2011

Automatic Code Homework Grading Based on Concept Extraction

I. Alhami

Izzat Alsmadi
Texas A&M University-San Antonio, ialsmadi@tamusa.edu

Automatic Code Homework Grading Based on Concept Extraction

Ikdam Alhami, and Izzat Alsmadi

Faculty of Computer Science and Information Technology,
Yarmouk University, Irbid, Jordan
ikdam@yahoo.com, ialsmadi@yu.edu.jo

Abstract

E-learning is taking more roles in the current methods of education. The automatic grading and assessment play a major role in both e-learning and traditional education as a method to reduce educational expenses and relief instructors from some of the lengthy tasks such as grading. In this paper, automatic grading for software code assignments or homework is described. A tool is developed to automatically grade students' code assignments. Concepts or code from Students' answers are first parsed. Key abstractions and keywords are extracted from students' assignments and compared with typical or expected answers. Weights are given to code keywords by the instructor based on their value and importance in the overall answer. Relating this grading with code plagiarism, similarities are also measured between students' assignments and an Euclidean distance method is developed and calculated between each assignment with all other assignments.

Results showed that automatic grading for code assignments can be automated due to the nature of expected answers where grader can set and expect a fixed number of possible keywords in each answer. Such formality may not exist for several other types of essay questions.

Keywords: *E-learning, on-line exams, automatic grading, concept extraction, and exams' assessment.*

1. Introduction

Code assignments are used in many computer science departments and courses to improve students' programming skills. On weekly or periodic bases, students are asked to work on those assignments and submit them. However, there are two major problem associated with code assignments. The first problem is the large amount of effort instructors are expected to put to evaluate and grade those assignments that are submitted weakly. The second major problem is the possibility that such code assignments are plagiarized from either local or Internet resources.

Many universities around the world are giving their students the option to take some classes or courses online or off campus. The evolution of e-learning technologies comes as a natural expansion of teaching techniques where it is more convenient to give students alternatives for traditional teaching. Online exams are considered as an important asset for both traditional and e-learning teaching methods.

As part of the e-learning tools, online exams are widely used to test students' skills through using computers and network services. Those exams can be executed locally or remotely. Online exams are more convenient and flexible relative to traditional exams. They reduce the overall expenses of processing exams especially in saving papers, storage, and materials' costs. The majority of online exams are executed using the multiple choice format due to the relative easiness of its automatic grading. Grading multiple choice questions is straightforward and does not require any Artificial Intelligence (AI) or Natural Language Processing (NLP) techniques or algorithms.

However, multiple choice questions can limit the skills of students in writing and expressing. Many educators prefer to have essay questions to grade, more realistically, students' skills.

In order to expand the utilization of online exams, many researchers are studying the implementation of different types of exams, other than the typical "multiple choices" format. In this paper, we will introduce a research to evaluate or grade automatically students' coding exercises. In this scope, coding exercises are evaluated as essay questions. One way that can facilitate the automatic grading of code assignments despite the fact that they are essay questions, is that programming languages are structured and formal. For example, to define an integer, you have to write it as Integer, int, etc depending on the programming language. Any change in this word, even if it is from capital to small letter (i.e. the case of the letters), is considered as a mistake to the compiler. As such, making predictable answers is possible where any slight change in the student answer relative to the typical answer set by the instructor can be considered as a mistake. This, however, can't be applied in all essay questions where a student may write a word that is a synonym to the word written in the typical answer.

2. Related Work

There are several research papers and projects on the automatic grading of essay questions. The majority of those papers discuss some natural language processing techniques to search for keywords in the answers and try to compare them with the typical answers written by the evaluator.

Chang et al [1] made a comparison study between the different scoring methods. They also studied the different types of exams and their effect on reducing the possibility of guessing in multiple choice questions. They investigated the performance difference among examinees between partial scoring by the elimination testing and the conventional dichotomous scoring methods.

Scott et al focused on a system for web homework assignments to be graded by a central server [2]. The developed systems (i.e. WATS) proposed ways for grading other types of assignments such as multimedia, etc. In another paper, Allen et al proposed using case based reasoning for the automatic grading of questions [3]. They proposed a hybrid format that combines the advantages of multiple choice questions to essay questions in order to allow it to be automatically graded. The proposed technique mixed between human and computer grading.

Rein proposed an intelligent system to help in mathematical problems. The system is similar in concept to the programming languages' technology that is called "Intellisense technology" where software developers will be assisted through programming by showing them possible actions and mistakes while typing [4].

Mu et al discussed an approach similar to the one followed in this paper for the automatic grading of code assignments [5]. The developed tool assesses some of the issues with the code such as evaluating the performance and logical errors.

Isidoro et al [6] addressed the combined use of automatic grading and the test-driven approach from a pedagogical view using the cognitive domain of Bloom's Taxonomy. This can be useful for instructors to smooth their learning curves.

Tuomo et al [7] presented a method for evaluating automatically the content of essays written in the Finnish language based on Latent Semantic Analysis (LSA). This technique used the text or course material as a lookup dictionary for evaluating questions.

Gerosa et al described a research on the automatic assessment of reading comprehension in young children [8]. An automatic speech recognition system, especially trained for children's speech, was used for tracking the reading passage. Results show the difficulty of grading essay questions for children as usually their answers are not structured or consistent.

3. Goals and Approaches

The main title (on the first page) should begin 1 3/16 inches (7 picas) from the top edge of the page, centered, and in Times New Roman 14-point, boldface type. Capitalize the first letter of nouns, pronouns, verbs, adjectives, and adverbs; do not capitalize articles, coordinate conjunctions, or prepositions (unless the title begins with such a word). Please initially capitalize only the first word in other titles, including section titles and first, second, and third-order headings (for example, "Titles and headings" — as in these guidelines). Leave two blank lines after the title.

As described earlier, the goal of the developed tool is to be able to automatically parse, organize, and grade students' assignments. The second goal is to be able to show any possible plagiarism in each code assignment.

In this stage, we will introduce an application that was written especially to grade students exercises in a C++ introductory course (i.e. the first task of those described earlier). The goal was to find a generic and automatic way to grade those exercises through comparing students' answers to typical answers written by the instructor. The tool is developed and implemented for several semesters on students' lab assignments. The tool searches for specific keywords in the student answer and start giving weights or grades depending on the number and the type of keywords found. Figure1 shows the main startup user interface of the tool.

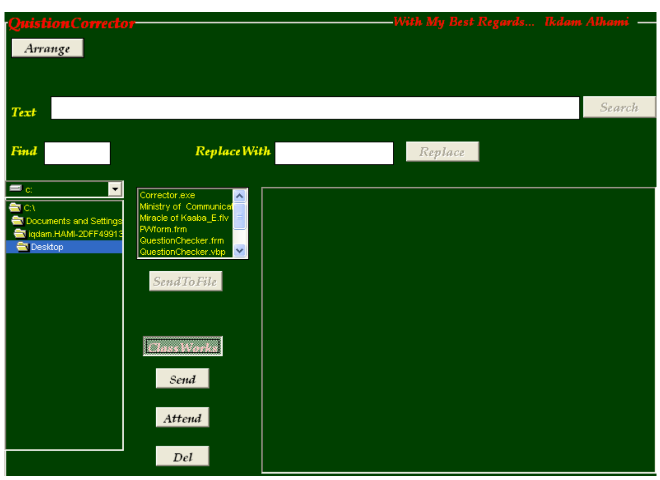

Figure1. The Code Evaluation Tool Main User Interface.

The tool collects students' files or folders according to their user name and student ID.

Figure 2 shows the collected folders from the students' assignments. The first test evaluates if the assignment can compile successfully or not. Students get their first grade on a successful compilation of their assignment.

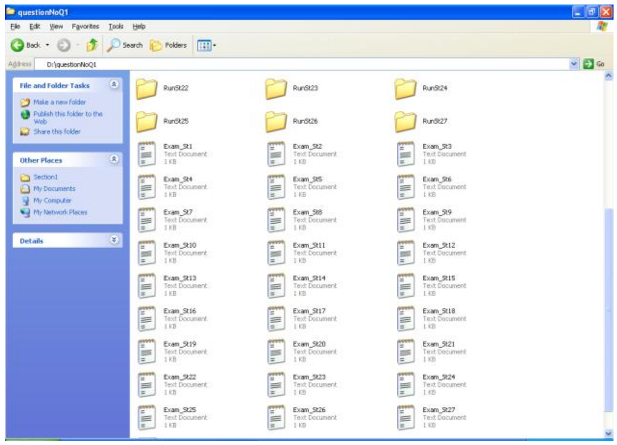

Figure 2. Students' Collected Assignments.

In the next stage, each assignment will be parsed line by line. Keywords that represent those typical answers set by the instructor are added and will be evaluated one by one. The tool will search through the assignment looking for those keywords. Each time a keyword is found, a grade is added to the student overall score. In the last step, results of each student are parsed to a Word file that includes the points they earned with detail description of where they got plus or minus points (Figure 3). The Figure shows the Word file which with the students answers and their relevant points or grades per weights.

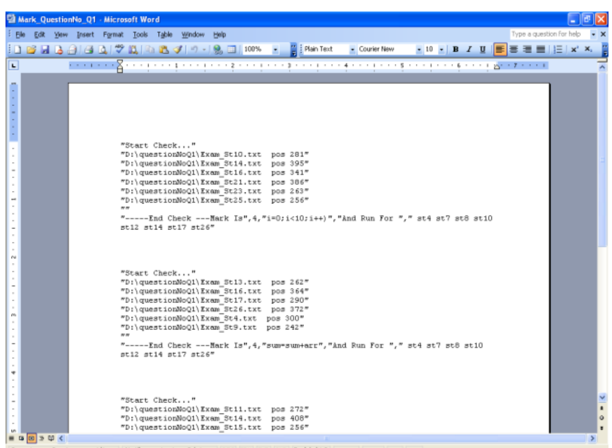

Figure 3. Code Assignment Evaluation Results.

4. A Case Study

In order to evaluate the developed tool using actual assignments, we have conducted a simple test for students of course C++ Dynamic array subject and the application direct as follows:

First question asks to establish a dynamic array and then find the following:
1 - The Most Repeated Digit In The Dynamic Array
2 - The Index Numbers Of That Most Repeated Digit. (With the assumption that the first index is 0). Figure 4 shows the screen of collected assignments.

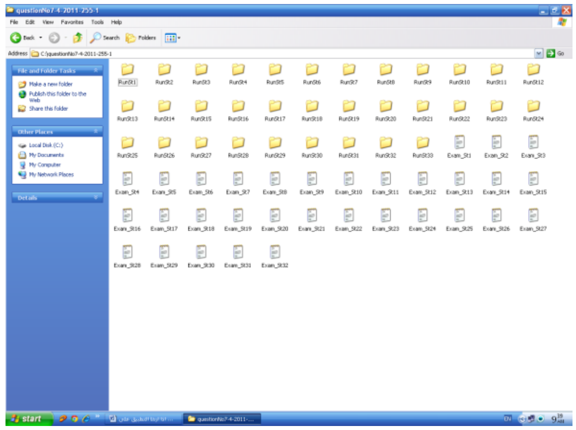

Figure 4. Collected Assignments

Figure 4 shows solution folders. Inside those folders, we will find the text files of questions and answers after performing some preprocessing analysis such as filtering spaces and empty lines, making all code lines consistent in terms of font, size, etc. Codes are also first evaluated for successful compilation. Successful compilation for each code is important in grading and the successful compilation is given a critical weight in the overall evaluation. Figure 5 shows an example of one text file of those inside solution folders.

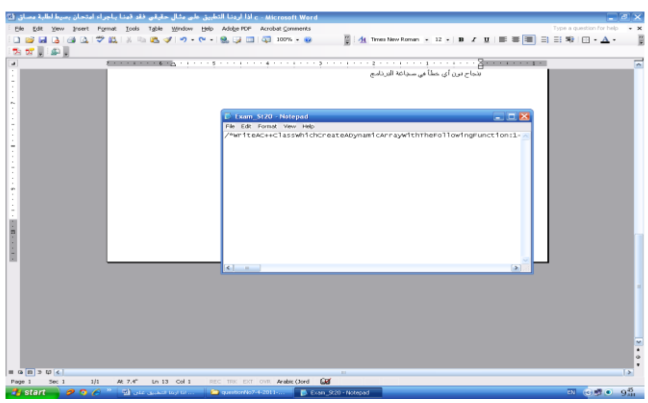

Figure 5. A Student Assignment Solution File

The text file shown in the screen is related to student No. 20 and shows the filtering process of code to facilitate the debugging process

After the gathering and filtering process and checking for successful compilation, we select the sentences are required to give the student a suitable mark for each phrase found in its proper place and that by calculating the number of characters prior to the beginning of the sentence, taking into account some variations in the number which are due to the letters number of the student name, for example, or use different style Somewhat in the programming, and we are keen to identify the sentence to be received and in the style of the solution

Figure 6 shows the screen of search results. The screen displays search results for the process of creating pointer of type "int" in the folder in which the work of students ready for the search, has been introduced numbers of students who were finding the sentence in their files to the right of the program with determining the location of this sentence; for example, student number 11 was the place of this sentence 311 and The student No. 12 in 314, this is what I meant by differences on site and thus

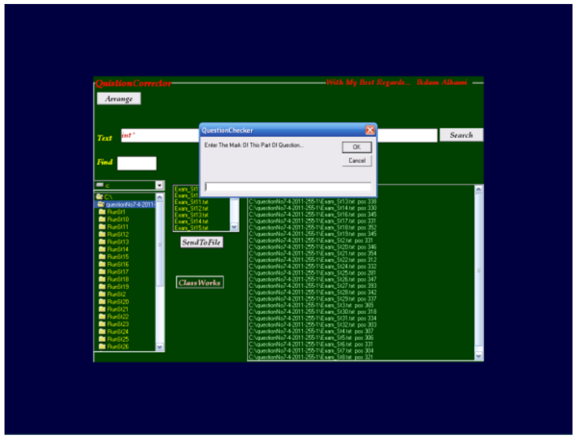

Figure 6. Search Results' Screen.

After pressing "SendToFile" button it will determine the value of the this sentence to be added for each student number appears on the list.
After that, the program creates a file with the name of the question (Figure 7), and it saves all the transactions of all queries on the first question as well as to select the students who carried out the program successfully.

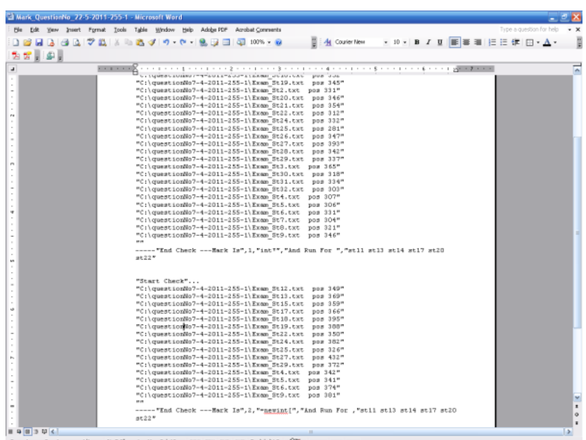

Figure 7. Transactions Results

5. Observations on the Application Process:

1 – In programming no matter of programming style or even language, certain keywords can be expected and hence preserved in the expected answer

2 – A lot of pre processing and improvement is required to make sure that the program can

eliminate unnecessary or irrelevant data along with its ability to detect correctly important program features and ignore any less important or irrelevant program features

6. Code Plagiarism

A second tool is under development for evaluating possible code plagiarism. The tool looks for possible plagiarism locally among students code. It also compare each student code and search through Google search engine to find exact matches for lines or paragraphs in the code.

A similarity index is calculated based on Euclidian distance between the subject code and all other code. This index is calculated based on the number of repeated words along with their frequency. Figure 8 shows a sample of the evaluation results of some of the code assignments. As shown in Figure 8, similarity weights index is divided into 3 levels; low (i.e. more than 25 % similarity, medium, more than 50 % similarity and high: more than 70 % similarity). Notice that due to the nature of code assignment up to 50 % similarity index is expected and may not raise plagiarism suspicion. However, this may not be true for research papers, for example.

file	more than 0.7	more than 0.5	more than 0.25
5463655	0.05	0.085	0.91
5463657	0.05	0.135	0.86
1240327	0.03	0.97	
666810	0.01	0.12	0.87
5306073	0.01	0.24	0.75
315757	0.005	0.04	0.955
870477	0.005	0.03	0.965
1235451	0.005	0.095	0.9
1487931	0.005	0.035	0.96
1568841	0.005	0.035	0.96
12139	0	0.003	0.996
12902	0	0.15	0.985
114022	0	0.217	0.782
114910	0	0.004	0.996
124818	0	0	1
138173	0	0	1
156156	0	0.015	0.985
161822	0	0.005	0.995
168140	0	0.015	0.985
168979	0	0	1
170077	0	0	1
205881	0	0.005	0.995
290812	0	0.025	0.975
292183	0	0.03	0.97

Figure 8. Similarity Index Between Different Students' Assignments

7. Study Evaluation

Results show that since programming introductory courses usually include weekly lab assignments, the process of manual grading can be time consuming and cumbersome. If keywords or typical answers were set carefully and correctly by instructors, they should be able to evaluate those assignments in a manner that is somewhat close to manual evaluation.

An experiment was implemented to study the correlation factor between students' grades that they got through the automatic grading by the tool and a manual grading by the instructor. Results indicate a high positive correlation factor between the two results. Computer grading is consistent and will help students write their code and program syntax free. Such techniques may not be valid for other types of essay questions where expressiveness is not limited as in programming assignments. Computer automatic assessment will not be able to accept spelling mistakes, synonymous words, or any answers that are not identical to those typical answers set by the instructor.

8. Conclusion and Future Work

In this paper, we proposed a simple technique to evaluate students' code assignments. A tool is built to collect student grades and evaluate them based on syntax, and keywords. Programming languages are formally structured which make it easy to grade essay questions or code assignments that are based on programming languages as examiners are supposed to write certain key words with the exact same format. Those keywords are the language constructs and data types, etc. Each correct word is given a weight that can add up to the total grading points of the question and the exam.

Instructors can such tool to include keywords that are highly relevant and can realistically distinguish a correct answer and give realistic deductions of possible corrections and mistakes.

We also proposed a code plagiarism tool to compare each student assignment with other students' assignments and/or compare the assignment with codes available in the Internet.

In future, we will use data mining techniques to evaluate the performance of the tool through the earlier course semesters. The goal is to train the tool and educate instructors of the best keywords to select that can best distinguish a correct answer from others and give proper deductions for errors or mistakes.

References

[1] Measures of Partial Knowledge and Unexpected Responses in Multiple-Choice Tests. Chang, S.-H., Lin, P.-C., & Lin, Z. C. International Forum of Educational Technology & Society (IFETS). 2007.

[2] Education research using web-based assessment systems. Scott W. Bonham, Aaron Titus, Robert J. Beichner and Larry Martin. Journal of Research on Computing in Education, 2000.

[3] Case-Based Grading: A Conceptual Introduction, Allen Briscoe-Smith, and Nicholas Evangelopoulos. ISECON 2002.

[4] Prospects of automatic assessment of step-by-step solutions in algebra, Rein Prank, 2009 Ninth IEEE International Conference on Advanced Learning Technologies.

[5] An Assessment Tool for Assembly Language Programming, Mu Lingling, Qian Xiaojie , Zhang Zhihong, Zhao Gang, Xu Ying, 2008 International Conference on Computer Science and Software Engineering.

[6] Testing-Based Automatic Grading: A Proposal from Bloom's Taxonomy, Isidoro Hernán-Losada, Cristóbal Pareja-Flores, and J. Ángel Velázquez-Iturbide. Eighth IEEE International Conference on Advanced Learning Technologies. 2008.

[7] Automatic Assessment of the Content of Essays Based on Course Materials, Tuomo Kakkonen, and Erkki Sutinen. 2004.

[8] Invistigating automatic assessment of reading comprehension in young children, M. Gerosa and S. Narayanan, ICASSP 2008.

[9] Automatic evaluation of users' short essays by using statistical and shallow natural language processing techniques, Diana P´erez Mar´ın, Advanced Studies Diploma Work, 2004.

[10] Automatic Essay Grading with Probabilistic Latent Semantic Analysis. Tuomo Kakkonen, Niko Myller, Jari Timonen, Erkki Sutinen. 2005. http://acl.ldc.upenn.edu/W/W05-/W050206.pdf

[11] A System to Assess the Semantic Content of Student Essays – Lemaire, Dessus – 2001.

[12] The debate on automated essay grading – Hearst – 2000.

[13] Automatic Assessment of the Content of Essays Based on Course Materials – Kakkonen, Sutinen – 2004.

[14] The computer moves into essay grading – Page, Petersen – 1995.

[15] On-line grading of student essays – Shermis, Mzumara, et al. – 2001.

[16] SemiAutomatic Evaluation Features in Computer-Assisted Essay Assessment, Kakkonen, Myller, et al. – 2004.

[17] A Hybrid Approach to Content Analysis for Automatic Essay Grading. Carolyn P. Ros´e, Antonio Roque, Dumisizwe Bhembe, and Kurt VanLehn. LRDC, University of Pittsburgh, 3939 O'hara St., Pittsburgh, PA 15260. rosecp@pitt.edu