A Dataset and an Approach for Identity Resolution of 38 Million Author IDs extracted from 2B Git Commits

Tanner Fry  
Tapajit Dey  
tfry2@vols.utk.edu  
tdey2@vols.utk.edu  
The University of Tennessee  
Knoxville, TN, USA

Andrey Karnauch  
Audris Mockus  
akarnauc@vols.utk.edu  
audris@mockus.org  
The University of Tennessee  
Knoxville, TN, USA

ABSTRACT
The data collected from open source projects provide means to model large software ecosystems, but often suffer from data quality issues, specifically, multiple author identification strings in code commits might actually be associated with one developer. While many methods have been proposed for addressing this problem, they are either heuristics requiring manual tweaking, or require too much calculation time to do pairwise comparisons for 38M author IDs in, for example, the World of Code collection. In this paper, we propose a method that finds all author IDs belonging to a single developer in this entire dataset, and share the list of all author IDs that were found to have aliases. To do this, we first create blocks of potentially connected author IDs and then use a machine learning model to predict which of these potentially related IDs belong to the same developer. We processed around 38 million author IDs and found around 14.8 million IDs to have an alias, which belong to 5.4 million different developers, with the median number of aliases being 2 per developer. This dataset can be used to create more accurate models of developer behaviour at the entire OSS ecosystem level and can be used to provide a service to rapidly resolve new author IDs.

KEYWORDS
Identity Resolution, Git Commits, Heuristics, Machine Learning, Data Sharing

ACM Reference Format:
Tanner Fry, Tapajit Dey, Andrey Karnauch, and Audris Mockus. 2020. A Dataset and an Approach for Identity Resolution of 38 Million Author IDs extracted from 2B Git Commits. In 17th International Conference on Mining Software Repositories (MSR ’20), October 5–6, 2020, Seoul, Republic of Korea. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3379597.3387500

1 INTRODUCTION
The studies of open source software ecosystems often rely on investigating the digital traces left by the software developers, for the reconstruction and quantification of the behavior of an individual [1, 4], a team [17], or an organization [14, 16]. However, the data obtained directly by mining software repositories often suffer from quality issues. In this paper, we focus on the challenge of identifying authorship of code changes (commits) based on the information that can be obtained from the multitude of open source repositories, which is a deceptively hard problem to address. Software developers often use multiple email addresses, different variations of their names, or aliases while creating commits, and being able to identify the different IDs used by an author is crucial for improving the quality of the data obtained through mining different software repositories.

A number of ways to address this problem are available in the literature, but they are either based on very simple heuristics with little room for tweaking, e.g. from simple name matching with manual inspection [11], to very complex techniques that require full pairwise comparisons [2]. In our case of 38M author IDs obtained from the World of Code [15] dataset, the full pairwise comparison involves \(10^{14}\) computationally expensive comparisons, which is beyond the reach of the fastest super computers. In this paper, we are proposing an approach for author identity resolution that requires nothing more than the first name, last name, and email address of a commit author, but is trained with a Random Forest model for improved accuracy. We start by first dividing all the author IDs into blocks of authors IDs using a combination of heuristic methods to ensure a high probability of in-block linkage and a low probability of out-of-block linkage between the IDs. To make the computation possible, we assume that author IDs belonging to different blocks are not linked, and use pairwise comparison followed by transitive closure to identify the author IDs belonging to a block that are linked with each other.

We applied our approach on 38,362,013 author IDs collected from the World of Code data [15]. Using a combination of three heuristics, we found 15,177,184 author IDs in 5,508,119 blocks of two or more IDs per block, with the remaining author IDs being in blocks of size 1. Finally, after applying pairwise comparison, followed by transitive closure, we found that 14,861,538 IDs had an alias, which were found to be associated with 5,427,024 different developers. We tested the accuracy of our method by enlisting the help of 44 developers, who identified 207 different IDs that belonged to them. In terms of identifying pairs of author IDs linked together, our method achieved in a precision of 0.99 and a recall of 0.84 using this test data.

We created a dataset with all the author IDs that were found to have an alias (alternate ID), which is a compressed CSV file with ‘;’
as the separator. If an author was found to have 2 different IDs: \textit{I1, I2}, then it is recorded in the file in 2 separate lines, with the lines being \textit{11;I1 and 11;I2}, i.e. the first column is the group identifier, which is one of the IDs in a group, and the second column contains the different author IDs in separate lines.

However, sharing this dataset publicly has some ethical implications. Although the data is collected from public git commits to public repositories, we cannot put the list of emails in a plain text form as any email harvester may be able to obtain the data. To enable replication and the ability to use data in MSR studies, we release all data with restricted access, so that anyone who adequately certifies that they will only use the data in a research study and not for commercial purposes can get access to the data. The data with restricted access is available at: https://zenodo.org/record/3653283. We also provide completely publicly the SHA1 values of author ID which is one of the IDs in a group, and the second column contains the git repositories. WoC also provides relationships among them. For the purpose of this analysis, we only use a single list of author IDs extracted from almost 2B commits in WoC.

WoC data is(versioned, with the latest version labeled as Q, containing 7.2 billion blobs, 1.8 billion commits, 7.6 billion trees, 16 million tags, 116 million projects (distinct repositories), and 38 million distinct author IDs. WoC has collected that data during November and December of 2019. For more information please consult the WoC website.

We used all the author IDs extracted from WoC for this analysis, which were extracted from the commits, and are represented by a combination of the authors’ names and email addresses, in the following format: \textit{first-name last-name<email-address>}. E.g. for an author whose first name is “John”, last name is “Doe”, and email address is “john@me.com”, the corresponding author id in the WoC dataset would be: “John Doe <john@me.com>”.

3 METHODS OF LINKING AUTHOR IDENTITIES

Studies of online activities such as collaboration in software development made identity resolution techniques important in the field of empirical software engineering research [3, 11, 12]. They help disambiguate identities of people in a software projects and ecosystems. Data mined from version control systems can be used for many purposes: [18] builds social diversity dataset, others measure output [13] and match developer identity across Open Source Software [20], and/or link it to mailing lists [19]. It can also be particularly useful in the context of bot detection [9, 10] and predicting which pull requests will be merged [8]. Knowing the actual number of developers who contribute to or use a project can also influence the calculations of the number of defects [5, 7], or the popularity of the project [6].

The most accurate methods for identifying connected author IDs, like ALFAA [2], require pairwise comparison of all author IDs, which is infeasible to apply on millions of author IDs. Therefore, we subdivide the problem by first creating blocks of author IDs that are likely to be connected, and then apply the pairwise comparison method for identifying all author IDs that are actually linked. The workflow used to generate this data is shown in Figure 1.

As mentioned earlier, we started the process with all 38M different author IDs. The idea behind creating the blocks is to maximize the chance that author IDs belonging to the same group could actually belong to one developer, while author IDs belonging to different groups are unlikely to belong to the same developer. We constructed 3 different types of maps for creating the blocks:

- **Author Email to Author ID map**: We created a map by linking the author IDs that have the same email address, after checking if the email address is valid using several heuristics.

---

1https://bitbucket.org/swsc/overview/src/master/
These heuristics are essential because git commits do not enforce any strict email field policy, allowing users or their system to use any arbitrary string as their “email”. This results in millions of author IDs that contain very common and uninformative emails, such as John<john@example.com>. Including emails of this type would result in groups that do not actually represent one developer. Therefore, to avoid including such email addresses, we use frequency analysis to determine the top “junk” emails that exist in the author IDs dataset and remove them. We filter these emails further through other heuristics such as email length and regular expression matching (the details are in the provided source code). The map is then generated by sweeping through this cleaner set of author IDs and grouping all of the IDs that share the same exact email field. The mapping is presented in the form of email;author_ID1;author_ID2;... The idea is that author IDs that share the exact same valid email address are more likely to belong together.

- **Author Name to Author ID map:** We created another map by linking the author IDs based on author names, using a similar approach to the author emails. This approach looks only at the first and last names of the author ID. Similar to the email field, the name field has just as much flexibility, allowing users to assign whatever they choose as their first and last name. As a result, a set of heuristics is applied on the name field to filter out short and uninformative names. However, we are much more strict with these heuristics because names are much more difficult to tell apart than emails (e.g. “John Doe” vs. “Johnathan Doe”). For example, after filtering out certain names, we also grouped together similar names and only included them in the final mapping if their group size was at most 12 author IDs (discussed in-depth below). This ensured that valid names that passed the original filter, such as “John Smith”, were ultimately thrown out to avoid one developer being responsible for all author IDs containing a common name. Similar to the approach above, the idea of this mapping is that author IDs that share the exact same, uncommon name are more likely to belong together.

- **Author’s GitHub handle to Author ID map:** Although the WoC data has author IDs connected to all git repositories, a large portion of authors use GitHub. Therefore, we decided to construct a map between author IDs and the GitHub handles of the authors, where available, using information from GHTorrent.

Finally, we used transitive closure on the first and last maps, along with the map for the uncommon author names to create blocks of author IDs. We provide the script to generate the blocks for all the three map combinations, and the data with the author blocks we generated, with the author IDs and the frequency of the first and last names, and we leave it to the community to decide if it is appropriate to run the second step (name based mapping) of the algorithm on all names for identity resolution. At the end of the process, we were left with 5,508,119 blocks of size 2 or more, containing 15,177,184 author IDs in total, with the maximum block size being 42.

After creating these blocks, we used pairwise comparison to identify which author IDs are actually linked together. In this way, we reduced the number of required pairwise comparison from $38M \times 38M \approx 1.4 \times 10^{15}$ to maximum $5.5M \times 42 \times 42 \approx 9 \times 10^9$. Moreover, since these blocks are independent by assumption, we can run these tasks in parallel, further reducing the computation time.

2http://ghtorrent.org/
We created a dataset with all the 14,861,538 author IDs that were found to have an alias, which is a CSV file with ‘;’ as the separator. If an author was found to have 2 different IDs: I1, I2, then it is recorded in the file in 2 separate lines, with the lines being I1; I1 and I1; I2, i.e. the first column is the group identifier, which is one of the IDs in a group, and the second column contains the different author IDs in separate lines. The restricted access version and the publicly available versions of the data are available at the links shared in Section 1.

We also shared the source code we used for creating the dataset, including the scripts, the pre-trained Random Forest model we used, and a README file describing how to use the scripts, at the link shared in Section 1. These models can be loaded into R language and used in the provided workflow.

The last part of data we share is the blocking done using all three heuristics and in a gzipped semicolon separated text file containing block id, frequency of the first name (number of times it appears in the 38M WoC version Q author IDs), frequency of the last name, full name, email, and Author ID. The largest block contains 993 Author IDs. The restricted access and publicly available versions of the data are available at the links shared in Section 1.

4 FUTURE WORK
We plan to (or hope someone from the MSR community would) further refine our heuristics to create the author ID blocks, and potentially increase the block sizes to ensure no potential matches are in different blocks.

We hope to improve upon the models we have now in three ways. First, train them on the larger training sample. Second, add information about name frequencies (see blocking description above) as there were found to be important in prior work [2] and other so called “behavioural fingerprints”. Finally, we would like to address the issue of homonyms, i.e., Author IDs used by multiple developers by assigning authorship at the commit level.

5 LIMITATIONS
We utilize WoC data collection with all associated limitations of using that repository and described there [15].

Our primary assumption while applying this method is that author IDs in different blocks do not match, which is not always true, but we sacrifice a little accuracy for heavily improved performance by using this method.

The task of identity resolution solely based on name and email ID is very difficult for the problematic cases, as described earlier. Looking at the activity of those authors can be helpful for identity resolution for such cases.

The accuracy of our approach is not easy to establish, since there is no sizeable Golden dataset that we can use for reference. We tried to circumvent this problem by enlisting the help of 44 developers who identified 207 IDs belonging to them. Still, the size of this test sample is very small, and might not be representative of the whole population.

6 CONCLUSION
We expect this dataset to spur research that relies on accurate author identities by eliminating the painstaking manual verification that is not even possible for large collections. We also expect to update the data as the information from more projects is collected and as larger training datasets and more accurate blocking and matching models are established.

ACKNOWLEDGEMENT
The work has been partially supported by the following NSF awards: CNS-1925615, IIS-1633437, and IIS-1901102.
REFERENCES

[1] Sadika Amreen, Bogdan Bichescu, Randy Bradley, Tapajit Dey, Yuxing Ma, Audris Mockus, Sara Mousavi, and Russell Zaretzki. 2019. A Methodology for Measuring FLOSS Ecosystems. In Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability. Springer, Singapore, 1–29.

[2] Sadika Amreen, Audris Mockus, Russell Zaretzki, Christopher Bogart, and Yuxia Zhang. 2019. ALFAA: Active Learning Fingerprint based Anti-Aliasing for correcting developer identity errors in version control systems. Empirical Software Engineering (2019), 1–32.

[3] Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, and Anand Swamianathan. 2006. Mining Email Social Networks (MSR ’06). ACM, New York, NY, USA, 137–143. https://doi.org/10.1145/1137983.1138016

[4] Tapajit Dey, Yuxing Ma, and Audris Mockus. 2019. Patterns of effort contribution and demand and user classification based on participation patterns in npm ecosystem. In Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering. ACM, 36–45.

[5] Tapajit Dey and Audris Mockus. 2018. Are software dependency supply chain metrics useful in predicting change of popularity of npm packages? In Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering. ACM, 66–69.

[6] Tapajit Dey and Audris Mockus. 2018. Modeling Relationship between Post-Release Faults and Usage in Mobile Software. In Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering. ACM, 56–65.

[7] Tapajit Dey and Audris Mockus. 2020. Deriving a usage-independent software quality metric. Empirical Software Engineering (2020), 1–46.

[8] Tapajit Dey and Audris Mockus. 2020. Which Pull Requests Get Accepted and Why? A study of popular NPM Packages. arXiv preprint arXiv:2003.01152 (2020).

[9] Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. 2020. Detecting and Characterizing Bots that Commit Code. arXiv preprint arXiv:2003.01772 (2020).

[10] Tapajit Dey, Bogdan Vasilescu, and Audris Mockus. 2020. An Exploratory Study of Bot Commits. arXiv preprint arXiv:2003.07961 (2020).

[11] Daniel German and Audris Mockus. 2003. Automating the measurement of open source projects. In Proceedings of the 3rd workshop on open source software engineering. University College Cork Cork Ireland, 63–67.

[12] Daniel M German. 2004. Mining CVS repositories, the softChange experience. In 1st international workshop on mining software repositories. Citeseer, 17–21.

[13] Mohammad Charehyazie, Daryl Posnett, Bogdan Vasilescu, and Vladimir Filkov. 2015. Developer initiation and social interactions in OSS: A case study of the Apache Software Foundation. Empirical Software Engineering 20, 5 (01 Oct 2015), 1318–1353. https://doi.org/10.1007/s10664-014-9332-x

[14] Randy L. Hackathar, Audris Mockus, John Douglas Palfiraman, and David M. Weiss. 2010. Assessing the state of software in a large enterprise. Empirical Software Engineering 15, 3 (01 Jun 2010), 219–249. https://doi.org/10.1007/s10664-009-9120-1

[15] Yuxing Ma, Chris Bogart, Sadika Amreen, Russell Zaretzki, and Audris Mockus. 2019. World of Code: An Infrastructure for Mining the Universe of Open Source VCS Data. In IEEE Working Conference on Mining Software Repositories. papers/WoC.pdf

[16] Audris Mockus. 2010. Organizational Volatility and Its Effects on Software Defects (FSE ’10). Association for Computing Machinery, New York, NY, USA, 117–126. https://doi.org/10.1145/1882291.1882311

[17] Audris Mockus, Roy T. Fielding, and James Herbsleb. 2000. A Case Study of Open Source Software Development: The Apache Server (ICSE ’00). Association for Computing Machinery, New York, NY, USA, 263–272. https://doi.org/10.1145/337180.337209

[18] Bogdan Vasilescu, Alexander Serebrenik, and Vladimir Filkov. 2015. A Data Set for Social Diversity Studies of GitHub Teams. In Proceedings of the 12th Working Conference on Mining Software Repositories. ACM, 514–517. https://dl.acm.org/citation.cfm?id=2820601

[19] I. S. Wiese, J. T. d. Silva, I. Steinmacher, C. Treude, and M. A. Gerosa. 2016. Who is Who in the Mailing List? Comparing Six Disambiguation Heuristics to Identify Multiple Addresses of a Participant. In 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME). 345–355. https://doi.org/10.1109/ICSM.2016.13

[20] Yunxiang Xiong, Zhangyuan Meng, Beijun Shen, and Wei Yin. 2017. Mining Developer Behavior Across GitHub and StackOverflow. In The 29th International Conference on Software Engineering and Knowledge Engineering. 578–583. https://doi.org/10.18293/SEKE2017-062