An Empirical Study on GitHub Repositories - Exploring the Types of Repositories

COMP 4520 - Undergraduate Honours Project Report
Supervisor: Shaowei Wang

Darshan Pandhi
Computer Science
University of Manitoba
Winnipeg, Manitoba, Canada
pandhid@myumanitoba.ca

1 Introduction

GitHub[1] is the largest and most advanced development platform in the world. Millions of developers and companies build, ship and maintain their software on GitHub. According to the company, it contains over 200 million repositories with over 65 million developers utilizing the platform, making it the largest source code host as of April 2020 [3]. As a result, it lends itself to data mining applications for several research studies, leveraging the invaluable open-source projects available in the website's massive database [7].

Being home to a large chunk of code hosted online does not take away from the fact that it is incredibly diverse. Hence, the goal of this study was to take a deep dive into all the source code available on GitHub in order to infer trends. This shall help quell the curiosity surrounding the proportions of various types of software and the type that enjoys the most support for open-source contributions. In addition, this would enable and act as a catalyst for future studies regarding software data collection in general.

A final sample of 100 projects having at least 50 stars and a minimum of 50 people watching was chosen. It was observed that libraries/frameworks are the most common type of software among 14 such categories whereas contrary to popular belief, over 20% of all projects are not even directly software related (docs, tutorials, empty, ..). Lastly, languages like HTML, Ruby and Shell lead the pack in terms of global usage on GitHub.

2 Background and Related Work

GitHub Data Mining has been a growing matter of interest over the years. Mathew Russell breaks down the entire process skillfully in their book "Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, Github, and More" to the point where it is almost akin to serving as a tutorial [13]. No wonder the methodology for this study is inspired by it.

Another paper, "The Promises and Perils of Mining GitHub" talks about how to approach the data in GitHub based on an initial qualitative assessment. It highlights the common mistakes researchers often make when going about this route [6]. However, since the paper was published in 2014 it will be interesting to see if some of their conclusions still hold.

Valerio et al. examine a number of research papers that use GitHub data. Their paper "Findings from GitHub: methods, datasets and limitations" talks about i) the empirical methods employed, ii) the datasets they used and iii) the limitations reported [18]. Lately, Michael Färber follows a similar approach to conduct a thorough analysis of the code repositories linked in scientific papers using the Microsoft Academic Graph as a data source [9].

The results across the board are unsettling. The dataset collection process, size, the low level of replicability, poor sampling techniques, lack of longitudinal studies and scarce variety of methodologies are a matter of concern. Şeker et al. attempt to solve problems like these by creating a single small, unified database that would allow researchers to work on many software engineering problems [15]. However, the sheer maintenance of the database to keep up with modern time, infinite breath of topics and unending hunger for more data makes it an unsustainable solution. Rather, emphasis should be shifted to data preprocessing and reduction [12].

The uncanny **assumption that GitHub is only for software projects** seems too naive to even take a closer look at. Given that superstition, most papers do not undertake adequate steps for data preprocessing and cleaning. This, together with the growing amount of research utilizing GitHub data, motivated me to assess the veracity of the claim.

While there are numerous (previous) studies that utilize GitHub data, no one has really recently explored what kind of projects does GitHub contain i.e., the quality of the dataset. This is what

[1] https://github.com/

will be the primary focus of this study. Furthermore, having a well-balanced dataset is crucial. For instance, research studies in the early 2000s often had an unreasonably high number of Java projects given the fact that the language was at its peak popularity [17]. Hence, papers during that time imbibed unintentional biases towards Java even though they had a more generic dataset. To uncover any potential spikes in the current scenario, this project will also involve analyzing the language distribution among the various projects.

With technology advancing at an unprecedented pace, the common approach needs to be constantly evolving in order to keep up. For that reason, the results for this study shall help in honing GitHub data collection and analysis process for the time to come.

3 Research Questions

The following are the 2 research questions that this paper aims to answer:

- **RQ1: What are the different types of repositories on GitHub?**

  While GitHub is commonly used to host open-source projects, understanding them in greater depth will go a long way in dealing with unintentional bias. The assumption is that most of the repositories on the version control platform are engineered software projects.

  Software can be divided into 4 main types: Application Software, System Software, Programming Software and Driver Software [19]. The goal is to break them down further using manual classification to gain a granular perception of the current state of things. Moreover, such detailed introspection will also allow for understanding the language mix.

- **RQ2: Can we build a classifier for GitHub repositories?**

  Why do we even need such a classifier in the first place? It will be an understatement to claim that GitHub contains a lot of software code. However, more often than not research projects focus on a narrow subsection, involving a particular type of software. For example, one of my recent projects was "What Are the Characteristics of Most-Popular Apps? - A Case Study on Android Applications" [11]. Undoubtedly, I examined quite a few mobile applications and not just any mobile applications but Android applications specifically. Therefore, studies like these would benefit immensely from a GitHub classifier that helps them collect the right subset of the available data quickly and efficaciously.

4 Data Collection and Experimental Setting

For this study, I needed a large number of software projects but more importantly supplementary data like the languages used in the codebase, ratio and lines of code, number of contributors, commit patterns, etc. in order to compare and contrast and infer trends.

I started with the GitHub Activity Data hosted on Google BigQuery. With over 3 million repositories[2], it is the largest released source of GitHub activity to date. However, the schema not only missed several key attributes but also was devoid of an easy way to download the repository contents inexpensively and efficiently. Consequent attempts included experimenting with various tools and datasets like GHTorrent [3], SEART[3]Tool [1], Kaggle[4], etc. Unfortunately, none of these attempts proved to be fruitful due to a plethora of reasons including but not limited to inconsistencies, obsolete and incomplete data, broken URLs, data skewness, lack of documentation, etc. Ultimately, GitHub Search API[5] was the one that outsmarts the competition since it offered the maximum information and was easy to use, thanks to the massive community support it enjoys. To see this in action, the Python programming language is used for implementing data mining and data processing. PyGitHub[6] complements this choice beautifully by making it very convenient to access the GitHub REST API. Finally, the code for the entire project is rightfully hosted on GitHub[7] itself.

The pertinent task of selecting the right data for further analysis was taken care of by the API itself as it returned a random assortment of the said number of repositories. For this experiment, 100 such repositories (sample A – Random Sample) were taken a closer look at. Despite that assurance, the first batch of repositories wasn't the most favourable for the task at hand. This is because a majority of those were outdated, unknown, incomplete, poorly documented and at times even empty / moved to new repositories. This made manual analysis hard and time-consuming. Accurately labelling them often involved a quick examination of the source code. Even after all this, there were ample repositories in the end that didn't have enough information to clearly decipher their intent. Hence, I started playing around with a variety of filters in order to obtain a higher-quality dataset. After several attempts, filtering out repositories with less than 50 stars and less than 50 people watching (sample B - Filtered Sample) seemed to do the job. In addition, only repositories that had a README file made the final cut. With these additional criteria in effect, getting the same 100 repositories involved programmatically scanning over 87,500 of them. The large number also meant data collection spanned over multiple sessions due to the rate-limiting

[2] Link to the Dataset: https://console.cloud.google.com/marketplace/product/github/github-repos
[3] https://seart-ghs.si.usi.ch/
[4] https://www.kaggle.com/
[5] https://docs.github.com/en/rest/reference/search
[6] https://github.com/PyGithub/PyGithub
[7] https://github.com/darshanpandhi/simplifying-development

restrictions imposed by the API. Be that as it may, the increased popularity of the repositories meant that there were more contributors to ensure the repositories are well maintained [5]. Simply scanning the project description and /or the README file was sufficient enough this time around and far worth it honestly to confidently label the given software.

4.1 Labels

While on one hand. I was finetuning data selection, I made sure to simultaneously refine the types of software (labels) [16] we started with to comprehensively represent all the data. After careful thought and multiple iterations, the following labels in Table 1 were finalized for software classification. Since the terms are often loosely defined and open to common interpretation, I have used the given connotations for the purposes of this study. The table, in addition, also contains an example for each label. This does not take away from the fact that a particular software can belong to multiple categories, i.e., multi-label classification.

Table 1: Types of Software

	Category	Description	Example
1	Library/Framework	A library works within the software while in the case of a framework, the software works within the framework	Bootstrap - https://getbootstrap.com/ Most popular HTML, CSS, and JavaScript framework for developing projects on the web
2	Language	Any and all types of languages like general-purpose programming languages, domain-specific languages (DSLs), Scripting Languages, etc.	Wren - https://github.com/wren-lang/wren A small, fast, class-based concurrent scripting language
3	Plugin	A software add-on that is installed on a program, enhancing its capability [4]	Adobe Flash Player - https://get.adobe.com/flashplayer/about/ A lightweight, robust runtime environment for rich media and rich internet applications [8]
4	Web Application	Application software that runs on a web server, accessed by the user through a web browser with an active network connection	jspaint - https://jspaint.app/ A nice web-based MS Paint remake
5	Doc/Tutorial	Only textual files including instructional documentation intended to impart knowledge	sindresorhus/awesome - https://github.com/sindresorhus/awesome Awesome lists about all kinds of interesting topics
6	Moved/Empty/ Archived	Any repository that has either been moved to a new place or is empty or has been made read-only by archiving	go-git - https://github.com/src-d/go-git A highly extensible git implementation library written in pure Go
7	Utility	Software that is designed to aid in analyzing, optimizing, configuring and maintaining a computer system	Hidden - https://github.com/dwarvesf/hidden An ultra-light MacOS utility that helps hide menu bar icons
8	Protocol	Set of rules or procedures for transmitting information between electronic devices, such as computers	Gun - https://github.com/amark/gun A cybersecurity protocol for syncing decentralized graph data
9	Database/Storage	Anything to do with database/persistence like Object-Relational-mapper (ORM)	Postgres - https://www.postgresql.org/ Free and open-source relational database management system
10	Mobile Application	Type of application software designed to run on a mobile device, such as smartphones or tablets	Signal - https://signal.org/en/ A messaging app for private communication with friends. Supported on both Android and iOS
11	Testing	Any software or tools that aid in ensuring code works as expected through written test cases	Jest - https://jestjs.io/ JavaScript Testing Framework maintained by Facebook
12	API	A software intermediary that facilitates data exchange between two applications	GitHub Search API - https://docs.github.com/en/rest/reference/search Lets you to search for the specific item efficiently on GitHub
13	Parser	A compiler or interpretor component that breaks data into smaller elements for easier translation into another language	argparse - https://docs.python.org/3/library/argparse.html Python parser module for command-line options, arguments and sub-commands
14	Other	All the remaining categories of software that had very little representation individually including user interface (code), operating systems, games,	HarmonyOS - https://www.harmonyos.com/en/ A distributed OS developed by Huawei to run on multiple devices

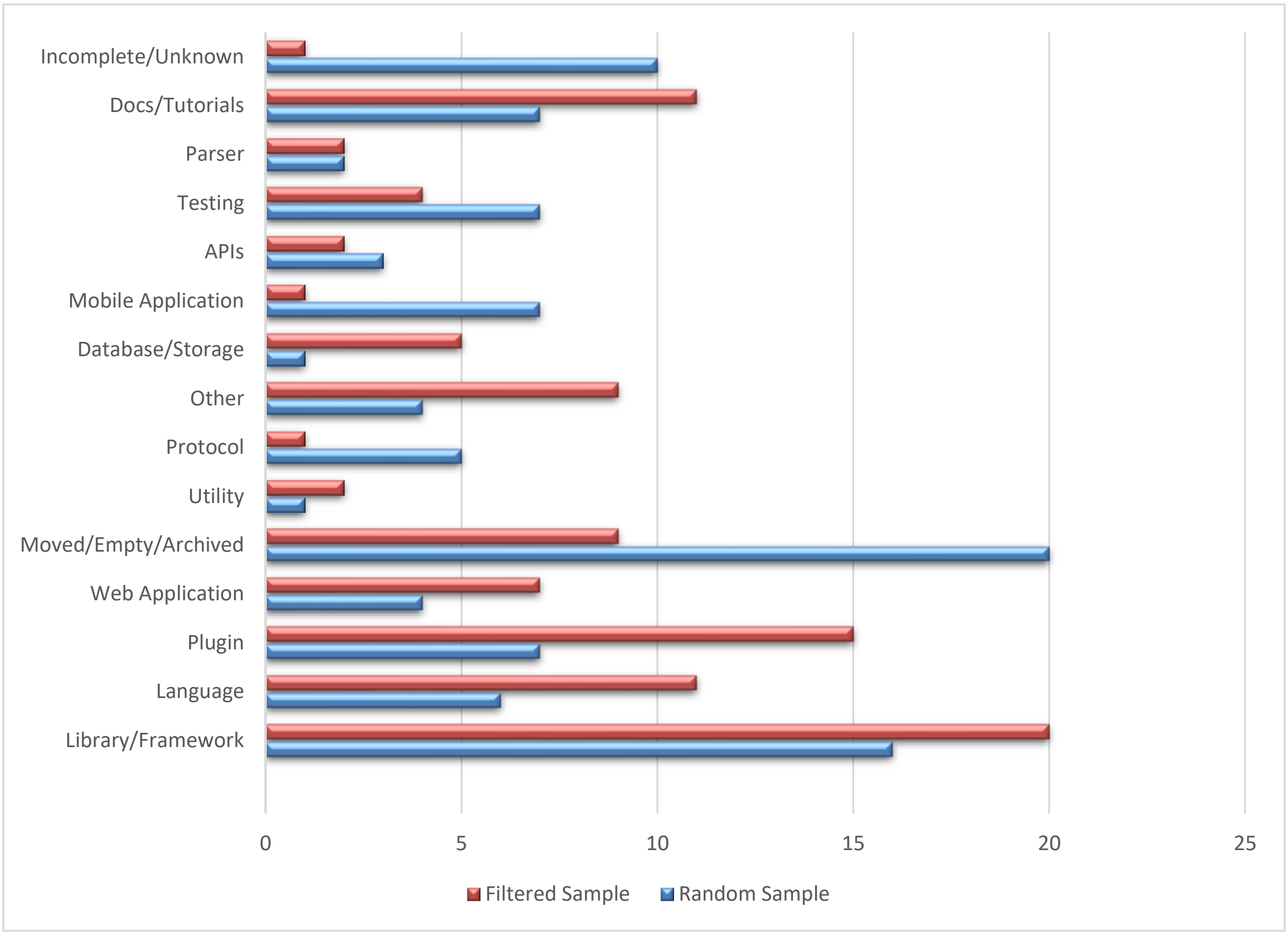

Figure 1: Software Label Comparison between Filtered Sample and Random Sample

5 Research Results

This section describes the results obtained to answer each of the 2 research questions outlined in section 3.

- **RQ1: What are the different types of repositories on GitHub?**

  Figure 1 compares the 2 samples: Random Sample and Filtered Sample (at least 50 stars and 50 watching). While this does not accurately represent the hardships and uncertainty faced in determining the right label for the random sample, the noteworthy increase in the number of repositories that were incomplete/unknown or Moved/Empty/Archived do hint in this direction. Needless to say, to inhibit these repos from distorting the analysis, sample B - filtered sample is used for the final results henceforth to maintain reliability. This is in line with the data cleaning steps taken in the article, "Analyzing the 'Biodiversity' of Open Source Ecosystems: the GitHub Case" where a vast majority of users are likewise excluded from the dataset [10].

  Table 2 contains a snapshot of 27 repositories and their types. The complete table with all the observed repositories is hosted on GitHub[8]. Alternately, Figure 2 is a graphical representation highlighting the distribution of the types of repositories found.

  Libraries/Frameworks are the clear winner with 20% whereas plugins occupy the runner-up spot with 14% to their name. 11% of GitHub is languages while docs/tutorials take up another 11%. Together, these four categories make up for more than half of GitHub.

[8] https://github.com/darshanpandhi/simplifying-development

Table 2: Snapshot of the Repository List (along with their labels)

Sr. No.	Repo Name	Download URL	Type
1	mojombo/grit	https://github.com/mojombo/grit	Library/Framework
2	rubinius/rubinius	https://github.com/rubinius/rubinius	Other
3	mojombo/god	https://github.com/mojombo/god	Library/Framework Utility
4	mojombo/chronic	https://github.com/mojombo/chronic	Plugin
5	engineyard/eycap	https://github.com/engineyard/eycap	Other
6	defunkt/facebox	https://github.com/defunkt/facebox	Library/Framework
7	haml/haml	https://github.com/haml/haml	Language
8	sferik/twitter	https://github.com/sferik/twitter	Other
9	IoLanguage/io	https://github.com/IoLanguage/io	Language
10	seven1m/onebody	https://github.com/seven1m/onebody	Web Application
11	mislav/will_paginate	https://github.com/mislav/will_paginate	Library/Framework
12	aasm/aasm	https://github.com/aasm/aasm	Library/Framework
13	dustin/java-memcached-client	https://github.com/dustin/java-memcached-client	Database/Storage
14	programming-nu/nu	https://github.com/programming-nu/nu	Language
15	preservim/nerdtree	https://github.com/preservim/nerdtree	Plugin
16	preservim/nerdcommenter	https://github.com/preservim/nerdcommenter	Plugin
17	spree/spree	https://github.com/spree/spree	Web Application
18	arclanguage/anarki	https://github.com/arclanguage/anarki	Language
19	psychs/limechat	https://github.com/psychs/limechat	Other
20	DotNetOpenAuth/DotNetOpenAuth	https://github.com/DotNetOpenAuth/DotNetOpenAuth	Library/Framework Protocol
21	mbleigh/acts-as-taggable-on	https://github.com/mbleigh/acts-as-taggable-on	Plugin
22	bborn/communityengine	https://github.com/bborn/communityengine	Plugin
23	prototypejs/prototype	https://github.com/prototypejs/prototype	Library/Framework
24	sstephenson/sprockets	https://github.com/sstephenson/sprockets	Moved
25	phusion/passenger	https://github.com/phusion/passenger	Web Application
26	jeremyevans/sequel	https://github.com/jeremyevans/sequel	Language Database/Storage
27	beanstalkd/beanstalkd	https://github.com/beanstalkd/beanstalkd	Other

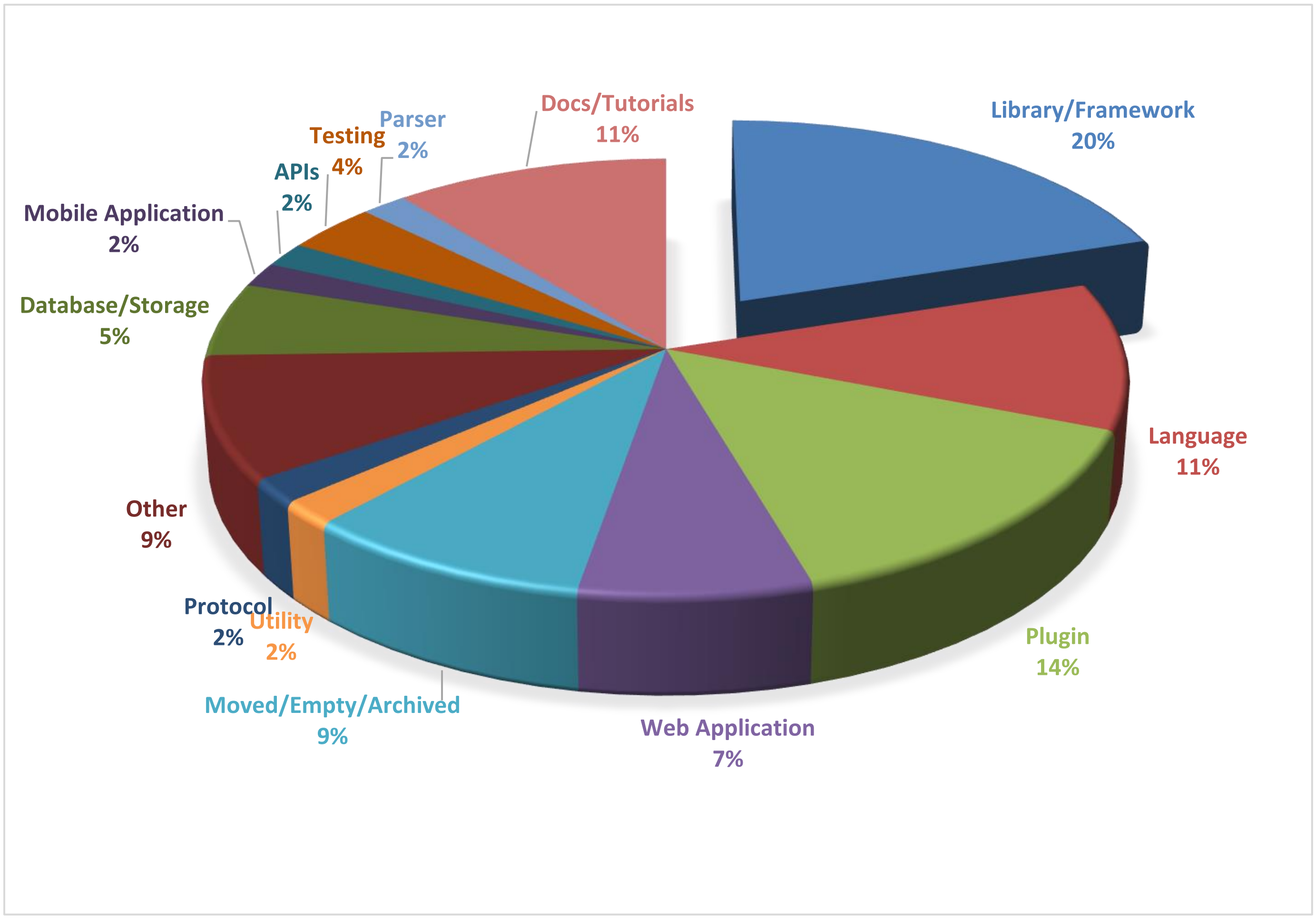

Figure 2: Types of GitHub Repositories

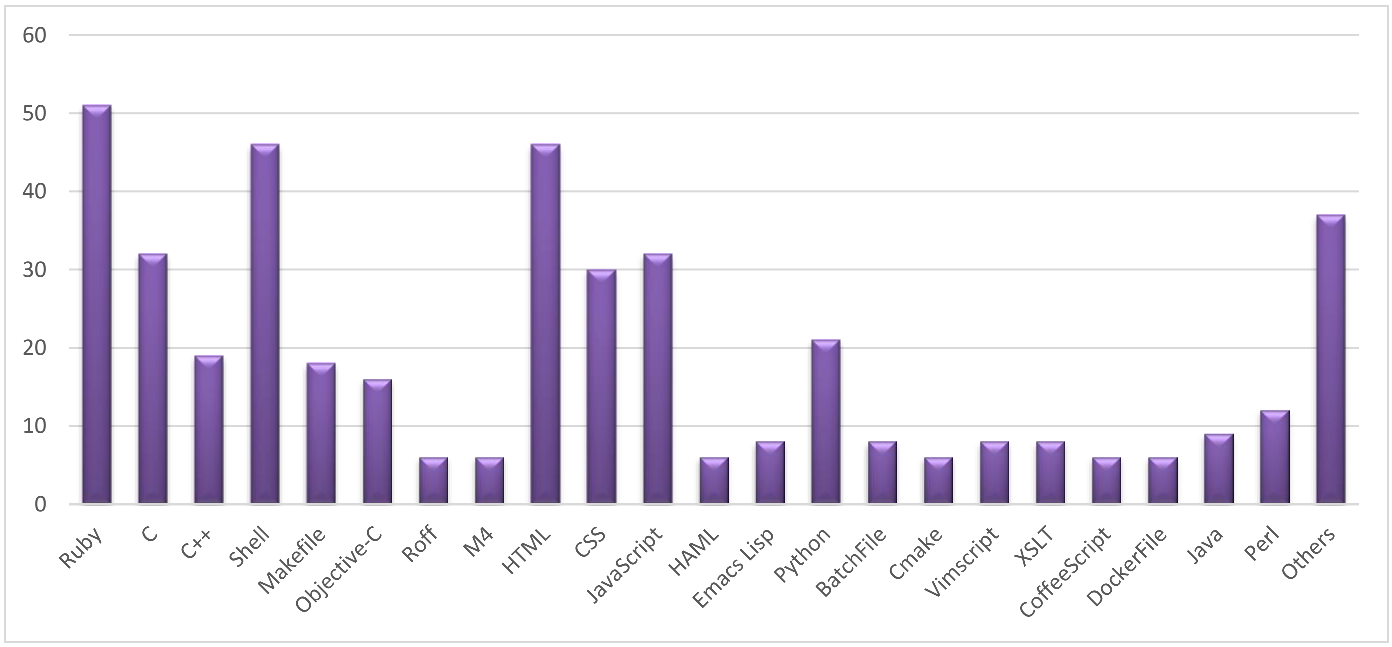

Figure 3: Language Mix

Next up, 9% repositories are empty now, have been moved or are archived. And this is what is the most surprising revelation of all. The fact that a significant part of GitHub is not pure software projects as one would expect. As a matter of fact, there were countless instances of only text files, abandoned projects, tutorials including class notes, half-baked academic assignments, archived repos, resumes hosted online, research papers, lists of popular movies, books, software companies, food and so on and so forth. Furthermore, the majority of the projects are personal and inactive. As you can see in the chart these two (docs/tutorials and moved/empty/archived) together accumulate for 20% of the lot i.e., 1 out of every 5 repos. To make things worse, many repositories are forks or forks of forks and most of the time the original project would be of greater interest for research purposes rather than their duplicated counterparts [2]. This is synonymous with Matragkas et al. who rightly point out that a common trend on GitHub is to fork a project but make little to no changes [10].

Web applications aren't too far behind with 7% and lastly, we have database/storage, testing, APIs, parsers, protocol, utility and mobile applications grabbing negligible shares between 2 – 4% each.

Lastly, remaining software like games, drivers, operating systems, firmware collectively forms 9% of all repositories (as Others).

All in all, the distribution isn't completely asymmetrical. There isn't one type of software that greatly dominates. Although there is a preference for libraries/frameworks, this is due to the fact that these two were combined as one category. Initially, both library and framework were separate labels. However, one cannot deny the fact that these are perhaps the most confusing terms, often used interchangeably. Hence for the sake of simplicity, these two were clubbed together.

The GitHub Search API also made it possible to compare the various languages used across the board as seen in Figure 3. Ruby is used the maximum with languages like HTML and Shell following closely behind. C is used in 32 out of the 100 repositories and so is JavaScript. CSS is next with 30 occurrences. Python is gaining popularity with while Java seems to be diminishing over the years with only 9 repositories actively using it. Languages like Objective-C and Makefile occupy the middle spot whereas BatchFiles, Cmake scripts, Vimscripts find themselves in single digits. The Others category combines languages that have nearly negligible representation in the grand scheme of things but tother constitutes about 37%. It includes the likes of TeX, R, VBScript, Dtrace, Pascal, Ada, Assembly, etc.

To summarize, the key results are:

1. **Filtered sample is used over random sample to increase validity**
2. **Over 20% of all repos are not software but rather docs, tutorials, moved, archived or empty**
3. **There is no one language clearly dominating (unlike the 2000s where Java was significantly ahead of the competition) but a rather even curve with Ruby, Shell and HTML garnering the most love**

RQ2: Can we build a classifier for GitHub repositories?

These results give more vigour to develop a robust classifier that could separate various types of software. It shall save lots of time and effort otherwise spent on preprocessing and filtering the data set. Features that can very well be used to develop the model are ratio/lines of code, programming languages (if any), the number of contributors, stars, forks, people watching, last commit, pull request comments, issues, task board, etc. Alongside, care should also be taken to exclude prehistoric data points by going over the repository activity.

As it happens, manual analysis of the repositories during the course of this study allowed me to form many preliminary associations between the available statistics and the type of software. One obvious one is to look for any presence of code. GitHub conveniently provides information regarding the languages used in a project. So, if a particular repository does not use any language, it has to be either documentation or an empty project or a project that has been moved to a new place.

6 Implications

The biggest takeaway from this research study hands down is the fact that GitHub is not only used for software but also merely as a means to store something for private use in the future or to host and share something with the entire world.

Therefore, this study should serve as a cautionary tale for subsequent research activities that rely on data from GitHub. After all, an accurate dataset is what makes up the foundation for any investigation. Corroding that will not only lead to invalid and biased results but also affect future dependent studies.

Just to secure a dataset of 100, easy enough for precise labelling saw me going over 87,500 repositories. That is nearly ~0.11%.

Clearly, cleaning up the dataset and filtering out all the noise is an absolute must.

Lastly, this paper can be used as a starting point for researchers or students that would like to learn how to collect GitHub data. Student skills in programming classes can be improved through practical exercises in gathering, preparing, visualization and analyzing data from GitHub. Experiments like these pave the way forward for the future generations to explore the field further.

7 Future Work

This experiment helped solidify the significance of a GitHub repository classifier. Such a model would have limitless applications. From recommendation systems to even search engine optimization (based on the type of software), the list is practically endless. And incorporating the language trends acquired here would be the icing on the cake. Speaking of the language spread, the comparison done in this study is at an elementary level. There is room for future expansion to understand the rationale behind the numbers we observe in the results. Besides, results obtained here can be used to fuel software engineering predictions for the upcoming decade as Chen et al. do in their paper "An Empirical Study of Programming Language Trends" [21]

Further, one of the threats to validity for this project was the sample size. GitHub rate limits to 1000 requests per hour for its API[9]. This combined with insanely high noise levels and time constraints made it difficult to manually analyze large samples of data, leaving room for future explorations.

Acknowledgement

I would like to thank Shaowei Wang under whose supervision I have been conducting this study. Without his constant support and guidance, this paper would simply not have been possible, to say the least. Further, I would also like to express my gratitude to Mathew Russell whose work on Social Web Mining paved the path forward with minimum obstacles while formulating the methodology for this study.

[9] This is when `GITHUB_TOKEN` is GitHub Actions is used.

References

[1] Dabic, O., Aghajani, E. and Bavota, G., 2021. Sampling Projects in GitHub for {MSR} Studies. IEEE, pp.560--564.

[2] Diomidis Spinellis, Zoe Kotti, and Audris Mockus. 2020. A Dataset for GitHub Repository Deduplication. In Proceedings of the 17th International Conference on Mining Software Repositories (MSR '20). Association for Computing Machinery, New York, NY, USA, 523–527. DOI:https://doi.org/10.1145/3379597.3387496

[3] Georgios Gousios, Bogdan Vasilescu, Alexander Serebrenik, and Andy Zaidman. 2014. Lean GHTorrent: GitHub data on demand. In Proceedings of the 11th Working Conference on Mining Software Repositories (MSR 2014). Association for Computing Machinery, New York, NY, USA, 384–387. DOI:https://doi-org.uml.idm.oclc.org/10.1145/2597073.2597126

[4] Hope, C., 2021. What is a Plugin?. [online] Computerhope.com. Available at: <https://www.computerhope.com/jargon/p/plugin.htm> [Accessed 31 August 2021].

[5] J. Han, S. Deng, X. Xia, D. Wang and J. Yin, "Characterization and Prediction of Popular Projects on GitHub," 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), 2019, pp. 21-26, doi: 10.1109/COMPSAC.2019.00013.

[6] Kalliamvakou, Eirini, et al. “The Promises and Perils of Mining GitHub.” Proceedings of the 11th Working Conference on Mining Software Repositories, ACM, 2014, pp. 92–101, doi:10.1145/2597073.2597074.

[7] Linnaeus University, 2013. Mining Git Repositories - An introduction to repository mining. [online] Available at: <https://www.diva-portal.org/smash/get/diva2:638844/FULLTEXT01.pdf> [Accessed 11 September 2021].

[8] McLaughlin, E., 2021. What is Adobe Flash Player?. [online] SearchCIO. Available at: <https://searchcio.techtarget.com/definition/Adobe-Flash-Player#:~:text=Adobe%20Flash%20Player%20is%20software,computer%20or%20supported%20mobile%20device.> [Accessed 31 August 2021].

[9] Michael Färber. 2020. Analyzing the GitHub Repositories of Research Papers. In Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020 (JCDL '20). Association for Computing Machinery, New York, NY, USA, 491–492. DOI:https://doi.org/10.1145/3383583.3398578

[10] Nicholas Matragkas, James R. Williams, Dimitris S. Kolovos, and Richard F. Paige. 2014. Analysing the 'biodiversity' of open source ecosystems: the GitHub case. In Proceedings of the 11th Working Conference on Mining Software Repositories (MSR 2014). Association for Computing Machinery, New York, NY, USA, 356–359. DOI:https://doi.org/10.1145/2597073.2597119

[11] Pandhi, D., n.d. What Are the Characteristics of Most-Popular Apps? - A Case Study on Android Applications. [online] Available at: <https://github.com/darshanpandhi/What-Are-the-Characteristics-of-Most-Popular-Apps-> [Accessed 11 September 2021].

[12] Ramírez-Gallego, Sergio, et al. “A Survey on Data Preprocessing for Data Stream Mining: Current Status and Future Directions.” Neurocomputing (Amsterdam), vol. 239, Elsevier B.V, 2017, pp. 39–57, doi:10.1016/j.neucom.2017.01.078.

[13] Russel, M., 2014. Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+ .... 2nd ed. Sebastopol, CA: O'Reilly.

[14] Schuwalow, M., Richter, F., Ludwig, T. and Nicolai, J., 2017. In: WeAreDevelopers Conference. [online] Available at: <https://www.youtube.com/watch?v=b3TqaDXWTNs> [Accessed 11 September 2021].

[15] Şeker, A. , Diri, B. , Arslan, H. & Amasyalı, F. (2020). Summarising big data: public GitHub dataset for software engineering challenges . Cumhuriyet Science Journal , 41 (3) , 720-724 . DOI: 10.17776/csj.728932

[16] squareboat, n.d. Different Types of Software with Examples. Available at: <https://squareboat.com/blog/different-types-of-software-with-examples> [Accessed 31 August 2021].

[17] Statistics and data, 2020. The Most Popular Programming Languages - 1965/2020. [video] Available at: <https://www.youtube.com/watch?v=UNSoPa-XQN0&t=1s> [Accessed 11 September 2021].

[18] Valerio Cosentino, Javier Luis, and Jordi Cabot. 2016. Findings from GitHub: methods, datasets and limitations. In Proceedings of the 13th International Conference on Mining Software Repositories (MSR '16). Association for Computing Machinery, New York, NY, USA, 137–141. DOI:https://doi.org/10.1145/2901739.2901776

[19] Wilcox, L., 2021. [Blog] The 4 Main Types of Software, Available at: <https://www.leadwithprimitive.com/blog/the-4-main-types-of-software> [Accessed 1 September 2021].

[20] Y. Tian, M. Nagappan, D. Lo and A. E. Hassan, "What are the characteristics of high-rated apps? A case study on free Android Applications," 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), Bremen, 2015, pp. 301-310, DOI: 10.1109/ICSM.2015.7332476.

[21] Yaofei Chen, et al. “An Empirical Study of Programming Language Trends.” IEEE Software, vol. 22, no. 3, IEEE, 2005, pp. 72–79, doi:10.1109/MS.2005.55.