Creating a Dataset of GitHub Repositories and their Analysis

Shreyansh Kumar Surana
201601112 - BTech ICT
DA-IICT Gandhinagar, India
201601112@daiict.ac.in

Smit Detroja
201601113 - BTech ICT
DA-IICT Gandhinagar, India
201601113@daiict.ac.in

Mentor: Prof. Saurabh Tiwari, DAIICT Gandhinagar

Abstract—**Open Source Development (OSD) is now increasing with a rapid increase in the release rate. One of the excellent sources for placing open source software is on GitHub. The repositories on the GitHub consists of information such as project contributors, the number of commits and its contributors, releases, pull requests, programming languages, and issues. However, no large dataset of open source projects exists which features detailed information about the repositories on GitHub. In the BTP work, we developed a tool which helps in creating a data-set of repositories based on the pre-defined schema. Out of initial 1680 repositories, the dataset hosts 620 repositories (with applied basic filters of stars and forks), and 247 repositories (after applying all filters). The tool extracts the information of GitHub repositories and saves the data in the form of CSV. files and a database (.DB) file. Subsequently, the extracted dataset is analyzed based on the five formulated research questions (RQs).**

Index Terms—**GitHub, repositories, watchers, stars, forks, issues, subscribers, contributors, pull requests, releases, watchers**

I. Introduction

GitHub is the world's largest Open Source Development (OSD) platform which provides online software development version control using Git and runs in the command line interface. GitHub integrates many features of social network coding such as forks, giving stars to the repository, making the repository private and so on. GitHub is the choice for developers from large companies too [] []. It can be easily connected to many project management tools such as Amazon and Google Cloud Accounts, and Android Studio. The main reason for it being so much widespread is that it allows multiple collaborators (e.g., developers, testers, analysts etc.) to work on a single project at the same time. A single command can make sure that all the collaborators are on the same page.

Due to the large use of the Git platform, a large amount of data has been generated. We tried to utilise some of the data to answer some questions. We created a tool which helps us in fetching the data from GitHub and used the same data to give insights on various questions such as the relation between watchers and contributors, relation between stars and forks, patterns of contributions among the GitHub users, the relation between programming languages and issues of the repository etc. The motivation behind the work to make a tool which could collect data according to the certain parameters and that data could be helpful in analysis for finding trending developers, languages, repositories, etc across the complete platform. This tool is useful to recruit the most talented developers and also in finding patterns in selecting the language for a project, issues related to that, the relation between contributors and the statistical insights which are helpful in showcasing the work done in the repository or by the organization.

In this BTP work, we have developed tool support to extract repositories along with the information related to contributors, issues, pull requests, releases, and subscribers. The tool has three different options: creating a data set of repositories, creating consolidated data files of individual users or repositories. Basic terminology used in the paper would be repository (a storage space where the project is live), forks (copying the repository to our own space for personal use), and stars (for following a specific repository to know about future developments and changes) etc. The tool uses GitHub API[1] in the back end, used with the help of python library requests, to get all the relevant data needed for creating the data-set. The tool uses a Tkinter python library[2] to create a UI so that the tool can be used without any hassle.

After the creation of data set from the developed tool support (named *GitHub analyser*), we have analysed the dataset based on five research questions (RQs). Specifically, RQ1 looks into the relationship between watchers and contributors in the repository, RQ2 focuses on how the stars can be determined, RQ3 relates the type of contributions made in the repositories, RQ4 determines the contributors pattern among multiple repositories, and RQ5 analyse issues, effect of programming language within the repositories.

The BTP report is organised as follows: Section II presents our tool support developed for dataset creation from GitHub. Section III describes the tool architecture. Section IV presents the research questions formulated for analysing various aspects from the generated dataset. Section V presents the analysis results of each RQs followed by the limitations and future directions in Section VI.

[1]https://developer.github.com/v3/
[2]https://docs.python.org/3/library/tkinter.html

II. Tool Support for Dataset Creation

The *GitHub Analyzer* tool helps in creating a data-set of multiple repositories, single repository, and single-user statistics. Multiple repositories data-set consists of repositories belonging to multiple users. This means that it takes up the data of repositories from many users that satisfy the search criteria, and creates a data-set. Whereas, a single repository means that it has only one owner, and the data would be only about that specific repository instead of multiple repositories. Single user data-set implies that the data-set would have all the information, including all the repositories, regarding a single user. Fig. 1 shows the main layout of the tool and divided into 3 parts:

- Searching whole GitHub for the repositories related to a specific domain and getting all the possible details such as details of contributors, commits, releases, pull requests, languages and issues. To generate the dataset, the name of the topic (domain), and filter details such as the minimum number of stars, forks, releases and contributors need to be specified in the tool.
- Getting all the details of a single repository with its commit details. For this, the tool takes two inputs: username of the owner of a repository and an access token of the same username[3].
- Getting all the details about a GitHub user. For this, the user needs to specify the username of the owner of a repository and an access token of the same username[3].

A. Features

The features of the GitHub analyser are as follows:

- Creating a data-set of multiple repositories, in CSV files and DB file, based on a specific domain and filters provided by the user of the tool.
- Showing percentage wise contribution of all the contributors of a single repository.
- Storing all the user information, including details about all repositories in CSV files.
- Showing the most prominent programming languages in the user's profile using percentage wise programming languages used in the user's all repositories.

B. Data-set Creation

A user need to set multiple parameters to create a data set. User needs to give a topic name and can set the minimum number of stars, contributors, releases and forks that need to be present in the repository to be included in the data set. The data set is created with keeping the schema shown in Fig. 2.

Step 1: We searched the repositories that a user wants using GitHub API and sorted it with respect to the number of stars of the repository. The search criteria are determined by the user where he/she decided the topic name (domain) to be searched and decide on threshold number of stars the repository must have and the minimum number of times the repository must have been forked.

[3]https://help.github.com/en/github/authenticating-to-github/creating-a-personal-access-token-for-the-command-line

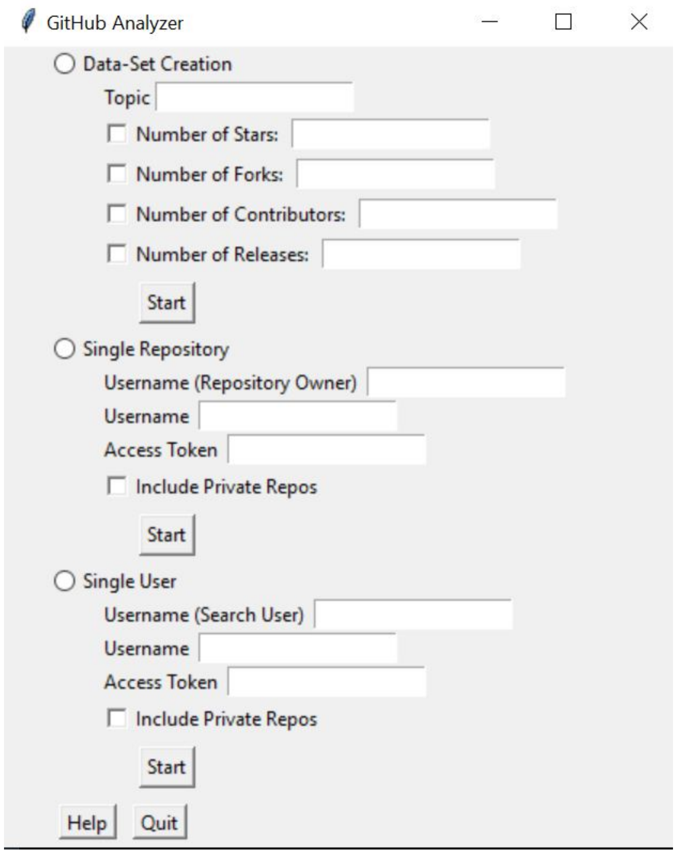


Fig. 1. *GitHub Analyzer* Tool

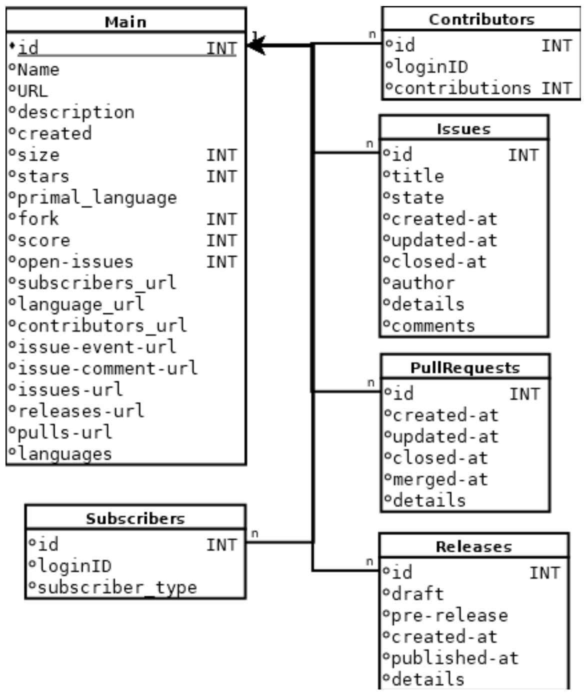


Fig. 2. Schema of Data Set

Step 2: In this step, we take each of the repositories and determine the prominent languages of each individual repository. This step enables us to collect the data regarding the languages that are used in a specific domain/field that has been searched. All the languages which are not prominent are also collected and saved. We also apply the second filter were, if user has given minimum number of releases, it is used here to filter out the repositories which have releases less than threshold minimum value given by the tool user.

Step 3: In this step, we extract more details about the repository, including details regarding the pull requests and collaborators of the project and filter the repositories based on inputs given by the user during the initial stage. We extract the issues that each repository has encountered. Using GitHub API, we collect all the details of each issue including the comments and details of the issue. This helps us in analysis questions that we have discussed further. We also fetch all the pull requests generated for the repository, and store it.

To understand better, the flow of data and the filters applied at a stage can be visualised using Fig. 3.

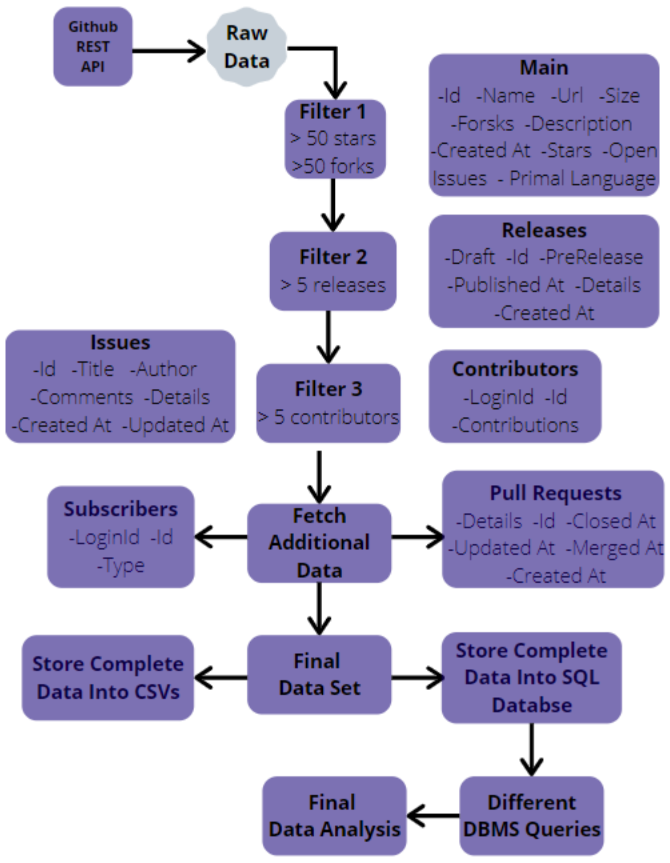


Fig. 3. *GitHub Analyzer* Data-Set Construction And Analysis Process Overview

C. Single User and Single Repository

Similar to above process, using GitHub API, we gave user of the tool more options to get all the data about a specific single repository and to get all the data about a specific GitHub user. A single repository analysis fetches all the basic details about the repository with all the commits whereas a single user profile search fetches all the data about the user. More details can be found in V-A and V-B.

III. Architecture Of GitHub Analyzer

GitHub Analyzer is an interactive tool developed in Python3.7. The tool uses Requests and HTTPBasicAuth library to fire the api queries to GitHub using the REST API for data extraction and uses Json file parser and Pandas' Data Frame to organize and store the data. Pandas' feature of converting Data Frames to csv is used to create CSV files. Python library SQLite3 is used to create a database out of the given formatted data (The database file can be properly opened only in SQLite3 software). Jupyter Notebook is used for detailed analysis of the data collected by the tool. Library MatPlotLib, WordCLoud and Seaborn is used to showcase the important results of the tool. Git, Gitpython and Pydriller is used to analyze the local repositories on the computer. The UI for ease of use is created using the help of python library Tkinter. Tkinter allows to create an interactive UI, and minimizes the use of directly running the main source file. Fig. 4 can be used to understand the basic data flow of the tool.

IV. Research Questions

Using the tool, we created a data set of these topics: distributed computing, embedded systems, software development, web development, bitcoin, cpp and data science. The parameters as mentioned in II-B, are shown in Table I.

TABLE I
Data-set Parameters

Stars	50
Forks	50
Releases	5
Contributors	5

Initially 1680 repositories were fetched without applying any of the filters. 247 repositories were collected after applying all the filters. Apart from the following questions, we carried out analysis of a single repository and a single user determining the ownership percentage of a repository or prominent languages used by the user.

RQ1: ***Is there any relation between the contributors and watchers in the repository of a specific domain? Is watcher in a specific repository contributing to some other repository?*** We look for proof about the watchers of a repository and their contribution. Do they take active roles in contributing to repository after watching it or they are just to receive the updates. We also try to look if they contribute to different repositories, other than the selected one.

RQ2: ***How can the stars be determined in the repositories? What are the parameters that contribute to the stars?*** Stars in GitHub can be considered as followers in social media.

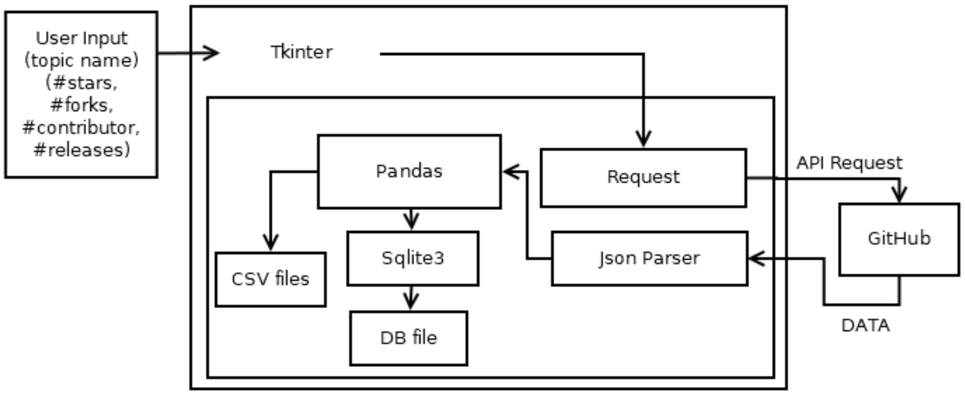


Fig. 4. Architecture of *GitHub Analyser*

It determines the popularity of a repository on the GitHub platform. We try to understand the correlation between stars and other important parameters that gives a complete idea about a repository such as number of contributors, forks, commits, issues, releases and age of the repository.

RQ3: ***How can the type of contributions can be determined in the repositories?*** The contributions on GitHub can be increased by various methods. As stars decides popularity of repositories, contribution determined popularity of a user on GitHub. We tried to separate out different type of events for contributions, namely, issues events, pull request event, pull request review event, and push event.

RQ4: ***Can we find a pattern of contribution among contributors of one repository?*** One repository can have many contributors suited to different parts, like the users who are developers, users who document the code, users who find issues etc. We try to find if a same user do multiple things in the same repository.

RQ5: ***What are open issues that lies within the repositories? Does the choice of programming language affect the count of open issues? How the popularity of the repository is affected by the choice of programming language?*** We try to visualise how much issues in general are encountered with respect to a particular programming language. We also try to associate the number of forks and contributions with programming languages to get a better idea of which language is more prominent in the current times.

V. Analysis and Results

Assumption:
To analyze the tool efficiently, we were in search of medium-sized private repositories. However, large repositories would have factored for long delays in analysis, while a small repository would be inefficient to cover all the parameters of analysis. Hence, we used one of our private repository for the analysis, which satisfied both the mentioned criteria.

A. Single Repository Analysis

The user gets the list of all the public and private repositories and can select any one of them to get insights about the repository. All the data related to the repository such as number of forks, issues, stars, watchers, repository license, important URLs, creation date and last update date, all the languages used in the repository, all the commit and the details of the commits are found at and stored in separate CSV files. Apart from this, a pie chart is created which shows how much of a contribution, a contributor has done in the given repository i.e., code ownership in the repository.

As an example, we used the tool to get insights about an old project named RCTMP which can be found at [2]. The total number of commits for the repository were 106 carried over the span of 4 months. Fig. 5 shows details of some of the commits of the given repository.

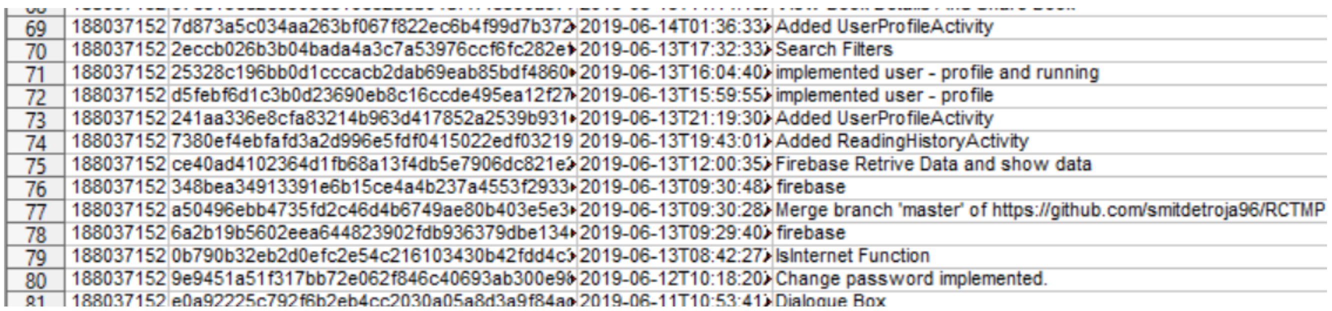
The repository has 3 contributors and their percentage wise contribution to the repository is show in the Fig. 6. The pie chart shows that contributor with username had most contribution in the RCTMP repository with a total of 37.7% share of contribution, followed by contributor shreyansh08 with 32.1% and gaurang-b with a contribution 30.2% in the repository.

If you manually look at a repository, you would find that many contributors are working simultaneously on different things such as new releases or new push request or solving issues. Hence, it becomes difficult to identify the contributors who are stand out among the other. This technique can be used to determine the major contributors of an ongoing or completed project.

B. Single User Analysis

The tool takes a username and returns all the details about the user's profile such as followers, following, company, all the details about all the repositories of the user. All the data is stored in form of CSV files.

Additionally, the tool takes up the data generated and uses it to find prominent programming languages in the user profile. A pie chart is generated denoting the percentage of each programming language used in the user's profile. Fig. 7 shows the percentage distribution of different programming languages that were present in GitHub profile of shreyansh08 [5]. The pie chart implies that the user has worked most on projects which has JAVA as base programming language taking up 48.8% of total code share, followed by programming language Python and MATLAB which takes up 22.3% and 16.7% of total code share. This can be used to determine if a user is comfortable



69	188037152	7d873a5c034aa263bf067f822ec6b4f99d7b372	2019-06-14T01:36:33	Added UserProfileActivity
70	188037152	2eccb026b3b04bada4a3c7a53976ccf6fc282e	2019-06-13T17:32:33	Search Filters
71	188037152	25328c196bb0d1cccacb2dab69eab85bdf4860	2019-06-13T16:04:40	implemented user - profile and running
72	188037152	d5febf6d1c3b0d23690eb8c16ccde495ea12f27	2019-06-13T15:59:55	implemented user - profile
73	188037152	241aa336e8cfa83214b963d417852a2539b931	2019-06-13T21:19:30	Added UserProfileActivity
74	188037152	7380ef4ebfafd3a2d996e5fdf0415022edf03219	2019-06-13T19:43:01	Added ReadingHistoryActivity
75	188037152	ce40ad4102364d1fb68a13f4db5e7906dc821e	2019-06-13T12:00:35	Firebase Retrive Data and show data
76	188037152	348bea34913391e6b15ce4a4b237a4553f2933	2019-06-13T09:30:48	firebase
77	188037152	a50496ebb4735fd2c46d4b6749ae80b403e5e3	2019-06-13T09:30:28	Merge branch 'master' of https://github.com/smitdetroja96/RCTMP
78	188037152	6a2b19b5602eea644823902fdb936379dbe134	2019-06-13T09:29:40	firebase
79	188037152	0b790b32eb2d0efc2e54c216103430b42fdd4c	2019-06-13T08:42:27	IsInternet Function
80	188037152	9e9451a51f317bb72e062f846c40693ab300e9	2019-06-12T10:18:20	Change password implemented.
81	188037152	e0a92225c792f6b2eb4cc2030a05a8d3a9f84a	2019-06-11T10:53:41	Dialogue Box

Fig. 5. Commits of the repository RCTMP from the CSV file

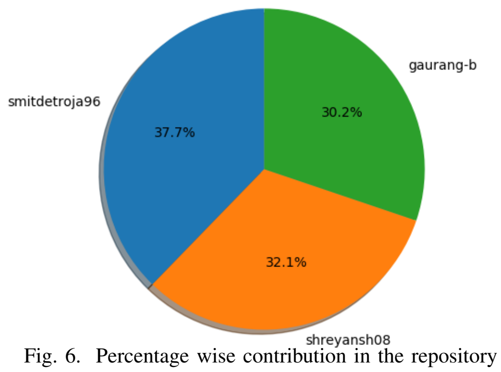


Fig. 6. Percentage wise contribution in the repository

and fluent in a specific programming language or not just by looking at user's github profile. Future scope, may include giving the tool user option to select the repositories to take into account for calculating the language share of the GitHub user or an option to discard some languages such as Jupyter Notebook or MATLAB for calculating the language share of the GitHub user.

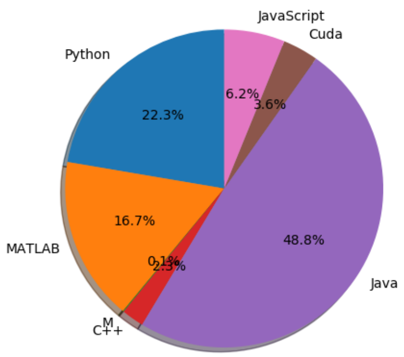


Fig. 7. Languages in profile of user: shreyansh08

C. Is there any relation between the contributors and watchers in the repository of a specific domain? Is watcher in a specific repository contributing to some other repository?

Watchers are those users who take a keen interest in the activity of a repository, with a probability that he/she will contribute to it. Any user can watch any public repository on GitHub but it does not mean that they would be able to contribute to it. Whereas a contributor is someone who contributes to the repository in a meaningful way like contributing to the main release code or pushing the new features to production code.

We found out that most of the watchers do contribute to some of the other ways to the repository. Most of the watchers contribute in the following type: issue reporting, issue assignment, commenting on issues, creating pull requests, commenting on pull requests. Percentage of watchers who become contributors is not large. But a large pool of watchers is important for the project's future. Watchers who become contributors contribute to the project over a long period. Watchers have an understanding of the project and its workflow before their contribution.

The Git user can watch and contribute in any number of repositories. Table II shows the part of analysis showing the relationship between contributors and watches with respect to all the repositories. Our analysis results found that the contributor in any of the repository can be a watcher of other or same repository. Our result of this RQ is also backed by the study conducted by Sheoran et al. [8] where they have found that the watchers begin contributing to the project and those contributors account for a significant percentage of contributors on the project.

TABLE II
QUANTITATIVE ANALYSIS ON RELATION BETWEEN CONTRIBUTORS AND WATCHERS

LoginId	Contributor In	Watcher In
AiusDa	1	6
Camille92	2	6
rustyrussell	4	6
traversaro	2	48
xuchen509	1	18

D. How can the stars be determined in the repositories? What are the parameters that contribute to the stars?

Starring a repository can be considered as an appreciation for the work done in that repository. GitHub's repository ranking is based on the number of stars a repository has.

From the Fig. 8, we can figure out how are the number of stars affected by different parameters such as number of commits, number of contributors, number of forks, number of releases, number of issues and the duration of the project. Table III and table IV depicts the numerical analysis of the parameters affecting number of stars.

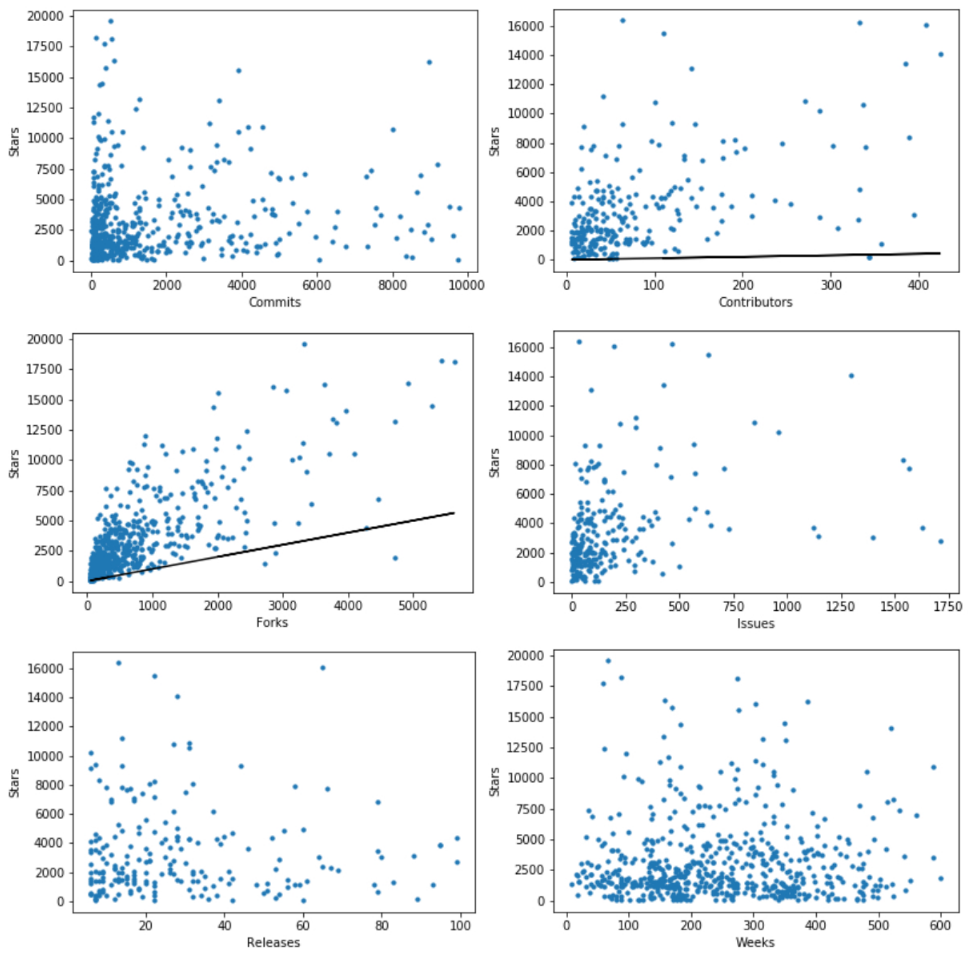


Fig. 8. Graphs showing relation of stars of a repository with commits, contributors, forks, issues, releases, weeks.

TABLE III
QUANTITATIVE ANALYSIS OF FILTERED REPOSITORIES

Average number of Stars	4249
Average Size	63658
Average number of forks	1067
Average number of issues	201

From Fig. 8, we can say that, the number of stars (in other words, popularity of repository), is not related to the age of repository. The correlation coefficient between number of stars and age of repository comes out to be 0.018. We cannot say anything about number of stars of repository based on its age. There are old repositories, who have very few stars and there are new repositories, with high number of stars. For example, a repository named *retrospectiva* is more than 10 years old, is one of the top repository of software-development but has only 520 stars whereas *gridstudio*, a web-based application is less than an year old and have more than 7500 stars already. The number of stars depends on value of the repository to an average user and cannot be dependent on the ”oldness” of the repository.

In Stars vs Contributors graph in Fig.8, the black line is x=y line. So, the repositories beneath the line have more contributors then the stars. The only such repository in out data set is *PlatON-Go*, which is based on distributed-computing. It has 343 contributors but only 180 stars. This can happen when repository belongs to an organisation and hence, they have large number of contributors from the same organisation but the tool itself is not as much useful to others and hence resulting in less number of stars. Number of stars and number of contributors are weakly related with correlation coefficient of 0.507. This means that we cannot predict popularity of repository based on how many contributors the repository have.

It can be seen from Fig.8, that as number of forks are increasing, the number of stars also increases. The correlation coefficient between number of stars and number of forks comes out at 0.753, which means high ¿75% correlation. This is due to the fact that, the more popular a repository is, more and more people would like to develop something related to it, and hence would fork it to develop on it. Also, there are 9 repositories with number of forks greater than number of stars in our data-set.

We can observe that number of commits are not relating to

TABLE IV
STATISTICAL ANALYSIS OF FILTERED REPOSITORY

No. of repos with no. of stars > avg	72
No. of repos with size > avg	69
No. of repos with no. of forks > avg	57
No. of repos with no. of issues > avg	61
Avg size of repos with no. of stars > avg	88950
Avg size for repos with no. of stars < avg	53252
Avg no. of stars for repos with size > avg	5992
Avg no. of stars for repos with size < than avg	3573
Avg no. of forks with no. of stars > avg	2502
Avg no. of forks with no. of stars < avg	477
Avg no. of stars with no. of forks > avg	10279
Avg no. of stars with no. of forks < avg	2440
Avg no. of stars with no. of issues > avg	8094
Avg no. of stars with no. of issues < avg	2977
Avg no. of issues with no. of stars > avg	357
Avg no. of issues with no. of stars < avg	138

number of stars. Number of commits depends on the use of repository. If repository belongs to organisation or is currently an ongoing project development, it would have large number of commits but less stars. The correlation coefficient of number of commits and number of stars is 0.147. Similarly, number of releases is also independent of number of stars. The correlation coefficient between number of stars and number of releases is 0.018. Correlation coefficient between number of stars and number of issues is 0.354. Number of stars and number of issues can be said to be weakly related. One important observation point is that high number of stars is not implying that there are high number of issues. The less issues even after so much users noticing of the repository means that the end product is more efficient in its function and hence more popular and reliable.

E. How can the type contributions be determined in the repositories?

GitHub uses events to specify which activity triggered the web hook. Each event has a specific JSON schema. Any type of contribution can be classified into 4 type of events :

1) ***Issues Event (IE):*** This event is triggered when an issue is opened, edited, deleted, pinned, unpinned, closed, reopened, assigned, unassigned, labeled, unlabeled, locked, unlocked, transferred, milestoned, or demilestoned.

2) ***Pull Request Event (PRE):*** This event is triggered when a pull request is assigned, unassigned, labeled, unlabeled, opened, edited, closed, reopened, synchronized, ready for review, locked, unlocked or when a pull request review is requested or removed.

3) ***Pull Request Review Event (PRRE):*** This event is triggered when a pull request review is submitted into a non pending state, the body is edited, or the review is dismissed.

4) ***Push Event (PE):*** This event is triggered on a push to a repository branch. Branch pushes and repository tag pushes also trigger web hook push events.

We used our already generated data to further fetch such events. We used the contributors from the topics- bitcoin, distributed computing(dist-comp)and web development(web-dev). Due to GitHub limitations [3], large data cannot be collected. Table V shows the collected data of type of contribution vs contributions of different contributors of the above mentioned topics.

TABLE V
TYPE OF CONTRIBUTIONS

Type of Contribution	**Contributions**		
	bitcoin	***web-dev***	***dist-comp***
PE	2424	3399	3662
IE	246	535	542
PRE	579	1147	1399
PRRE	0	0	0

Table V can be used to infer that the push events are most occurring events i.e., users generally tends to push the data to respective repository. We can also notice that issue events are not as frequent as pull request events among the included contributors. If we take the count of push and pull request events against the issue events, we can say that the contributors who actively take part in development of a repository are not that much keen on finding the issues in their own code base. Generally, commercially developed tools are more used in place of open source software tools. One of the big reason can be the low number of users working on the issues as shown by the Table V.

Another important observation that can be inferred is that no one from our included set of contributors uses the pull request review events. This can be attributed to the fact that most of the users, who are regular contributors to the given repository, are not denied the pull requests. As the contributors taken into consideration in the data-set are contributors to the repository, there is really low chance that their requests are denied. Only if you are a new contributor, your requests can be denied on the basis of lack of clean code or proper documentation or any other reason. If the request is denied, the user would not be counted as a contributor of the repository. This is also the reason why there is no *Pull Request Review Event* type of contribution among the given contributors.

F. Can we find a pattern of contribution among contributors of one repository?

From the count of contribution for each type of contribution we can make a correlation between contributors of the same repository. If count for each type of contribution is nearly equivalent then we can say that there is a strong correlation between that users which can strongly infer that they may be doing the same type of work in the repository. If count for each type of contribution is not equivalent then we can say that there is a weak correlation between that users which can strongly infer that they may not be doing the same type of work in the repository. This analysis can further be extended to set correlation between issues in the repository also.

TABLE VI
PATTERN OF CONTRIBUTIONS FOR REPO: **YATIN2410 / WHATEVER**

User	Type Of Contribution	Contribution
SyntaxC4	Push Event	57
SyntaxC4	Issues Event	2
SyntaxC4	Pull Request Event	24
SyntaxC4	Pull Request Review Event	0
berndverst	Push Event	58
berndverst	Issues Event	16
berndverst	Pull Request Event	14
berndverst	Pull Request Review Event	0
cephalin	Push Event	5
cephalin	Issues Event	0
cephalin	Pull Request Event	4
cephalin	Pull Request Review Event	0
microsoftopensource	Push Event	255
microsoftopensource	Issues Event	0
microsoftopensource	Pull Request Event	0
microsoftopensource	Pull Request Review Event	0
smitdetroja96	Push Event	69
smitdetroja96	Issues Event	0
smitdetroja96	Pull Request Event	0
smitdetroja96	Pull Request Review Event	0

From the Table VI, we can see that user **smitdetroja96** and **microsoftopensource** are doing same kind of contribution, **SyntaxC4** and **berndvest** are nearly contributing in the same pattern. While **cephalin's** pattern of contribution doesn't match with anyone else. From this we can arrange the contributors into 3 groups working with different methods for the repository.

G. What are open issues that lies within the repositories? Does the choice of programming language affect the count of open issues? How the popularity of the repository is affected by the choice of programming language?

Most of the software projects have some kind of bug tracker to keep the software in its best performance. GitHub has it's own tracker known as **issues**. Issues are the best way to keep the track of bugs, pending tasks and improvements of the project. GitHub's issues are special because of excellent formatting, collaboration, references, features for organizing issues, crossing connecting other issues.

From Table VII, we can see that Python is having the maximum number of open issues with less amount of contribution while C++ is having the maximum contribution with respect to comparative amount of issues. Considering the number of forks Python ranks first with C++ on the second. The number of stars determine the popularity of a repository. We can see that the repositories with Python as primal language have largest number of stars. This is followed by JavaScript programming language. This corresponds to the fact that most popular repositories on GitHub are built on Python and JavaScript. Those 2 have vast library base and applications, resulting in most programmers using one or the other. The other languages' low number of stars signifies the use of programming languages in only specific domains where there are no alternatives.

TABLE VII
FACTORS AFFECTING CHOICE OF LANGUAGE

Primal Language	Stars	Contribution	Forks	Open Issues
C	51232	58	14415	5471
C++	198127	10450	63970	9527
Go	36295	15	5894	1516
JavaScript	221635	111	44547	5217
Julia	2079	3	341	324
PHP	14153	6	2863	727
Python	336722	66	86333	15859
Rust	26247	16	2512	666
Shell	1517	2	409	123
Typescript	23043	166	6853	1970

VI. LIMITATIONS AND FUTURE SCOPE

A. Limitations

- GitHub API allows maximum of 5000 requests per hour for a logged in account. This shortens the maximum amount of data that can be retrieved in a single run of program.
- The details of contributions of user are only kept for the past 90 days, or to a maximum of 300 events.
- The API is subjected to change anytime in the future without any notice period, making many of the applications unable to adapt in a short span of time.
- In order to keep the API fast for everyone, pagination is limited. We cannot traverse the complete data which increase the number of requests to fetch the complete data.. This reduces the amount of data that can be collected [3].

B. Future Scope

- The data set can be used to answer many other questions. The database format can be used to execute many SQL queries that can give many interesting results.
- With the use of deep learning algorithms, more details can be obtained such as finding patterns in the issues, finding choice of language for any specific topic etc.

REFERENCES

[1] https://github.com/shreyansh08/GithubAnalyzer
[2] https://developer.github.com/v3/
[3] https://developer.github.com/v3/#pagination
[4] https://help.github.com/en/github/authenticating-to-github/creating-a-personal-access-token-for-the-command-line
[5] https://github.com/shreyansh08
[6] S. D. Joshi and S. Chimalakonda, "RapidRelease - A Dataset of Projects and Issues on Github with Rapid Releases," 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), Montreal, QC, Canada, 2019, pp. 587-591.

[7] M. Martinez and M. Monperrus, ”Coming: A Tool for Mining Change Pattern Instances from Git Commits,” 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Montreal, QC, Canada, 2019, pp. 79-82, doi: 10.1109/ICSE-Companion.2019.00043.

[8] Jyoti Sheoran, Kelly Blincoe, Eirini Kalliamvakou, Daniela Damian, and Jordan Ell. 2014. Understanding “watchers” on GitHub. In Proceedings of the 11th Working Conference on Mining Software Repositories (MSR 2014). Association for Computing Machinery, New York, NY, USA, 336–339. DOI:https://doi.org/10.1145/2597073.2597114

[9] H. Lee, B. Seo and E. Seo, ”A Git Source Repository Analysis Tool Based on a Novel Branch-Oriented Approach,” 2013 International Conference on Information Science and Applications (ICISA), Suwon, 2013, pp. 1-4, doi: 10.1109/ICISA.2013.6579457.