A Tool for Mapping Topics to Public GitHub Repositories

Nima Tayefeh
EECS
University of Tennessee
Knoxville, United States
ntayefeh@vols.utk.edu

Neh Patel
EECS
University of Tennessee
Knoxville, United States
napatel79@vols.utk.edu

Kyung Han
EECS
University of Tennessee
Knoxville, United States
kyumhan@vols.utk.edu

Chris Lee
EECS
University of Tennessee
Knoxville, United States
clee95@vols.utk.edu

***Abstract*— With more and more public repositories being published onto GitHub, the need for a simple way to search for a tool to search for repositories related to specific topics becomes ever more important. Though the GitHub website does contain a search option, it does not do well at finding repositories that are related to topics that are not named after the search query. As such, in this experiment, we attempted to answer the question of whether public repositories can efficiently be queried and searched to find desired topics and keywords. We propose a package that allows for a way for users to map their search queries to repos based off file content and names. This package utilizes the World of Code infrastructure and Oscar Python Library to facilitate efficient repository querying. With over 128 million public repositories posted on the website by 23 million different authors, it can be a difficult task to search all these repositories [1]. With a modern tool that facilitates this task for you, the research and open-source community can more effectively complete their tasks. Despite being a challenging and timely task, this package effectively is able to map keywords to project SHAs, Blob SHAs, and commit SHAs. This experiment focuses primarily on COVID related search queries to test the success and effectiveness of the tool.**

Keywords—GitHub, open source, world of code, COVID-19, public repository, mapping

I. Introduction and Motivation

With source and version control becoming such a common and necessary task for developers, the need for a cloud based service becomes obvious. GitHub has served as the main solution for this problem for the past couple of years and has the majority of the market. In fact, GitHub has almost 80% of the market share with the next biggest only being used by less than 5% of developers [2]. With such a large market share, the number of authors contributing to public repositories on the site is extremely large and will only continue to grow. On the current web client, there is a search option that allows for search queries to find authors or projects based off the keywords provided; however, this tool is not very efficient or useful since it relies on files and projects to contain the search string in their names.

To create a more extensive search function and mapping tool, this experiment utilizes World of Code [3] to provide a command line based tool to map keywords to the public repositories. By using a tool similar to our library, developers can easily maintain collections and references to more easily find reliable sources and documentation. With an increased access to necessary and useful information, more productive and significant development can continue.

II. Related Work

Though there are not many published works on mapping repositories to their content, there is much research conducted on analyzing the content of the files in public repositories. One example of this is the DejaVu package [4]. This package allows for the removal of duplicate and non-unique files in GitHub, as only around 30% of the files published on GitHub are unique [4]. This tool utilizes token hashes of the files to measure the similarity of files, since a token hash can contain information regarding whitespace and ordering [4]. This project strictly utilizes tokens to facilitate more efficient indexing due to the large input space of the number of files. Though this methodology works well for comparing similarities, more information regarding the files and repositories is needed to know if the project has some context of specific topics.

Another related paper contains a similar objective as our experiment. In the paper2repo project, research papers are mapped to relevant public repositories on GitHub [5]. In this experiment, the authors combine text encoding and graph convolutional neural networks to find the main text concepts in the corpus [5]. Similar models are used for the repos where the graph structure of the repo is used as the content of the repo. Because their strategy utilizes machine learning, there needs to be ground truths to test their results on. As such, they utilized human graders to provide scores of the similarity of the repos and paper [5]. As humans are required to compare the results to, the results were not as high as intended. As such, in our experiment we chose to not utilize machine learning and instead create a complete searching and mapping tool based on located strings.

III. Dataset

To retrieve information regarding the repositories on GitHub, the World of Code API is used to easily contain access to maps containing information regarding the repositories. These maps contain information regarding projects, commits,

blobs, files, etc. The maps used for this experiment are explained in the table below.

TABLE I. DATA MAPS

Map	Description	Size
P2p	Mapping containing project and deforked project names	Finds 209,048,151 projects
P2c	Mapping project names to commit hash SHA	Finds 131,175,609 commits
P2b	Mapping commits to blob SHA	Finds 6,602,635,571 blobs
b2f	Mapping blob SHA to filenames	Not saved into a temporary file due to large file size

Using these three mappings, our tool is able to gain extensive information regarding the specific repository. These mappings are saved into their own text files such as allProjects.txt or allCommits.txt for easier access.

IV. METHOD AND EXPERIMENTAL SETUP

A. Search Queries and Options

When running the tool, there are numerous different options that can be entered to pick the style of mapping. These options and functionalities are outlined in the table below.

TABLE II. CHOICES OF OPERATION

Option	Functionality
-c [outputFile.txt]	- Maps search queries to all commit SHAs that contain any of the queries in their commit messages. - Writes to the optional output file or found_commits.txt
-p [outputFile.txt]	- Maps search queries to all project repository names that contain any of the queries in their name - Writes to the optional output file or found_projects.txt
-r [outputFile.txt]	- Maps search queries to all blob SHAs contain any of the queries in the content of the readme - Writes to the optional output file or found_readmes.txt

Upon selection of the options, the tool will ask for desired search queries to look for. It will continue to accept new queries until *Control-D* is entered, when it will then begin the mapping functionalities. In the case of our experiment, we utilized the queries for COVID-19 mapping: ['vaccin', 'covid', 'quarantin', 'corona', 'lockdown', 'social dist']

B. Finding Projects

When the *-p* option is selected in the running the tool, a Python function will run that utilizes the grep tool in Unix to search for each of the desired search queries. This would involve running the command:

zcat allProjects.txt | grep "query1\|query2 \|query3" (1)

Upon completion the command will output the project names to found_projects.txt or the optional filename entered in the beginning.

C. Finding Commits

When the -c option is selected, a Python function runs that reads each commit SHA. This function utilizes the OSCAR API to retrieve World of Code functionality in Python. It takes each hash and converts it to a Commit object, where it can then access its *message* attribute to obtain the commit message that was entered before being pushed to GitHub. The function then iterates through each search query and if any are found in this commit message, it saves the commit SHA to write to the output file.

D. Finding Related Readme Files

When the -r option is selected, a Python function runs that reads each project blob SHA. This function utilizes the OSCAR API to retrieve World of Code functionality in Python. It will run the b2f mapping on each blob SHA to obtain the filenames in that blob. Any file with “readme” in its name is saved in a temporary list. When this list contains 5000 separate readme files, the showCnt tool in the World of Code API is run on these blob SHAs. This command when run with “1” as its last command line argument writes the base64 encoded string of the entire file content to a temporary output file. A new function can then access this output file and decode each base64 into a string that can be searched for the queries. If any of the queries are found, that project blob SHA is saved to the output file. This process then continues for the next chunk of 5000 readme files.

V. RESULTS

When running these different operations, the finding of relevant projects based off readme files took the longest by far, which was expected. To retrieve results in time, the input file was truncated to containing xxx project blobs. Out of these blobs, 23,217 project readme files were found to contain any of the query strings related to COVID-19. This number ended up being smaller than we had predicted, as over xxx readme file blobs were found.

TABLE III. README FILE MAPPING RESULTS

All Project Blobs	Potential Related Files	Found Readme Files
6,602,635,571	4,130,286	23217

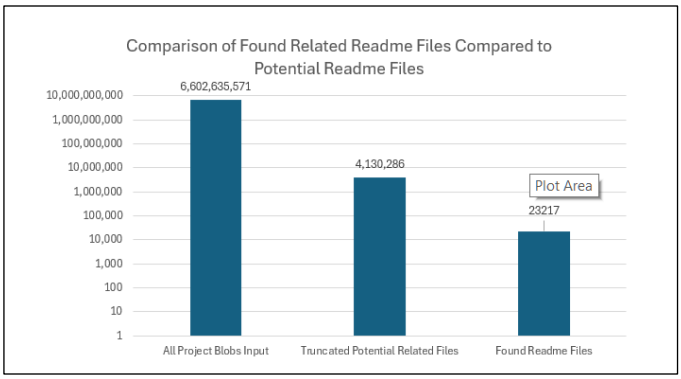

Figure 1: Comparing the Numbers in the Readme Operation

Finding the number of project names that contain these strings was the easiest and most efficient of these operations, as it just relied on the grep tool and took less than an hour to complete. Upon completion, 327,020 projects were found related to COVID-19 based off the project name.

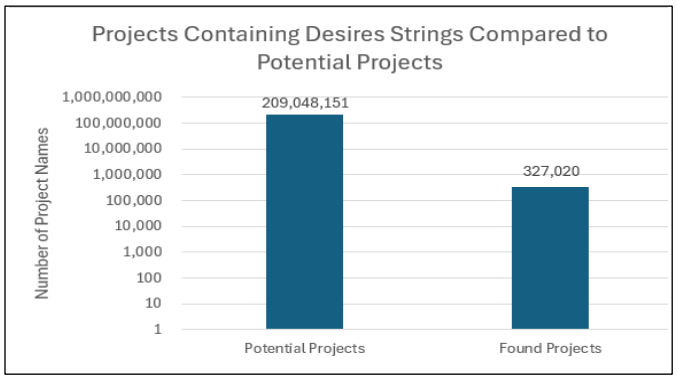

Figure 2: Comparing the Numbers in the Project Name Operation

Finding the commits containing the strings containing the query strings was the other operation that ended up taking a quite long time. To get example results quicker, this operation was also truncated and exited early. Upon truncation, xxx commit blobs were checked with over xxx commits containing messages that used the search queries.

TABLE IV. COMMIT MAPPING RESULTS

All Commits Input	Truncated Commits Input	Found Commits
131,175,609	39,686,000	529,767

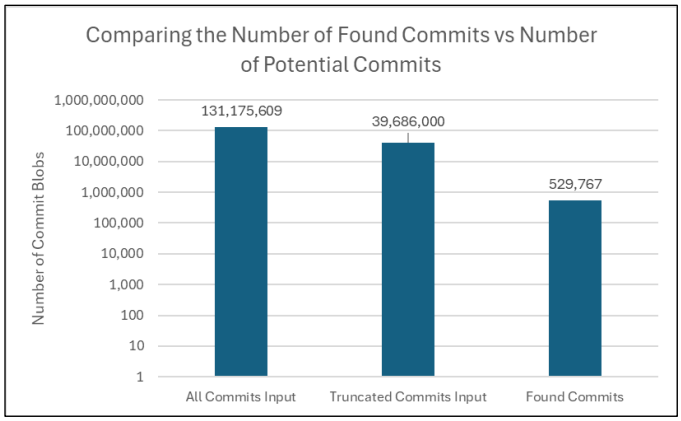

Figure 3: Comparing the Numbers in the Commit Message Operation

VI. CONCLUSION AND DISCUSSION

Upon looking at the different results found in the plots and tables, the performance of this tool can be seen. Despite being very effective at mapping the search queries to the correct repositories, the greater problem is the time taken. Because a linear search is essentially needed to conduct these mappings, most operations will take very long and will continue to increase as the number of public repositories and commits increases. Further, for this tool to become portable and more accessible the data maps would need to be more accessible. The majority of this experiment was conducted on a shared system accessed with SSH, the results were much slower than if performed on a local system, as the resources are shared, and the API requires multiple SSH calls to different systems.

These elements served as some of the main complications in contributing as a group to the development. To get around these implications, the group spent time together using the collaborative Live Share feature in Visual Studio Code allowing for all members to write to the same shell and filesystem. This facilitated easier collaboration in the development of the tool and bash scripts to modify results and truncated input.

VII. FUTURE WORK

Looking at the previous related works, it makes sense why machine learning approaches were used to calculate similarity, as the content of something like commit messages or readme files can be quite large, resulting in a linear search to take even longer. It would be interesting to apply this machine learning encoding approach to our problem by potentially generating a basic document about the search queries through a GAN and then use an encoding approach to find readmes or commit messages that have a very high similarity score, which would likely indicate that the repository content contains related materials.

In a future case of this experiment, the main priority should be to focus on increasing the portability and efficiency of these operations. Some options to accomplish this could be to use more efficient data structures such as Tries and using different algorithms like regular expressions. Further, using more experimental substring search algorithms like the Knuth-Morris-Pratt algorithm which utilizes partial match tables to avoid unnecessary comparisons [6]. The Boyer-Moore also uses pre-processing tables to facilitate more efficient searching by changing the search patterns dynamically [7]. Lastly, parallel computing can be used through libraries like PySpark to process numerous chunks of the input data maps at the same time or when searching the readme content for the desired strings. With these changes, the tool should operate much quicker than its current version

VIII. CONTRIBUTIONS

TABLE V. CONTRIBUTIONS

Contributor	**Tasks**
Nima Tayefeh	Readme Mapping Logic, bash scripts for splitting large files for quicker lookup
Neh Patel	Commit Mapping Logic, error and exception handling
Kyung Han	Commit Mapping Logic, generating truncated input files
Chris Lee	Project Mapping Logic, command line parsing

REFERENCES

[1] N. Kashyap, "GitHub's Path to 128m Public Repositories," Medium, https://towardsdatascience.com/githubs-path-to-128m-public-repositories-f6f656ab56b1

[2] S. Ravoof, "GitLab vs Github: Explore Their Major Differences and Similarities," Kinsta, https://kinsta.com/blog/gitlab-vs-github/

[3] A. Mockus, "World of code (WOC)," WoC, https://worldofcode.org/

[4] C. V. Lopes *et al.*, 'DéjàVu: a map of code duplicates on GitHub', *Proc. ACM Program. Lang.*, vol. 1, no. OOPSLA, Oct. 2017.

[5] H. Shao *et al.*, 'paper2repo: GitHub Repository Recommendation for Academic Papers', in *Proceedings of The Web Conference 2020*, Taipei, Taiwan, 2020, pp. 629–639

[6] X. Lu, 'The Analysis of KMP Algorithm and its Optimization', *Journal of Physics: Conference Series*, vol. 1345, no. 4, p. 042005, Nov. 2019.

[7] O. Danvy and H. K. Rohde, 'On obtaining the Boyer–Moore string-matching algorithm by partial evaluation', *Information Processing Letters*, vol. 99, no. 4, pp. 158–162, 2006.