ISSN:0975-9646

Author Identification: An Approach Based on Style Feature Metrics of Software Source Codes

Rohit R. Joshi, Rajesh V. Argiddi

Computer Science Department,
Walchand Institute of Technology,
Solapur, India.

***Abstract*— Today the field of software is facing the problem of software theft on a major scale. Also it has many more challenges in front such as plagiarism detection, copyright disputes, piracies and legal allegations. In most of the cases, lack of strong evidences is the only issue to resolve the problems. So in such cases how to resolve the problem is the question? The field called as author identification helps us to resolve these kinds of problems. The field leads in the right direction to find the closest author of the source code under the above discussed problems. This field has many different methods to find the likely author. But, the one used in this paper is stylometry. The stylometry means the study of linguistic styles of the author of the code. So, the method involves capturing the writing styles of code of the known authors and based on these precaptured styles the codes to be tested are classified to their respected authors. The study involves 9 different metrics for a software source code. The sample source codes are obtained from different sources such as educational assignments of students, code snippets of study books and open source projects of some developers. The study is carried out on 1500 sample source codes providing 102621 lines of codes to be scanned.**

***Keywords*— Software Source Code Metrics, Stylometry, Style metrics, Author Identification**

I. Introduction

As discussed above the field of software is facing lots of problems such as plagiarism, copyright disputes and legal allegations etc. Author identification is the technique that helps in resolving these problems. It's the technique that deals with the identification of the likely author of the source code. Though the technology is same the methodology differs. This paper concentrates on the stylistic approach for doing author identification. Style is that part of human life which is developed over the years and some part of it persist throughout the life. If we study those styles carefully then it is possible to derive some fixed patterns of an individual. The best example of it is our handwriting. Once we learnt the shapes of the letters in childhood it persists throughout the life and becomes our style of handwriting and is unique in some respects. This means that every person has its own style of handwriting. Also, based on the handwriting the signatures of every other person are different. This will be useful when it comes to the verification of legal documents to avoid frauds. The work has been previously done in case of articles, novels, essays etc. Also, some work has been done in case of e-mails, blogs etc. to identify their authors. Arvind Narayanan et al. [1] presented some stylometric features used for author identification of textual material. It include several features such as length, vocabulary richness etc. Also they have considered some features in terms of their frequencies such as word shape, word length, letters, digits etc. The features such as punctuation, special characters, function words and syntactic categories are also considered. So from the above discussion the question arises in the mind whether this can be possible in case of software source code? Yes, this can be possible in case of software source code. The writers of the code also have some specific style of theirs and the only restriction for them is to write the code by following the standard grammar of the language. Here, the author identification of the software source code can be done on the basis of the writing styles of the authors of the source code. Each style is called as metric as it is the kind of the measurement of the piece of a software source code. The basic procedure of author identification using styles of the authors involves the following steps:

1. Capture different styles of the known authors
2. Save these captured styles for future use
3. Take the codes to be tested of unknown authors and capture their styles
4. Based on the saved precaptured styles of the authors, predict the likely authors for the codes to be tested.

This is what discussed the basic of the field author identification from the viewpoint of the stylistic approach. So, which are different style metrics to be captured from the code? What is the methodology of the author identification? And more details about it are discussed in further sections.

II. Related Work

In the above section we have seen basics of author identification, stylometry etc. This section discusses about the work carried out by different people in this field. Lots of work has been done in this field in finding the metrics, doing their extraction and following it by the author identification. Jay Kothari et al in [2] discussed the method of probabilistic approach for doing the authorship identification. For that they have used style based metrics as well as Text distribution metrics.

Style based metrics of theirs include following:

1. distribution of line size
2. leading spaces
3. underscores per line
4. Semicolons, commas and tokens per line.

Here the main consideration of there is the probability of the metric. Suppose if 'x' is a metric under consideration, then metric 'x' will be classified to class/author 'i'. They have worked out this concept by taking two terminologies as below:

a. Individual Consistency: measure of consistency of author to use the particular metric

b. Population Consistency: measure of consistency of the metric used by number of authors. Based on these two values the selection criterion is built and the classification tools such as bays and VFI can be used to classify the author of the unknown code.

Maxim Shevertalov et al in [3] also discussed the method of discretization of source code metrics for author identification. They have discussed the metrics such as follows:

1. Leading Spaces: it's the measurement of the white spaces at the beginning of the each line
2. Leading Tabs: it's the measurement of the tab characters at the beginning of the each line
3. Line Length: it's the measurement of length of each line of the code
4. Line Words: it's the measurement of words in a single line of code

Their process of author identification involves extraction of above metrics from the inputted source codes of known authors. Then form the author profiles of known authors with the help of discretized metrics. Based on these author profiles classify the unknown source codes to predict the likely author.

MacDonell S.G. et al [4] have discussed three types of metrics such as style metrics, structure metrics and layout metrics. From those metrics some of the styles metrics are:

1. Capital and Small letters metric: Captures writing style of the author for capital and small letters, say for variables or method names
2. Lines of Code (LOC): Captures total no. of lines of code
3. Words per Line: Captures no. of words on per line basis

Rohit R. Joshi et al in [5] discussed various metrics such as below:

1. leading spaces, leading tabs, trailing spaces, trailing tabs,
2. line length, lines of code,
3. brace positions, average indentations,
4. No. of methods etc.

Also, they have introduced the concept of Boolean metrics where they have considered only the presence or absence of some metrics such as conditional operators, naïve variable names, i-as-iterator, method chaining, try statements etc. The proposed approach of author identification of theirs involves the methodology of decision trees. They extract the above metrics from the source codes of the known authors as well as from source codes to be tested. Then, preparation of classification model has been done based on results of extraction of known authors. Based on that classification model the codes of unknown authors are classified accordingly. R. A. Vivanco and N. J. Pizzi in[6] discussed the use of genetic algorithm for identification of effective metrics. They have discussed following metrics:

1. Lines of code
2. No. of lines containing comments
3. No. of lines containing white spaces
4. Ratio of no. of comment lines to total no. of lines of code.etc

Frantzeskou G et al [7] discussed three different approaches such as Neural Network, Discriminant Analysis, Case Based Reasoning for doing source code authorship analysis. The table (see Table I) below shows the comparative study of the existing techniques. Jay Kothari et al in [2] have taken the sample source codes from open source projects. They have got 61% of the correct classification of unidentified samples using Bays classifier, while got 76% of correct classification using VFI classifier.

TABLE I
COMPARATIVE STUDY OF EXISTING TECHNIQUES

Approach	**Technology**	**No. of authors**	**Result**
Probabilistic approach[2]	Bays/VFI	12	61%/76% [Bays/VFI]
Discretization approach[3]	GA	20	54.3%/75% [Files/Projects]
GA[6]	LDA	-	62.7%

Maxim Shevertalov et al in [3] have also taken the sample source codes from open source projects. They considered in total 60 projects of over 20 developers i .e. 3 projects for each developer. 2 out of 3 projects are taken for training purpose while 1 project is left for testing purpose of each developer. They have got 54.3% of the correct classification in case of files while 75% of the correct classification in case of projects. R. A. Vivanco and N. J. Pizzi in[6] have used the GA approach for identifying effective metrics. They have used LDA algorithm to get 62.7% as the classification result for 338 code samples.

III. METHODOLOGY

This is the section where it clears the questions like what are the metrics to be extracted from source code? How they are extracted? How the actual author identification of the likely author can be done? First we will start with what is the meaning of the term metrics?

Metrics: It is nothing but measurement of something taken at a time. Here, we are going to take the measurements of a piece of software source code at a given time. Hence, they are called as software metrics. In the previous section of related work many metrics from the past literature of this field have been discussed. The main goal of this paper is to workout with the style metrics and hence the style based metrics considered are as follows:

A. *Style Based Metrics:*

1) *i-as-iterator:* Almost all the authors of the program have the habit of taking the 'i' as there iterator variable. Some people to make the differentiation intentionally avoid this. So, taking the count of 'i' from the program is not useful as it finds in most of the cases. So, here the consideration is that only the presence or absence of this variable is checked and the differentiation is made between the two authors, one who used this and the other who are not.

2) *Line-Length*: It deals with no. of characters in the line. Many authors give most of their preference to better representation and clear view of their code for better understanding. Hence they have the tendency to

restrict the line-length up to some specific limit by writing the single statement in to multiple lines by breaking it. So, this peculiarity can be captured to distinguish the two authors.

3) *Comments:* Every author has its own style of commenting. Some may use the single liners only, while others may use the multiple line comments. Many of the times the mixture of both can be seen resulting in some new pattern. This attribute captures these commenting styles of the author.
4) *Average Procedure Length:* Again this metric captures the average procedure length for each author. It involves counting average no. of lines per method. The value of this metric may vary from author to author and again depends on the level of expertise. This metric is calculated on per class basis for each author.
5) *Methods*: This is also a useful feature to be captured from a code. While writing the code the author many times write it as a whole. i.e. doing all the operations of the program in to the main part of their code. This will create the problem when some error occurs, because tracing the error in an entire code will be problematic. Also, when such code is given to the other developer working on different module it will be hectic job for him to understand the function of it. For avoiding these problems many authors use to write the code by implementing methods. This metric count for no. of methods from a source code.
6) *No. of Arrays*: Storing style is also one of the styles of the author. Suppose an author has to store 10 values. The option in front of him/her is either to store it in 10 different variables or in arrays. This metric deal with no. of arrays on per author basis.
7) *Object Creations:* Every author has to create the objects of a class for accessing its contents. But the differentiation can be made between different styles of object creations such as object creation with its declaration and the constructor, direct creation using new keyword without its declaration.
8) *Single Literal Variables*: Many of the authors have the habit of declaring the single literal variables. Though this can be the unprofessional names given to the variables, but often found in material such as student assignments.
9) *Double Literal Variables:* This metric covers the double literal variable count such as int aa, int bb etc. Most of the authors have the habit of declaring these kinds of variables.

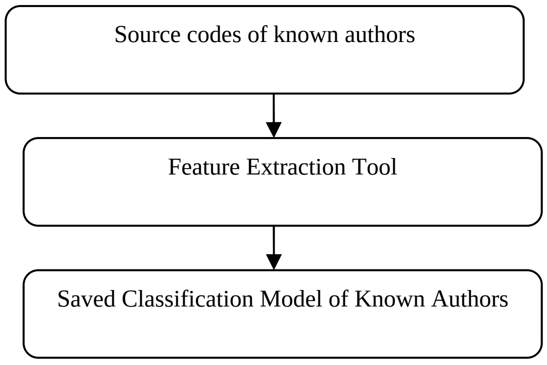
Source codes of known authors
Feature Extraction Tool
Saved Classification Model of Known Authors

Figure 1 Process of building classification model

B. Process of Author Identification

This section discusses about the actual process of author identification. The method involves 2 phases:

1) Building of classification model of known authors.
2) Doing prediction on the basis of above saved classification model.

1) Building Classification Model of Known Authors:

Figure 1 shows the block diagram of the process of building classification model.

a. *Source Code of The Known Authors*: Here the process involves taking the source codes of the known authors and give it to the metric extraction tool. Actually the type of input we are giving here is an arff file which consists of the file names and the name of the known authors respectively. The arff file is giving as input because a tool called as WEKA is used for the classification ahead. These source codes of known authors are then given to feature extraction tool in the next step.
b. *Feature Extraction Tool*: This tool functions similarly as the filtering tool. It accepts the source codes of known authors and extracts many different features discussed above. This tool uses the japa 1.5 parser which works on AST technique to extract those features. The japa has different visit () methods which can be overridden to get the values of these features and these values are then forwarded to the classification tool to build classification model of known authors. The combination of steps (a) and (b) are altogether called as preprocessing.
c. *Classification Model Of Known Authors*: As we have seen the different features have been extracted by using feature extraction tool, the values of theirs are then forwarded to the classification tool to build the classification model. In the classification tool we have used the decision tree based classifier with C4.5 algorithm. WEKA has provided J48 as its implementation. It's a pruning tree algorithm meaning that the unnecessary branches are cut down by replacing them with their leaf nodes giving the same equivalent results. This classification model has been saved for the future use of identification of likely author.

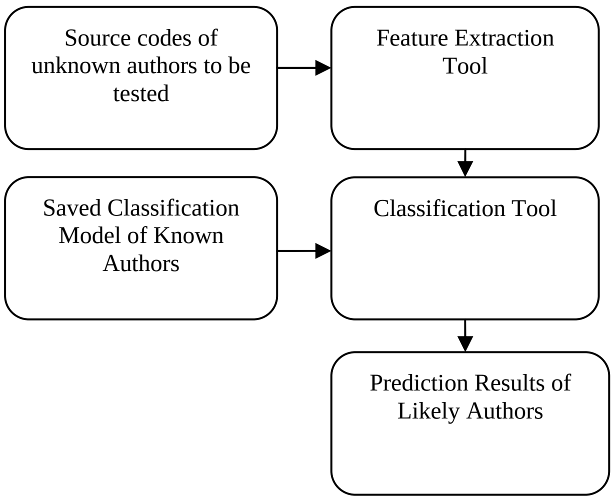
Source codes of unknown authors to be tested
Feature Extraction Tool
Saved Classification Model of Known Authors
Classification Tool
Prediction Results of Likely Authors

Figure 2 Process of Predicting Likely Authors

2) Doing Prediction on the Basis of Above Saved Classification Model:

Figure 2 shows the second phase of our methodology showing the actual process of predicting the likely author.

a. *Source codes of unknown authors*:
These are the source code which are of unknown authors or rather are the ones which are going to be tested for their likely authorship. The input format of this provided source codes are again an arff file with instances having the values as author name and file name. This input is again given to the next phase of feature extraction for further process.

b. *Feature Extraction Tool*:
This unit in the second phase functions the same as in the first one. It extracts the same metrics that are extracted for the source code of the known authors. It also uses japa 1.5 parser working on AST technology to extract those metrics. The visit () method is overridden to extract those metrics of japa parser. It then sets the values of these metrics and sends those values to the classifier.

c. *Classification Tool*: This is very important unit in this entire process. It accepts the values of extracted metrics from feature extraction tool of unknown codes. Then by using J48 algorithm and the saved classification model of known authors from the first phase, the classifier classifies these unknown codes to their respected likely authors. The tool called as WEKA is used for doing all these activities.

d. *Prediction Results*:
This unit then shows the final result of likely predicted authors of unknown codes with the help of confusion matrix built by the classifier. It also, shows the correctly classified instances and the non-correctly classified instances from the provided set of instances. As we are using decision tree based algorithm, it then builds the tree of the feature values.

IV. Experimental Results

To study the practical aspect of our study we have applied the technique to the source code samples collected. The source code samples are collected from different sources such as student study assignments, code snippets of study books and some open source projects. The total sample consists of overall 1504 source codes with 102621 lines of code scanned. This technique is especially used for all java code samples. As mentioned above the tool called as japa which is a java parser is used for metric extraction and it works on AST technique. As far as metrics are concerned we have used 9 style based metrics as follows:

1. i-as-iterator
2. Line-length
3. Comments
4. Average procedure length
5. Methods
6. No. of arrays
7. Object creations
8. Single literal variables
9. Double literal variables

For this work there are in total 8 authors are considered. Among 1504 total source codes 1128 sample source codes are taken for the first phase of the methodology i.e. for building classification model of known authors, while remaining 376 source codes are taken for prediction of likely author identification. After applying the methodology we got total 68.89% correctly classified instances while 31.11% as incorrectly classified instances. The tabular representation of the result is shown below. (See Table II).

TABLE II.

RESULT OF STYLE BASED APPROACH

Approach	**Classification Algorithm**	**Training Samples**	**Testing Samples**	**Result %**
Style Based Approach	C4.5	1128	376	68.89

V. Conclusions and future work

This paper represents the basics of the field called author identification. Author identification is the technique that helps in finding the likely author of the unknown source codes. First it introduces this field with the help of different examples showing that how the habits of the people can be captured and is used for tracing the frauds. Then it moves to the actual topic of this paper i.e. author identification of the source codes. Then it introduces the concept of stylometry saying that it is the study of linguistic styles of the authors and the differentiation can be made using these styles between the two authors. Also, it has discussed different applications of it such as plagiarism detection, copyright disputes and legal allegations etc. Then it discussed the work previously done by different people in this field. Then it is followed by the methodology in which the 9 style metrics are discussed in detail. Also the actual process of author identification is discussed in detail in methodology. The process involves two phases from which classification model of known authors is built in the first phase while in the second phase the actual likely author identification of the unknown codes is carried out. Then the experimental results of this work are discussed. Total 1504 source codes are taken for the study from which 1128 source codes are utilized for training purpose i.e. for the first phase of the methodology and 376 source codes are utilized for testing purpose i.e. for the second phase of the methodology. Future work of this study focuses on finding different layout metrics and doing study of those metrics in order to perform author identification with the help of those metrics.

References

[1] Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, Dawn Song. "*On the Feasibility of Internet-Scale Author Identification*". IEEE Symposium on Security and Privacy (SP), IEEE Conference Publication, 2012.

[2] Jay Kothari, Maxim Shevertalov, Edward Stehle, and Spiros Mancoridis."*A probabilistic approach to source code authorship identification*", 4th International Conference on Information technology, IEEE Conference Publication, 2007.

[3] Maxim Shevertalov, Jay Kothari, Edward Stehle, and Spiros Mancoridis, *On the Use of Discretized Source Code Metrics for Author Identification*, Ist International Symposium on Search Based Software Engineering,2009.

[4] MacDonell S.G., Buckingham D., Gray A. R., and Sallis P. J. (2002) , *Software Forensics : Extending Authorship Analysis Techniques to computer programs* , Journal of Law and Information Scienece, 13(1) , pp. 34-69

[5] Rohit R. Joshi, Rajesh V. Argiddi, Sulabha S. Apte, *Author Identification: An Approach based on Code Feature Metrics using Decision Trees*, International Journal of Computer Applications, Volume 66– No.4, March 2013 (2002)

[6] R. A. Vivanco, N. J. Pizzi, *Identifying Effective Software Metrics Using Genetic Algorithm* , Canadian Conference on Electrical and Computer Engineering, 2003, IEEE CCECE 2003.

[7] Frantzeskou G, Gritzalis S., & MacDonell S., (2004) ,*Source Code Authorship Analysis For Supporting the Cybercrime Investigation Process* , 1st International Conference on E-Business and Telecommunication networks. Setubal, Portugal, INSTICC Press, pp. 85-92.