CSEDU 2019
11th International Conference on
Computer Supported Education

Requirements for Author Verification in Electronic Computer Science Exams

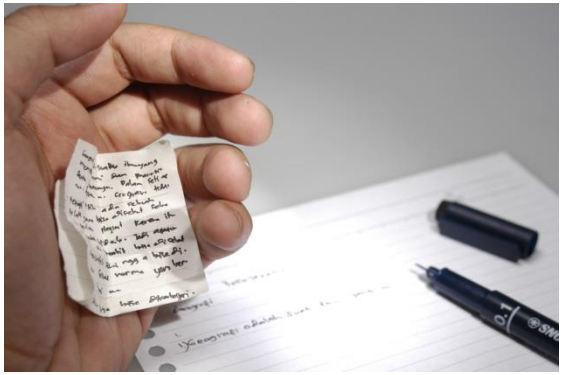
Julia Opgen-Rhein, Bastian Küppers, Ulrik Schroeder

RWTH AACHEN UNIVERSITY

Requirements for Author Verification in Electronic Computer Science Exams

Motivation

- Cheating is a problem in examinations and can have many forms [1]
- Electronic exams come with an increased danger of impersonation and illegal communication between students [2]



By Hariadhi, myself - Own work, CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=2206309

Our Project: FLEX

FLEX (Framework for FLExible Electronic EXaminations)

Requirements for Author Verification in Electronic Computer Science Exams

- Programming exams require more simple text input or multiple choice questions
- BYOD – students work on a familiar device
- Different operating systems and configurations
- Untrusted device

Requirements for Author Verification in Electronic Computer Science Exams

- Current approach: Plagiarism Detection
  - Comparison with work of other students or internet sources
  - Finds complete and partial matches
  - Solutions exist for written texts and source code

- Fails, if the actual source is not available

A-posteriori cheating detection that goes beyond plagiarism detection is required

Requirements for Author Verification in Electronic Computer Science Exams

Individual Programming Style

- Idea from Caliskan-Islam et al. [3]
- Assumption: Every student has a unique programming style
- This style is reflected in the layout of the code as well as in the way classes, methods and variables are named and which elements of the programming language are frequently used
- These personal styles are distinguishable, even if students work on the same tasks and have the same lecturer

```
public double calc_pi(int it) {
        double pi = 0;

        for (int i = 1; i < it; i++) {
            double r = 1.0 / (2.0 * i - 1);
            if (i % 2 == 0) {
                r *= -1.0;
            }
            pi += r;
        }
        return pi * 4.0;
    }
```

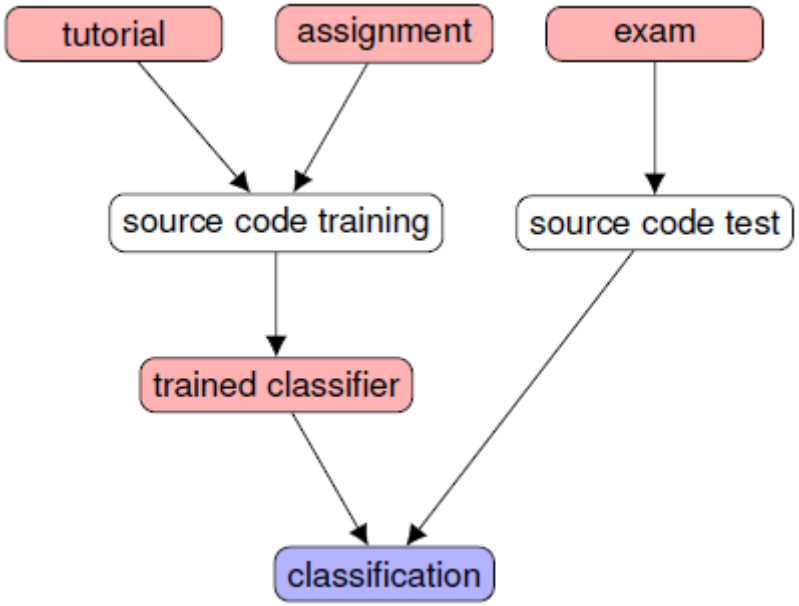
```
private double calcPi(int it){
        double pi=0;
        int k=1;
        while(k!=it){
            if(k%2==0)
            {
                pi+=-1.0/(2.0*k-1);
            }
            else
            {
                pi+=1.0/(2.0*k-1);
            }

            k++;
        }

        return 4.0*pi;
    }
```

Previous Work [4]

- Application to verify the author of source code
- Extraction of features from the code
- Classification via Random Forest, Linear Support Vector Machine or Artificial Neural Network (binary and multi-class classification)



Requirements for Author Verification in Computer Science Exams

Examples of Features

Layout Features	Syntactic Features
Number of inline comments	Number of underscores/method name
Number of block comments	Number of prefix operators
min/max/avg number of consecutive empty lines	Number of consecutive calls (e.g. do().doNext())
Number of spaces	Number of short notations (e.g. +=)
Number of { in new line	Number of classes
Number of { in same line	Number of chars/variable name

F: [number of chars, average line length, standard deviation of line length, number of lines, [term frequency of all words], frequency of ’else if ’, max node depth in AST, [TF, TFIDF, avg node depth of AST node types], [term frequency of AST node bigrams], syntactic features, semantic features]

Requirements for Author Verification in Electronic Computer Science Exams

- Tested on data of 12 students and on non-student data

1) 12 on-campus assignments from 13 students of a first semester programming course in Java

2) A set of files from the best 40 (Java) programmers that participated in Google Code Jam 2017 contest

- the files are randomly split into training and test data via 10-fold cross validation

Previous Results

	DNN	SVM	RF
Top 1	34.69%	44.54%	58.36%
Top 3	37.78%	57.69%	67.15%

Data Selection

- Keep all files, regardless of their length
- Files that do not consist of valid Java code or do not compile are discarded
- Omit ’hello world’ programs and simple text output
- Files that are not written by the students themselves like test classes are removed manually from the data set
- 47 of 437 files from the raw data set not being used for classification.

Pre-Processing

- Outlier detection
- Removal of duplicate (test) files
- Check for a sufficient amount of training data

- Principal Component Analysis is beneficial for the results of SVM but reduced the accuracy of RF and NN classifiers
- Removal of features with low variance and zero mutual information

Results

Student Data Set

	RF	lin. SVM	NN
multi	89.15%	77.18%	78.49%
binary	88.72%	80.15%	78.85%

GCJ Data Set

	RF	lin. SVM	NN
multi	100.00%	99.34%	99.69%
binary	99.41%	100.00%	99.81%

‘Relaxed‘ Author Attribution

- The author is verified, if one of the conditions holds
  - The classifier gives the highest probability to the class of the presumed author
  - The probability of the class of the presumed author is bigger than 50%, even if it is not the highest probability (this is possible in case of binary classification)
  - The probability of the class of the presumed author is the second or third highest probability but amounts to at least 0.8 times the highest probability

	RF	lin. SVM	NN	comb.
multi	91.18%	81.64%	79.10%	91.46%
binary	90.44%	83.54%	80.41%	90.95%

Conclusions

- With careful data selection and pre-processing, author verification is a useful method to detect cheating

- Assignments should produce at least ten source files
- Testing in an extra file
- Pure ’data containers’ with pre-defined names for classes and methods are not useful
- Files need to be free of parts that are written be e.g. the lecturer (test cases)
- No group work

Conclusions

- Code from all parts of a programming course can be used for cheating detection as long as it can be reasonably assumed that it is indeed authored by one particular student
- Source code does not need to be particularly complex as long as it is long
- Best feature set depends on the data set, no general reduced feature
- Students must agree that their tasks will be used for fraud detection.

Thanks for your attention! ☺
Σας ευχαριστώ για την προσοχή σας! ☺

Are there any questions or comments?

Sources

[1] Judy Sheard and Martin Dick (2012). Directions and dimensions in managing cheating and plagiarism of IT students. In Proceedings of the Fourteenth Australasian Computing Education Conference - Volume 123, ACE'12, pages 177–186, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

[2] Ingrid Noguera, Ana-Elena Guerrero-Roldán, and M. Elena Rodríguez. (2017). Assuring authorship and authentication across the e-assessment process. In Technology Enhanced Assessment, pages 86–92. Springer International Publishing.

[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). USENIX Association, Washington, D.C., 255–270. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/caliskan-islam

[4] Julia Opgen-Rhein, Bastian Küppers, and Ulrik Schroeder (2018). An application to discover cheating in digital exams. In Proceedings of the 18th Koli Calling International Conference on Computing Education Research, Koli Calling'18, pages 20:1–20:5, New York, NY, USA. ACM.