Code Mapper: Mapping the Global Contributions of OSS

Thomas Le Tourneau
CY Tech
Cergy, France
letourneau@cy-tech.fr

Jasmine Latendresse
Data-driven Analysis of Software (DAS) Lab
Concordia University
Montreal, Canada
jasmine.latendresse@concordia.ca

Ahmad Abdellatif
University of Calgary
Calgary, Canada
ahmad.abdellatif@ucalgary.ca

Emad Shihab
Data-driven Analysis of Software (DAS) Lab
Concordia University
Montreal, Canada
emad.shihab@concordia.ca

ABSTRACT

Free and Open Source Software (FOSS) has reshaped the software landscape. Software developers from around the world contribute to the development and maintenance of these projects. The geographic diversity within FOSS offers insights into community dynamics, collaboration patterns, and inclusivity. Despite the rich insights that can be gained from this geographic diversity, there remains a scarcity of research in this area. One possible reason for this gap in studies is the lack of tools that can identify and visualize the geographic distribution of contributions in OSS projects.

We present Code Mapper, a tool that identifies the location of contributors in GitHub projects. To enable users to explore the global influence of their projects, Code Mapper visually presents the geographic distribution of project contributors. To accelerate future research in this area, we have deployed Code Mapper at https://codemapper.alwaysdata.net and have made our source code publicly available online. A demonstration of Code Mapper can be viewed at https://www.youtube.com/watch?v=AtARvrBJbVM.

KEYWORDS

Open source, Machine Learning, Software Development

ACM Reference Format:
Thomas Le Tourneau, Jasmine Latendresse, Ahmad Abdellatif, and Emad Shihab. 2024. Code Mapper: Mapping the Global Contributions of OSS. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3639478.3640030

1 INTRODUCTION

Free and Open Source Software (FOSS) has become a cornerstone in the evolving world of software development. The global OSS market was estimated to be valued at USD 27.73 billion, with expectations to reach USD 75,209.66 million by 2028.[1] Collaborative platforms like GitHub[2] have played a crucial role in bolstering the FOSS movement over the last decade. GitHub is a cloud-based service for software development, enabling over 100 million developers to host, share, and collaborate on open-source software projects through *repositories*. The collaborative nature of FOSS facilitates geographically dispersed volunteers in contributing to software development and improvement [6]. While the geographic diversity in FOSS is a crucial aspect, it has not been extensively explored with practical tools. Our tool introduces a novel approach to address this gap, making the study of geographic diversity in FOSS more accessible and applicable.

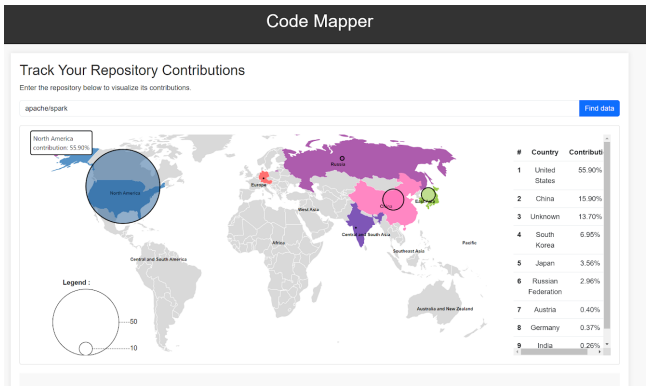

Figure 1: Visual Representation of Repository Contributions.

In a recent study focused on understanding the geographic diversity of contributors to public code and its evolution over 50 years [6], the authors analyzed 2.2 billion commits from 160 million GitHub projects. They employed heuristics (e.g., name, email addresses and UTC offsets of commit timestamps) to determine the contributors' locations. The results of this study reveal a consistent trend of increasing geographic diversity in contributions to public code. Nevertheless, such heuristics may not be available on all contributor profiles, which limits the applicability of this approach. Furthermore, to the best of our knowledge, there is currently no

ICSE-Companion '24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0502-1/24/04…$15.00
https://doi.org/10.1145/3639478.3640030

[1] https://www.benzinga.com/pressreleases/23/09/34518020/open-source-software-market-overview-2023-and-forecast-till-2030-report-pages-113
[2] https://github.com

tool capable of identifying and visualizing the geographic locations of project contributors.

Thus, we present Code Mapper, a novel tool that visually depicts the geographic distribution of contributions in an open-source repository hosted on GitHub. Through its interface, users can explore the global footprint of any given project, as depicted in Figure 1. Code Mapper leverages machine learning to infer contributors' probable geographic locations and can be utilized by developers, project managers, and organizations looking to integrate open-source code into their systems. By understanding the geographical distribution of contributions, one can make more informed decisions, ensuring the stability and security of one's software projects. We have made both our source code and dataset accessible to the public to facilitate further research in this area.[3]

2 A WALKTHROUGH OF CODE MAPPER

Code Mapper offers developers a geographical view of GitHub contributions for open-source repositories, highlighting the primary regions or countries of development. Figure 2 showcases an overview of Code Mapper, which has two main components: 1) a **front-end** for visualizing the region or country of the contribution, and 2) a **backend** for retrieving and analyzing contributions and metadata.

2.1 Front-end of Code Mapper

Upon visiting https://codemapper.alwaysdata.net, users are shown the Code Mapper homepage depicted in Figure 3. This page allows users to explore contributions by region or country for any open-source repository. For this, users input the repository name (e.g., `apache/spark`) and click the "Find data" button. Then, the front-end component forwards the repository name to the backend component to be analyzed. Once the repository is analyzed, contributions are visualized on a map with circle sizes proportional to the contribution volume from each country or region, as shown in Figure 1. Hovering over a circle reveals the percentage contribution for that specific region or country. In addition to the map that presents the contributions, the front-end features a table displaying the exact contribution percentages for each respective location.

2.2 Backend of Code Mapper

To provide a visual representation of the contributions by the front-end component, Code Mapper's backend performs two sequential steps: 1) obtaining contributor information, and 2) location inference. Code Mapper's backend component receives a user's request for a specific repository from the frontend component. The backend then interacts with the GitHub API to obtain the list of contributors and their meta-data (e.g., name and commits) for the designated repository. Next, the backend processes the collected contributor information to determine the geographical location of the repository's contributions. Then, the results are forwarded back to the front-end where they are rendered and presented to the end-user. Finally, we store the results of each request in an internal database for future requests.

Step 1: Obtaining Collaborator Information. The first step in the backend is to collect user data. To achieve this, we leverage GitHub GraphQL[4] to retrieve a list of the top 100 contributors (based on commit count) who have contributed more than 10 commits to a repository. This threshold is to mitigate GraphQL's resource limitations.[5] The rationale behind capping at 100 contributors is based on our observation that, beyond this threshold, contributors typically have a significant drop in commit frequency, minimizing their impact on the repository's contributions [7, 8]. After obtaining the list of contributors, we extract information that is publicly available from their GitHub profiles, such as their GitHub username, name, location, and commit timestamps. Next, we process the extracted information to determine likely country or regional affiliations, as detailed below.

- **Contributor Profile Location**. Every user on GitHub has an account profile that contains personal information such as name. GitHub users can enter their location in their profile.
- **Contributor Name**. Names can be a significant cultural marker and can provide insights into a person's likely geographical origin. This is because certain names or name combinations are more prevalent in specific regions or countries [6].
- **Contribution Commit Timestamps**. A contributor's activity on GitHub, particularly the times at which they make code contributions (commits), can hint at their geographical location. This is based on the assumption that people generally work during specific hours that align with their local time zones. By analyzing the UTC offsets of commits, one can estimate the contributor's likely time zone. Thus, we extract the UTC offsets from the last 100 commits of a contributor for a given repository. We use the median of these offsets to get a stable estimate. This inferred time zone can then be used in conjunction with the other data points to improve the accuracy of the location inference.

The above-described data features are then used in the subsequent location inference step.

Step 2: Location Inference The location inference mechanism of Code Mapper is done through two distinct methods, ensuring that Code Mapper not only captures user-centered locations but also possesses the capability to predict the location with considerable accuracy in the absence of direct information. We detail each method below.

Direct inference from GitHub Profiles. When a contributor provides their location on their GitHub profile, we prioritize this information as our primary source of truth. However, the entered locations can be broad or specific, ranging from country names to city identifiers. To decipher this, the backend component cross-references user input against an extensive database containing over 140,000 global cities (sourced from OpenDataSoft[6]). Similar to prior work [6], we prioritize cities with the most probable country based on their population (i.e., cities with higher populations), in instances of ambiguous matches that have several potential matches. For example, 'London', which can be found in both Canada and England

[3] https://github.com/jaslatendresse/codemapper

[4] https://graphql.org/

[5] https://docs.github.com/en/graphql/overview/resource-limitations

[6] https://www.opendatasoft.com/en/

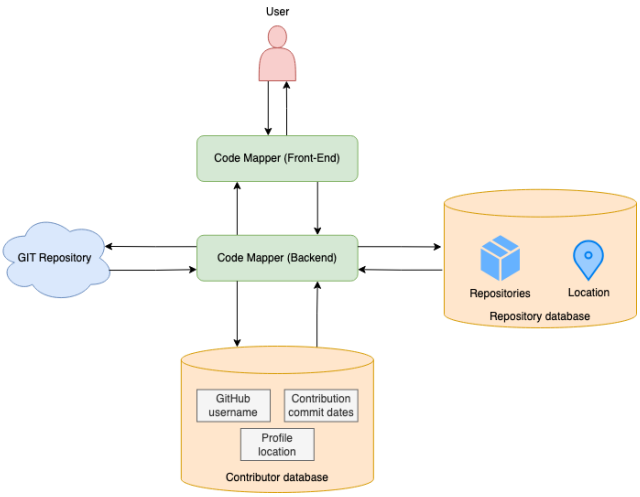

Figure 2: Overview of Code Mapper.

– we select the country with the highest population to derive the country, in this scenario, England. This approach is adopted based on methodologies established in prior work [6]. Then, to further ensure the accuracy of the derived location, we cross-verify the user-provided location by examining the median UTC offsets from the contributor's commits. For example, if someone claims to be from Paris, France, and their median commit times align with typical working hours in Central European Time (CET), this reinforces our confidence in the stated location. In the case of an inconsistency between the user-provided location and the commit time analysis, we resort to the next described approach.

Machine Learning-based Prediction. In cases where we cannot infer the location from the user's profile or when the contributor's profile does not specify the location, we leverage a machine learning model to predict the contributor's location. We train the model on the contributor's UTC offsets and name-based country probabilities from the data obtained in the previous step. To compute the name-based country probability, we first dissect names into subnames (e.g., "Jean Luc" becomes "Jean" & "Luc"). Next, we determine the subname's prevalence per country using a World of Code database, which leverages Forebears[7] data (a repository of over four billion unique names) [6]. If multiple subnames are associated with a single country, we first sum the probabilities of all these subnames. Then, we adjust this combined probability by factoring in the country's population. This provides a more precise representation of the likelihood that a set of subnames originates from a particular country.

3 EVALUATION

As mentioned in Section 2.2, when the contributor's location isn't explicitly mentioned on their GitHub profile, a machine learning model is used to determine the location. The prediction is based on the contributors' UTC offsets and name-based country probabilities. Below we describe the process of curating the dataset of contributors, selecting the best model for our use case, evaluating the selected model, and the deployment of the selected model.

[7]https://forebears.io/

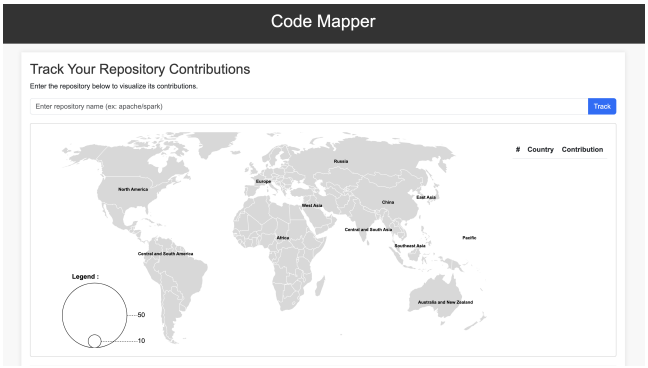

Figure 3: Code Mapper's Home Page.

3.1 Dataset Curation

We experiment with various ML classifiers to identify which ones most accurately predict a contributor's location using a representative dataset composed of 73,144 distinct GitHub repositories. These repositories are sourced from SEART GitHub Search Engine [5], a platform that allows to sample GitHub repositories statistics and metadata using various selection criteria, and have been filtered based on a minimum of 100 commits and 20 contributors. These selection criteria help ensure the relevance and robustness of our data in several ways. First, repositories with at least 100 commits are more likely to be real projects rather than experimental or test repositories. Second, a minimum requirement of 20 contributors increases the likeliness of diversity of contributions, which can offer a broader picture of geographic distribution.

From this pool of repositories, we collect data (name, profile location, commits timestamps) from the top 100 contributors of each, yielding information on 45,247 unique contributors. Then, we exclude the contributors who had not specified the location of their profiles, leaving us with 26,010 contributors.

Finally, to obtain our final training dataset, we validate the stated locations in profiles by comparing them to the median UTC offsets derived from the contributors' commit timestamps. This step is crucial for ensuring that the stated location is not stale, incorrect, or far-flung (such as a location mentioned in jest). We treat this location as the ground truth in our oracle, which is composed of 18,579 contributors on GitHub.

3.2 Model Selection

To identify the best machine learning model for Code Mapper's location inference component, we use a k-fold cross-validation technique and compare four machine learning classifiers: Logistic Regression (LR), K Nearest Neighbors (KNN), Decision Tree (DT) and Random Forest (RF). These classifiers possess various assumptions regarding the analyzed data and can deal with overfitting [4]. Furthermore, they have been commonly used in software engineering work [1–3].

Specifically, we employ a stratified 5 folds-cross-validation on the dataset curated in Section 3.1, meaning that we randomly partitioned the oracle into 5 equal folds that maintain a consistent distribution of regions. Next, we train the classifier on four folds and use the remaining fold to evaluate the classifier's performance.

This process is repeated five times for each classifier, and the average performance of these runs presents the overall for that classifier. The models are evaluated on four key metrics:

- **Precision**. Measure of the number of correctly predicted positive observations out of all predicted positives.
- **Recall**. Measure of the number of correctly predicted positive observations out of all the actual positives.
- **F1 Score**. The harmonic mean of precision and recall, providing a balance between the two metrics.
- **AUC-ROC**. Represents the Area Under the Receiver Operating Characteristic Curse. A higher AUC-ROC indicates a superior model performance.

The results of our evaluation are presented in Table 1 shows the performance metrics of the four selected models. The RF model consistently outperforms the other three models across all metrics. Specifically, the RF model achieved an AUC-ROC of 0.83, which is notably higher than the AUC values of the rest of the models. This indicates that the RF model has a better ability to distinguish between positive and negative classes, making its predictions more reliable. The F1 score of RF (0.64) was also the highest, which suggests that it provides the best performance in terms of precision and recall of predictions compared to the other models. This could be explained by the fact that RF combines multiple decision trees to generate its final output, allowing it to capture more complex patterns and relationships in the data that might be missed by simpler models like LR or DT. Furthermore, RF's inherent ability to protect against overfitting by averaging out the results of multiple decision trees make the predictions more stable and generalized, ensuring that the model doesn't get too specialized on the training data and does well on unseen data.

Table 1 presents the performance metrics of the four selected models. From the table, we find that the RF model consistenly outperforms all other models across all metrics. Specifically, the RF model achieved an AUC-ROC of 0.83, which is notably higher than the AUC values of the rest of the models. This indicates that the RF model has a better ability to distinguish between positive and negative classes, making its predictions more reliable. We manually inspect the results to gain insights into the reasons that led our approach to misclassify the regions. First, we find that our approach tends to misclassify countries sharing the same timezone. A case in point is the misclassification of a contributor's location as China when they were actually from Singapore. Given that China and Singapore are in the same timezone, and China's instances (10.13%) in our dataset outnumber Singapore's (0.58%), our model is biased towards China in this particular case. Second, we find instances where the absence of name-based probability data led to misclassifying the contributor's location.

3.3 Model Deployment

To put our approach into the hands of practitioners and researchers, we deploy Code Mapper on https://codemapper.alwaysdata.net/index.html. Code Mapper uses a Random Forest model to classify GitHub contributors by location. We trained this Random Forest on all GitHub contributors in our oracle (18,579 instances).

To avoid excessive API requests, we preemptively infer the location of the contributors of 2,046 repositories sourced from our dataset, sorted by the number of stars. Thus, when Code Mapper receives a user request, it first checks its database to see if the geographical contributions for the specified repository have already been recorded. If a contributor isn't in the database or if the stored data is over a month old, the results of the new request are saved. This method guarantees that the data provided to users is both current and precise.

Table 1: Performance of the classifiers on identifying the contributor locations.

	Precision	Recall	F1 Score	AUC-ROC
LR	0.40	0.45	0.39	0.67
RF	**0.62**	**0.67**	**0.64**	**0.83**
KNN	0.45	0.48	0.46	0.66
DT	0.59	0.59	0.59	0.66

4 CONCLUSION AND FUTURE WORK

In this paper, we present our tool, Code Mapper, that allows to visualize geographically the contributions of any open source project hosted on GitHub. We also propose a machine learning-based approach that uses the commit UTC offsets and the name-based country probabilities proposed in prior work [6] to infer probable locations of contributions.

Using Code Mapper can help practitioners in several ways. First, researchers can use Code Mapper's output to analyze geographic trends in open-source contributions, aiding in academic studies (e.g., studying the correlation between the geographic diversity of contributors and the innovativeness of projects), or informing decisions related to hiring strategies in the information technology (IT) market [6]. Moreover, our tool supports practitioners in the task of tracking and promoting diversity in open-source projects by shedding light on areas with underrepresented contributors. The ability to identify the location of contributions enhances software performance and security. By monitoring contributions on a regional basis, it becomes possible to promptly identify potential security threats or irregular activities and optimize infrastructure, including server locations, to enhance user experiences in those regions. Finally, identifying the geographical distribution of contributors equips software development companies with essential insights for making well-informed decisions regarding market expansion and product localization.

In the future, we plan to enhance Code Mapper by incorporating new features into our machine learning model, such as the top-level domain from email addresses and personal websites, to refine our location inferences. Another key focus will be to expand our training dataset, aiming to increase the model's accuracy and reduce biases, particularly in underrepresented regions. Furthermore, we plan to develop an API for Code Mapper, making it more accessible to researchers and practitioners for automated, large-scale analyses. In the same vein, we aim to introduce the capability to export results in various formats, like CSV files, to facilitate easier integration with other analysis tools and workflows.

REFERENCES

[1] Rabe Abdalkareem, Suhaib Mujahid, and Emad Shihab. 2020. A Machine Learning Approach to Improve the Detection of CI Skip Commits. *IEEE Transactions on Software Engineering (TSE)* (2020), To Appear.

[2] Ahmad Abdellatif, Mairieli Wessel, Igor Steinmacher, Marco A Gerosa, and Emad Shihab. 2022. BotHunter: an approach to detect software bots in GitHub. In *Proceedings of the 19th International Conference on Mining Software Repositories*. 6–17.

[3] Lingfeng Bao, Zhenchang Xing, Xin Xia, David Lo, and Shanping Li. 2017. Who will leave the company?: a large-scale industry study of developer turnover by mining monthly work report. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 170–181.

[4] Rich Caruana and Alexandru Niculescu-Mizil. 2006. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*. 161–168.

[5] Ozren Dabic, Emad Aghajani, and Gabriele Bavota. 2021. Sampling Projects in GitHub for MSR Studies. In *18th IEEE/ACM International Conference on Mining Software Repositories, MSR 2021*. IEEE, 560–564.

[6] Davide Rossi and Stefano Zacchiroli. 2022. Geographic diversity in public code contributions: an exploratory large-scale study over 50 years. In *Proceedings of the 19th International Conference on Mining Software Repositories*. 80–85.

[7] Kazuhiro Yamashita, Yasutaka Kamei, Shane McIntosh, Ahmed E Hassan, and Naoyasu Ubayashi. 2016. Magnet or sticky? Measuring project characteristics from the perspective of developer attraction and retention. *Journal of Information Processing* 24, 2 (2016), 339–348.

[8] Kazuhiro Yamashita, Shane McIntosh, Yasutaka Kamei, and Naoyasu Ubayashi. 2014. Magnet or sticky? an oss project-by-project typology. In *Proceedings of the 11th working conference on mining software repositories*. 344–347.