An Anti-Plagiarism Add-on For WebCAT

Lawan Thamsuhang Subba

lawansubba@gmail.com

Department Of Computer Science

Maynooth University

Co.Kildare, Ireland

Dr Aidan Mooney and Hugh Maher

amooney@cs.nuim.ie, hugh.n.maher@gmail.com

Department Of Computer Science

Maynooth University

Co.Kildare, Ireland

Abstract

Plagiarism is a major problem in every discipline and Computer Science courses are no different. It is very common for students to submit their peers programming assignment as their own. This practice is unfair and also halts the learning process of the students who choose to copy. This research investigates the performance of various software plagiarism detection tools such as MOSS, JPlag and Plaggie. Controlled changes were made to a code file, and the sensitiveness of the various tools to those changes was determined. Plaggie with its algorithm of tokenisation followed by string comparison was found to have acceptable performance for our tests. It is also open source whereas the other tools are proprietary and web based and we decided to incorporate Plaggie with Web-CAT.

Web-CAT is a flexible, automated grading system designed to process computer programming assignments. It serves as a learning environment for software testing tasks and helps automatically assess student assignments. The developed system was tested using submissions to a real class assignment and also by a variety of potential future users in a number of tests and the feedback received was very positive. In addition the tests presented a number of possible future enhancements.

Keywords

Plagiarism, Computer Science, Automated Plagiarism Detection, Plaggie, Web-CAT, Web-CAT Add-on

1. Introduction and Motivation

In academic environments, students are required to put substantial effort into understanding the course material. Academics then evaluate the students by assessing assignments given as part of the course and examinations. However, there are some students who decide to plagiarise rather than put in the required work. According to Higgins et al. (2006) plagiarism is the attempt to pass off other peoples' work as one's own. This work can be ideas, data, graphs, images, essays, publications, computer programs to name a few. When students plagiarise, they cannot be assessed correctly as they will have formed a deceitful impression of their understanding. It is also unfair to hard working students who spend time and effort on the course material. Then there is also the issue that the original author will not be given due credit. In general if instances of plagiarism are common, the academic integrity of the entire program will be at risk. Therefore plagiarism is considered a serious offense in academic environments.

All institutions have their own guidelines for dealing with instances of plagiarism, but the penalty normally depends on factors like the students reluctance to admit the offense, the extent to which the student has gone to disguise his/her actions and the institutional policy. With the rise in popularity of the Internet it has become easier for students to find content online to cheat, consequently making it difficult for teachers to check and detect plagiarism. This has led to the rise of automated plagiarism detection tools such as Turnitin (2014), which compares the student's submission with a repository of already submitted documents, online documents, and documents from various journals and publications.

In Computer Science, like in other disciplines, it is integral that students not only understand the presented material but are also able to apply what they have learned in a practical setting. Therefore, most computer science courses expect students to submit programs as part of their laboratory works and assignments. These assignment submissions may account for some portion of the overall marks of the course. At Maynooth Univeristy, computer modules normally run over a period of 12 weeks and include 10 lab sessions. Depending on the module, assignments range from simple programming or algorithmic problems to complex data structure and implementation problems. Generally, students are required to take 60 credits worth of modules across

the university and each module normally accounts for five credits. During each assignment, students may pose questions and ask assistance from lab instructors and the lecturer. Students are encouraged to discuss the assignments with each other informally without any guidelines, as discussions form an integral part of the learning process; however, their submissions must entirely be their own.

Nevertheless, plagiarism is prevalent in computer science, where students use the copy/paste method, sources from the Internet and even make alterations to disguise other peoples programs. Furthermore the large number of students and files involved in computer science courses make it difficult for instructors to check for plagiarism rigorously. For example, at Maynooth University there are regularly over 450 students in first year computer science courses. When researchers at the Dublin City University used an automated plagiarism detection tool on a programming class they discovered that more than 50 percent of the students were involved in plagiarism (Daly and Horgan, 2005). Consequently students who did not cheat did much better in end of semester examinations than students who did cheat making it very important to detect and tackle the reasons for plagiarism. There are a number of automated plagiarism detection tools available for computer programs, each having its own strengths, weaknesses and associated problems (Hage, Rademaker & van Vugt, 2011).

We were interested in learning the perspectives of the Computer Science faculty at Maynooth University on plagiarism. Therefore we created an online questionnaire, where 10 lecturers provided feedback and based on this we were able to determine that Java was the most commonly used programming language in the Computer Science department. It was also found that the majority of respondents use a non-automated hand to eye approach for identifying plagiarism in their assignment submissions. Also the majority of the respondents believe that plagiarism is only a minor problem in their classes but all the respondents expressed an interest in using an automated plagiarism detection system.

One of the primary goals of computer science has always been to ensure that students are able to write correct programs. One of the ways to do this is by following test driven development (TDD), where students are expected to test their own programs. This process forces them to think first before writing code. Web-CAT, an open-source automated grading system champions the usage of TDD. It is a web application

tailored specifically for programming assignments. It forces students to test their own programs and after submission grades them depending on the quality of their tests. As tutor assessment and feedback is an integral part of the learning process Web-CAT allows instructors to provide prompt feedback to students after they submit their assignments. Web-CAT won the 2006 premier award, recognising high-quality, non-commercial course-ware for engineering education (Edwards & Perez-Quinones, 2008). Various institutions are using Web-CAT at the moment including the University of Pisa (Italy), Virginia Tech (United States of America) and the University of the Basque Country (Spain).

Although Web-CAT processes hundreds of student's assignments it does not have the feature to check students' submissions for plagiarism. Therefore, this papers aims to study the various existing plagiarism detection tools and develop a plagiarism detection add-on which can be ported to Web-CAT. We believe that institutions already using Web-CAT will greatly benefit with this feature and tutors from the University of Basque Country and Lynbrook High School have shown an interest in using our add-on.

2. Literature Review

In any academic environment, tutor assessment and feedback is a fundamental component of the learning process. It provides instructors a means to guide the learning process and also returns valuable information on learning about the whole course and about individual students on subjects being taught (Ihantola, Ahoniemi, Karavirta, and Seppälä, 2010). However, the disproportionate ratio of students to teachers in today's programming classes means that assessment and feedback is not always timely and meaningful. These problems can be reduced by using automated assessment tools. These tools will run the students programs, compare the results with a reference solution and provide immediate feedback to the students. A study undertaken by Ihantola et al. (2010) discuss the various different automated assessment tools available and their features and identified Web-CAT as a good candidate for use in computer science.

In order to detect plagiarism in computer programs, early systems used an attribute counting mechanism (Ottenstein, 1976) where various attributes of programs were

counted and these attributes were compared to detect instances of plagiarism. Later structure metric systems like MOSS (Aiken, 2005) and JPlag (Prechelt, Malpohl and Philippsen, 2002) were developed, which compared programs according to their structure. Whale (1990) concluded that attribute counting mechanisms alone are not sufficient for plagiarism detection and that structure based approaches were shown to have better detection capabilities.

There are many plagiarism detection tools available for computer programs and two of the most peer reviewed ones are JPlag and MOSS. JPlag was developed by Prechelt, Malpohl and Phlippsen at the University of Karlsruhe (Prechelt et al., 2002). It uses a structure based approach to compare programs and check for plagiarism. MOSS was developed in 1994 by Aiken (Aiken, 2005) at Stanford University and it uses a document fingerprinting approach to compare programs and check for plagiarism. Both of these tools are proprietary but are publicly available for use as a web service. In contrast, Plaggie which was developed by Ahtiainen, Surakka and Rahikainen (2006) uses a detection approach similar to JPlag but it is open source and can be installed locally.

Lancaster and Culwin (2004) compared eleven different source code detection engines and recommended the usage of JPlag and MOSS. In addition, a study by Hage, Rademaker and van Vugt (2011) found that the performance of JPlag and MOSS were similar. Both of these tools are proprietary and require the user to send files to their servers. Due to the sensitive nature of plagiarism we decided that such an option would not be appropriate. Therefore we decided to integrate Plaggie with Web-CAT and provide plagiarism detection functionality as an add-on.

3. Implementation

Both Web-CAT and Plaggie are open-source applications and are covered by the GNU General Public License (GPL). This license allows people to study, modify and share these applications but the GPL also enforces software to be copyleft. Copyleft (Heffan, 1997) means every modified or extended version of the software will also retain the license, thus making the code and license legally inseparable.

Web-CAT supports three different types of users: administrator, instructor and student. The administrators are the super users of the system, enabling them to create

courses, add students and add instructors in addition to assigning instructors and students to courses. The instructors can then create assignments and set up assignment processing rules. Finally the students will see the call for assignment and submit their work. We added plagiarism detection functionality into Web-CAT to allow the administrator and instructors to execute Plaggie on the assignments submissions. They can then view the results and indentify students who are involved in plagiarism.

Plaggie works by converting each program to its representation in token format. These tokens are in fact words, phrases or any other element that holds meaningful value by breaking up a stream of text by following a set of rules. Each program is compared to every other program, resulting in (n*(n-1)/2) comparisons, where n is the number of total programs. For example, if there are 5 programs the tool will need to perform ((5*(5-1))/2) = 10 comparisons to check for plagiarism. Finally, the similarity values of programs are calculated in percentage and the results are published as a list. Prechelt et al. (2002) suggest that a threshold of 50 percent is sufficient to give good results.

Student A	Student B	Similarity
Bob	Alice	100%
Bob	Robin	90%
Joe	John	80%

Table 3.1 – Sample Results Of Plaggie

Table 3.1 shows a sample representation of the results produced by Plaggie. The inherent problem with this manner of representing results is that the list will grow exponentially with the number of students, thereby making it difficult for instructors to analyse results and increasing the risk that some cheating students may be missed.

We therefore decided to modify Plaggie and change the manner in which it shows results. Firstly we sort the results so that the instructor is able to see every other student that shares similarity with a single student, for example:

Bob: Alice, Robin and

Joe: John.

Next we show groups of students who share similarity with each other, for example:

Group1: Bob, Alice, Robin and

Group2: Joe, John.

We believe that these changes will greatly assist the instructor and avoid any cheaters being missed. As Web-CAT is deployed with a subsystem based architecture, any other administrator of Web-CAT can use our changes and integrate plagiarism detection functionality with their instance of Web-CAT.

4. Participants and setting

In order to determine the performance of Plaggie in terms of plagiarism detection we tested it with real world assignment submissions. For this purpose we selected a second year computer science course *Algorithms & Data Structures 2 (CS211)* at Maynooth University. This course comprised of students from five different streams

- MH140 - Bsc in Computer Science Software Engineering (Arts)
- MH203 - Bsc in Computer Science and Software Engineering
- MH211 - Bsc in Multimedia, Mobile Web Development
- MHG54 - Higher Diploma In Information Technology
- MH214 - Bsc in Computational Thinking

The course required students to submit a final assignment to solve a travelling salesman problem. The assignment required students to submit a Java program to find the shortest route that visited all towns in a closed loop, given the GPS co-ordinates of 80 towns in Ireland. Students were warned not to cheat and that their submissions would be checked for plagiarism using an automated detection tool. The students submitted the assignment in two groups, MH214 and MHG54 in the first group and MH211, MH203 and MH140 in the second.

5. Results

After the deadline had passed for each group to submit their assignment, the submissions were sorted and the submission details are shown in Table 5.1. In total 132 students submitted their work. If an instructor wanted to rigorously check for plagiarism s/he would have to perform 8,646 ((132*(132 - 1))/2) manual comparisons. All submissions were presented to our modified version of Plaggie along with JPlag and MOSS. The goal was to determine the effectiveness of Plaggie and compare its results with the other leading tools.

Group	Streams	Number Of Submissions
1	MH214 and MHG54	59
2	MH211, MH203 and MH140	73
Total		132

Table 5.1 – Submission Details For CS211 Final Assignment

For Group 1, all three tools identified four pairs of students as having similar programs, as shown in Figure 5.1. When the submitted programs were investigated, it was found that students in pairs A, B and D had similar programs. However, in the case of pair C the students were placing the GPS co-ordinates in the program itself causing them to be highly similar.

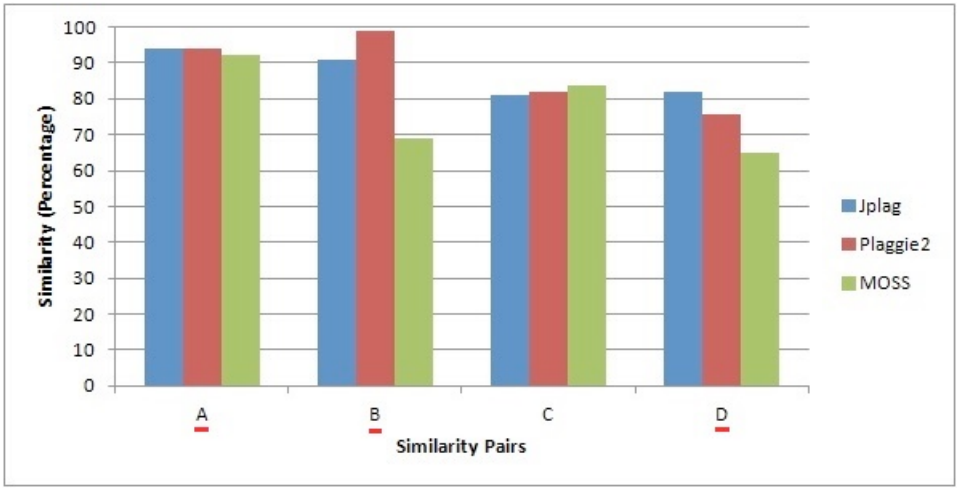

Figure 5.1 – Similarity Values for Group 1

For Group 2, all three tools again identified four pairs of students as having similar programs, as shown in Figure 5.2. Again we investigated the students programs and determined that students in pair N and O had similar programs, whereas the programs submitted for pairs M and P were shown as similar because the students had submitted small and incomplete assignments.

We then wanted to check for plagiarism between the two groups. Therefore, we ran the three plagiarism detection tools on the combined submissions of Group 1 and Group 2. However we were unable to detect any additional similar programs.

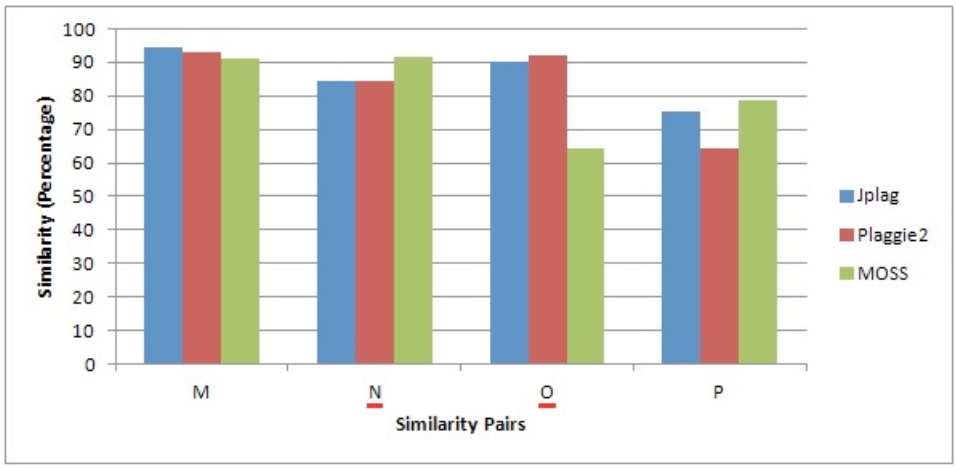

Figure 5.2 – Similarity Values for Group

Finally we gave the results of our findings to the lecturer of the CS211 course. The lecturer then asked the students in highlighted pairs to explain why their programs were flagged as being similar. Their response and subsequent actions were as follows:

- In pair A, one of the students admitted to copying from his friend and the copying student was given zero for the assignment.
- In the case of pairs B, D and N, the students involved admitted to discussing and working together but stated that they could defend each part of the program. Therefore no one in those pairs were penalised.
- In the case of pair O, the students admitted to sharing some code, but the lecturer deemed that the shared code did not contribute much to the assignment and neither of the students were penalised.

As it can be seen the automated plagiarism tool developed is not truly automated, as we had to verify each flagged pair of programs. However, instead of having to manually carry out 8,646 comparisons we had to carry out eight, which is a massive saving in terms of instructor time. In some cases, the programs were similar but could not be considered as being plagiarised. Finally as the pairs detected by Plaggie have also been flagged by the other two tools, our decision to integrate Plaggie with Web-CAT has been vindicated.

6. Feedback from faculty

In order to release our add-on for other interested users, we first wanted to determine its effectiveness and ease of use. We performed a usability study of the plagiarism detection add-on in Web-CAT with three teaching assistants and three lecturers at Maynooth University. We created scenarios where they would be using Web-CAT to support two programming classes and were required to use the add-on to find plagiarising students. By dividing the scenario into multiple tasks, we were able to determine which parts of the system the user found difficult to use.

We also performed a qualitative survey to document the users experience with the add-on. When asked if the add-on was easy to use some of the comments received were: *"It took a while to find some features but overall I found the system intuitive", "The system provides clear instructions and overall a good user interface"* and *"I found it easy to navigate and operate"*

Participants were then asked whether the results of the add-on were easy to interpret and some of the responses were: *"The system was difficult at first but it took a few tries before I got used to the different data being reported.", " I could not properly interpret the results, I think more help and demo files would help"* and *"Some of the features are difficult to grasp."*

Finally we asked them whether they would consider using this plagiarism detection tool in the future. Some of the responses were: *"It would provide great assistance to correcting, grading and giving feedback to students particularly in the early years of programming courses", "It is useful, and I may use it if the tool is improved to be*

more user friendly", *"Yes I would use it as automatic plagiarism detection can greatly enhance staff productivity"* and *"For large groups of students this would be useful. The teaching assistants can't catch it in the labs."*

In general the feedback of our system was positive. There were issues highlighted around the user interface and the display of results. Due to the complex nature of the results it may be useful to have a department wide presentation on how to use the add-on and interpret the results before any department takes it on.

7. Conclusion and Future work

We were successful in integrating plagiarism detection functionality with Web-CAT by using an open source plagiarism detection program Plaggie. After checking the performance of our modified version of Plaggie on real world assignments and comparing its results with the results of other tools, we determined that the system had acceptable performance. However, we also realised that the process is not entirely automated as instructors do have to verify the results. The feedback from members of the faculty showed promise and provided sufficient feedback for future changes.

We acknowledge the fact that a single assignment is not a large enough test bed to say that Plaggie is infallible; however it is a good starting point. Therefore in the future we would like to run Plaggie for a diverse set of programming assignments to get a better understanding of its performance. Finally, a revised version of the add-on has been provided to instructors at University of Basque Country and Lynbrook High School and we are in communication with them as they provide more suggestions on improving the add-on.

Also currently our work focuses only at aiding instructors at identifying students who are involved in plagiarism. We are yet to investigate why students cheat and whether better teaching methodologies or more effective assistance can help counter plagiarism. These issues have been kept aside for future work.

References

Ahtiainen, A., Surakka, S., & Rahikainen, M. (2006, February). Plaggie: GNU-licensed source code plagiarism detection engine for Java exercises. *In Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006* (pp. 141-142). ACM.

Aiken, A. (2005). Moss: A system for detecting software plagiarism. *University of California–Berkeley*. See www. cs. berkeley. edu/aiken/moss. html, 9.

Daly, C., & Horgan, J. (2005). A technique for detecting plagiarism in computer code. *The Computer Journal*, 48(6), 662-666.

Edwards, S. H., & Perez-Quinones, M. A. (2008, June). Web-CAT: automatically grading programming assignments. In *ACM SIGCSE Bulletin* (Vol. 40, No. 3, pp. 328-328). ACM.

Hage, J., Rademaker, P., & van Vugt, N. (2011, April). Plagiarism detection for Java: a tool comparison. In *Computer Science Education Research Conference* (pp. 33-46). Open Universiteit, Heerlen.

Heffan, I. V. (1997). Copyleft: licensing collaborative works in the digital age. *Stanford Law Review*, 1487-1521.

Higgins, M., Ohri, L., Scheirton, L., Malone, P., Dash, A. and Padilla R. (2006). Appropriate Citation Methods:How to Avoid Plagiarism. Retrieved September 22, 2014, from http://www.studymode.com/essays/Plagiarism-804610.html.

Ihantola, P., Ahoniemi, T., Karavirta, V., & Seppälä, O. (2010, October). Review of recent systems for automatic assessment of programming assignments. *In Proceedings of the 10th Koli Calling International Conference on Computing Education Research* (pp. 86-93). ACM.

Lancaster, T., & Culwin, F. (2004). A comparison of source code plagiarism detection engines. *Computer Science Education*, 14(2), 101-112.

Ottenstein, K. J. (1976). An algorithmic approach to the detection and prevention of plagiarism. *ACM Sigcse Bulletin*, 8(4), 30-41.

Prechelt, L., Malpohl, G., & Philippsen, M. (2002). Finding plagiarisms among a set of programs with JPlag. *J. UCS*, 8(11), 1016.

Turnitin. (2014). Retrieved September 21, 2014, from http://turnitin.com/

Whale, G. (1990). Identification of program similarity in large populations. *The Computer Journal*, 33(2), 140-146.