Poster: Git Blame Who?: Stylistic Authorship Attribution of Small, Incomplete Source Code Fragments

Edwin Dauber
Drexel University

Aylin Caliskan
Princeton University

Richard Harang
Sophos Data Science Team

Rachel Greenstadt
Drexel University

ABSTRACT

Program authorship attribution has implications for the privacy of programmers who wish to contribute code anonymously. While previous work has shown that complete files that are individually authored can be attributed, these efforts have focused on ideal data sets such as the Google Code Jam data. We explore the problem of attribution “in the wild,” examining source code obtained from open source version control systems, and investigate if and how such contributions can be attributed to their authors, either individually or on a per-account basis. In this work we show that accounts belonging to open source contributors containing short, incomplete, and typically uncompilable fragments can be effectively attributed.

CCS CONCEPTS

• **Security and privacy → Pseudonymity, anonymity, and untraceability**;

KEYWORDS

Source code authorship attribution, stylometry, machine learning

1 INTRODUCTION

The attribution of source code, be it in the form of entire single-authored files or small fragments from collaborative files either individually or linked to an account, has numerous potential applications, from employer-mandated non-compete clauses to the (future) attribution of the authors of malicious code. While the problem of attributing source code samples written by a single author has been examined in some depth using Google Code Jam[1] datasets, extending these techniques to more complex problems – such as deanonymizing the account of an open source contributor – is much harder, and has received less attention [2].

Collaboratively written code (or any code which has passed through the hands of multiple programmers) introduces several novel complications to the problem of attribution not present in previous work focused on single files. Files may have a primary author who has contributed most of the code, with other authors making relatively inconsequential additions, or a block of code may be written by one author and later modified by another. Individual author contributions may be scattered across multiple files and consist of fragments as small as a single line of code. Using real world source code collected from the online collaborative platform GitHub[2] we show that these problems can be overcome, particularly when multiple modifications can be linked to a single unknown identity (for example, a single pseudonymous version control system account).

2 METHODOLOGY

We collected C++ repositories on GitHub, and then split collaborative files from those repositories into single authored pieces using git blame. We have a set of 106 programmers with 150 samples each. For each sample, we extract the AST and use that to extract a feature vector as in [3]. The AST is a tree representation of code with nodes representing syntactic constructs in the code, and we used the fuzzy parser joern to extract them, allowing us to extract ASTs even for incomplete and uncompilable code fragments [4]. We have a total of 369,097 non-constant features, of which an average of 365,690 are zero-valued for any given sample (so an average of 3,407 features are non-zero valued). These features include both raw and TFIDF values of AST node types, bigrams, node depths, and bigram depths as well as word unigrams, C++ keywords, API symbols, and maximum and average AST depth. TFIDF is a measure combining frequency with how many authors use the feature.

We then proceed to perform attribution of each sample using a random forest with 500 trees. Random forests are ensemble classifiers made of decision trees which vote to assign a class to an instance [1]. When we have linked samples, we then average the output probability distributions across the linked samples, and take as the predicted author for the whole set the one with the highest averaged probability. We also create a *calibration curve* for our classifier, binning the samples based on the highest classifier probability in increments of 10%, and report the accuracy for samples in each interval. We use this curve both when our samples are not linked and to address the open world problem. For our open world experiments, we have four rounds each with a set of 25 programmers as our suspect set S, and the remaining 81 programmers as the set of unknown authors U. We perform 10-fold cross-validation on S, adding all samples from U to the evaluation set of each fold.

3 RESULTS

Figure 1 shows the calibration curve constructed from attempting to attribute single samples. The overall accuracy was 73%. While

[1]Google Code Jam http://code.google.com/codejam

[2]GitHub repository hosting service http://www.github.com

ICSE ’18 Companion, May 27-June 3, 2018, Gothenburg, Sweden
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195007

most samples fell into the lower confidence levels, we show that even mid-confidence attributions can be trusted to a high degree.

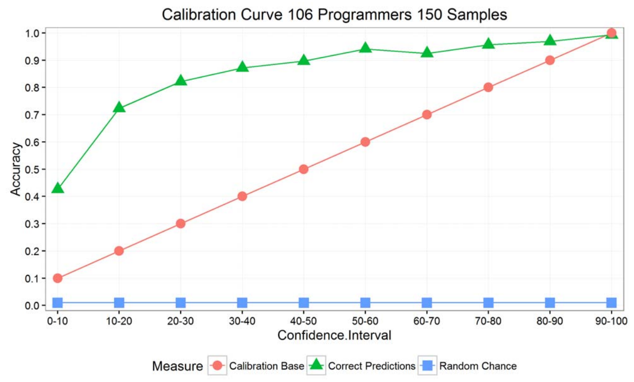

Figure 1: This calibration curve compares bin accuracy based on confidence level to random chance for single samples.

Figure 2 shows the results for linked samples. We achieve 70% accuracy for single samples, 95% accuracy for pairs of samples, and 99% accuracy for sets of 15 samples. We note that in many cases, a single collaborative code file will contain at least two samples for most programmers, and 15 samples is not unreasonable for larger projects or for programmers who contribute to multiple projects.

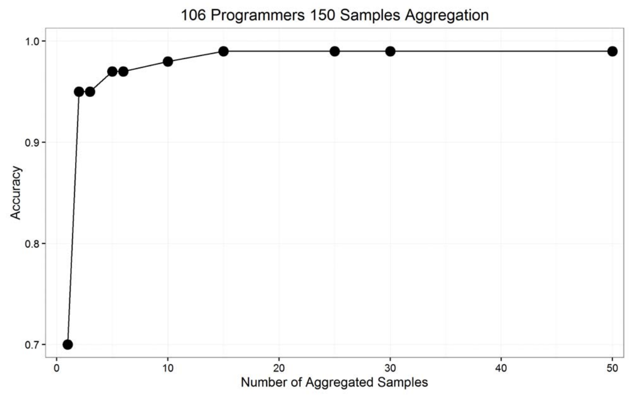

Figure 2: These are the results for aggregating classifications. We range from individual samples to sets of 50 samples.

Figure 3 shows a receiver operating characteristics (ROC) curve for identification of false attributions using a threshold, from our open world experiments. ROC curves plot true positive rate on the y-axis and false positive rate on the x-axis. This can help evaluate acceptable trade-offs and assist in choosing thresholds. For this curve true positives are correctly identified false attributions, while false positives are correct attributions labeled false.

We note that while there are cases in which our method may mistake out of world samples as belonging to one of the suspects, this is rare. As we increase the threshold we start with a few large cuts to the percentage of out of world samples above the threshold, correctly identifying over 90% of such samples in only a few increments for even single sample attribution and continuing to identify about 97% and then over 99% as we continue to increment.

Our open world experiments use a weaker version of the classifier, with fewer trees of limited depth in the random forest, and with 97% of evaluation samples by programmers outside of the suspect set. We expect a stronger version of the classifier would reach the levels of success found in our experiments at lower thresholds.

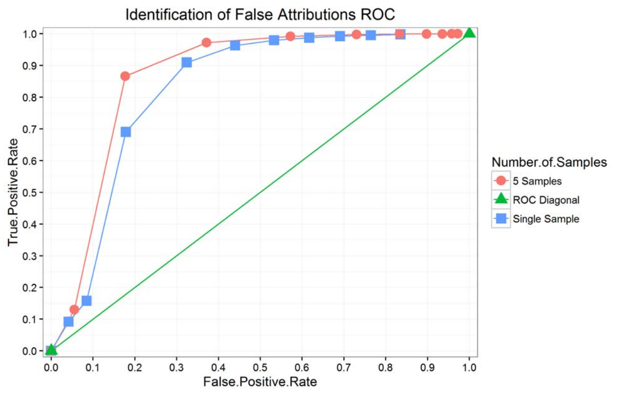

Figure 3: This ROC curve compares false and true positive rates for identifying false attributions through thresholding.

4 CONCLUSION

We show that it is possible to identify the author of even small, single code contributions and easy to attribute collections of such contributions, such as those belonging to an account. We further show that calibration curves can be used to perform attribution in open world cases and in cases with lower overall accuracy.

In light of our results, we recommend avoiding open source contributions if attribution posses a risk. If contributions must be made, single line patches are preferable to blocks of code. We caution that a single pseudonymous account is not sufficient as having more than a single line of contribution creates the possibility of linking samples to attribute the account, as even two linked samples can make attribution far easier.

ACKNOWLEDGMENT

We thank DARPA Contract No. FA8750-17-C-0142 and the United States Army Research Laboratory Contract No. W911NF-15-2-0055 for funding this work, and Dr. Frederica Nelson, Dr. Michael Weisman, and Gregory Shearer for their roles in continuing it.

REFERENCES

[1] Leo Breiman. 2001. Random Forests. *Machine Learning* (2001).
[2] Steven Burrows. 2010. *Source code authorship attribution.* Ph.D. Dissertation. RMIT University.
[3] Aylin Caliskan-Islam, Richard Harang, Andrew Liu, Arvind Narayanan, Clare Voss, Fabian Yamaguchi, and Rachel Greenstadt. 2015. De-anonymizing programmers via code stylometry. In *24th USENIX Security Symposium (USENIX Security 15).* 255–270.
[4] Fabian Yamaguchi, Nico Golde, Daniel Arp, and Konrad Rieck. 2014. Modeling and Discovering Vulnerabilities with Code Property Graphs. In *Proc. of IEEE Symposium on Security and Privacy (S&P).*