Program Authorship Identification

Mei Duanmu

Department of Mathematics and Statistics
University of Massachusetts Amherst

October 11, 2024

Outline

- Introduction
- Related Work
- Methodology
- Experiments and Results
- Conclusion

Introduction

• Objective: Different Programers have different habits of coding. The objective of the project is to identify the author of the code.

• Hypothesis: Programmers are unique and that this uniqueness can be observed in the code they write.

• Application: Plagiarism detection, author tracking, ownership disputes etc.

Related Work

Previous work on author identification usually base on two levels of code: source-code level and binary-code level.

• Source-code based approach: extract features at source code level.
E.g. In the work of Macdonell etc, stylish features are extracted from C++ source files and achieve accuracy from 81% to 88% with different methods.

• Binary-code based approach: extract features at binary level.
E.g. In the work of Rosemblum etc, features are extracted from binary code and achieve 81% prediction accuracy for 10 authors.

• Our work is source-code based. We use both textual and stylish features and achieved prediction accuracy 91.87%..

Data

- Source code (C and C++)

One project may be written by multiple authors.

The number of authors of one project varies from 1 to 6.

The number of valid source files of one project varies from 5 to 125.

Table: Data Source

# authors	#projects	# files	# files per author
9	9	812	90.2

- Data Preparation

Each file (.cc, .c, .h, .cpp) serves as a data unit.

Label each file with an author.

Feature Construction

- Textual Features

Extract textual features from comments of the code.

- Stylish Features

Extract stylish features from the body of the code.

Textual Features

Textual features are extracted from textual part of the code. Here we use comments of all code as textual part.

• N-gram sequence
N-gram is a continuous sequence of n words of comments.
E.g. Who is the author of the code?
Unigram: 'Who', 'is', 'the', 'author', 'of', 'the', 'code'
2-gram: 'Who is', 'is the', 'the author', 'author of', 'of the', 'the code'

• Spelling and Grammer
24 categories errors are counted with library from LanguageTool, an Open Source proofreading software.

Stylish Features

Totally 27 stylish features are extracted.

Stylish feature examples:

Frequency of keywords(e.g. if and while) with whitespace on the left side?

Frequency of operators(e.g. '=' and '+') with whitespace on either side?

Methodology

Weka 3.7 is a software with a collection of machine learning algorithm for data mining. Three methods are tested and compared with Weka.

- Libsvm
- Liblinear
- Random Forest

Libsvm and Liblinear

Libsvm is a library for Support Vector Machines(SVMs).
Characteristics of Libsvm include various kernels, weighted SVM for unbalanced data, efficient multi-class classification and so on.

Liblinear is a library for large scale linear classification.

- Support logistic regression and linear support vector machine.
- Liblinear may not use kernels. For large data, Liblinear is more efficient.

The Liblinear is developed based on Libsvm and share similar usage.
There may be great difference between the two when kernel in Libsvm is not linear.

Random Forest

- An Ensemble classifier consisting of decision trees.
- Bootstraping aggregating improves the stability and precision.
- At each split of the node, select the best split from a random subset of features.
- Efficient for large datasets.

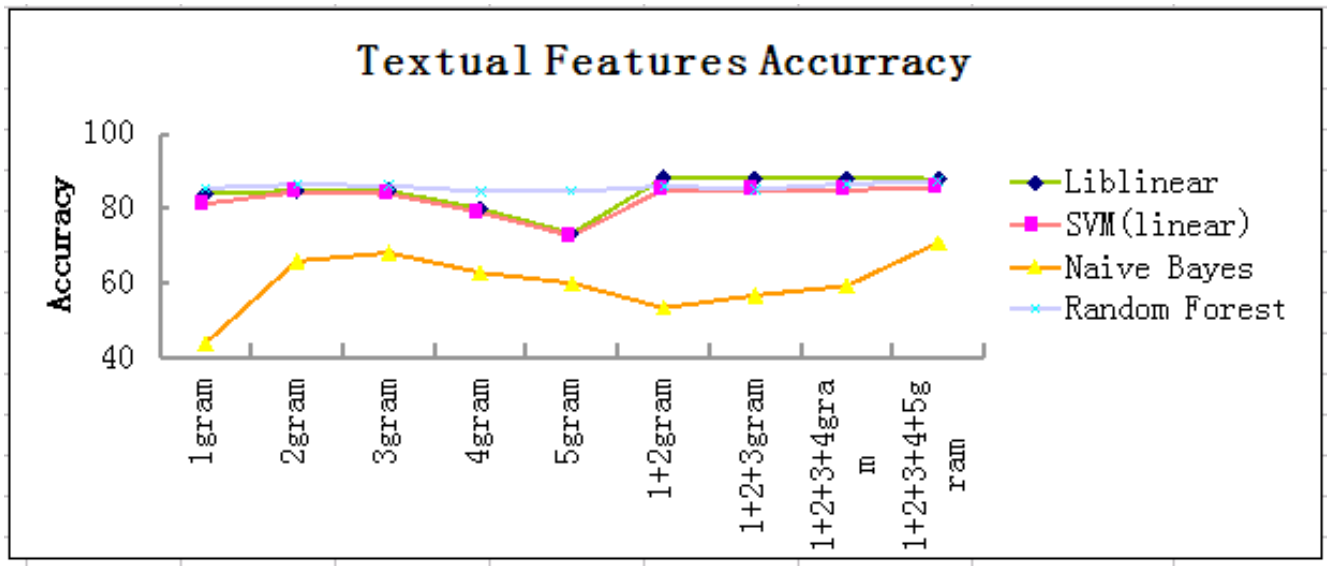
Experiment and Results

- Experiments with only textual features
- Experiments with both textual and stylish features
- Experiments with selected features

Textual Part Results

Table: Textual Feature Results

Classifier	1-gram	2-gram	3-gram	4-gram	5-gram	1-2 gram	1-3 gram	1-4 gram	1-5 gram
Liblinear(L2-loss SVM)	83.99	84.73	84.73	79.93	73.28	88.18	88.05	88.18	87.93
SVM(linear)	81.28	84.48	83.87	78.94	72.66	85.10	84.98	84.98	85.47
Random Forest	85.47	86.47	86.08	84.61	84.61	85.96	85.22	86.58	87.44



Stylish Part Results

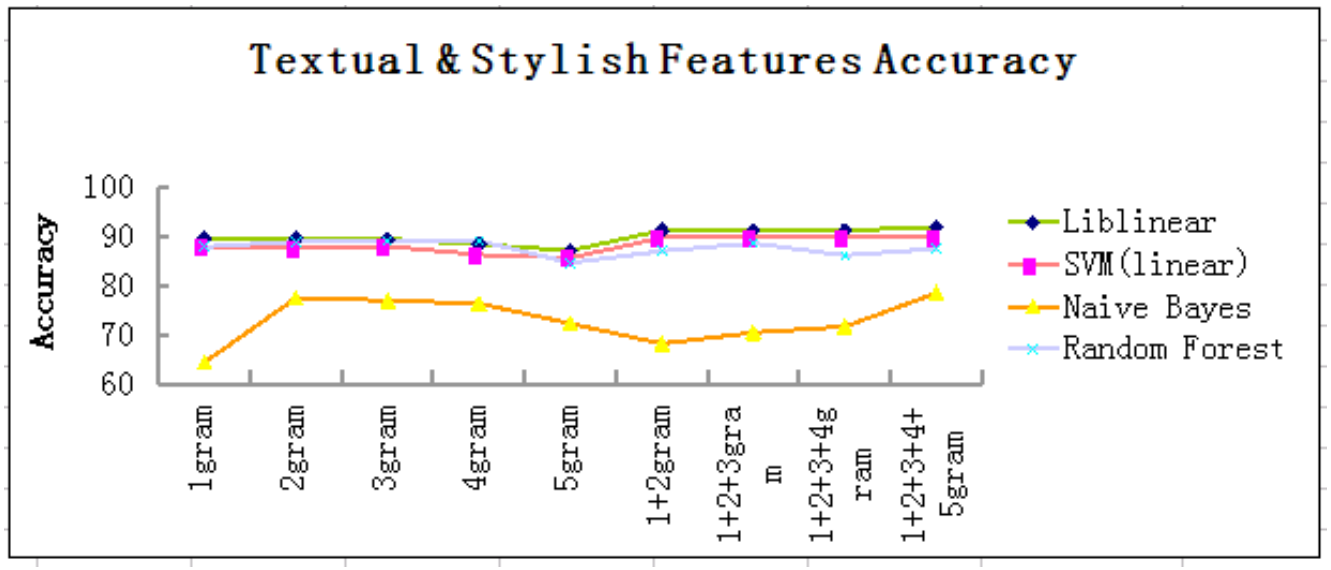
Table: Stylish Feature Results

Classifier	Accuracy (%)
Liblinear(L2-loss SVM)	63.1
SVM(linear)	62.4
Random Forest	71.7

Textual and Stylish Results

Table: Textual and Stylish Combined Results

Classifier	1-gram	2-gram	3-gram	4-gram	5-gram	1-2 gram	1-3 gram	1-4 gram	1-5 gram
Liblinear(L2-loss SVM)	89.41	89.53	89.41	88.30	87.07	91.13	91.13	91.13	91.87
SVM(linear)	87.93	87.68	87.93	86.21	85.59	89.66	89.66	89.66	89.66
Random Forest	87.81	88.92	89.04	89.04	84.61	86.95	88.55	86.08	87.44



Feature Selection

- Use information gain to rank the attributes.

Expected Information gain evaluates the change in information entropy from a prior state to a state with additional information.

- Select features based on the ranking and test the results.

Feature Selection Results

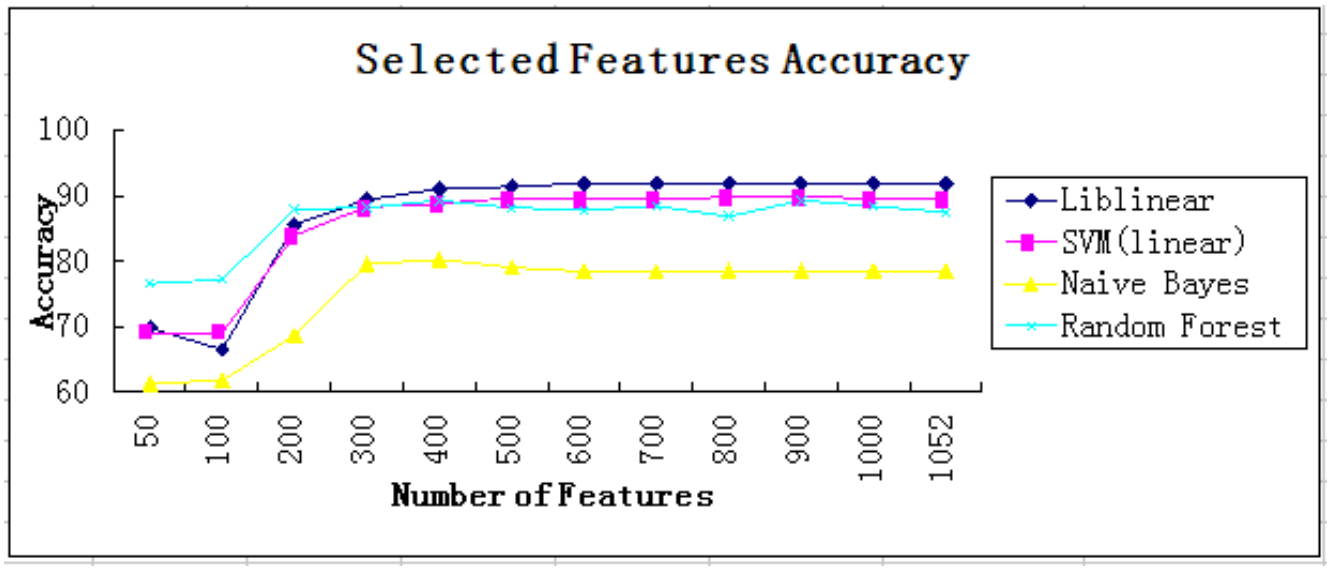
Table: Top Feature Selection Results

Features	top 50	top 100	top 200	top 300	top 400	top 500	top 600
# Textual Features	50	100	196	292	387	484	573
# Stylish Features	0	0	4	8	13	16	27

Feature Selection Results

Table: Feature Selection Results

Classifier	50	100	200	300	400	500	600	700	800	900	1000	1051
Liblinear(L2-loss SVM)	69.83	66.50	85.59	89.41	91.01	91.38	91.87	91.87	91.87	91.87	91.87	91.87
SVM(linear)	68.97	69.09	83.99	88.30	88.79	89.66	89.41	89.41	89.78	89.78	89.66	89.66
Random Forest	76.60	77.22	87.93	88.18	89.29	88.18	87.81	88.42	86.82	89.29	88.42	87.44



Conclusion

- Textual and Stylish features are extracted and combined in our work.
- Use feature selection to remove redundant and irrelevant features.
- Achieve higher accuracy of 91.87%.

Appendix 1

Rules in LanguageTool:

1. "HAVE PART AGREEMENT",
2. "PHRASE REPETITION",
3. "COMMA PARENTHESIS WHITESPACE",
4. "DOUBLE PUNCTUATION",
5. "NON3PRS VERB",
6. "MASS AGREEMENT",
7. "COMP THAN",
8. "TO NON BASE",
9. "DT DT",
10. "WHITESPACE RULE",
11. "EN UNPAIRED BRACKETS",
12. "EN A VS AN",
13. "THREE NN",
14. "UPPERCASE SENTENCE START",
15. "EN QUOTES",
16. "GENERAL XX",
17. "ENGLISH WORD REPEAT BEGINNING RULE",
18. "MORFOLOGIK RULE EN US",
19. "HE VERB AGR",
20. "ENGLISH WORD REPEAT RULE",
21. "POSSESIVE APOSTROPHE",
22. "ALL OF THE",
23. "CD NN",
24. "BEEN PART AGREEMENT"

Appendix 2

1. Proportions of lines that are blank.
2. Proportions of keywords with whitespace on both sides.
3. Proportions of keywords with whitespace only on the left side.
4. Proportions of keywords with whitespace only on the right side.
5. Proportions of keywords with whitespace on neither side.
6. Proportions of keywords with operators on both sides.
7. Proportions of keywords with operators only on the left side.
8. Proportions of keywords with operators only on the right side.
9. Proportions of keywords with operators on neither side.
10. Average number of characters per non-comment line.
11. Average number of letters that are upper case per non-comment line.
12. Proportions of lines that contain only line comments.
13. Proportions of lines that contain only close comment (/*).
14. Proportions of lines that start with start comment (/*).
15. Proportions of lines that begin with opening bracket ().
16. Proportions of lines that contain only opening bracket.
17. Proportions of lines that contain only closing bracket ().
18. Proportions of keyword "new".
19. Proportions of keyword "const".
20. Proportions of keyword "for".
21. Proportions of keyword "if".
22. Proportions of keyword "static".
23. Proportions of keyword "struct".
24. Proportions of keyword "switch".
25. Proportions of keyword "while".
26. Proportions of keyword "else".
27. Average McCabe's cyclomatic complexity number per method. If there is no method in a source file, the cyclomatic complexity number is 0.

Reference

- L. Breiman. Random forests. Machine learning, 45(1):5 32, 2001.
- Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1-27:27, 2011. Software available at http: //www.csie.ntu.edu.tw/ cjlin/libsvm.
- Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research, 9:1871-1874, 2008.
- Joachim Diederich etc. Authorship attribution with support vector machines. Applied Intelligence, 19(1-2):109-123, May 2003.
- J.H. Hayes and J. Offutt. Recognizing authors: an examination of the consistent programmer hypothesis. Software Testing, Verification and Reliability, 20(4):329-356, 2009.
- Moshe Koppel, Jonathan Schler, Shlomo Argamon, and Eran Messeri. Authorship attribution with thousands of candidate authors. In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '06, pages 659-660, New York, NY, USA, 2006. ACM.
- S.G. Macdonell, A.R. Gray, G. MacLennan, and P.J. Sallis. Software forensics for discriminating between program authors using case-based reasoning, feedforward neural networks and multiple discriminant analysis. In Neural Information Processing, 1999. Proceedings. ICONIP '99. 6th International Conference on, volume 1, pages 66-71 vol.1, 1999.
- N. Rosenblum, X. Zhu, and B. Miller. Who wrote this code? identifying the authors of program binaries. Computer Security-ESORICS 2011, pages 172-189, 2011.
- N.E. Rosenblum. The provenance hierarchy of computer programs. PhD thesis, UNIVERSITY OF WISCONSIN, 2012.
- E.H. Spaord and S.A. Weeber.Software forensics: Can we track code to its authors? Computers and Security, 12(6):585-595, 1993.

Thank you!