Research in Engineering Education Symposium &
Australasian Association for Engineering Education Conference
5 – 8 December, 2021 - Perth, WA

A Customized and Automated As Management and Marking System Student Performance in the STE

Ashkan Shokri[a], Veronica Halupka[b], Michael Crocco[b], and Valentijn Pauwels[c]
Bureau of Meteorology, Melbourne, Victoria, Australia[a], Faculty of Engineering, Monash University, Clayton, Victoria, Australia[b] Department of Civil Engineering, Clayton, Victoria, Australia[c]
Corresponding Author's Email: Valentijn.Pauwels@monash.edu

CONTEXT

A strong increase in student numbers for CIV3204, an introduction to statistics unit taught at Monash University within an undergraduate engineering course, has been accompanied by decreased performance in the final exam. Anecdotal evidence suggests that this is caused by cheating in the in-semester assignments. Based on evidence in the literature that individualized assignments result in reduced cheating, and that automated marking allows for the completion of more assignments by the students (leading to more practice and feedback), a system to generate and automatically mark individualized assignments has been developed. A closed-form solution does not exist for most questions, so existing methods such as Moodle quizzes could not be used. This paper provides an overview of the system and the very positive results of the implementation.

PURPOSE OR GOAL

The objective of this study is to improve the students' performance in the final exam for CIV320 and their learning in the unit. The hypothesis is that automatically marked individualized assignments lead to reduced cheating and the completion of more assignments, and consequently an improved performance in the final exam. A user-friendly system working through Moodle has been developed for this purpose.

APPROACH OR METHODOLOGY / METHODS

An individualized assignment was generated for each of the 11 topics in the unit, which was automatically marked. Detailed feedback was provided to the students afterwards. This level of assessment would have been impossible to achieve with manual marking. The performance of the cohort in the final exam and the Student Evaluation of Teaching and Units (SETU) are used to evaluate the system.

ACTUAL OR ANTICIPATED OUTCOMES

The system has led to very positive results. 66% of the students appreciated that the assignments were the most effective aspects of the unit. The unit received its highest overall satisfaction in eight years. The failure rate in the final exam decreased from 22% in 2019 to 11% in 2020, even though the final exam was more difficult.

CONCLUSIONS / RECOMMENDATIONS / SUMMARY

The greatest surprise from the study was that the students were very positive about the large number of assignments, and the automated marking. The students suggested to improve the python-based Graphical User Interface system, which we will replace with a website. The system improved the students' learning through more practice and feedback, evidenced by their achievement in the exam. Based on the positive outcome, we suggest that automated marking should be further developed and implemented in the STEM disciplines.

KEYWORDS

Automated marking, individualized assessment, statistics

Introduction

Marking of assignments and exams is a very labour-intensive and thus, costly, task (Vista et al., 2015) spurring decades of effort to develop automated marking systems, initially focusing on computer code. Fleming et al. (1988) compared the results of automated versus manual marking of Fortran-77 codes, concluding human markers placed undue emphasis on cosmetic (versus functional) aspects and Jackson (1996) showed that automation led to faster and more comprehensive marking. However, Cheang et al. (2003), who developed a system to automatically mark student C++ codes for a class of more than 700 students, argued that there were not only positive outcomes of accuracy and more focused assessment, but also complexity limitations and inadequate feedback.

More recently, generic code markers, like that of Blumenstein et al. (2008), and spreadsheet and database management evaluation systems of J. Kovacic and Green (2012) have been developed. Both Naude et al. (2010) and Vujosevic-Janicic et al. (2013) demonstrated high correlation between manual and automated marking using graph similarity. Partial marking of Structured Query Language codes was enabled by Chandra et al. (2015), while Kiraly et al. (2017) describe a system to automatically mark JAVA codes for Massive Open Online Courses (MOOCs). Further advances have been seen from Conejo et al. (2019), who developed an approach based on well-founded assessment theories (Classical Test Theory and Item Response Theory) and Janicic and Maric (2020) used regression verification. Ma et al. (2020) developed a machine-learning-based peer tutor recommender system, concluding that student learning improved with the automated system; mainly because students could complete more assignments in the available time. In their study of automated computer code marking, Aldriye et al. (2019) concluded that two issues persisted: the systems tend not to work on all operation systems and feedback quality was poor.

Automation of essay marking has also received significant attention. Reilly et al. (2014) compared two automatic systems to human markers for approximately 15,000 student submissions, concluding further improvements were still needed. Systems, such as that developed by Vista et al. (2015) can analyse an essay based on predefined rubrics and reduce workload by highlighting important content to the expert marker. Meanwhile, a comparison of automated and manual marking of open-ended writing assignments by Reilly et al. (2016) showed that the automated system disadvantaged non-native English speakers; evidencing human markers' tolerance for imperfect language and willingness to focus on content. Later, Zupanc and Bosnic (2017) developed improvements to such systems by incorporating additional semantic coherence and consistency attributes, demonstrating increased accuracy compared to nine existing systems.

Several attempts have made to automate the marking of assignments in the exact sciences. Stockburger (1999) developed a web-based system to automatically generate and mark homework for statistics units. A machine-learning based method to automatically mark open-ended questions in creative problem-solving was developed by Wang et al. (2008) and demonstrated high correlation with manual marks. Donnelly et al. (2015) developed a system to automatically provide tailored feedback to middle school students of thermodynamics, depending on the quality of the original essay and concluded that the guidance was more effective for students with lower prior subject knowledge. A system to automatically score students' graph construction learning was developed by Vitale et al. (2015). Barana and Marchisio (2016) strongly advocated to automate formative assessment in mathematics and science. For applications in chemistry, a web-based automated marking system is described in Munoz De La Pena et al. (2013) and a spreadsheet-based method was introduced by Carberry et al. (2019). Lee et al. (2019) developed a more advanced method of automating marking of open-ended questions and concluded that the system caused significant improvements to uncertainty-infused scientific argumentation from pre-test to post-test. A different approach was used by Zhu et al. (2020), who examined the effect of automated feedback on students' revision of scientific arguments. While all studies mentioned above

focused on assignments in English, Cinar et al. (2020) developed a machine learning algorithm for grading open-ended physics questions in Turkish. Machine learning has also been used to automatically mark assignments in a number of different fields, including medicine (Gierl et al., 2014), computer programming (Blikstein et al., 2014), and physics (Zhang et al., 2020). Zhai et al. (2020) provide an overview of the application of machine learning in the assessment of scientific assignments, concluding that it can significantly improved the automaticity of examining and scoring complex constructs such as explanation, argumentation, scientific inquiry and problem-solving, and thus is promising for next generation science assessments.

At Monash University, the delivery of CIV3204 (Engineering Investigation), an introduction to statistics unit in an undergraduate engineering course, has faced a number of problems over the last few years. Along with nearly 3-fold growth in enrolments, we have witnessed an increased rate of academic infringements (cheating) in continuous assessment; logically leading to inferior understanding and subsequent poor summative results which is evidenced by the increase in failure rate seen during this period from approximately 10% to as high as 27%. This situation has forced the Unit Coordinator to drastically restructure the approach to continuous assessment to one which discourages cheating and ensures students invest time in the formative assessment tasks.

This paper focuses on the development and operation of an automated generation and marking system of individualized assignments for this unit and the lessons that have been learned in its first application. An overview of the results of the implementation of the system, including an evaluation by the students and an analysis of the impact on their final examination performance, is also provided.

Purpose or Goal

The objective of this study is to improve the students' performance in the final exam for CIV304 and their learning in the unit. The hypothesis is that individualisation of assessments will discourage cheating, while automatic marking will enable more immediate feedback on a greater number of assignments completed by students; consequentially, increased student engagement (practice) should improve final exam performance and knowledge acquisition. A user-friendly system working within the Moodle Learning Management System (LMS) has been developed for this purpose.

Approach or Methodology/Methods

The Workflow

The system was developed based on a number of prerequisites. These are:

1. Each student must work with individualized data.
2. The system must allow a range of question types, including questions of stochastic nature, which inherently include uncertainty in the answers.
3. The students must be able to submit their answers in a user-friendly manner.
4. The system must work through the e-learning system, more specifically Moodle.
5. The system must lead to challenging questions, and not provide direct information on how to solve the problems.
6. The system must work on all operating systems (more specifically Windows, Mac, and Linux).
7. The students must receive constructive feedback on their submissions.

The system has been coded in python, ensuring it can work on all platforms. The overall principle is that all information exchange from assessment generator to student and to marker is passed through Moodle. To begin, questions, random numbers, and input files are

generated for each student. The input files contain the random numbers that make up the question being asked and any further data that students need to answer the question in the form of input files. If a question, for example one on probability, can be constructed using different random numbers contained in the pdf, students do not need a separate input file. On the other hand, students may be asked to calculate a sampling distribution of a specific data set; in which case they will need the input file.

It is important to note that the random number generator is a pseudo-random number generator, using a seed that is based on the student ID number. This has the advantage of the same random number set being obtained for each student every time questions are regenerated.

The system creates a directory (folder) for each student following the conventions of MOODLE. These directories can later be zipped into one file which can be uploaded on MOODLE. The uploaded files include, for each student, the pdf with the assignment questions, a Graphical User Interface, and the input file for each question (if this is necessary).

The students can then download their assignment files and enter the answers in the Graphical User Interface, which generates Excel files following a naming convention which includes the student ID number, for later traceability. Students then submit their spreadsheet files via Moodle, and markers download all student submissions at once. The marking script then marks all submissions and writes all feedback to a single file per student. Feedback files are then batch uploaded to Moodle for review by individual students.

Generating Questions

In order to generate a unique question, two Python subroutines must be modified. The first, “input maker” creates random numbers and data. These could be, for example when doing hypothesis testing, the sample sizes, the means and standard deviations of the samples, and the confidence level. The upper and lower limits for these random numbers need to be specified, as well as the distributions from which they are drawn. The second subroutine that needs to be modified is “tex maker”, which uses these random numbers to generate the question to be solved by the students (“tex” referring to LaTeX, the standard markup language used in scientific writing). The system then uses these subroutines to typeset a pdf of questions for the students.

The Graphical User Interface

The next step is setting up the Graphical User Interface (GUI) for the students. Two short python codes need to be modified for this purpose. The first is structureMaker.py, which generates a YAML file with the structure of the GUI. Essentially, this program can be set up by copying from example programs, and lists, for each row in the GUI, which variable needs to be entered, and how it should be entered (from a drop-down menu or by manually entering the number). The second is the program for the GUI itself. This program specifies the assignment and question number, the order in which the variables entered by the students must be saved, and can be developed by copying from example programs. Figure 1 shows an example of the resulting GUI for a question on hypothesis testing.

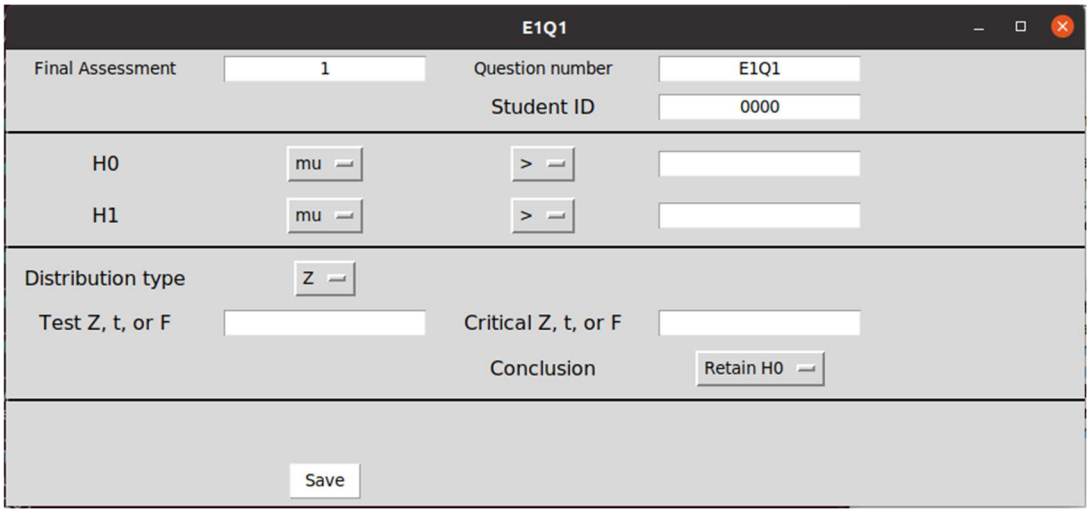

Figure 1 Example of a Graphical User Interface (GUI) for a question on hypothesis testing. This was used on the final examination, hence the question number E1Q1.

Marking Questions

This is the part of the system in which the user has the most freedom. Three subroutines must be modified here.

1. input loader: Here the inputs written in the csv file (for example, the random numbers described in Section II-B) are read in and stored in a vector (or matrix) “inputs”.
2. result loader: Here the results from the students are read in, and entered in the vector or matrix “results”.
3. marker: This subroutine uses the vectors or matrices “inputs” and “results”, and generates a LaTeX string “feedbacks” and a number “mark”. As the name suggests, “feedbacks” contains the feedback that is written to the feedback file for the student. “mark” contains the student’s mark for this specific question. In this subroutine, the user needs to calculate the correct answers to the
4. question: compare the student’s answers to the correct answers, and calculate the mark for the question and generate the feedback.

It should be noted that the system makes consequentially marking questions very easy. If an intermediate error when solving the question is made, the system can calculate the pseudo-correct answer, compare the student’s response to this number, and assign a lower mark.

Types of Questions Enabled by the System

The most straightforward questions are those that require simple calculations. The marking software can compute the correct answer, compare the correct answer to the student’s response, and mark the answer as correct or incorrect using a specified tolerance. These type of questions include hypothesis testing, analysis of variances (ANOVA), regressions, etc. Other straightforward questions are multiple choice questions, which can be developed using the dropdown menu option. A further benefit of the system is that it can also work with questions that are stochastic in nature and, thus, must support uncertainty in the answers. One such question is the calculation of a sampling distribution. In one instance, students were provided an individualized data set from which they had to calculate the sampling distribution of the mean, using a sample size of three, and 10,000 repetitions. They were then provided six different sampling distributions, of which one was correct.

Actual or Anticipated Outcomes

By operationalizing the system for the unit in the second semester of 2020, a number of

lessons could be learned.

A first lesson was that the marking system needed to be made more robust with respect to the numbers entered by the students. Even though it was made very clear to the students to only enter numbers in the GUI cells, and no letters or special characters, in the beginning of the semester this request was consistently ignored. The marker crashed if a string was being read in while a number was expected. A short subroutine was then written that checked, for each cell where a number was expected, if a number was entered.

A second lesson what that students Students would often submit files which did not comply with set guidelines, leading the marking software no to recognize submissions, and consequently return a mark of zero. A short code was then written to list the missing files for each student. The zip files were then manually unzipped, and the file names with errors manually corrected.

Student Evaluation of the System

Table 1. The Student Evaluation of Teaching and Units (SETU)

Question	Responses (2019 / 2020)	Median (2019 / 2020)	% Strongly Agree or Agree (2019 / 2020)
University Wide Items (Summary)			
The Learning Outcomes for this unit were clear to me	112 / 67	3.72 / 4.01	59.82 / 79.10
The instructions for the assessment tasks were clear to me	111 / 67	3.51 / 4.03	50.45 / 76.12
The Feedback helped me achieve the Learning Outcomes for the unit	111 / 67	3.75 / 3.90	60.36 / 70.15
The Resources helped me achieve the Learning Outcomes for the unit	112 / 67	3.32 / 4.01	45.54 / 79.10
I attempted to engage in this unit to the best of my ability	111 / 67	3.74 / 4.22	61.26 / 86.57
Overall, I was satisfied with this unit	112 / 67	3.51 / 3.95	50.45 / 74.63
Faculty Wide Items (Summary)			
The assessment tasks helped me to develop the knowledge and skills required for this unit	112 / 67	3.76 / 4.10	59.82 / 82.81
I understood the grading criteria used in assessing my work	112 /67	3.73 / 3.89	59.82 / 68.66
This unit contained a good mix of theory and practical	112 / 67	3.68 / 3.89	58.04 / 68.66
The Moodle site was engaging and enhanced the learning experience	112 / 67	3.65 / 3.97	56.25 / 74.63
The lectures were valuable for my learning	112 / 67	3.53 / 4.02	50.89 / 74.63

Individualized assignments were generated for each of the 11 topics in the unit; they were automatically marked and detailed feedback was provided for each. This level of assessment would have been impossible to achieve with manual marking. The performance of the cohort in the final exam and the Student Evaluation of Teaching and Units (SETU) were used to evaluate the success of system.

The implementation of the system has led to very positive results for the unit. Table 1 shows an overview of the Student Evaluation of Teaching and Units (SETU) for the Clayton campus. Of the 350 students enrolled, 67 participated in the evaluation. The result for the overall satisfaction question is the highest for the eight years in which the responsible academic taught the unit. Important as well are the results of the qualitative analysis. The first question is “Which aspect(s) of this unit did you find most effective?” 42 Students answered this question, and 28 students stated they appreciated the nine relatively short assignments, and four students stated clearly that they appreciated that this forced them to keep on track with the unit.

The second question is: “Would you suggest any changes to enhance this unit in the future?”, which 44 students answered. Seven students replied that the setup of the GUI’s could be improved. These have now been replaced with html-based GUI’s, which are much

more user-friendly.

Another advantage is that student complaints regarding unprepared tutors have disappeared. A week earlier than for the students, an individualized assignment was also generated for the tutors, which they also had to generate the answers for, and for which they also were marked. This forced the tutors to prepare themselves for the tutorials.

A final advantage of the system was that it led to a strongly improved performance of the students in the final exam, even though this was more difficult than in the previous years. For the students enrolled in CIV3204 in 2019, 107 out of the 475 did not pass the unit. In 2020, after the implementation of the system, 40 out of the 350 students enrolled in CIV3204 failed the unit. The failure rate thus decreased from 22% to 11%. One explanation, which is suggested by the answers to the SETU questions, is that the individualization of the assignments has forced the students to do them, and consequentially they were better prepared for the exam.

In response to the students' comments, an improvement to the GUI's has been made. These are now written in html, and the students can activate them by double-clicking, upon which the GUI's appear in the browser of their choice. This eliminates the need to install python and type in the command line interface.

Conclusions/Recommendations/Summary

The greatest surprise from the study was that the students were very positive about the large number of assignments and the automated marking. The students suggested to improve the python-based Graphical User Interface system, which we have now solved using html-based GUI's. The system improved the students' learning through more practice and feedback, evidenced by their achievement in the exam. Based on the positive outcome, we suggest that automated marking should be further developed and implemented in the STEM disciplines.

References

Aldriye, H., A. Alkhalaf, and M. Alkhalaf (2019), Automated grading systems for programming assignments: A literature review, International Journal of Advanced Computer Science and Applications, 10(3), 215-221, doi:10.14569/IJACSA.2019.0100328.

Barana, A., and M. Marchisio (2016), Ten Good Reasons to Adopt an Automated

Formative Assessment Model for Learning and Teaching Mathematics and Scientific Disciplines, Procedia - Social and Behavioral Sciences, 228 (June), 608-613, doi:10.1016/j.sbspro.2016.07.093.

Blikstein, P., M. Worsley, C. Piech, M. Sahami, S. Cooper, and D. Koller (2014), Programming Pluralism: Using Learning Analytics to Detect Patterns in the Learning of Computer Programming, Journal of the Learning Sciences, 23(4), 561-599, doi:10.1080/10508406.2014.954750.

Blumenstein, M., S. Green, S. Fogelman, A. Nguyen, and V. Muthukkumarasamy (2008), Performance analysis of GAME: A generic automated marking environment, Computers and Education, 50(4), 1203-1216, doi:10.1016/j.compedu.2006.11.006.

Carberry, T. P., P. S. Lukeman, and D. J. Covell (2019), Bringing Nuance to Automated Exam and Classroom Response System Grading: A Tool for Rapid, Flexible, and Scalable Partial-Credit Scoring, Journal of Chemical Education, 96(8), 1767-1772, doi:10.1021/acs.jchemed.8b01004.

Chandra, B., M. Joseph, B. Radhakrishnan, S. Acharya, and S.Sudarshan (2015), Partial marking for automated grading of SQL queries, Proceedings of the VLDB Endowment, 9(13), 1541-1544, doi:10.14778/3007263.3007304.

Cheang, B., A. Kurnia, A. Lim, and W. C. Oon (2003), On automated grading of programming assignments in an academic institution, Computers and Education, 41(2), 121-131, doi:10.1016/S0360-1315(03)00030-7.

Conejo, R., B. Barros, and M. F. Bertoa (2019), Automated Assessment of Complex Programming Tasks Using SIETTE, IEEE Transactions on Learning Technologies, 12(4), 470-484, doi:10.1109/TLT.2018.2876249.

Cinar, A., E. Ince, M. Gezer, and O. Yilmaz (2020), Machine learning algorithm for grading open-ended physics questions in Turkish, Education and Information Technologies, doi:10.1007/s10639-020-10128-0.

Donnelly, D. F., J. M. Vitale, and M. C. Linn (2015), Automated Guidance for Thermodynamics Essays: Critiquing Versus Revisiting, Journal of Science Education and Technology, 24(6), 861-874, doi:10.1007/s10956-015-9569-1.

Fleming, W., K. Redish, and W. Smyth (1988), Comparison of manual and automated marking of student programs, Information and Software Technology, 30(9), 547-552, doi:10.1016/0950-5849(88)90133-4.

Gierl, M. J., S. Latifi, H. Lai, A. P. Boulais, and A. de Champlain (2014), Automated essay scoring and the future of educational assessment in medical educa tion, Medical Education, 48(10), 950-962, doi:10.1111/medu.12517.

J. Kovacic, Z., and J. Green (2012), Automatic Grading of Spreadsheet and Database Skills, Journal of Information Technology Education: Innovations in Practice, 11, 053-070, doi:10.28945/1562.

Jackson, D. (1996), A software system for grading student computer programs, Computers and Education, 27(3-4), 171-180, doi:10.1016/s0360-1315(96)00025-5.

Janicic, M. V., and F. Maric (2020), Regression verification for automated evalu- ation of students programs, Computer Science and Information Systems, 17(1), 205-227, doi:10.2298/CSIS181220019V.

Kiraly, S., N. Karoly, and O. Hornyak (2017), Some aspects of grading Java code submissions in MOOCs, Research in Learning Technology, 25(1063519), 1-16.

Lee, H. S., A. Pallant, S. Pryputniewicz, T. Lord, M. Mulholland, and O. L. Liu (2019), Automated text scoring and real-time adjustable feedback: Supporting revision of scientific arguments involving uncertainty, Science Education, 103(3), 590-622, doi:10.1002/sce.21504.

Liu, X. (2013), A new automated grading approach for computer programming, Computer Applications in Engineering Education, 21(3), 484-490, doi:10.1002/cae.20494.

Ma, Z. H., W. Y. Hwang, and T. K. Shih (2020), Effects of a peer tutor recommender system (PTRS) with machine learning and automated assessment on vocational high school students' computer application operating skills, Journal of Computers in Education, (300), doi:10.1007/s40692-020-00162-9.

Manoharan, S. (2017), Personalized assessment as a means to mitigate plagiarism, IEEE Transactions on Education, 60(2), 112-119, doi:10.1109/TE.2016.2604210.

Menk, K. B., and S. Malone (2015), Creating a cheat-proof testing and learning environment: A unique testing opportunity for each student, Advances in Accounting Education: Teaching and Curriculum Innovations, 16, 133-161, doi:10.1108/S1085-462220150000016007.

Munoz De La Pena, A., D. Gonzaez-Gomez, D. Munoz De La Pena, F. Gomez-Estern, and M. Sanchez Sequedo (2013), Automatic web-based grading system: Application in an advanced instrumental analysis chemistry laboratory, Journal of Chemical Education, 90(3), 308-314, doi:10.1021/ed3000815.

Naude, K. A., J. H. Greyling, and D. Vogts (2010), Marking student programs using graph similarity, Computers and Education, 54(2), 545-561, doi: 10.1016/j.compedu.2009.09.005.

Reilly, E. D., R. E. Stafford, K. M. Williams, and S. B. Corliss (2014), Evaluating the validity and applicability of automated essay scoring in two massive open online courses, International Review of Research in Open and Distance Learning,15(5), 83-98, doi:10.19173/irrodl.v15i5.1857.

Reilly, E. D., K. M. Williams, R. E. Stafford, S. B. Corliss, J. C. Walkow, and D. K. Kidwell (2016), Global times call for global measures: Investigating automated essay scoring in linguistically-diverse MOOCs, Online Learning Journal, 20(2), doi:10.24059/olj.v20i2.638.

Stockburger, D. W. (1999), Automated grading of homework assignments and tests in introductory and intermediate statistics courses using active server pages, Behavior Research Methods, Instruments, and Computers, 31(2), 252-262, doi:10.3758/BF03207717.

Vista, A., E. Care, and P. Griffin (2015), A new approach towards marking large-scale complex assessments: Developing a distributed marking system that uses an automatically scaffolding and rubric-targeted interface for guided peer-review, Assessing Writing, 24, 1-15, doi:10.1016/j.asw.2014.11.001.

Vitale, J. M., K. Lai, and M. C. Linn (2015), Taking advantage of automated assessment of student-constructed graphs in science, Journal of Research in Science Teaching, 52(10), 1426-1450, doi:10.1002/tea.21241.

Vujosevic-Janicic, M., M. Nikolic, D. Tocic, and V. Kuncak (2013), Software verification and graph similarity for automated evaluation of students' assignments, Information and Software Technology, 55(6), 1004-1016, doi: 10.1016/j.infsof.2012.12.005.

Wang, H. C., C. Y. Chang, and T. Y. Li (2008), Assessing creative problem-solving with automated text grading, Computers and Education, 51(4), 1450-1466, doi:10.1016/j.compedu.2008.01.006.

Zhai, X., Y. Yin, J. W. Pellegrino, K. C. Haudek, and L. Shi (2020), Applying machine learning in science assessment: a systematic review, Studies in Science Education, 56(1), 111-151, doi:10.1080/03057267.2020.1735757.

Zhang, Y., C. Lin, and M. Chi (2020), Going deeper: Automatic short-answer grading by combining student and question models, User Modeling and User-Adapted Interaction, 30(1), 51-80, doi:10.1007/s11257-019-09251-6.

Zhu, M., O. L. Liu, and H. S. Lee (2020), The effect of automated eedback on revision behavior and learning gains in formative assessment of scientific argument writing, Computers and Education, 143(September 2018), 103,668, doi:10.1016/j.compedu.2019.103668.

Zupanc, K., and Z. Bosnic (2017), Automated essay evaluation with semantic analysis, Knowledge-Based Systems, 120, 118-132, doi:10.1016/j.knosys.2017.01.006.

KEYWORDS

Automated marking, individualized assessment, statistics

Acknowledgements

We wish to thank the Department of Civil Engineering at Monash University for providing the funding for this project.

Copyright statement

The following copyright statement should be included at the end of your paper. Substitute authors' names in final (camera ready) version only.