# Poster: Source Code Authorship Attribution

Aylin Caliskan Islam, PhD Student – Advisor: Dr. Rachel Greenstadt
Drexel University, Department of Computer Science, Philadelphia, PA 19104
Webpage: https://www.cs.drexel.edu/ ˜ac993 E-mail: ac993@drexel.edu

## I. Problem and Motivation

As information becomes widely available and easily accessible through the Internet and other sources, the trend of plagiarism has been increasing. Plagiarism and copyright infringement are issues that come up in both academic and corporate environments. We need author classification techniques to inhibit such unethical violations. Source code is also intellectual property and reflects individual style. It is important to be able to identify the author of source code. Building a tool to detect the author of a program in an automated way aids in resolving copyleft, copyright and plagiarism issues in the programming fields. Making authorship attribution tools available to the public will also raise consciousness and decrease any possible tendency to plagiarize.

Code stylometry methods can also attribute authors to malware or malicious code. De-anonymization techniques could be used to identify where the malware originates and link collaborating authors of malicious code. Obtaining this information will enable us to analyze the interaction graphs of malware authors. We could infer which author holds certain tools and how malware spreads and evolves.

In this project, I perform authorship attribution on source code that is written in C. I investigate how well we can classify authors within and across projects. How much source code, which machine learning classifier and which features are required for an accurate analysis of this specific problem? Answering these questions could provide proof of authorship in court or automate the process of finding a cyber criminal from the source code left in an infected system.

## II. Background and Related Work

Stylometry is a field that relies on linguistic information found in a document to perform authorship recognition. Authorship attribution is the problem of determining a text's author, which could be accomplished using stylometric analysis. A classifier is trained with the necessary features to detect the authors' footprints to attribute an author to anonymous text. Source code authorship attribution could be called code stylometry and performed in a similar manner.

There has been a great amount of work done on authorship attribution of unstructured or semi-structured text. In this research, we are interested in structured text, source code in particular. Burrows and Tahaghoghi [1] classified source code authors by looking at n-grams. Burrows et al. [2] also investigated C code of 1,597 student programming assignments. Their most successful approach reached 76.78% classification accuracy in a 10 class problem, which we outperform in this work. Rosenblum et al. [3] present a novel program representation and techniques that automatically detect the stylistic features of binary code. Frantzeskou et al. [4] investigated the high-level features that contribute to source code authorship attribution in Java and Common Lisp. They determined the importance of each feature by iteratively excluding one of the features from the feature set. They showed that comments, layout features and naming patterns have a strong influence on the author classification accuracy. There is also a great deal of research on plagiarism detection which is carried out by identifying the similarities between different programs. For example, there is a widely used tool called Moss that originated from Stanford University for detecting software plagiarism. Moss [5] is able to analyze the similarities of code written by different authors.

## III. Approach

The experiments were automated by incorporating JStylo [6] which is an authorship recognition analysis tool. JStylo is free software open to improvements and it is available in git. It allows for loading a training set of documents with known authors to test a set of documents with unknown authors extracting selected features of interest. JStylo uses a variety of WEKA [7] classifiers to analyze the problem set, which enables us to find out the optimum classifier for source code authorship attribution. JStylo allows for the option of performing authorship attribution using a set of features as opposed to a single feature in feature extraction and analysis stages. These properties make JStylo a crucial tool for automating code stylometry experiments. We analyze the different combinations of classifiers and lexical, syntactic, and character features.

I cloned eleven git repositories in C on my local machine and I formed three different problem sets. Forsyth and Holmes [8] show that a minimum of 250 words are required to attribute a document to an anonymous author. After combining all the source code of a particular author in a text file, I re-chunk the document into separate 500-word text files. In the first dataset, I extracted the list of contributors and put them in descending order by the number of commits they made in each project. From each project, I took one author who had the highest number of commits. I saved the contents of the commits of each author in a file, including the comments they committed. In this work, I am using a public dataset and I do not need to exclude the comments, which potentially contain personally identifiable information. Including the comments minimizes the preprocessing efforts on such structured data. I excluded the formerly written and then removed source code, removed comments, as well as commit messages. The purpose of this data selection was forming the first problem set to perform authorship attribution on authors from different projects. In this case, the content of the source code should be quite different. The second problem set included all the C files from all these projects. The C files were combined and then re-chunked to 500-word documents as well. I wanted to see

how accurately we can attribute a project to its original git repository. The third dataset was formed to perform source code authorship attribution within one project. I listed the number of contributors in each project. The github project libgit2 had the highest number of contributor accounts, which is 152. Some authors had more than one git account. I extracted the commits of 11 unique contributors that had the highest number of commits in libgit2. I limited the number of classes to 11 in all the problem sets so that I can compare the accuracy of the three datasets. The third dataset poses the most difficult problem. I am trying to differentiate authors within one project where the source code content is similar.

Previous stylometric research shows that authorship attribution requires at least 5,000 words that belong to an author in order to perform accurate authorship attribution. I use 10 to 40 of the re-chunked 500-word text files for each author to see how much source code is required for the highest accuracy.

The classification step consists of 10-fold cross-validation. The results of classification are evaluated by means of accuracy which is the true-positive rate.

## IV. Results and Contributions:

| Dataset | Accuracy |
|---|---|
| 11 authors across projects | 85.91% |
| 11 projects | 90.91% |
| 11 authors within projects | 70.31% |

TABLE I. Source code authorship attribution in 3 datasets

My first contribution is showing that using a support vector machine with sequential minimal optimization leads to the highest accuracy in code stylometry. My second contribution is showing that code stylometry requires 15,000 words per author for accurate analysis which can be observed in Figure-1. There is no increase in classification accuracy if we use more than 15,000 words (the red toned bars in Figure-1) per author. This is an interesting finding when compared to the 5,000 words that are required per author for regular authorship attribution.

We train the classifier with a great variety of features and show that a language independent feature set performs almost as well as using the Writeprints_Limited feature set which is the gold standard for English stylometry. The 'Language Independent Feature Set' consists of the features: characters, punctuation, special characters, top character bigrams and trigrams, words and word lengths. The fundamental comparisons are done by using this feature set. The 'words' feature is used to represent reserved words in the C programming language. Writeprints is the most effective feature set in English stylometry and it was formed by Abbasi and Chen [9]. The Writeprints_Limited feature set was designed by Zheng et al. [10]. Using Writeprints_Limited led to better accuracy compared to using language independent features. This is not surprising since the English is the common language for coding and commenting in these projects. I extended the Writeprints_Limited feature set with a variety of n-grams and keywords in C to form the 'Source Code Feature Set' and to obtain the highest accuracy. Table IV reflects the correct classification rate when the classifier was trained with the 'Source Code Feature Set'. Incorporating these features led to the highest accuracy in all three of the datasets, compared to other feature combinations that were used. Considering the fact that the chance of randomly attributing the 11 classes correctly is 9.09%, the accuracy we see in Table IV is quite plausible.

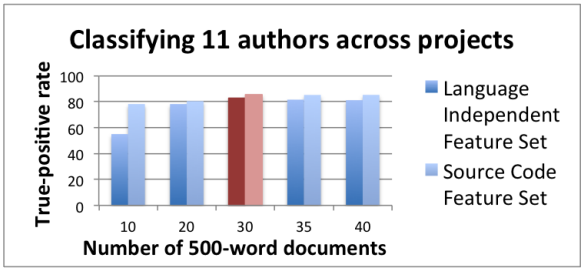


Fig. 1. Number of 500-word documents required for code stylometry

I also show that source code authorship attribution across projects is an easier problem than source code authorship attribution within a project. The content across projects varies greatly and the representative features of authors in different projects become more distinct. The more diverse the feature values, the easier it becomes to classify authors. Identifying authors within a project is a more difficult task since the content among the authors is similar. This makes the extracted feature values from each author's source code less distinct and renders classification more challenging. As a result, there is a 15% decrease in classification accuracy within a project. In future work, we would like to study source code from varying programming languages and binaries, with a focus on malware of known authorship.

## References


[1] S. Burrows and S. Tahaghoghi, "Source code authorship attribution using n-grams," in *Proceedings of the Twelth Australasian Document Computing Symposium, Melbourne, Australia, RMIT University*, 2007, pp. 32–39.

[2] S. Burrows, R. L. Uitdenbogerd, and A. Turpin, "Application of information retrieval techniques for source code authorship attribution."

[3] N. Rosenblum, X. Zhu, and B. Miller, "Who wrote this code? identifying the authors of program binaries," *Computer Security–ESORICS 2011*, pp. 172–189, 2011.

[4] G. Frantzeskou, S. MacDonell, E. Stamatatos, and S. Gritzalis, "Examining the significance of high-level programming features in source code author classification," *Journal of Systems and Software*, vol. 81, no. 3, pp. 447–460, 2008.

[5] A. Aiken *et al.*, "Moss: A system for detecting software plagiarism," *University of California–Berkeley. See www. cs. berkeley. edu/aiken/moss. html*, vol. 9, 2005.

[6] A. McDonald, S. Afroz, A. Caliskan, A. Stolerman, and R. Greenstadt, "Use fewer instances of the letter "i": Toward writing style anonymization," in *Privacy Enhancing Technologies*. Springer, 2012, pp. 299–318.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[8] R. S. Forsyth and D. I. Holmes, "Feature-finding for test classification," *Literary and Linguistic Computing*, vol. 11, no. 4, pp. 163–174, 1996.

[9] A. Abbasi and H. Chen, "Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace," *ACM Transactions on Information Systems*, vol. 26, no. 2, p. 7, 2008.

[10] R. Zheng, J. Li, H. Chen, and Z. Huang, "A framework for authorship identification of online messages: Writing-style features and classification techniques," *Journal of the American Society for Information Science and Technology*, vol. 57, no. 3, pp. 378–393, 2005.