Plagiarism at Scale

Identifying and Reducing Code Plagiarism in an Online Master's in Computer Science Program

Anonymous Author(s)

ABSTRACT

In this work, we evaluate the impact of several common and distinct anti-plagiarism efforts implemented in an introductory course on operating systems (OS) in an online Master's in Computer Science program. Over ten semesters (Spring 2015 to Spring 2018) we objectively measured plagiarism within each semester's programming projects. We explored multiple mechanisms for reducing plagiarism in the projects and experienced varying degrees of success.

We define plagiarism within the context of our analysis, describe how we measure it, and then review the techniques that we trialed in attempting to reduce it. Two plagiarism-reduction techniques showed a statistically significant (e.g.: M=0.06385, SD=0.001, $p < 0.05$) decrease in detected plagiarism rates. While we found two effective mechanisms, one stands out for being effective, requiring minimal instructor time and resources, and decreasing the time required for the *other*. more traditional, effective mitigation approach.

Specifically, we demonstrate that an honor code *combined* with a simple student evaluation is effective at reducing plagiarism rates. This represents a novel "middle ground" between prior work that shows honor codes are **not** effective in online classes and prior work that shows a comprehensive course in academic honesty, with evaluation *was* effective.

CCS CONCEPTS

• **Applied computing** → *Interactive learning environments*; *Distance learning*; *E-learning*;

KEYWORDS

MOSS, Academic Honesty, Honor Codes, Plagiarism

ACM Reference Format:

Anonymous Author(s). 1997. Plagiarism at Scale: Identifying and Reducing Code Plagiarism in an Online Master's in Computer Science Program. In *Proceedings of ICER 2018*. ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Academic honesty is a critical component of those engaged in research. Because we build upon the work of others, it is imperative that we be able to rely on the integrity of work done by others:

> *As a society, we rely on the academic and journalistic integrity of other people's work. The whole point of academic research is to share knowledge with others and learn from one another. Since knowledge and ideas are the primary product produced by academic communities, it is essential that this knowledge is accurate and gives credit to those who created it. [43]*

The concept of intellectual integrity is not restricted to academic work; dishonest behavior spills over into the non-academic world. Cheating is a difficult habit to break, and forms of cheating (such as plagiarism) spread easily when left unchallenged. [13]

Instructors for complex technical classes focus on conveying the material to promote mastery. Students, on the other hand, often focus on achieving the highest possible marks, and when they have not achieved mastery of the material, they may submit work that is not their own.

Started in 2014, our part-time program grants a Master's of Science in Computer Science to students who complete it. All courses and student interactions are conducted online. The average student is in their mid-30s and employed full-time. Many work in the technology industry. Our university seeks to maintain its reputation for high academic standards, thus evaluating students' work for integrity is essential. Online courses are expected to show plagiarism rates comparable to traditional, in-person courses (in which plagiarism does occur, even within high-ranking institutions) [5, 24, 29, 33, 41]. However, courses in which the students tend to be employed full-time (such as ours) generally show higher rates of plagiarism than courses taken by full-time students [36].

This work evaluates plagiarism rates for the three programming projects included in an online, graduate-level introductory operating systems course that is taught at scale, with typical class sizes of 200 or more students and a trend to larger class sizes. We have full-semester data available for all but the first and the most recent (incomplete) semesters.

Using the available data, we evaluate plagiarism rates for those projects across the various semesters. Over time, the class instructors evaluated and implemented specific intervention mechanisms with the goals of understanding and reducing plagiarism rates.

Over the course of this study, we worked with the instructional staff to implement five different interventions (see §4). Of those five, we identified two that are effective (see §4.2 and §4.5). We describe our interventions, and our evaluation and analysis of the results.

Of the two techniques we identify, one (§4.5) demonstrates a "middle ground" between prior work that showed honor codes do not work, which we confirmed (§4.3) was not effective for our class, and prior work that showed a full course with assessment did work for online clases. Notably, our technique requires nominal instructor effort to implement *and* substantially decreases the effort required to implement the other successful intervention strategy (§4.5).

ICER 2018, August 13-15 2018, Espoo, Finland

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

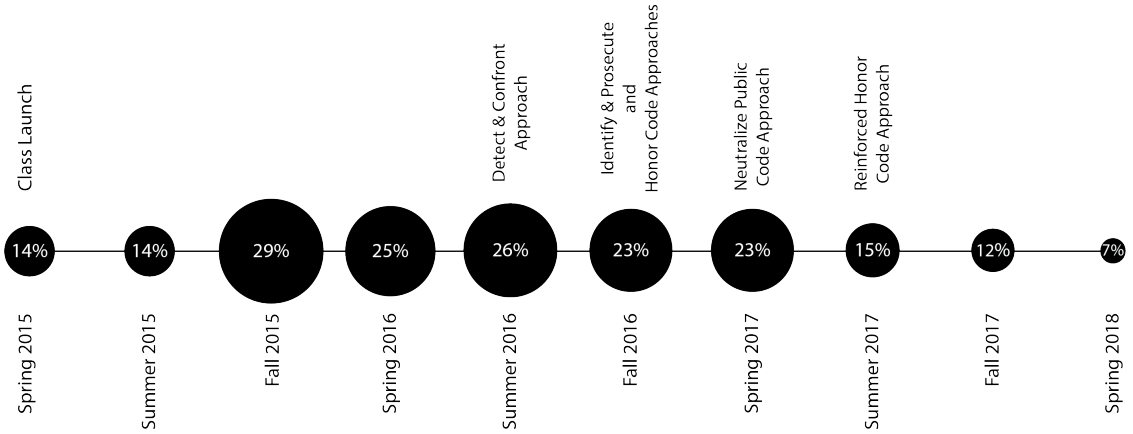


Figure 1: Course Timeline
Intervention timeline and percentage of unique students detected via automatic plagiarism checks. A student flagged multiple times is only counted *once*.

2 PRIOR WORK

While we are still early in the understanding of online-instruction dynamics, the available evidence suggests that plagiarism occurs in both online and in-person programs at scale. [6, 16, 24, 33, 41, 48, 56, 59] Much of the prior work focuses on *undergraduate* academic dishonesty. To the best of our knowledge, this is the first multi-year analysis reporting results from trialing actual intervention strategies in an online computer science graduate course.

Motivating students to adhere to academic honesty policies and the instructional strategies that are effectively achieve this goal, particularly with respect to plagiarism in computer science classes remains an open area of research and practice. [15]. Although some prior work has explored root-cause analysis for plagiarism, our goal was less to understand *why* students plagiarize and more to understand how to discourage it through a combination of detection and education mechanisms. [19] Our approach is consistent with prior work suggesting a move away from a moralistic view of plagiarism, instead reframing discussion of plagiarism as an integral part of the learning process. In other words, *even if* students do not understand what plagiarism is when starting the program, educating them about it is important and beneficial, albeit secondary to the instructors' primary goals. [1] That said, our work also supports the prior observation that simply informing students about an honor code *does not* lead to a decrease in plagiarism in online programs. [37] Active education about academic honesty through formal training, however, *may be* effective. [14]

We admit that we do not know the optimal mechanism for achieving the goal of motivating students to adhere to academic honesty policies. We have not explored the specific issue of motivations behind plagiarism (which the literature suggests can be cultural, economic, or perception-based), but we do note that our successful interventions are consistent with educating students, regardless of their cultural background. [4, 9, 12]

There is a strong body of research describing techniques for reducing plagiarism, but there is a much smaller pool of evaluations of applying these techniques. [20, 34, 52] One side effect of this paucity of research is that the most common strategy for combating plagiarism is to shift responsibility to the individual instructors, which results in haphazard enforcement and knowledge-sharing. By providing rigorous review of techniques applied over time to the same class, with similar student populations, we contribute solid evidence of techniques that work. We hope that universities can incorporate this evidence into a broader initiative toward improvement for *all* courses, rather than rely on anecdotal knowledge shared haphazardly by individual instructors. [40, 51]

3 BACKGROUND

As noted in the introduction, the subject of our study is an online graduate-level introductory course in operating systems. The course consists of a set of video lectures, two proctored examinations, three programming projects, and one narrative evaluation project. Examinations are modified on a per-semester basis; some prior examinations are provided as study guides so students can review the format and general scope of questions. All programming is done in the C programming language. [31] Instructors evaluate student projects and provide feedback on their deficiencies. Because our specific program provides a credential, it must take prudent steps—including preventing plagiarism—to ensure its integrity. [21]

3.1 Class Information

The university collects and publishes data about the courses in the program. Our analysis covers data collected between the Summer 2015 and Spring 2018 semesters, though we have only partial data for Summer 2015 and Spring 2018 as of this writing. From course inception in Spring 2015 through Spring 2017, 1,691 students enrolled in the course and 632 students subsequently withdrew. This is the fourth highest withdrawal rate (39.5%) at the university. A popular student-review website unaffiliated with the university describes the course as challenging and requiring a substantial work commitment.

Table 1: Project 1 Submission & Plagiarism Data

Project 1 submissions consist of four distinct units; each unit is independently analyzed. Per semester, we report the total number of submissions and the percentage flagged for plagiarism (as defined in §5).

Semester	PR1-A		PR1-B		PR1-C		PR1-D	
	Submissions	% Plagiarism	Submissions	% Plagiarism	Submissions	% Plagiarism	Submissions	% Plagiarism
Summer 2015	152	3.95	145	4.14	121	3.31	129	5.43
Fall 2015	161	11.80	154	9.74	136	6.62	142	9.15
Spring 2016	148	6.76	137	8.76	136	4.41	133	6.02
Summer 2016	141	1.42	142	4.93	119	5.04	123	4.07
Fall 2016	173	6.36	176	6.82	141	4.26	154	5.84
Spring 2017	137	8.03	136	11.03	107	11.21	111	13.51
Summer 2017	72	1.39	70	0.00	62	3.23	68	0.00
Fall 2017	217	2.76	211	1.90	192	2.60	204	1.47
Spring 2018	255	4.31	252	4.76	218	1.83	228	2.63

3.2 History

The first presentation of this course was in Spring 2015. Our ability to evaluate plagiarism for this semester was limited because much of the relevant data was unavailable; we did have access to a subset of the data. Since prior students' solutions to the assignments for this first semester were not available, our presumption is that the rate of plagiarism was likely low. From Spring 2015, some students placed their solutions to the projects in publicly accessible code repositories. We expected to observe some increase in the rate of plagiarism as a result.

The course has evolved during its lifetime. From Summer 2015 through Spring 2016, code was submitted via either the university's learning management system or a custom web-based interface provided by the content provider. In many cases, students submitted to *both*; such submissions were flagged as plagiarism.

Instructors evaluated the initial submission of each project by running a simple set of tests and then providing feedback to the students. Formative analysis of the projects was done after the submission deadline by using a set of black-box tests that were not available to students. Instructors used the results of those tests to provide feedback to the students. Beginning in Summer 2016 and continuing through Spring 2018, students submitted code to a centralized grader via an automated submission system that saved copies of each student's submission. Grading was based on the results of the *last* submission prior to the submission deadline. The automated system allows instructors to submit all student code submissions to the Stanford MOSS service (see §5), which simplifies the process of performing routine plagiarism checks. Using those checks, the instructors identified evidence of plagiarism, which led to our involvement with the instructors in doing further evaluation the available data. Subsequently, most of the historical data was also made available to us for this study.

4 INTERVENTIONS

Figure 1 is a course timeline marked with the interventions that we describe within this section as well as the observed student plagiarism rate *across all projects* — in other words, each student detected as a case of plagiarism, based on our definition (see §5).

Having observed evidence of plagiarism in Summer 2016, in consultation with the instructors, we initiated a series of intervention strategies aimed at proactively reducing the plagiarism rate in the course. Most interventions were trialed independently. Two interventions were trialed at the same time, but we did not observe any effect from those. The interventions that were successful were introduced independently, which permits us to do this retrospective analysis (see also §3.2).

4.1 "Detect and Confront" (Summer 2016)

In Summer 2016, a student reported their code repository on the university server had been duplicated ("forked") by another student. As a result of that report, the instructors initiated an investigation, which ultimately led to the implementation of systematic evaluations of student submissions for plagiarism. MOSS data [2] (see §5) was generated as part of the normal automated code-evaluation system, making this investigation straightforward. We worked with the instructional team from that point to evaluate the plagiarism data. Our analysis of prior semesters is retrospective.

The plagiarism rate varied between 8% and 10% between Fall 2015 and Summer 2016, with a notable increase after Summer 2015, when the plagiarism rate was measured at 5%. We hypothesize that the initial low plagiarism rate was the result of the course being new; no students had yet publicly posted code for the projects.

The instructional team discussed strategies for handling this newly discovered problem. They decided to implement an amnesty intervention. The instructors made a public post on the class discussion forum (Piazza) noting the discovery of plagiarism and offering the option of coming forward:

> For these students in question, we will give 48 hours (12:00PM 16 JUL 16) to put a private note on Piazza for the Instructors' team with information of where you got the answers from and will only receive a zero for the assignment. Otherwise, those who do not self report will still get a zero and be reported to the Office of Student Integrity and called for a hearing.

Interestingly, a number of students *did* come forward. **None** of them had, in fact, committed anything bordering on substantial plagiarism. The instructors investigated these cases and concluded

that the confessing students had consulted with standard reference manuals or online assistance sites and typically were concerned over small pieces of code they had taken from online sources, such as Stack Overflow [26], but had failed to cite in their submissions. None had been suspected of plagiarism, and their conduct did not rise to a level the instructors considered plagiarism. No action was taken against *any* of the students that attempted to claim the amnesty. Conversely, those students who had been suspected of plagiarism did not take advantage of the amnesty.

4.2 "Identify and Prosecute" (Fall 2016)

After the amnesty experience, the instructors took an aggressive stance on plagiarism in Fall 2016. However, enforcement took place *after* the second project submission, and university time limits on reporting suspected academic dishonesty foreclosed the option for enforcement on the first project.

Enforcement was rigorous. Instructional staff met with the students, advised them they were investigating a *different* student for potential plagiarism, and asked for insight on how the other student might have gained access to the copied code. In most cases the students confessed to their inappropriate behavior, were given a failing score for the individual assignment, and continued in the class. Some students refused to respond to instructor inquiries, and others denied they had plagiarized their submissions. Those cases were referred to the university for further investigation. No decrease in the plagiarism rate was observed.

The following semester (Spring 2017), the instructional staff began enforcing the anti-plagiarism policy shortly after the completion of the first project, using the same strategy as in Fall 2016: meeting with students, granting a failing grade to students who confessed, and referring the remaining students to the university. Unfortunately, the two students who did not respond were *not* reported to the university, and *both* of those students repeated their plagiarism for Project 3. None of the other offending students repeated their inappropriate behavior, and there was a notable drop in the plagiarism rate for the second project. However, this approach of meeting with students is resource intensive. We discuss the resource requirements of this approach in §6.2.

4.3 "Honor Code" (Fall 2016)

In Fall 2016, the instructors posted a public message on the class forum (Piazza), "pinning" it so it appeared near the top of the forum messages. The post established a definition of collaboration for the course, encouraging discussion of techniques, problem-solving mechanisms, concepts, and reference materials. The post also defined plagiarism: using someone else's code *without* attribution. It stated that students could use small blocks (10 lines or fewer) of code from public sources provided they properly cited the source.

A copy of the Piazza post was sent (via e-mail) to every student in the class. This step was unusual but consistent with the instructors' intent to clearly communicate their expectations to students.

We did not observe any meaningful decrease in the plagiarism rate for the course for Fall 2016. We did observe a notable decrease in Spring 2017, but we attribute that to the aggressive enforcement action also undertaken at that time (see §4.2. With that in mind, we conclude that the intervention undertaken in Fall 2016 did not have a material impact on the plagiarism rate. This observation is consistent with prior work indicating that honor codes *by themselves* are not effective in online programs. [24]

4.4 "Identify & Neutralize Public Code" (Spring 2017)

From analyzing the entire body of reported cases of plagiarism in this course, we observed that there is a flow of code from semester to semester as students make their code available publicly, whether through a code repository such as github.com or a homework-sharing service such as coursehero.com — both of which contain code from former students of *this* specific course (as well as many others).

Based on our understanding that this sort of cross-semester code-sharing was occurring, in Spring 2017 instructors agreed to work with us to implement a new intervention: removing prior students' code from public repositories. We used uniquely identifiable information from the project, such as project-specific function names within the code itself, to locate over a dozen separate public locations for course code. Whenever a public repository was identified, one of the instructors would contact the owner of that repository and request they remove the students' code. In most cases, the repository owner agreed.

One reason former students commonly gave for sharing their code publicly was to allow them to show the code to prospective employers as an example of their abilities; in those cases, the instructors asked the students to keep the code private and only share it as necessary. This is another case in which the goals of the instructors and students (albeit *former* students) were at odds.

Despite these efforts, new repositories continued to appear even as older repositories were removed. As of the date of this writing, there are five easily identified public code repositories on github.com. We have suggested that the "DMCA Takedown" mechanism might be useful in removing public code, but that effort has not been evaluated. [23] Instructors hesitate to purse this approach because it would require utilizing instructor time to work with the university lawyers.

Because requesting the removal of course-related code from public repositories has been ongoing and because new repositories continue to appear, we maintain that this enforcement method is unlikely to be a factor in the improvements observed in §4.2 and §4.5.

4.5 "Honor Code with Reinforcement" (Summer 2017)

As previously noted, the university has a written honor code. Reminding students of the honor code was not an effective intervention strategy in and of itself (§4.3). Prior work has found that *teaching* academic integrity, including formal evaluation, may be effective for online classes. [14] With this in mind, we proposed a new approach in Summer 2017: constructing a short educational module describing the honor code and its purpose, as well as requiring that students take a short quiz in the first two weeks of the class acknowledging they had read, understood, and agreed to abide by the honor code. The quiz was entitled **Project Collaboration Policy** and consisted of two questions. The first displayed a

Table 2: Project 2 Submission & Plagiarism Data

Project 2 consists of two separate submission units. Per semester, we report the total number of submissions and the percentage flagged for plagiarism (as defined in §5).

Semester	Part 1		Part 2	
	Submissions	%Plagiarism	Submissions	%Plagiarism
Summer 2015	136	7.35	132	7.58
Fall 2015	146	9.60	133	9.00
Spring 2016	140	1.43	136	2.94
Summer 2016	134	8.21	128	7.81
Fall 2016	150	6.67	145	8.97
Spring 2017	122	4.10	121	3.31
Summer 2017	71	0.00	69	2.90
Fall 2017	199	1.01	193	0.00

Table 3: Project 3 Submission & Plagiarism Data

Project 3 consists of two separate submission units. Per semester, we report the total number of submissions and the percentage flagged for plagiarism (as defined in §5).

Semester	Part 1		Part 2	
	Submissions	%Plagiarism	Submissions	%Plagiarism
Spring 2015	176	14.20		
Summer 2015	129	17.05	131	14.50
Fall 2015	137	23.36	136	17.65
Spring 2016	130	16.92	130	12.31
Summer 2016	122	13.11	122	13.11
Fall 2016	140	20.71	140	20.71
Spring 2017	114	13.16	114	13.16
Summer 2017	60	10.00	60	10.00
Fall 2017	173	9.83	173	9.83

letter from the Dean of the College and asked the student to *Please confirm that you have read and understood the following message sent by the Dean.* The second question referred to the instructors' "Collaboration versus Cheating" policy as posted on Piazza: *Please confirm that you have read and understand the clarification on Collaboration vs. Cheating posted in Piazza.* This policy statement was the same one the instructors had posted in prior semesters. A small number of students followed up after this quiz, asking clarifying questions in the course-discussion forum.

When we began analyzing the plagiarism data for Summer 2017, we were pleasantly surprised to observe a *noticeable* decrease in detected plagiarism rates. However, the Summer 2017 class was the smallest in the program's history, so we theorized that the lower rates of plagiarism might be simply a natural variation due to the (relatively) small sample size. We did find these results encouraging and anxiously awaited the results for Fall 2017. No new interventions were trialed that semester.

The Fall 2017 class was the *largest* class since the program's inception. This addressed our concern about the relatively small class size in the previous semester. Our review of the data supported our prior observation that the rate of plagiarism had materially decreased.

The instructional team has continued this intervention into Spring 2018; no new intervention has been added. Spring 2018 is the largest presentation of the course to date, and although the course is still in progress as of the time of this writing, preliminary results are good.

5 PLAGIARISM DETECTION

This work relies on the ability to detect plagiarism. We utilized multiple automated techniques for detecting academic dishonesty. The primary tool used throughout the lifetime of the course is **MOSS**, or **Measure of Structural Similarity**. [2] This service is freely available through Stanford University's Computer Science department and is a popular tool for detecting plagiarism in programming assignments such as those used in the class we are evaluating. [18, 46, 49] Routine plagiarism evaluation is performed by submitting the current semester's students' code in its entirety. Each student's code is compared against all other code submitted in the same batch, and MOSS permits for submission of any skeleton ("common") code that is provided to students by the instructors. The resulting automated report is a set of web pages that display regions of structural similarity between any two submissions from the submitted code.

Students are provided with a basic skeleton program, written in the 'C' programming language [31], that implements command line arguments and basic initialization. The code is built using standard Linux tools and the code builds, but does not provide any functionality. The projects themselves attempt to introduce students to basic operating-systems concepts: threads, inter-process communications, queues, stacks, I/O, etc. These low-level systems implementation tools are often unfamiliar to students who are used to working with higher-level languages and abstractions.

At our urging, instructors introduced a second technique in Fall 2016: "watermarking" the student starter code. Although watermarking did not prevent plagiarism, it did provide us with a tool for establishing provenance. Thus far, watermarking has not identified instances of plagiarism that were not independently flagged by **MOSS**.

A third technique, introduced in Fall 2017, was to calculate and include in the information submitted to the automatic grading system the **SHA-2** checksum of files within the student environment; one of the key components was the actual submission script used to send student code to the automated evaluation system. This permitted us to determine whether students were modifying the tools in order to spoof the grader. While we have observed separate checksums, our investigations thus far have not identified more nefarious academic dishonesty. Because this approach also reports the checksums of files that are not submitted by students, we can use it to flag student code from prior semesters.

A member of our research team suggest that one possible explanation for the observed decrease in the plagiarism rate was that students became better at hiding their plagiarism. While we cannot definitively exclude this possibility, our analyses of watermarks and checksums do not suggestions any irregularities not detected by MOSS. Because we have not previously disclosed our use of watermarks and checksums, we think it unlikely students would be aware of it. A further advantage of evaluating checksums is that SHA-2 is known to be difficult to spoof; a student brilliant enough to do so is unlikely to need to plagiarize the code for an introductory graduate operating-systems course. [45]

Table 4: Aggregate Plagiarism Results

MOSS-reported similiarity across all semesters: breakdown by project/sub-project, reporting numer of flagged submissions, percentage (of total), plus average similarity % and average line count similarity (with standard deviation).

Project	Total Submissions	Plagiarism		MOSS			
		Detected	Percent	Average %	Standard Deviation	Average Lines	Standard Deviation
PR1-A	1456	77	5.29	11.35	15.39	43.67	50.62
PR1-B	1423	83	5.83	11.03	16.32	30.76	36.46
PR1-C	1232	54	4.38	11.44	10.67	35.70	27.01
PR1-D	1292	66	5.11	13.22	14.13	37.51	32.51
PR2-A	1094	52	4.75	12.05	11.16	30.82	25.78
PR2-B	1056	54	5.11	8.36	16.32	64.43	107.43
PR3-A	1181	184	15.58	24.05	15.75	26.28	18.16
PR3-B	1012	136	13.44	22.83	16.50	28.01	23.14

Table 5: "Honor Code with Reinforcement" T-Test Results

See §4.5. One-tailed two-sample t-test, assuming unequal variances: all but **PR3-B** demonstrates statistical significance of intervention from pre-intervention classes.

Project	P (T<=t)
PR1-A	0.036
PR1-B	0.006
PR1-C	0.020
PR1-D	0.004
PR2-A	0.004
PR2-B	0.044
PR3-A	0.005
PR3-B	0.051

In the unlikely case that students have developed the level of insight necessary to circumvent the various plagiarism checks in the system, they have done so independently of other students — MOSS detects collaborative plagiarism. Thus, to circumvent MOSS, a student must find a candidate code base and then substantially alter its structure to avoid detection. Even if a student were to actively use MOSS to confirm their code no longer appeared structurally similar to the original code they plagiarized, they would then need to also make sure their modified code *works*, which in turn means fixing it and debugging any problems that arise due to the substantive changes. Given that our pedagogical goal is to ensure students understand the underlying mechanisms, this goal is achieved if the student must understand their now modified code base sufficiently well to pass the tests of their code. This effect has been separately noted, particularly with respect to MOSS and programming classes. [22]

6 EVALUATION

In our retrospective evaluation of the entire corpus of student code submissions for the course, we chose to minimize the potential for bias in our findings by defining *plagiarism* in a manner that can be automatically detected. Our basis for this choice was to mitigate the possibility of subjective bias on the part of the researchers at excluding specific cases of potential plagiarism. In terms of actual enforcement actions in a real class, we would not rely solely upon automated identification: we must *also* engage our expertise to verify that the flagged submissision legitimately represents an instance of plagiarism. **All** of the cases flagged in our analysis would have been evaluated for further action; indeed, many of them were further evaluated.

For the purposes of our study, an instance of plagiarism is defined by a MOSS-reported similarity score of 30% or more, with more than 30 lines of structurally similar code. While this may exclude some cases of plagiarism, it is broadly consistent with the data and represents a balance between false positive and false negative rates.

Since our goal is measuring *effectiveness*, a reasonable model should indicate whether our interventions work. During our analysis, we considered several different thresholds, as well as a sieve model in which potential cases were reviewed, to ensure that this was the case. That review process led us to change the way we processed MOSS data and eliminate duplicate student submissions by providing the student prototype code to MOSS so that overlap would not show up as plagiarism among the small group of students who submit the base code.

Each project consists of discrete components: Project 1 has four, Projects 2 & 3 have two. Each component is individually submitted for grading. Our use of MOSS analyzes each component separately. Table 1 reports our results for Project 1 from Summer 2015 through Spring 2018. Table 2 reports our results for Project 2 from Summer 2015 through Fall 2017. Table 3 reports our results for Project 3 from Spring 2015 through Fall 2017. Table 4 summarizes all presentations of the course.

We have also provided detailed information about the quantitative nature of similarity reported by MOSS for *all* submissions from the available historical data. We break this down by the average structural similarity, both as a percentage and as a line count, and we provide the standard deviation for both of those values.

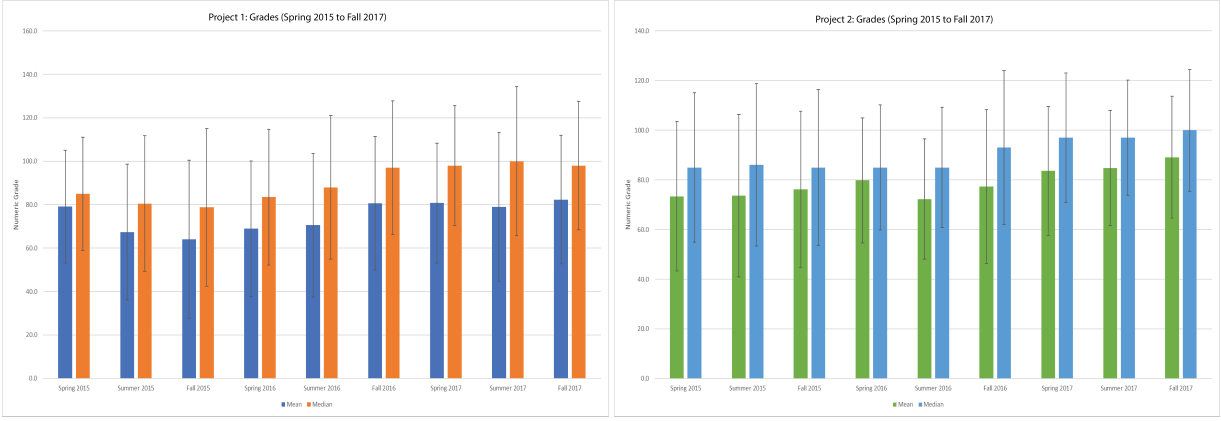
This aggregate data provides insight into the varying nature of the projects. Project 3, for example, utilizes a code generator for part of the student-submitted code, corresponding to a substantially higher level of similar code. We consider this is a limitation of the design of the project rather than an indicator of improper student behavior.

6.1 Effectiveness

Because we noted that results were comparable for all but two of the interventions, we did not perform a statistical analysis of the comparable interventions. While we did find the "Identify and Prosecute" intervention effective, we do not see that as providing novel insight — though it may provide some insight into the lower bounds of what is reasonably achievable.

We did perform a statistical evaluation of our "Honor Code with Reinforcement" intervention. For each separate component that was submitted, we performed a one-tailed two-sample t-test, assuming unequal variances. We decided this approach was appropriate for this evaluation because we were only interested in interventions that yielded lower plagiarism rates. We were testing distinct groups of students, but they had similar backgrounds. We tested against the null hypothesis with a 95% confidence ($p < 0.05$).

In evaluating Project 1 cases (four of the eight total components per class) we used data from Summer 2015 through Spring 2018,



(a) Project 1 Grades **(b) Project 2 Grades**

Figure 2: Projects 1 & 2 per-semester mean and median grades
Decreased plagiarism has not cause decreased student grades.

with the pre-intervention data coming from Summer 2015 through Spring 2017, and the post-intervention data coming from Summer 2017 through Spring 2018. This represented four distinct data sets to analyze. We used the same data set for Project 2 cases (two of eight total components), with Summer 2015 through Fall 2017 for the pre-intervention case, and Summer 2017 through Fall 2017 data for the post-intervention case. We omitted the Spring 2017 data for Project 3 because it represented the "Identify and Prosecute" intervention. For Project 3, we used data from Spring 2015 through Fall 2017. Spring 2015 to Fall 2017 data was used for the pre-intervention case, and Summer 2017 through Fall 2017 were used for the post-intervention case. We also omitted the Spring 2017 data for Project 3 under the same rationale as we omitted it from the Project 2 data.

Our interpretation of these results is that the samples in question are unlikely to come from the same population ($p < 0.05$). We conclude that our intervention in this case is significant.

6.2 Resource Requirements

An important consideration for the instructional staff is the level of resources required for implementing these interventions. Beginning in Spring 2018, we collected detailed information about the level of effort required for the two effective interventions. Our findings, while preliminary, indicate that, on average, the preparation of the honor code and the formal evaluation of student understanding takes less than one hour of instructional time; the University learning-management system saves time by permitting them to be copied from prior semesters to the current semester. Responses to follow-up questions from students and clarification of the policy required approximately an additional hour of time.

The time required for students to complete the evaluation is generally less than 15 minutes. Fewer than a dozen students asked follow-up questions; most questions were one or two paragraphs long.

Preliminary analysis to review and report on suspected cases of plagiarism took approximately 10 hours. Much of this work is now done via automated scripts, which can extract the relevant information. We have created additional scripts that will decrease the amount of evaluation time required during future presentations of the class.

The instructional staff spends three to four hours *per case* of suspected plagiarism to review the material, schedule a meeting with the student, post a private message to the student on the course-discussion forum, and meet with the student (if the student chooses to respond). Students who admit to plagiarism are offered the option of resolving the issue by taking a score of zero on the plagiarized assignment; the agreement is reported to the university. When a student denies plagiarizing, the instructional staff must spend approximately five hours constructing a detailed description of the evidence of plagiarism, which includes noting the code and the MOSS output, identifying watermark indicators, pointing to the original source code or other student's code, preparing the formal complaint process for the university, etc.

We have not formally quantified the overall cost, but even the most superficial analysis indicates that any intervention that minimizes cases of plagiarism substantially improves over the enforcement model.

6.3 Observations

Prior to this work, our evaluation was restricted to doing intra-term analysis of student code submissions. As we previously noted, integration of MOSS into the automatic grading system (Summer 2016) made this analysis simpler to perform. One conclusion of a thorough review of the output of that process is that the automated system, while convenient, can provide a high degree of false-positive results. Excessive false positives demonstrate the need to improve the performance of the tools used within the course.

We also took the opportunity to perform a broad cross-sectional observation of student submissions over time. We observed patterns of data sharing that suggest students find and use public repositories. While we saw no real decline in plagiarism rates when we removed existing repositories, the sharing pattern indicates that it is important to convince students not to share their code publicly.

Table 6: Cross-semester Plagiarism Cases Identified

MOSS analysis: all course submissions, all semesters. Exclude same-semester flagged instances; student counted at most once: report new students and previously reported students. Single-semester analysis fails to detect some plagiarism.

Project	Unique Instances	Pre-existing Instances
PR1-A	15	29
PR1-B	17	30
PR1-C	5	18
PR1-D	3	24
PR2-A	6	11
PR2-B	14	24
PR3-A	25	32
PR3-B	0	1

One interesting insight that emerged from our review is that not only did the absolute level of MOSS-detected plagiarism decrease, but the relative *severity* of plagiarism has decreased. To illustrate this point, we computed the *average* similarity score, as reported by MOSS, for all instances that met our definition of plagiarism for the purposes of this study (see Table7). These averages appear to have decreased substantially.

We were curious whether the decrease in plagiarism corresponded to a decrease in student grades, so we analyzed project scores for Summer 2015 through Fall 2017 (see Figure 2). Our conclusion is that the decrease in plagiarism *did not* correlate with decreased student grades.

7 ADDRESSING LIMITATIONS

One member of the University instructional team questioned the value of doing cross-semester analysis, citing anecdotal wisdom to us: "If one student cheats from a given source, itś almost certain that another will use the same source." We decided to analyze our data and determine if this claim was supported by the data.

Thus, we analyzed our full-cohort data, as shown in Table 6. We reviewed all identified students and compared them with the intra-semester data to determine how many of the cases discovered in this fashion were not identified by the intra-semester analysis. In doing this analysis, we **excluded** all same-semester code matches. The cross-semester approach identified 85 additional cases, in addition to 189 cases that duplicated the intra-semester findings.

There were 47 unique users from the 85 cases, and 73 unique users from the 189 duplicate cases. We note that we had previously identified **293** unique students in the cross-semester analysis, thus indicating that the cross-semester analysis failed to identify **220** of those unique students: we were surprised at this finding. We suggest further research may provide additional insight in §8.

Thus, the intra-semester analysis identified 293 unique students across all projects and semesters, while the cross-semester analysis identified 47 unique students that had not been previously identified. While we will recommend continuing with the cross-semester analysis, the 16% increase in identified cases may not strike other practitioners as justifying the added effort required to do cross-semester analysis.

Table 7: Average Similarity for MOSS identified cases

Notable decrease in *average* post-intervention similarity: decreased cheating level for flagged students.

	Project 1				Project 2		Project 3	
	Part A	Part B	Part C	Part D	Part A	Part B	Part A	Part B
Summer 2015	76.50	51.25	54.29	43.20	65.10	60.91	61.68	
Fall 2015	60.00	69.90	47.89	55.23	45.33	65.36	52.66	57.21
Spring 2016	69.20	63.25	43.50	49.75	31.00	48.50	48.18	52.81
Summer 2016	44.50	43.86	39.33	41.20	49.36	62.70	41.56	62.70
Fall 2016	53.27	64.67	44.67	59.89	47.40	56.46	48.38	48.38
Spring 2017	85.09	69.07	56.67	67.27	59.40	73.75	49.80	49.80
Summer 2017	30.0	0.00	36.50	0.00	0.00	55.00	35.33	35.33
Fall 2017	41.33	42.00	51.20	52.33	33.50	0.00	40.76	40.76
Spring 2018	53.09	51.25	32.25	49.67				

8 FUTURE WORK

We can see a number of areas for future work: (1) We can augment the mechanisms available for detecting plagiarism. While MOSS is an excellent tool, it is not necessarily applicable to all programming classes. Evaluating new mechanisms for detecting code plagiarism will be useful, and this is an active area of research. [7, 8, 17, 18, 25, 28, 32, 38, 39, 55, 58] (2) We can work to identify new ways to modify the behavior of students to achieve our goals of further improving academic honesty. (3) we can improve our own analysis of the output from our existing tools. Our work on this project has already encouraged us to develop a number of tools we can use to ease the evaluation of student code in the future, and we suspect there is further work to do in this area, particularly with an eye toward simplifying the process for others.

We realize that future classes will now be aware of the techniques that the course instructors use to detect plagiarism. As a result, course instructors will need to find ways to further improve their detection abilities. At some point, the work required to circumvent these checks becomes greater than the effort of simply doing the assignment — and hopefully also achieves the instructors' pedagogical goal. [22]

In §6.3 we noted the surprising number of unique students identified by the per-semester analysis that *were not* identified in the cross-semester analysis. We suggest further research into this surprising finding may provide additional insight into the patterns of plagiarism in classes, as we suspect this suggests **collaboration** may be a significant source of academic misconduct. Finally, we note that we excluded all same-semester matches from the cross-semester analysis; it may be sufficient to *only* perform cross-semester analysis because presumably it should also identify same-semester matches — provided that we do not exclude them.

9 CONCLUSION

Although prior work demonstrated that honor codes alone are not an effective mechanism for reducing code plagiarism, we have demonstrated in this work that explaining and reinforcing lessons in academic honesty resulted in statistically significant decreases ($p < 0.05$) in plagiarism rates across seven of eight distinctive programming submissions in a large online graduate computer science course. In addition, this technique required minimal effort from the instructors — in contrast to a strict enforcement policy, which substantially increases the burden on the instructional staff.

REFERENCES

[1] Lee Adam, Vivienne Anderson, and Rachel Spronken-Smith. 2017. "It's not fair": policy discourses and students' understandings of plagiarism in a New Zealand university. *Higher Education* 74, 1 (2017), 17–32.

[2] Alex Aiken. 1994. MOSS: A System for Detecting Software Similarity. (1994). http://theory.stanford.edu/~aiken/moss/ (Accessed January 6, 2018).

[3] Giora Alexandron, José A Ruipérez-Valiente, Zhongzhou Chen, Pedro J Muñoz-Merino, and David E Pritchard. 2017. Copying@ Scale: using harvesting accounts for collecting correct answers in a MOOC. *Computers & Education* 108 (2017), 96–114.

[4] Ruth Baker-Gardner and Cherry-Ann Smart. 2017. Ignorance or Intent?: A Case Study of Plagiarism in Higher Education among LIS Students in the Caribbean. In *Handbook of Research on Academic Misconduct in Higher Education*. IGI Global, 182–205.

[5] Rashad A Bennett. 2017. *Online Academic Integrity: An Examination of MBA Students' Behavioral Intent of Engaging in Plagiarism*. Ph.D. Dissertation. University of Phoenix.

[6] Russell Boyatt, Mike Joy, Claire Rocks, and Jane Sinclair. 2014. What (Use) is a MOOC?. In *The 2nd international workshop on learning technology for education in cloud*. Springer, 133–145.

[7] 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), Washington, DC*.

[8] Aylin Caliskan-Islam, Fabian Yamaguchi, Edwin Dauber, Richard Harang, Konrad Rieck, Rachel Greenstadt, and Arvind Narayanan. 2015. When coding style survives compilation: De-anonymizing programmers from executable binaries. *arXiv preprint arXiv:1512.08546* (2015).

[9] Shih-Chieh Chien. 2017. Taiwanese College Students? Perceptions of Plagiarism: Cultural and Educational Considerations. *Ethics & Behavior* 27, 2 (2017), 118–139.

[10] Henry Corrigan-Gibbs, Nakull Gupta, Curtis Northcutt, Edward Cutrell, and William Thies. 2015. Deterring cheating in online environments. *ACM Transactions on Computer-Human Interaction (TOCHI)* 22, 6 (2015), 28.

[11] Henry Corrigan-Gibbs, Nakull Gupta, Curtis Northcutt, Edward Cutrell, and William Thies. 2015. Measuring and maximizing the effectiveness of honor codes in online courses. In *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*. ACM, 223–228.

[12] Georgina Cosma, Mike Joy, Jane Sinclair, Margarita Andreou, Dongyong Zhang, Beverley Cook, and Russell Boyatt. 2017. Perceptual comparison of source-code plagiarism within students from UK, China, and South Cyprus higher education institutions. *ACM Transactions on Computing Education (TOCE)* 17, 2 (2017), 8.

[13] Victoria L Crittenden, Richard C Hanna, and Robert A Peterson. 2009. The cheating culture: A global societal phenomenon. *Business Horizons* 52, 4 (2009), 337–346.

[14] Guy J Curtis, Bethanie Gouldthorp, Emma F Thomas, Geraldine M O'Brien, and Helen M Correia. 2013. Online academic-integrity mastery training may improve students' awareness of, and attitudes toward, plagiarism. *Psychology Learning & Teaching* 12, 3 (2013), 282–289.

[15] Cesur Dagli. 2017. *Relationships of first principles of instruction and student mastery: A MOOC on how to recognize plagiarism*. Ph.D. Dissertation. Indiana University.

[16] Charlie Daly and Jane Horgan. 2005. Patterns of plagiarism. In *ACM SIGCSE Bulletin*, Vol. 37. ACM, 383–387.

[17] Tomche Delev and Dejan Gjorgjevikj. 2017. Comparison of string matching based algorithms for plagiarism detection of source code. (2017).

[18] Xuliang Duan, Mantao Wang, and Jiong Mu. 2017. A Plagiarism Detection Algorithm based on Extended Winnowing. In *MATEC Web of Conferences*, Vol. 128. EDP Sciences, 02019.

[19] Sarah E Eaton, Melanie Guglielmin, and Benedict Otoo. 2017. Plagiarism: Moving from punitive to pro-active approaches. (2017).

[20] Helen Ewing, Ade Anast, and Tamara Roehling. 2016. Addressing plagiarism in online programmes at a health sciences university: a case study. *Assessment & Evaluation in Higher Education* 41, 4 (2016), 575–585.

[21] Magda Fonte, Pedro Cabral, Ana Pedro, João Pie-dade, and Abel Silva. 2017. Learning scenarios in the initial teacher education: designing a MOOC. (16 November 2017). http://ceur-ws.org/Vol-1993/5.pdf

[22] genchang1234. 2015. genchang1234/How-to-cheat-in-computer-science-101. (Sep 2015). https://github.com/genchang1234/How-to-cheat-in-computer-science-101 (accessed January 6, 2018).

[23] Inc. GitHub. 2018. Guide to Submitting a DMCA Takedown Notice. (2018). https://help.github.com/articles/guide-to-submitting-a-dmca-takedown-notice/ (Accessed January 20, 2018).

[24] Therese C Grijalva, Clifford Nowell, and Joe Kerkvliet. 2006. Academic honesty and online courses. *College Student Journal* 40, 1 (2006).

[25] Daniël Heres. 2017. *Source Code Plagiarism Detection using Machine Learning*. Master's thesis.

[26] Stack Exchange Inc. 2018. Stack Overflow - Where Developers Learn, Share, & Build Careers. (2018). stackoverflow.com (Accessed January 21, 2018).

[27] Aylin Caliskan Islam, Fabian Yamaguchi, Edwin Dauber, Richard E. Harang, Konrad Rieck, Rachel Greenstadt, and Arvind Narayanan. 2015. When Coding Style Survives Compilation: De-anonymizing Programmers from Executable Binaries. *CoRR* abs/1512.08546 (2015). arXiv:1512.08546 http://arxiv.org/abs/1512.08546

[28] Hasan M Jamil. 2017. Automated personalized assessment of computational thinking MOOC assignments. In *Advanced Learning Technologies (ICALT), 2017 IEEE 17th International Conference on*. IEEE, 261–263.

[29] Christine L Jocoy and David DiBiase. 2006. Plagiarism by adult learners online: A case study in detection and remediation. *The International Review of Research in Open and Distributed Learning* 7, 1 (2006).

[30] David A Joyner, Ashok K Goel, and Charles Isbell. 2016. The Unexpected Pedagogical Benefits of Making Higher Education Accessible. In *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*. ACM, 117–120.

[31] Brian W Kernighan and Dennis M Ritchie. 2006. *The C programming language*.

[32] Hiroshi Kikuchi, Takaaki Goto, Mitsuo Wakatsuki, and Tetsuro Nishino. 2015. A source code plagiarism detecting method using sequence alignment with abstract syntax tree elements. *International Journal of Software Innovation (IJSI)* 3, 3 (2015), 41–56.

[33] Lisa M. Krieger. 2016. Stanford finds cheating âĂŤ especially among computer science students âĂŤ on the rise. (Aug 2016). https://tinyurl.com/yan2sull (accessed January 6, 2017) originally published February 6, 2010.

[34] Bruce Macfarlane, Jingjing Zhang, and Annie Pun. 2014. Academic integrity: a review of the literature. *Studies in Higher Education* 39, 2 (2014), 339–358.

[35] Hjalti Magnussion. 2015. MOSSUM: A tool for summarizing results from MOSS. (15 December 2015). https://github.com/hjalti/mossum (Accessed January 21, 2018).

[36] Leah Marks, Sarah Meek, Camille Huser, and Louise Blakemore. 2016. Playing the Numbers Game: Students as Assessors in a MOOC Context. (2016).

[37] David F Mastin, Jennifer Peszka, and Deborah R Lilly. 2009. Online academic integrity. *Teaching of Psychology* 36, 3 (2009), 174–178.

[38] Xiaozhu Meng. 2016. Fine-grained binary code authorship identification. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 1097–1099.

[39] Xiaozhu Meng, Barton P Miller, and Kwang-Sung Jun. 2017. Identifying multiple authors in a binary program. In *European Symposium on Research in Computer Security*. Springer, 286–304.

[40] Holmes Miss and J Evelyn. 2017. Development and Leadership of a Faculty-led Academic Integrity Education Program at an Ontario College. (2017).

[41] Hannah Natanson. 2017. More than 60 Fall CS50 Enrollees Faced Academic Dishonesty Charges. (3 May 2017). http://www.thecrimson.com/article/2017/5/3/cs50-cheating-cases-2017/ (Accessed January 6, 2018).

[42] Erik Nilsson. 2012. *Abstract Syntax Tree Analysis for Plagiarism Detection*. Master's thesis.

[43] University of Ontario. 2016. Why is Academic Integrity and Honesty Important? (10 September 2016). https://secure.apa.uoit.ca/academic_integrity/module1/Module13.html

[44] Inc. Piazza. 2017. Why Piazza Works. (2017). https://piazza.com/product/overview Accessed December 31, 2017.

[45] Zulfany Erlisa Rasjid, Benfano Soewito, Gunawan Witjaksono, and Edi Abdurachman. 2017. A review of collisions in cryptographic hash function used in digital forensic tools. *Procedia Computer Science* 116 (2017), 381–392.

[46] Saul Schleimer, Daniel S Wilkerson, and Alex Aiken. 2003. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. ACM, 76–85.

[47] Gaurav Shah. 2012. JADE based Virtual Checker to Avoid Plagiarism in MOOC's. *International Journal of Computer Applications* 60, 16 (2012).

[48] Heather B Shapiro, Clara H Lee, Noelle E Wyman Roth, Kun Li, Mine Çetinkaya-Rundel, and Dorian A Canelas. 2017. Understanding the massive open online course (MOOC) student experience: An examination of attitudes, motivations, and barriers. *Computers & Education* 110 (2017), 35–50.

[49] Safwan Shatnawi, Mohamad Medhat Gaber, and Mihaela Cocea. 2014. Text stream mining for Massive Open Online Courses: review and perspectives. *Systems Science & Control Engineering: An Open Access Journal* 2, 1 (2014), 664–676.

[50] Dana Sheahen and David Joyner. 2016. TAPS: A MOSS Extension for Detecting Software Plagiarism at Scale. In *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*. ACM, 285–288.

[51] Raghu Naath Singh and David Hurley. 2017. The Effectiveness of Teaching and Learning Process in Online Education as Perceived by University Faculty and Instructional Technology Professionals. *Journal of Teaching and Learning with Technology* 6, 1 (2017), 65–75.

[52] Alison Smedley, Tonia Crawford, and Linda Cloete. 2015. An intervention aimed at reducing plagiarism in undergraduate nursing students. *Nurse education in practice* 15, 3 (2015), 168–173.

[53] Sami W Tabsh, Hany A El-Kadi, and A Abdelfatah. 2015. Past and present engineering studentsâĂŹ views on academic dishonesty at a middle-eastern

university. *International Journal of Engineering Education* 31, 5 (2015), 1334–1342.

[54] Secil Tisoglu and Kadir Yucel Kaya. 2017. EXPLORING THE USE AND CREATION OF A MOOC ENVIRONMENT: A CASE STUDY. (January 2017). https://tinyurl.com/y8e8lkxw

[55] RL Victor and Farica P PUTRI. 2016. A Code Plagiarism Detection System Based on Abstract Syntax Tree and a High Level Fuzzy Petri Net. *DEStech Transactions on Materials Science and Engineering* mmme (2016).

[56] Yoav Yair. 2014. I saw you cheating. *ACM Inroads* 5, 3 (2014), 36–37.

[57] Shu Ching Yang, Feng Kuang Chiang, and Chiao Ling Huang. 2017. A comparative study of academic dishonesty among university students in Mainland China and Taiwan. *Asia Pacific Education Review* 18, 3 (2017), 385–399.

[58] Xinyu Yang, Guoai Xu, Qi Li, Yanhui Guo, and Miao Zhang. 2017. Authorship attribution of source code by using back propagation neural network based on particle swarm optimization. *PloS one* 12, 11 (2017), e0187204.

[59] Youdan Zhang. 2013. Benefiting from MOOC. In *EdMedia: World Conference on Educational Media and Technology*. Association for the Advancement of Computing in Education (AACE), 1372–1377.