Plagiarism at Scale: Identifying and Reducing Code Plagiarism in an Online Master's Program

Leave Authors Anonymous
for Submission
City, Country
e-mail address

Leave Authors Anonymous
for Submission
City, Country
e-mail address

Leave Authors Anonymous
for Submission
City, Country
e-mail address

ABSTRACT
One challenge in presenting complex technical classes is that the goals of instructors often conflict with the goals of students. As instructors, our focus is on achieving pedagogical milestones: gaining understanding of complex material and demonstrating that mastery. Students often focus on achieving the highest possible marks, which may lead them to submit work that is not their own. In this work, we examine the anti-plagiarism efforts made in an introductory course on Operating Systems (OS) in an online Master's in Computer Science program. One of the key assessments in this class is a series of programming projects that facilitate student understanding of basic OS concepts. Within this class we have explored five mechanisms for reducing plagiarism in these projects: (1) increased detection and reporting, (2) identifying and removing online code repositories; (3) improved community building, (4) improved communications with students; (5) explicitly evaluating student understanding of expectations. Over the seven semesters during which we explored these mechanisms, plagiarism rates decreased from a peak of 14% to 2%. Post hoc analysis indicated a statistically significant decrease in detected plagiarism when comparing the pre-intervention groups (M=0.0740, SD=0.0020, t=5) to the post-intervention groups (M=0.0275, SD=.0008) ($p < 0.01$).

Author Keywords
Learning@Scale, Plagiarism, ACM proceedings

INTRODUCTION
Academic honesty is a critical component of those engaged in research. When reading another researcher's work, it is imperative that the reader be able to rely upon the integrity of those results because it is the very nature of academic work to build upon the work of 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. As a student, you are part of the academic community. [33]*

Learning@Scale 2018, June 28-30, 2018, London, UK

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 123-4567-24-567/08/06. . . $15.00

DOI: http://dx.doi.org/10.475/123_4

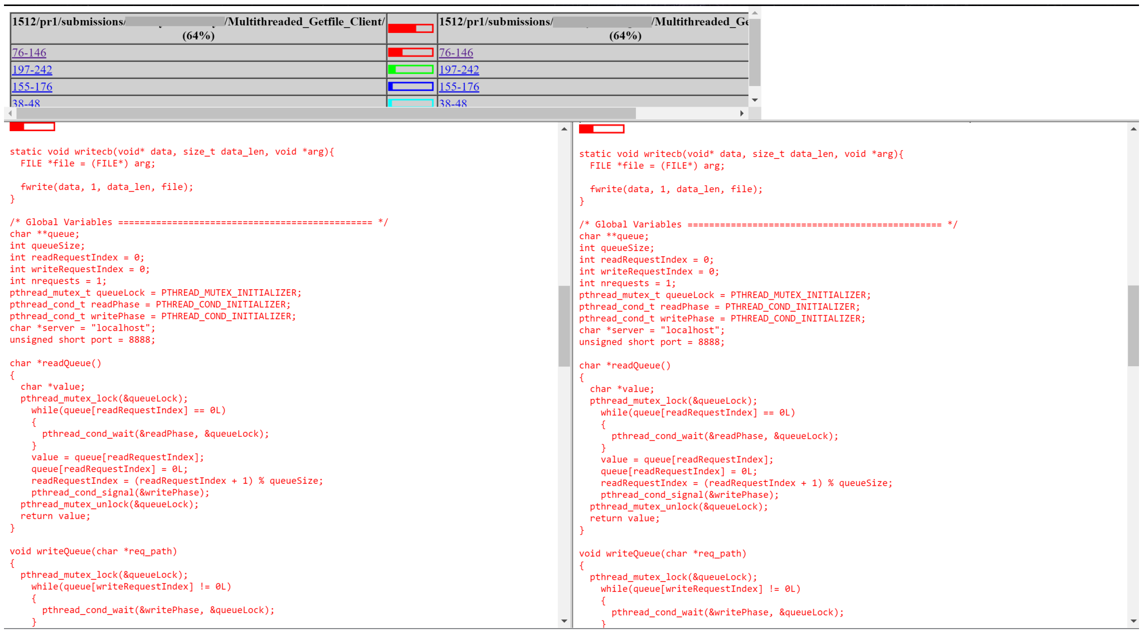
Indeed, one might argue that teaching students the importance of academic honesty is *more* important than the immediate subject of a single course, as it has broader applicability. Further, the literature observes that plagiarism is a habit that once acquired is difficult to break and insidious, as it encourages others to engage in similar behavior. [11] Thus, understanding the extent of plagiarism, the underlying drivers for it (when possible), and finding potential strategies to minimize the incidence is essential to preserving academic integrity.

Beyond the concerns of the instructors, the University is also concerned about its academic reputation and thus find monitoring and as necessary reducing plagiarism rates is inherent in its desire to maintain program integrity. While there is concern that the incidence of plagiarism is higher in online courses, prior work suggests it is comparable to that found in traditional in-person courses [21]. However, we note academic dishonesty does exist with in-person courses, including highly ranked ones, underscoring our argument that continued vigilance is necessary. [29, 32]. An additional concern for this program is that most of the students have full-time employment and the literature indicates plagiarism rates for such students may be higher than for traditional students. [31].

Our study involves the detection and prevention of plagiarism in the program's introductory course in Operating Systems. This course is taught at scale: typical class sizes are 200 or more students and the recent trend is for larger class sizes; the most recent semester of the course, which started in January 2018, had 420 students at the end of the enrollment period.

Students completing the program receive an MSCS degree. All courses are delivered online and student interaction is via online mechanisms. The average student is in their mid-30s and engaged in full-time employment, with many working in the technology industry. The rate of plagiarism that we found at the start of our research was in-line with that commonly reported in the literature — and below the level reported by some programs. [5, 25]

In this work, we contribute a systematic analysis of five intervention methods, evaluated against the same course, instructors and project specifications in different semesters. In addition, we observe a statistically significant decrease (($p <$



1512/pr1/submissions/[redacted]/Multithreaded_Getfile_Client/ (64%)	🟥	1512/pr1/submissions/[redacted]/Multithreaded_Ge (64%)
76-146	🟥	76-146
197-242	🟩	197-242
155-176	🟦	155-176
38-48	🟦	38-48

```
static void writecb(void* data, size_t data_len, void *arg){
  FILE *file = (FILE*) arg;

  fwrite(data, 1, data_len, file);
}

/* Global Variables ============================================== */
char **queue;
int queueSize;
int readRequestIndex = 0;
int writeRequestIndex = 0;
int nrequests = 1;
pthread_mutex_t queueLock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t readPhase = PTHREAD_COND_INITIALIZER;
pthread_cond_t writePhase = PTHREAD_COND_INITIALIZER;
char *server = "localhost";
unsigned short port = 8888;

char *readQueue()
{
  char *value;
  pthread_mutex_lock(&queueLock);
    while(queue[readRequestIndex] == 0L)
    {
      pthread_cond_wait(&readPhase, &queueLock);
    }
    value = queue[readRequestIndex];
    queue[readRequestIndex] = 0L;
    readRequestIndex = (readRequestIndex + 1) % queueSize;
    pthread_cond_signal(&writePhase);
  pthread_mutex_unlock(&queueLock);
  return value;
}

void writeQueue(char *req_path)
{
  pthread_mutex_lock(&queueLock);
    while(queue[writeRequestIndex] != 0L)
    {
      pthread_cond_wait(&writePhase, &queueLock);
    }
```

```
static void writecb(void* data, size_t data_len, void *arg){
  FILE *file = (FILE*) arg;

  fwrite(data, 1, data_len, file);
}

/* Global Variables ============================================== */
char **queue;
int queueSize;
int readRequestIndex = 0;
int writeRequestIndex = 0;
int nrequests = 1;
pthread_mutex_t queueLock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t readPhase = PTHREAD_COND_INITIALIZER;
pthread_cond_t writePhase = PTHREAD_COND_INITIALIZER;
char *server = "localhost";
unsigned short port = 8888;

char *readQueue()
{
  char *value;
  pthread_mutex_lock(&queueLock);
    while(queue[readRequestIndex] == 0L)
    {
      pthread_cond_wait(&readPhase, &queueLock);
    }
    value = queue[readRequestIndex];
    queue[readRequestIndex] = 0L;
    readRequestIndex = (readRequestIndex + 1) % queueSize;
    pthread_cond_signal(&writePhase);
  pthread_mutex_unlock(&queueLock);
  return value;
}

void writeQueue(char *req_path)
{
  pthread_mutex_lock(&queueLock);
    while(queue[writeRequestIndex] != 0L)
    {
      pthread_cond_wait(&writePhase, &queueLock);
    }
```

Figure 1: This demonstrates a "true positive" — two students have submitted substantially identical code. This case is counted as an instance of plagiarism.

0.01) in the plagiarism rate on one of the projects over two presentations of the class, in which the only known change was to introduce the "explain and verity" method provides support for further research in evaluating this method in other online classes and programs.

PRIOR WORK

The literature demonstrates that in-person programs at scale experience cheating and plagiarism. [42, 32, 13, 29] While we are still early in the understanding of online program dynamics, the available evidence suggests these issues are present in online programs at scale as well. [6, 36, 21, 43] While similar, we note that the mechanisms for motivation and instructional strategies for coping with these challenges at scale remain an open area of research and practice. [12]. Further, because the program provides a credential, it is important that the program ensures it is taking prudent steps to ensure its integrity, including preventing plagiarism. [18]

Online programs, such as the subject of this work, have unique challenges. They are often designed to appeal to non-traditional students, who tend to be older. [26] The literature suggests plagiarism rates may be higher for students engaged in full-time employment. [31] Thus, vigilance is well-justified for this program.

Prior work has explored root-cause analysis for plagiarism. [16] We contribute here by looking at these issues in a program at scale, over a period of several years, which yields further useful insights. Prior work has suggested moving away from a moralistic view of plagiarism and re-framing it as an integral part of the learning process. Thus, even if students do not understand what plagiarism is, educating them about it can be beneficial. [1] Our work appears to further support this position, though the optimal mode of action remains to be determined. The literature suggests a broad range of reasons for plagiarism, including cultural, economic, and perceptive reasons. [10, 4, 7] We have not explored the specific issue of cause, but do note that our successful interventions are consistent with educating students, regardless of their cultural background. We are not the first research team to report positive results in reducing plagiarism via interventions. [17, 9] Our problem domain differs from this prior work, though ultimately our findings support the idea that programs can develop "best practices" works for online programs at scale, as well as in-person programs. [8]

The need to continue evaluating educational programs to prevent academic dishonesty is clear — as a community we must remain vigilant as the problem, social norms, and techniques continue to evolve. [39]

BACKGROUND

In this work, we analyze two programming projects that are an integral part of the summative assessment of the course. As such, the outcome of these projects directly impacts student grades and students are highly motivated to obtain the best possible scores on those projects in order to earn a passing grade for the course. In some cases, students require specific grades to receive reimbursement or gain access to specific job opportunities, further creating grade pressure.

The inaugural session of this course was delivered in Spring 2015. Our analysis includes all but the first session of the course. Over the course of the more than two years being studied, we have been able to trial a number of interventions with some notable success. We report all interventions and their outcomes here.

Intervention History

Over the course of this study, we have trialed five different intervention mechanisms: (1) a "detect and report" model in which we looked for plagiarism and when found dealt with it directly with students; (2) a "search and destroy" technique of finding and eliminating public code repositories; (3) a "community building" model in which we have attempted to foster stronger social ties between class participants as a means of suppressing the impulse to cheat against fellow classmates; (4) a model of "setting expectations" that consisted primarily of explaining to people what plagiarism was and the penalties associated with it; and (5) "explain and verify", a formative assessment of students' understanding of the expectations regarding academic honesty combined with (4).

Of these five mechanisms, two were linked to a measurable decrease in plagiarism rates. For two, we were unable to demonstrate any impact. For the fifth, while we were unable to demonstrate any quantitative impact, we did collect qualitative evidence suggesting it helped decrease the plagiarism rate.

We found that "detect and report" and "explain and verify" were effective in reducing plagiarism rates. We reached a contrary conclusion for "search and destroy" and "setting expectations" as neither appeared to lead to any perceptible decrease in the plagiarism rate. For "community building" we did not demonstrate any obvious improvement, but we did collect subjective evidence suggesting that for those participating in the voluntary community building activities the plagiarism rate was low; we hope to clarify this further in future work.

With the introduction of "explain and verify" we observed a marked decrease in the plagiarism rate — 2.7% in the recently completed Fall 2017 class presentation. Further, even for cases that qualified as plagiarism there is a noticeable decrease in the magnitude of the transgression — a *qualitative* improvement not captured by the binary measurement of detected plagiarism.

Course Background

The course we evaluated for this work is an introductory course in Operating Systems delivered online and part of an Online Master's in Computer Science program. It has been offered each semester since Spring 2015; we analyze data collected between Summer 2015 through Fall 2017 (eight semesters). Lectures are delivered online. Summative analysis is conducted via two online proctored examinations. There are four projects, three of which are focused primarily on programming in the C language. The project provide a combination of formative and summative assessment: student code is "auto-graded" and pass/fail information is provided to them. After the project due date student submissions are graded via a combination of automatic and manual grading components and results are provided back to students with both a numeric score as well as feedback on their submission.

The University collects and publishes data about the courses in the program. For data from course inception through Spring 2017, 1,691 students had enrolled in the course at the end of the relevant registration period. 632 students withdrew from the course in that time frame. As a result, this course is tied for the dubious position of fourth highest for "most withdrawals" in the program by percentage (39.5%). A popular unofficial website collects anonymous student reviews. Those reviews describe it as a challenging course demanding a substantial work commitment. Thus, this course has a reputation as being challenging for students.

Evaluation Structure

This evaluation focuses on the two projects in this class that have demonstrated the presence of plagiarism. The project themselves help students learn more about fundamental operating systems concepts by writing components of a simplified HTTP-like server using the C programming language. [27] They involve learning about and utilizing network communications, multi-threading using standard operating system mechanisms, and utilizing inter-process communications mechanisms such as message queues and shared memory.

Because the programming projects are revised each semester, the instructors for the course have greater latitude to modify them. For us, this was an ideal opportunity to trial various intervention mechanisms. Beginning in Fall 2016, we began contemporaneously evaluating student submissions for evidence of plagiarism. The instructional team deployed a number of interventions and we evaluated them. Because early interventions did not demonstrate the desired improvement, we iterated through a variety of approaches.

The instructors' primary tool for evaluation of suspected plagiarism was the **Measure of Structural Similarity** (MOSS). [2] This is a standard tool used to detect plagiarism in programming assignments. [37] Despite having been in active use for more than twenty years, MOSS's basic approach continues to be employed and enhanced successfully. [35, 15]

After the project submission deadline, the instructors submit the corpus of student code to the MOSS service provided by Stanford University's Computer Science department. Within a short period of time the results are returned as a short-lived URL hosted by that service. The instructors archive the results made available at the proffered web link; our subsequent analysis has been done against that data as well as by referring to the original code submitted by students.

Plagiarism Detection Mechanisms

MOSS is not a simple text comparison utility. Instead, it goes beyond textual analysis and compares the underlying code structure to determine similarity. Thus, if a student changes the names of program components (e.g., program variables, routine names, etc.) MOSS will still detect the similarity. MOSS itself **is not** proof of plagiarism. Thus, once MOSS identified potential plagiarism, someone with domain knowledge should evaluate it manually to determine if the similarity is in fact actual plagiarism. However, by default, MOSS is conservative in its reporting, so reported cases were often in fact plagiarism. In the numbers that we have used for this work, our figures report code that has been manually inspected. In some cases we have reviewed the code more than once; our concern was that we wanted to ensure our significant improvement was not demonstrating our confirmation bias.

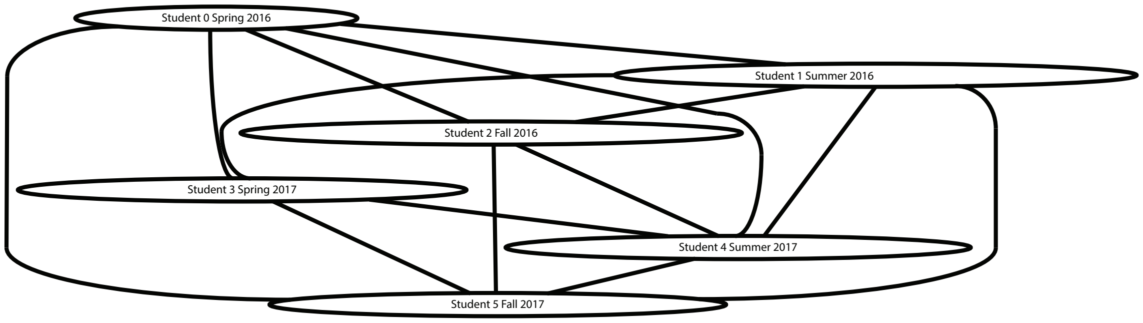


Figure 2: MOSSUM visualization of code sharing across semesters and students. Note that linkage indicates strong pairwise similarity.

In some cases MOSS will detect similarity amongst submissions because students have submitted the base starter code. We have omitted these clearly false positive reports from our analysis.

Thus we use MOSS as a front-line screening tool for suggesting that we perform further evaluation of the code. The actual process of determining something is likely plagiarized is done via manual analysis. For example, we look to various code characteristics, such as style, comments, sequence of operations, and combine that with judgment as to whether such similarity "makes sense" in the context of the programming assignment. This does add a level of subjective analysis to the process, but the instructors also have a fairly high bar here: when they report someone for plagiarism to the University, they must do so knowing that the evidence must clearly demonstrate plagiarism.

MOSSUM [30] is a tool for generating graphs from MOSS data. In MOSSUM's graphs, nodes represent individual project submissions, and edges represent significant code sharing; we used a threshold of 20% and 12 lines of code. In our analysis of the aggregate program MOSS data, which combined all semesters from Summer 2015 through Fall 2017, we could see clustering amongst student submissions, both within a single semester as well as across semesters. A visualization of one instance of this is shown in Figure 2, with this example being particularly rich, showing common links to submissions from multiple students spanning different semesters. In some cases, we have observed submissions that MOSS reports are 99% similar — a level so high that it is difficult to imagine any pair of 500 line programs written by the same programmer would be so similar. An example of this can be seen in Figure 1. Note that MOSS reports the size of the common code regions. We do not explicitly capture this qualitative analysis, though we discuss it later in this work as it serves as an insightful qualitative marker.

Sometimes, instructors find that students have, in fact, plagiarized, but have also cited to their plagiarism in the project documentation. Normal practice for the instructors is to use that as an opportunity for educating the student as to acceptable use and **why** their use is not acceptable. Note that for the purposes of our evaluation in this paper, we have counted all such cases as being incidents of plagiarism.

At our request, beginning in Fall 2016, instructors started actively watermarking the initial "starter" code given to students by the instructors. They accomplished this by changing aspects of the base code that do not impact the actual behavior of the program. For example, one approach here was to change the order of the header files in the skeleton code. They also changed default constants within the program, or moved functional elements between files. The purpose of this was to aid in providing better code provenance as well as creating additional mechanisms for finding plagiarism; the fear was that students might be figuring out how to bypass MOSS. Thus, this provided an alternative mechanism for finding suspicious code submissions.

Beginning in Fall 2017, instructors have added checksums to the code as well, so that they can detect various kinds of tampering with the scripts used to submit code (for example) or when students are using versions of the non-submitted code that are not from the current semester's code base. Again, the goal is to create further tools and mechanisms for detecting academic dishonesty. Thus far, while we have detected anomalous student submissions, investigations by instructors have only led to cases that were separately identified by MOSS or were explainable in some other fashion.

Result Observations

In reporting our results, we report manually reviewed cases for likely plagiarism. To be considered, we must observe at least 20% similarity with another work, ignoring common code elements, and at least 12 lines must be in a single segment between the two being considered. We then manually review the code and record a positive result if we would normally report this instance as suspected plagiarism.

Prior to this work, our evaluation was restricted to doing intra-term analysis of student code submissions; this was integrated into the automatic grading systems beginning in Summer 2016, making this analysis easy to conduct. In preparing this work, we analyzed data for all but the first semester of this course (for which the code submissions were not available). We also went beyond the review provided by instructors and performed a cohort analysis combining code from all semesters between Summer 2015 and Fall 2017.

Our evaluation includes:

- Census data for each presentation of the course: number of registered students at the end of course registration, and number of students that submitted each project component.
- Our estimates of suspected plagiarism from the code submissions.
- Visualizations of code sharing "flow" through the semester (based upon MOSS data).
- Cohort analysis reviewing our overall level of suspected plagiarism; this can help us identify cases of plagiarism that are not detected with our term-by-term analysis approach.

INTERVENTION 1: DETECT AND REPORT

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 systematic introduction of student submissions for plagiarism. MOSS data was generated as part of the normal automated code evaluation system, making this investigation straight-forward. We worked with the instructional team at that point, and since, to evaluate the plagiarism data for this class. Our analysis of prior semesters is retrospective.

Data analysis identifies the plagiarism rate varied between 8% and 10% for Fall 2015 through Summer 2016. There is a notable increase after Summer 2015 (5%). We hypothesize this is because the course was new and student code from prior semesters had not yet begun to be publicly available.

We discussed potential 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, such as typically they were concerned with small pieces of code they had used from online sources, such as Stack Overflow [23] but failed to note that in their submissions. None were suspected of plagiarism and their conduct did not rise to a level the instructors considered constituting plagiarism. Thus, no action was taken on any of the students that attempted to exploit the amnesty.

Based upon our suggestion, beginning in Fall 2016 the instructors began to systematically watermark the "starter code" given to students. The purpose of this watermarking was to make it easier to establish the provenance of subsequent code identified in suspected student plagiarism. This simple, yet effective mechanism greatly simplified mapping student code back to the original submitting student. It also provided a mechanism for detecting plagiarism that was not dependent on MOSS.

In Fall 2016 the instructors repeated this process: the student submissions were analyzed. Enforcement was rigorous albeit somewhat duplicitous: when meeting with students, instructors would advise them they were investigating a *different* student for potential plagiarism and they were trying to ascertain how the other student gained access to their 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. A few students simply refused to respond to inquiries; they were referred to the University for further investigation.

Note that due to the legal rights of students, the results of such investigations are not released to us or the instructors. However, Indirect evidence does suggest that several of those students no longer appeared in the student directory, suggesting they left the institution. We cannot say if this was a result of the University investigation initiated by the instructors.

In Spring 2017 the instructors once again went through this process, with similar results for the first project. One important difference was that we provided our analysis to the instructors quickly and were able to action it prior to the student submission of Project 2. The plagiarism rate for Project 2 dropped substantially — from 13.2% to 2.5% (16 students for Project 1 and 3 students for Project 2.) Interestingly all three cases for Project 3 were students that had also been involved in Project 1 and had refused to follow up with instructors and thus they'd already been referred to the University.

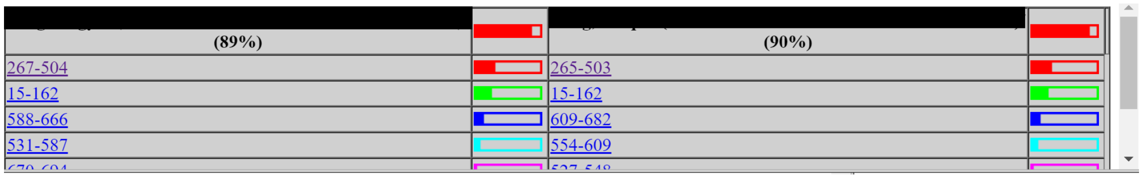
Thus, we demonstrated that a clear enforcement strategy can identify and punish students for cheating, which in turn resulted discouraged them from cheating a second time.

INTERVENTION 2: SEARCH AND DESTROY

We have observed that there is a natural flow of code from semester to semester when students make their code available publicly, whether it is 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.

Based upon our understanding that this sort of cross-semester code sharing was occurring, in Summer 2016 we suggested and instructors agreed to work with us to implement a new intervention: seeking out pubic repositories containing prior students' code and removing that code. 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 this code.

Thus began the second intervention approach: **Search and Destroy**. Whenever a public repository was identified, one of the instructors would contact the owner of that repository and request they remove it. In most cases the repository owner agreed. A common reason was to allow them to show the code to prospective employers as examples of their abilities;



[redacted] (89%)	🟥	[redacted] (90%)	🟥
267-504	🟥	265-503	🟥
15-162	🟩	15-162	🟩
588-666	🟦	609-682	🟦
531-587	□	554-609	□

Figure 3: Spring 2017 Example. Note largest matching block is at the top — 237 lines

in those cases the instructors requested the students keep them private and only share them as necessary.

What we have found is that despite these efforts, new repositories continue to appear, even as older repositories are removed. As of the date we write this paper 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. [20]

One reason we choose to describe this (largely) ineffective strategy is because the instructors continued to implement it throughout the period Summer 2016 to Fall 2017. While it is possible it had some impact, we have no evidence that it has been efficacious. Further, there is no evidence it has contributed to the more recent improvement in plagiarism rates we have identified.

INTERVENTION 3: COMMUNITY BUILDING

In Summer 2016, the instructors moved to a new code submission/grading system in which students would submit their code to an automated grader process and receive immediate feedback. As that cohort of students were working with a new system, the instructors strove to increase the interaction rate with them beyond the existing Piazza framework. A group of students in the program had established an unofficial discussion group using the interactive online forum system(`slack.com`). Surprisingly, while analyzing the plagiarism data for that semester, we noted that **none** of the students that were active in the Slack community were on the plagiarism list. This certainly was unexpected, and we noticed that this trend continued in subsequent semesters, to such an extent that when one did show up on the list it was quite a surprise to us. Subsequent investigation concluded there was insufficient evidence to support further action.

Independent of the impact on plagiarism rates, the instructors have embraced this community building channel as a further supplement to the existing community building mechanisms present in the overall program. We note that limited sense of community has been noted to be related to plagiarism. [40] The program officially utilizes Piazza [34] and instructor office hours as the primary means of communications between students and instructors. Piazza supports asynchronous conversations amongst the entire class. Instructors actively encourage students to collaborate in gaining **understanding** of the material, even though the instructors expressly prohibit them from working together on joint submissions. This is consistent with the instructors' stated pedagogical goal of facilitating understanding amongst the students of the material, yet requiring they demonstrate their mastery via a submission of their own work. Slack supplements these mechanisms by creating an interactive environment as well.

These results were so encouraging that the instructors set up a separate "workspace" on `slack.com` so they could utilize the paid tier. Students have found this sufficiently useful that they have contributed towards the cost of maintaining this community, independent of fees paid to the University for their courses. The Fall 2017 class had an initial enrollment of 300 students. At one point the online Slack community had registrations for more than 50% of the registered class and we were able to observe activity throughout the 24 hour period, reflecting the global nature of the students in the class. Indeed, we recently noted that the class online community has students in 17 different time zones.

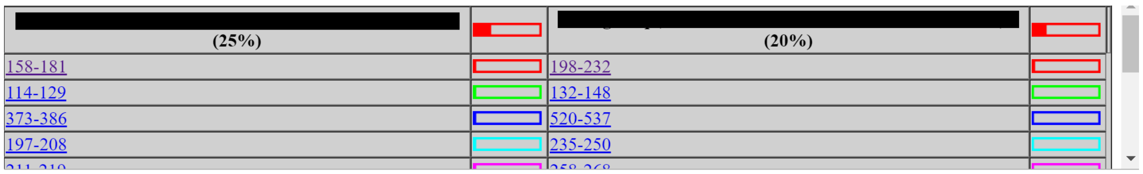
Our hope is that we will be able to analyze our anecdotal observation with respect to plagiarism now that we have access to a complete data set. However, as we will see, the success of another intervention strategy has pushed plagiarism rates sufficiently low that it may confound our ability to determine if this community building might have a positive impact as a separate intervention strategy.

INTERVENTION 4: STANDARDS COMMUNICATION

Beginning in Fall 2016, the instructors posted a public message on the class forum (Piazza). It attempted to clearly establish the distinction between collaboration and plagiarism, along with examples of each, and descriptions of expected behavior regarding citing work that the student used.

A copy of this Piazza post was sent (via e-mail) to every student in the class, which was an unusual step, but part of the instructors' intent on clearly communicating their expectations to students. The post in which this was discussed was "pinned" so that it always appeared near the top of the forum messages. It established a definition of collaboration appropriate for the course: discussions of techniques, problem solving mechanisms, concepts, and reference materials was encouraged. Similarly, it established a definition of plagiarism: using someone else's code **without** attribution. It stated that students could use small blocks of code (10 lines or less) so long as they properly cited their source for the code. Some students are well aware of plagiarism and the concerns about it; they engaged with the instructors to clarify the policy.

We note that we did not observe any meaningful decrease in the plagiarism rate for the course for Fall 2016 or Spring 2017. Thus, we concluded that this intervention did not seem to have a material impact on the plagiarism rate.



[redacted] (25%)	🟥	[redacted] (20%)	🟥
158-181	🟥	198-232	🟥
114-129	🟩	132-148	🟩
373-386	🟦	520-537	🟦
197-208	🟦	235-250	🟦

Figure 4: Fall 2017 MOSS Example: Note largest matching block is at the top — 23 lines

INTERVENTION 5: EXPLAIN AND VERIFY

In Summer 2017 we proposed a new approach to the instructors: to construct a short quiz and require students in the class take it during the first two weeks of the class. It was entitled **Project Collaboration Policy** and it consisted of two questions. The first displayed a letter from the Dean of the College and asked the student *Please confirm that you have read and understood the following message sent by the Dean*. The second referenced 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*. Note that this policy statement was the same that the instructors had posted in prior semesters as part of the prior intervention we had suggested.

When we began analyzing the plagiarism data for Summer 2017, we were surprised that we observed a *noticeable* decrease in MOSS-identified similarity. However, the Summer 2017 class was the smallest in program history, so we were concerned that we were seeing the vagaries of sampling from a (relatively) small group. Thus, while encouraging, we cautiously watched the results for Fall 2017.

At the time of the Fall 2017 class it was the **largest** class since the program inception. Thus, our prior concerns about small class size were mollified. In addition, our review of the data reflected stronger data leading to the same conclusion: the rate of plagiarism was noticeably decreased.

No other change we made in Summer 2017 or Fall 2017 could reasonably have been expected to decrease our plagiarism rate. Indeed, we have continued to add additional mechanisms for detecting academic dishonesty but despite these efforts we observed the noticeable decline that led us to reporting our results.

In addition to this noted decrease in the incidence of plagiarism, we also noted a *qualitative* difference. For example, Figure 3 shows the worst case example of plagiarism for Spring 2017 — 95% identical, with 237 matching lines in the largest matching block. Compare this with Figure 4 where the worst case example of plagiarism was 25% identical, with 23 matching lines in the largest matching block. Thus, not only is there a decrease incidence, but that decrease appears to have eliminated the most flagrant violations of the academic honesty policy.

One member of our research team suggest that one possible explanation for this decrease is that class students have become substantially better at hiding their plagiarism from us. Admittedly, we cannot exclude this possibility. However, our analysis of the watermarks and other benign indicators of code provenance did not yield any suggestions of irregularities not detected by MOSS. This was in spite of the fact we spent *more* time looking at those watermarks for the Fall 2017 data than any previous semester. While the instructors use of MOSS in the program is well-known at this point, the presence of code watermarks is something that was not previously well known.

Thus, we argue that the suggestion students have suddenly developed a new set of skills to overcome plagiarism detection techniques beyond MOSS is not a plausible explanation — we appeal to Occam's Razor in suggesting our hypothesis is a more plausible explanation of the observed behavior in the system.

Finally, we note that in the unlikely case where students have developed the level of insight necessary to circumvent the various plagiarism checks in the system, they would need to have obtained a level of domain knowledge and done so independently of other students — MOSS will detect collaborative plagiarism. If each student has gained sufficient domain knowledge to modify their code, then they have learned something, which is in keeping with the larger pedagogical goals of the instructors. This effect has been separately noted, particularly with respect to MOSS and programming classes. [19]

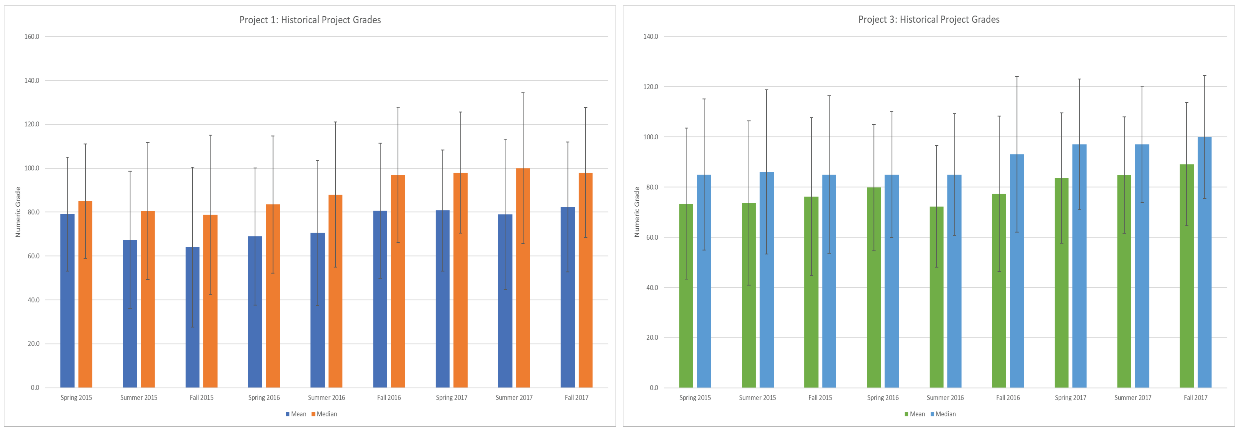
CONCLUSION

Our work on minimizing plagiarism at scale has allowed us to evaluate a number of different techniques for minimizing the impact of plagiarism within a degree-granting for-credit online course situation. We achieved this by proposing and evaluating a series of interventions that we hypothesized would achieve our goals.

We met with mixed result for two of the interventions — neither appears to have had a noticeable decrease in the plagiarism rate, though we note there was a modest decrease over time that might have been attributable to some of these changes.

One of the intervention techniques may have contributed to the decline, but we had insufficient data to support our hypothesis.

We did identify two interventions that appeared to correlate strongly with reduced plagiarism rates. One required active enforcement and thus by itself required actively confronting students on the first project. The second required up-front communications with students and no active after-the-fact confrontation. The aggregate plagiarism rate in the post-intervention classes was 4.4% for Project 1 and 2.6% for Project 2, substantially below the 11.9% rate previously detected. For project 1, this did not correspond with a statistically significant decrease between pre-intervention (M=11.95, SD=4.54) and post-intervention (M=6.45, SD=5.59) groups



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

Figure 5: Decreased plagiarism has not cause decreased student grades

(t(1) = 7.16, p=0.05). For project 2, this did correspond to a statistically significant decrease between pre-intervention (M=7.40, SD=0.02) and post-intervention (M=2.75, SD=0.01) groups (t(1)=7.16, p=0.01). In both cases a one-tail, two sample t-test with unequal variances was used.

Summary

In reviewing our results, we note that the most significant gain appears to have come from the policy of clear communications combined with the assessment of student's understanding. Further, in terms of the effort required, it was one of the lower cost intervention mechanisms to implement, consisting of a written policy and a short online quiz. We expect the instructors will continue to use this approach in future presentations of the course due to the efficacy and relative ease of implementation.

We note that simply using a written policy without any form of "forcing function" to demand student engagement did not seem to be an effective mechanism by itself.

We found that "Detect and Report" was partially effective when applied quickly enough to impact student behavior early in the semester: for Spring 2017 we observed a dramatic drop in plagiarism for Project 2. Our hypothesis is that rapid enforcement of the reporting process for Project 1 was sufficient to cause the notable drop for the later project, since we did not observe a noticeable drop for Project 2 in the previous semesters where the instructors pursued an aggressive reporting policy later in the semester. This intervention method is the most consumptive of instructor resources, requiring several hours per case to assemble the information, disseminate it to the instructional team, determine the appropriate remediation, meeting with students, and reporting it to the University authorities. In addition, we observe that student sentiment towards this approach was noticeably hostile in the exchanges on the Piazza forum, as well as some of the comments left in the unofficial public course reviews.

The "Search and Destroy" approach is also resource intensive, as it requires finding the resources. Some are hosted with commercial sites that specialize in disseminating this type of information for a fee. From what we can tell, this becomes a cat-and-mouse game, in which new repositories appear almost as fast as they are removed. From what we can tell, the only way this approach can be successful is if all of the easily located public repositories are removed.

One question that arose during our analysis was whether or not the elimination of student plagiarism has materially impacted the course grades. According to our analysis (Figure 5, this is not the case and grades do not appear to be negatively affected by the increased plagiarism detection techniques employed by the class.

Discussion

As we noted, we have not definitively demonstrated that our interventions are the cause for the decrease observed in plagiarism rates. For example, it is possible that the students have become substantially better at hiding their plagiarism. One obvious mechanism would be to withdraw the interventions and see if the plagiarism rate increases as a result. The challenge with this approach would be in convincing the instructional staff to remove these interventions — their goal is class and program integrity and they are satisfied with the current results decreasing the detected plagiarism rate to below 3%. Another approach would be to identify other classes that might benefit from trialing these interventions and determine if they are successful in other settings.

There are challenges in attempting to intervene and analyze any complex undertaking; this class has been no exception to the rule. While our deployment of interventions was spread out over a period of more than two years, we did not set out with a clear plan of deployment uniformly over that time period. In future work, we suggest more clearly establishing the intervention goals up front, which will yield stronger evidence of the efficacy of specific interventions.

We also note that we have not considered more insidious types of academic dishonesty. Fortunately, some, such as multi-accounting, do not appear to apply to our course. [3] But others, such as "work for hire" services, could be another option and one that we presently have no automated way to handle.

Takeaways

Our most significant observation is that a simple policy of clearly establishing policy and aligning student interests in learning and understanding that policy seems to have successfully led to a notable decrease in detected plagiarism rates. In addition, this is a low-cost intervention technique.

While our findings did not clearly support community building, the instructional staff have actively worked towards increasing student engagement in this area; it seems to have other benefits as well, as former students in the course actively continue to engage with current classes, providing an interesting form of continuity. We posit that this may be a completely different avenue for future research unrelated to plagiarism.

Future Work

Thus, we have several possible areas in which our results can be verified in the future: (1) We can carefully track the correlation between the online community participants and the students suspected of plagiarism to see if we can disprove the null hypothesis; (2) we can continue to monitor the rate of plagiarism in the class to confirm that our findings continue to be efficacious over time; (3) we are concerned about the subjective nature of evaluating student submissions, particularly in light of the significantly diminished obviousness we saw in the most recent cases. Future work should aim to eliminate those biases.

In deciding to write this paper, we realize that future classes will be aware of the techniques that the course instructors use to detect plagiarism. The result of this is that course instructors will need to continue to find new ways to further improve their detection abilities. Of course, at some point the work required by students to circumvent these checks becomes greater than the effort of simply doing the assignment on their own – and hopefully this also achieves the instructors' pedagogical goal. [19]

In the future, we hope to automate the process of performing historical analysis. Tools for facilitating this with MOSS are available [38] and we expect to gain further insight into longer term trends by further improving our own reporting process. Additionally, there are numerous tools available for detecting plagiarized code. [24][15] [14] [41][28] [22] We have defaulted to MOSS because it is well known and freely available for use.

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. Shih-Chieh Chien. 2017. Taiwanese College Students? Perceptions of Plagiarism: Cultural and Educational Considerations. *Ethics & Behavior* 27, 2 (2017), 118–139.
8. Henry Corrigan-Gibbs, Nakull Gupta, Curtis Northcutt, Edward Cutrell, and William Thies. 2015a. Deterring cheating in online environments. *ACM Transactions on Computer-Human Interaction (TOCHI)* 22, 6 (2015), 28.
9. Henry Corrigan-Gibbs, Nakull Gupta, Curtis Northcutt, Edward Cutrell, and William Thies. 2015b. 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.
10. 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.
11. 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.
12. Cesur Dagli. 2017. *Relationships of first principles of instruction and student mastery: A MOOC on how to recognize plagiarism*. Ph.D. Dissertation. Indiana University.
13. Charlie Daly and Jane Horgan. 2005. Patterns of plagiarism. In *ACM SIGCSE Bulletin*, Vol. 37. ACM, 383–387.
14. Tomche Delev and Dejan Gjorgjevikj. 2017. Comparison of string matching based algorithms for plagiarism detection of source code. (2017).
15. 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.

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

17. 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.

18. 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`

19. 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).

20. 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).

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

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

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

24. 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.

25. 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).

26. 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.

27. Brian W Kernighan and Dennis M Ritchie. 2006. *The C programming language*.

28. 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.

29. 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.

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

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

32. 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).

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

34. Inc. Piazza. 2017. Why Piazza Works. (2017). `https://piazza.com/product/overview` Accessed December 31, 2017.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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`

41. 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).

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

43. 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.