant large-scale open source code datasets exist. In this paper, we present the Public Git Archive – dataset of 182,014 top-bookmarked Git repositories from GitHub. We describe the novel data retrieval pipeline to reproduce it. We also elaborate on the strategy for performing dataset updates and legal issues. The Public Git Archive occupies 3.0 TB on disk and is an order of magnitude larger than the current source code datasets. The dataset is made available through HTTP and provides the source code of the projects, the related metadata, and development history. The data retrieval pipeline employs an optimized worker queue model and an optimized archive format to efficiently store forked Git repositories, reducing the amount of data to download and persist. Public Git Archive aims to open a myriad of new opportunities for "Big Code" research.

CCS CONCEPTS

• **Human-centered computing** → *Empirical studies in collaborative and social computing*; • **Software and its engineering** → *Software configuration management and version control systems*; Software libraries and repositories;

KEYWORDS

source code, git, GitHub, software repositories, development history, open dataset

ACM Reference Format:
Vadim Markovtsev and Waren Long. 2018. Public Git Archive: a Big Code dataset for all. In *MSR '18: MSR '18: 15th International Conference on Mining Software Repositories , May 28–29, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3196398.3196464

1 INTRODUCTION

Big code is revolutionizing software development. The revolution has begun with GitHub, whose collection of Git repositories is not just big, but vast: more than 24 million developers collaborating on over 67 million projects in 2017 [11]. GitHub has made version control accessible, therefore universal. The next stage of the revolution is permitting the automatic analysis of source code at scale, to support data-driven language design, to infer best (and worst) practices, and to provide the raw data to data hungry machine learning techniques that will be the basis of the next generation of development tools [3, 15]. It requires source code archives that are both big and programmatically accessible for analysis.

The GHTorrent project [12] took first steps in this direction, focusing on metadata in order to be scalable. Current source code datasets typically contain tens of thousands of projects at most [3] and are dedicated to particular programming languages such as Java and JavaScript [25], thus lacking diversity and attracting critics [6]. Software Heritage [7] is a recent attempt to archive all the open source code ever written, however no public dataset has been published yet by them.

We present the Public Git Archive, the first big code dataset amenable to programmatic analysis at scale. It is by far the biggest curated archive of top-rated[1] repositories on GitHub, see Table 1 for comparison. The Public Git Archive targets large-scale quantitative research in the areas of source code analysis (SCA) and machine learning on source code (MLoSC). The dataset is made available via HTTP as a separate index file together with files in the Siva format, a novel archive format tailored for storing Git repositories efficiently [29]. Every GitHub repository can be forked; forks typically introduce subtle changes not necessarily merged into the origin. The naive way to obtain forks is to clone them separately, requiring additional time and storage space. We describe the data retrieval pipeline which places forks into the original repository without mixing identities by reusing the existing Git objects [5]. The dataset size becomes thus smaller, requiring users to download and store less data.

The main contributions of the paper are:

- The Public Git Archive dataset, which is the largest collection of Git repositories to date available for download.
- The data retrieval pipeline which produces this dataset. Each part of that pipeline can scale horizontally to process millions of repositories.
- The Siva repository archival format used in this dataset. This format allows to efficiently store forks.

2 DATASET PRODUCTION

The dataset production consists of three steps, as follows.

MSR '18, May 28–29, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5716-6/18/05.
https://doi.org/10.1145/3196398.3196464

[1] We use "top-rated", "top-starred", 'top-bookmarked' and "having most stargazers" interchangeably. The number of stargazers is a proxy on the degree of public awareness and project quality within the community.

	Qualitas Corpus [31]	Sourcerer [18]	GitHub Java Corpus [3]	Public Git Archive
Number of projects	111	19,233	14,807	182,014
Year of release	2013	2014	2013	2018
Code language	Java	Java	Java	455 distinct
Development history	No	No	No	Yes
Number of files, 10^6	0.177	1.9	1.5	54.5 (HEAD)
Lines of code, 10^6	37.1	320	352	15,941 (HEAD)
Storage size	1.3 GB	19 GB	14 GB	3.0 TB

Table 1: Datasets comparison

2.1 Compiling the list of repositories

Similarly to existing research on GitHub mining [24], the focus of the dataset is put on the top-starred repositories. To compose the list of repository URLs, we make use of the metadata provided by GHTorrent [12]: a scalable, queryable, offline database generated from listening events through GitHub API, and available to the research community as a data-on-demand service [13]. The list for the Public Git Archive is based on GHTorrent's MySQL dump dated from January 1^{st}, 2018.

We created a command-line application which streams the compressed GHTorrent MySQL dump, reads and processes the needed files and stores the intermediate tables on disk. This tool can also be used to filter repositories based on the number of stargazers and chosen programming languages by taking the intermediate tables for input. For the Public Git Archive, there is no filtering by language performed and the minimum number of stargazers is set to 50. The resulting list contains 187,352 unique Git repository URLs.

2.2 Cloning Git repositories

Once the list repository URLs is produced, we fetch the actual data with `borges` [26]: a container-friendly distributed system that clones and stores Git repositories at scale.

Borges is designed as two separate standalone services: the producer reads URLs and determines which repositories should be processed next and adds new jobs for the consumer into the message queue; the consumer dispatches jobs to its thread worker pool. Multiple producers and consumers can be running; the message queue is also scalable. A job is a request to update a repository, new or existing. Each Git remote is fetched and each reference is pushed to the corresponding *rooted repository*. This way, we store all references (including all pull requests) from different repositories that share the same initial commit – *root*, Fig. 1. As a consequence, forks go into the same physical Git repository, thus being stored more efficiently.

Subsequently, Borges consumer's workers put Git packfiles belonging to rooted repositories into Siva files. Siva is an archival format [29] similar to `tar` and `zip`: it allows constant-time random file access, concatenation of the archive files, and seekable access to the contained files which are written verbatimâĂŤsince packfiles are already compressed with `zlib`. Siva makes possible to store rooted repositories in an efficient and convenient way with minimal storage overhead. Internally, placing Git repositories inside Siva is implemented as `git push`. Borges can store Siva files in the local file system or Hadoop Distributed File System (HDFS).

The repositories belonging to the Public Git Archive were cloned in late January-February, 2018. A total of 8 consumers ran 32 threads each, taking one week under a 1Gbps internet connection. The "smart HTTP" Git protocol was used. The bulk download of 3.0 TB of data at the same connection speed takes under 8 hours âĂŤ less than 1% of the initial retrieval time normalized to single consumer. The exact storage space saved due to fork embedding is computationally expensive to calculate because a part of the pull requests are merged and the corresponding forks are not needed to be fetched. This is left for the future work.

From the initial 187,352 URLs, 3,156 had become inaccessible by the Git clone time, including 82 removed for legal reasons (HTTP 451 error messages returned by the server). The final amount of repositories cloned is 182,014 since several outliers [21] could not be processed by our pipeline. 90% of the repositories were cloned within the first 24 hours and they constitute 50% of the final dataset size.

2.3 Generating the index file

Users of the dataset may prefer to work with a smaller subset of the terabytes collected of Siva files. A frequently observed use case is to triage files of certain programming languages. To address this and more preferences or restrictions, we generate from the finalized Siva files a CSV-type file including: per-repository metadata, detected license information, plus aggregate statistics on the number of files, lines and bytes per programming language. Each line in the CSV links to the corresponding Siva files, which in turn contain Git references of the corresponding repository. It becomes therefore possible to query the index file and choose which Siva files to download. The columns of the CSV file are explained in Table 2.

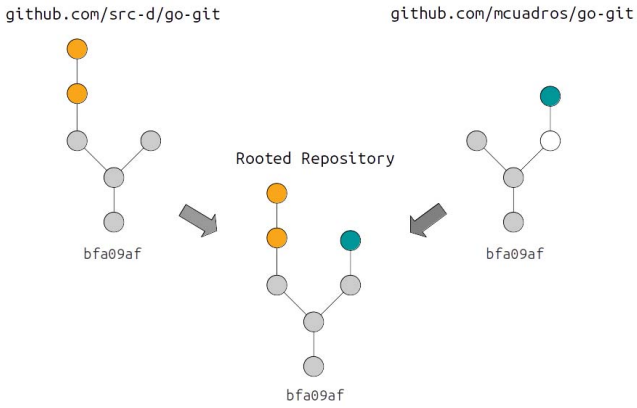

Figure 1: Rooted Repository

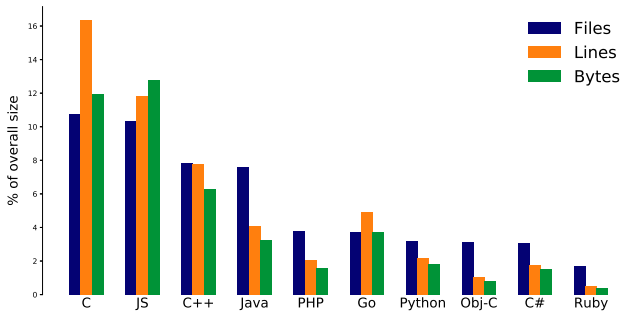

Figure 2: Statistics of 10 most popular programming languages in PGA.

3 USING PUBLIC GIT ARCHIVE

Links to the dataset download, as well as to all the relevant tools and scripts to reproduce it, are hosted in the official repository on GitHub[2]. The Public Git Archive consists of (a) 248,043 Siva files (3.0 TB) with Git repositories and (b) the index file in CSV format. We also provide a command-line application to automate, accelerate and simplify downloading (a) and (b). In the columns of the CSV file, *HEAD* is used to denote the latest commit of a branch; *default HEAD* means the latest commit of the default branch. The default branch corresponds to the reference which is marked main on GitHub. Languages were detected using **enry** [28]. Licenses were detected using **go-license-detector** [30]. Lines of code were calculated using **gocloc** [14]. **GitHub API was not used**, as it is planned to extend the dataset to sources beyond GitHub.

Fig. 2 shows the aggregated programming language statistics.

After the selected Siva files are downloaded, users can work with the dataset using **engine** [27]. The engine is an extension for Apache Spark which adapts Siva files as a Spark data source, allowing users to execute conventional Spark or PySpark queries to analyze Git repositories. It is also possible to unpack Git repositories from Siva files by using its Go language API or through the command line interface.

4 SIGNIFICANCE

Analysis of source code has recently received significant research attention. The areas which can benefit from the Public Git Archive are statistical machine learning and natural language processing on source code. For example, source code modeling studies [3, 15] have shown that the performance of n-gram models critically depends on the dataset size. That's why the presented dataset can enhance research in topics like automatic naming suggestion [2], program prediction [25], topic modeling and semantic clustering [17], bug detection [8], and automated software transpilation [4]. It can also provide valuable insights for compiler developers into how the programming languages are used by the open source community.

Another promising research direction is inter-project source code clone detection. Social programming platforms with minimal boundaries between projects like GitHub have facilitated code reuse across multiple projects. A number of studies has been carried out about those ecosystems in the recent years [23]. Code clones were first studied within single projects, but as GitHub grew further, different reasons for the appearance of duplicated code snippets have been explored, e.g. "accidental" clones due to imprecise API usage protocols [1] or automatic program repair [19]. The Public Git Archive enables the research community to study source code clones across project boundaries, not limited to a single language and having large graphs with over 10,000 projects [9].

[2]github.com/src-d/datasets

Column name	Description
url	URL of the GitHub repository.
siva_filenames	Siva files which contain parts of that repo.
file_count	Number of files in default HEAD reference.
langs	Languages encountered in default HEAD.
langs_{byte, lines, files}_count	Byte, line, file counts per each language, in the same order as *langs*.
commits_count	Number of unique commits in the Siva files which refer to that repository.
branches_count	Number of references, tags excluded.
fork_count	Number of remotes in the referring Siva files.
{empty, code, comment}_lines_count	Number of empty, code, commented lines in default HEAD.
license	License names and corresponding confidences.

Table 2: CSV columns

5 UPDATES

In order to evolve along the constantly changing open-source landscape, The Public Git Archive needs to be regularly updated. Several technical challenges arise from this requirement. The typical way to organize dataset updates is to provide regular snapshots, as GHTorrent does. However, every snapshot of our dataset would require considerable disk space. The solution is to manage incremental updates consisting of the differences from the previous snapshot. Two ways to implement this solution are:

- To pull changes into every packfile in every Siva file of the dataset. The `git pull` operation requires the whole new packfile to be read, and this is precisely what one would like to avoid.
- To generate binary diffs of the Siva files. However, diffing Git packfiles is not straightforward. They are compressed and even a single Git object which is removed at the beginning of a packfile changes the whole binary stream, making it necessary to retain the old objects which are no longer referenced in the new packfile. GitHub always returns a single packfile during `git clone` and runs garbage collection from time to time, effectively breaking the binary diffs.

The challenges of the first implementation are harder to resolve technically. The second implementation seems more feasible and has ongoing research. The current plan to update the Public Git Archive is to publish complete snapshots, limiting their lifetime. There are going to be Long Term Support (LTS) snapshots with

extended lifetime and researchers are encouraged to focus on them. The exact schedule is subject to change and is updated on the Public Git Archive website.

6 PRIVACY AND LICENSING

The Public Git Archive contains the full commit history for each public repository, including commit messages, timestamps, author names and emails. GitHub Terms of Service (GHTS) explicitly allow passing such public information to third parties as long as the goal is doing research or archiving [10]. The Public Git Archive is maintained solely for research purposes, and the collected credentials are not to be used in any way except as allowed by the GHTS.

Despite the public nature of the information, some developers may prefer to take their projects down or private, making repositories inaccessible as noticed in section II. We do provide a communication channel for repository removal requests, the full details provided on the official Public Git Archive website.

Each rooted repository inside Siva files is licensed separately and according to the manifested project license. Projects which do not have an explicit license are distributed under the same terms as stated in GHTS exclusively for research purposes. The index file is licensed under a *Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International* license.

7 LIMITATIONS

Dataset miners should take into account several potential threats to validity [16].

Regarding the data collection process and the traditional trade-off between freshness and curation [6], we choose to emphasize the curation of the dataset rather than a limited amount of data up-to-date. We rely on GHTorrent for the list of repositories to retrieve, thus our update schedule depends on the upstream. Consequently, the dataset and the pipeline to collect it are entirely transparent. The output is never exactly the same, though, as GitHub is a dynamic environment and repositories may change or become inaccessible over time.

Other notable concern is about the generalization of the dataset. Selecting repositories based on the number of stargazers is arguable and may introduce bias. Fair probabilistic sampling of the complete list of repositories should improve the diversity, e.g. stratified random sampling [22]. Other popularity indicators can be explored, as the number of forks or accepted pull requests [3]. By focusing on the number of stars as a measure of people's interest and awareness of the project, there is a risk to miss quality samples. As a result, various source code quality metrics should be considered. Finally, there will be duplicate files across different repositories [20]. Mentioned suggestions constitute the basis for the future work.

8 CONCLUSION

In this paper, we presented the Public Git Archive, the largest source code dataset of top-starred Git repositories, described the novel scalable pipeline to reproduce it and the tooling to download and use it. The Public Git Archive is made available through HTTP and includes the source code of the projects, their metadata, and their development history. The retrieval pipeline is efficient and the dataset size is optimal thanks to the distributed cloning system and the custom Git repository archive format. We believe that the Public Git Archive âĂŤ over ten times larger than any of the currently available datasets âĂŤ has the potential to boost the quality, confidence and diversity of the software engineering and mining research.

REFERENCES

[1] Raihan Al-Ekram, Cory Kapser, Richard Holt, and Michael Godfrey. 2005. Cloning by accident: An empirical study of source code cloning across software systems. In *ISESE*. IEEE, 376–385.
[2] Miltiadis Allamanis, Earl T. Barr, Christian Bird, and Charles Sutton. 2015. Suggesting Accurate Method and Class Names. In *FSE*. ACM, 38–49.
[3] Miltiadis Allamanis and Charles Sutton. 2013. Mining Source Code Repositories at Massive Scale Using Language Modeling. In *MSR*. IEEE, 207–216.
[4] Earl T. Barr, Mark Harman, Yue Jia, Alexandru Marginean, and Justyna Petke. 2015. Automated Software Transplantation. In *ISSTA*. ACM, 257–269.
[5] Marco Biazzini, Martin Monperrus, and Benoit Baudry. 2014. On Analyzing the Topology of Commit Histories in Decentralized Version Control Systems. In *ICSME*. IEEE, 261–270.
[6] Valerio Cosentino, Javier Luis, and Jordi Cabot. 2016. Findings from GitHub: methods, datasets and limitations. In *MSR*. ACM, 137–141.
[7] Roberto Di Cosmo and Stefano Zacchiroli. 2017. Software Heritage: Why and How to Preserve Software Source Code. In *iPRES*. ACM, 10.
[8] Zheng Gao, Christian Bird, and Earl T. Barr. 2017. To Type or Not to Type: Quantifying Detectable Bugs in JavaScript. In *ICSE*. IEEE, 758–769.
[9] Mohammad Gharehyazie, Baishakhi Ray, and Vladimir Filkov. 2017. Some from Here, Some from There: Cross-project Code Reuse in GitHub. In *MSR*. IEEE, 291–301.
[10] GitHub. 2017. GitHub Terms of Service. goo.gl/yeZh1E. (2017).
[11] GitHub. 2017. The State of the Octoverse. octoverse.github.com. (2017).
[12] Georgios Gousios. 2013. The GHTorrent dataset and tool suite. In *MSR*. IEEE, 233–236.
[13] Georgios Gousios, Bogdan Vasilescu, Alexander Serebrenik, and Andy Zaidman. 2014. Lean GHTorrent: GitHub Data on Demand. In *MSR*. ACM, 384–387.
[14] Hideo Hattori. 2016. hhatto/gocloc. github.com/hhatto/gocloc. (2016).
[15] Abram Hindle, Earl T. Barr, Zhendong Su, Mark Gabel, and Premkumar Devanbu. 2012. On the Naturalness of Software. In *ICSE*. IEEE, 837–847.
[16] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2016. An In-depth Study of the Promises and Perils of Mining GitHub. *Empirical Softw. Engg.* 21, 5 (2016), 2035–2071.
[17] Adrian Kuhn, Stéphane Ducasse, and Tudor Gírba. 2007. Semantic Clustering: Identifying Topics in Source Code. *Inf. Softw. Technol.* 49, 3 (2007), 230–243.
[18] Davy Landman, Alexander Serebrenik, and Jurgen Vinju. 2014. Empirical Analysis of the Relationship Between CC and SLOC in a Large Corpus of Java Methods. In *ICSME*. IEEE, 221–230.
[19] Claire Le Goues, ThanhVu Nguyen, Stephanie Forrest, and Westley Weimer. 2012. GenProg: A Generic Method for Automatic Software Repair. *IEEE Trans. Softw. Eng.* 38, 1 (2012), 54–72.
[20] Cristina V. Lopes, Petr Maj, Pedro Martins, Vaibhav Saini, Di Yang, Jakub Zitny, Hitesh Sajnani, and Jan Vitek. 2017. DéJà Vu: A Map of Code Duplicates on GitHub. *Proc. ACM Program. Lang.* 1, OOPSLA (2017), 84:1–84:28.
[21] Kate Murphy. 2017. Katee/git-bomb. github.com/Katee/git-bomb. (2017).
[22] Meiyappan Nagappan, Thomas Zimmermann, and Christian Bird. 2013. Diversity in Software Engineering Research. In *FSE*. ACM, 466–476.
[23] Hoan Anh Nguyen, Anh Tuan Nguyen, Tung Thanh Nguyen, Tien N. Nguyen, and Hridesh Rajan. 2013. A Study of Repetitiveness of Code Changes in Software Evolution. In *ASE*. IEEE, 180–190.
[24] Rohan Padhye, Senthil Mani, and Vibha Singhal Sinha. 2014. A Study of External Community Contribution to Open-source Projects on GitHub. In *MSR*. ACM, 332–335.
[25] Veselin Raychev, Martin Vechev, and Andreas Krause. 2015. Predicting Program Properties from "Big Code". In *SIGPLAN Not.* ACM, 111–124.
[26] source{d}. 2017. src-d/borges. github.com/src-d/borges. (2017).
[27] source{d}. 2017. src-d/engine. github.com/src-d/engine. (2017).
[28] source{d}. 2017. src-d/enry. github.com/src-d/enry. (2017).
[29] source{d}. 2017. src-d/siva. github.com/src-d/go-siva. (2017).
[30] source{d}. 2018. src-d/go-license-detector. github.com/src-d/go-license-detector. (2018).
[31] Ewan Tempero, Craig Anslow, Jens Dietrich, Ted Han, Jing Li, Markus Lumpe, Hayden Melton, and James Noble. 2010. The Qualitas Corpus: A Curated Collection of Java Code for Empirical Studies. In *APSEC*. IEEE, 336–345.