SimGrade: Using Code Similarity Measures for More Accurate Human Grading

Sonja Johnson-Yu, Nicholas Bowman, Mehran Sahami, and Chris Piech
Stanford University
{sonja, nbowman, sahami, piech}@cs.stanford.edu

ABSTRACT

While the use of programming problems on exams is a common form of summative assessment in CS courses, grading such exam problems can be a difficult and inconsistent process. Through an analysis of historical grading patterns we show that inaccurate and inconsistent grading of free-response programming problems is widespread in CS1 courses. These inconsistencies necessitate the development of methods to ensure more fairer and more accurate grading. In subsequent analysis of this historical exam data we demonstrate that graders are able to more accurately assign a score to a student submission when they have previously seen another submission similar to it. As a result, we hypothesize that we can improve exam grading accuracy by ensuring that each submission that a grader sees is similar to at least one submission they have previously seen. We propose several algorithms for (1) assigning student submissions to graders, and (2) ordering submissions to maximize the probability that a grader has previously seen a similar solution, leveraging distributed representations of student code in order to measure similarity between submissions. Finally, we demonstrate in simulation that these algorithms achieve higher grading accuracy than the current standard random assignment process used for grading.

Keywords

similarity, code embeddings, embeddings, assessment, grading, human, simgrade, grade

1. INTRODUCTION

Free-response coding questions are a common component of many exams and assessments in programming courses. These questions are popular because they give students the opportunity to show their understanding of course material and demonstrate their coding and problem-solving skills [16]. However, the flexible nature of these problems introduces unique challenges when it comes to grading student responses, which are compounded in situations where the scale of the course necessitates a team of graders working together ("group grading"). The difficulty of consistent application of grading criteria by a group of graders stems from the incredible diversity of student submissions that are generated for free-response coding questions. In particular, it has been previously shown that the space of different student solutions to free-response programming problems follows a long-tailed Zipf distribution [18]. For this reason, it is challenging to develop automated systems for grading and providing feedback and thus human grading remains the gold standard for grading such free-response problems. However, even a team of human graders with extensive experience can struggle to consistently and accurately apply a single, unified criteria when grading. This is problematic as it can result in negative impacts on students in the form of incorrectly assigned grades and inaccurate feedback. Our goal in this paper to explore the frontier of techniques improving the process and outcomes of the exam grading experience.

Our main insight in developing improved approaches for grading is that *it is easier for graders to grade in a consistent manner if they are able to grade similar submissions one after another*. First, we examine historical data to provide concrete evidence of a relationship between grader accuracy and the similarity of previously graded submissions to the current submissions a grader is grading. Then, we propose algorithms that group and order similar submissions in different ways to minimize grader error. Finally, we show that these algorithms perform better than current baseline methods for grading. This work's primary contributions are:

1. Reporting of grader errors in a CS1 course
2. Using historical data to demonstrate the potential benefits of similarity-based grading
3. Three algorithms for grading using code similarity

1.1 Related Work

Autograding One commonly used approach to scale grading is the use of autograders [6]. While useful for comparing program output for correctness or matching short snippets of code, autograders are more problematic for free-response questions in exam settings. In such contexts, the subtlety of understanding that human graders provide is often essential to providing appropriate feedback to students and properly assessing the (partial) correctness of their solutions. While promising, fully autonomous AI solutions are not ready for

grading CS1 midterms [14, 11, 18, 12] especially for contexts with only hundreds of available student submissions [17].

Grading by Similarity The idea of grouping and organizing student submissions in order to improve grading outcomes has been previously proposed for a variety of problem types. Merceron and Yacef [9] use vectors that encode students' mistakes in order to group together students who make similar mistakes when working on formal proofs in propositional logic. Gradescope, designed by Singh et al. [15] offers functionality for grading similar solutions, which is currently most effective on multiple-choice-type questions. This approach has also been applied to short answer questions, as explored by Basu et al. [2], as well as math problems, as demonstrated by Mathematical Language Processing [8]. In this paper, we identify "similar" student responses on free-response programming questions to improve grading quality.

Code Similarity In order to define similarity metrics for student code submissions, we apply techniques for generating numerical embeddings for student programs. Henkel et al.what [5] created abstracted symbolic traces, a higher-level, light-syntax summary of the programs, and embedded them using the GloVe algorithm [13]. Alon et al. [1] pioneered code2vec, an attention-based embedding model specifically used to represent code. Recently, further advances have been made to improve code embeddings by training contextual AI models on large datasets from Github [7]. For this application, we favor simpler unsupervised embedding strategies that do not require human-generated labels by adapting the popular NLP technique Word2vec [10], in which "word" representations are derived from surrounding context.

1.2 Dataset

Our analysis focuses on the student submissions and grader logs from four exams for an introductory programming (CS1) course taught in Python. The breakdown of summary statistics across the four exams is presented in Table 1. As a note, a "submission" is defined as one student's written answer to one free-response problem – thus, the total number of submissions for a given exam is roughly the number of students times the number of coding problems on the exam. In total, we analyze 11,171 student submissions across 1,490 students. Additionally, we have grading logs for every student submission, which consists of information about the grader, the criteria items applied, the final score, and the amount of time that the grader spent on the submission. 199 graders contributed to grading these four exams. As discussed below, the same student submission is sometimes graded by more than one grader for validation purposes. Thus, our dataset contains 14,597 individual grading log entries.

Our grading data comes from a grading software system that randomly distributes student submissions to graders. Among the standard student submissions for grading, this software also inserts "validation" submissions that have already been graded by senior teaching assistants. Every grader assigned to a specific problem will grade all "validation" submissions for that problem. The presence of these special submissions creates opportunities for assessing grader performance, both relative to their peers and relative to "expert" performance.

Exam #	# Students	# Submissions	# Graders
1	533	3,731	53
2	259	1,813	52
3	247	2,470	51
4	451	3,157	43
Total	1,490	11,171	199

Table 1: Exam Grading Dataset Summary Statistics

2. NATURAL GRADING ERROR

While anecdotal experience of grading inconsistency is a common trend in our experience as educators, our first focus is to quantify the inconsistencies present in historical grading sessions in a rigorous manner. In particular, our analysis focuses on the aforementioned "validation" submissions that were specially handled by the grading software and assigned to every grader working on a specific problem. As a result, we had a subset of the grading logs for which we knew both the true grade (as defined by an expert) and the "validation" grade assigned by each grader. Plotting these values against one another is shown in Figure 1, which reveals troubling inconsistencies in the grades assigned by graders. With an RMSE of 7.5 (i.e., average error of 7.5 percentage points per problem), we see that grading error is significant, nearly on the order of what would translate to a full letter grade. Linear regression on this plot yields an R-squared coefficient of 0.947 indicating that while the error may be high, the direction of errors is generally unbiased. In other words, there is not systematic over/under-grading. Rather, the grading errors tend to be randomly distributed around the true grade. Thus, the rest of this paper focuses on methods for decreasing this demonstrated inconsistency (absolute error) in human grading.

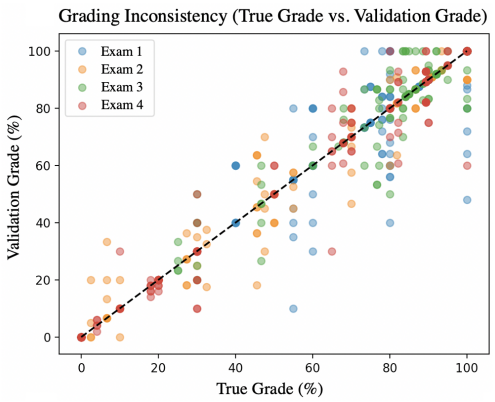


Figure 1: True grade assigned by expert vs. validation grade assigned by human grader

3. METHODS

In this section, we will first outline methods for answering key questions about the problem of improving human grading using similarity scores. Then, we will present three novel algorithms for improving human grading.

3.1 Can code similarity be accurately captured?

We generate program embeddings for all student submissions in our corpus. Word embeddings are an established

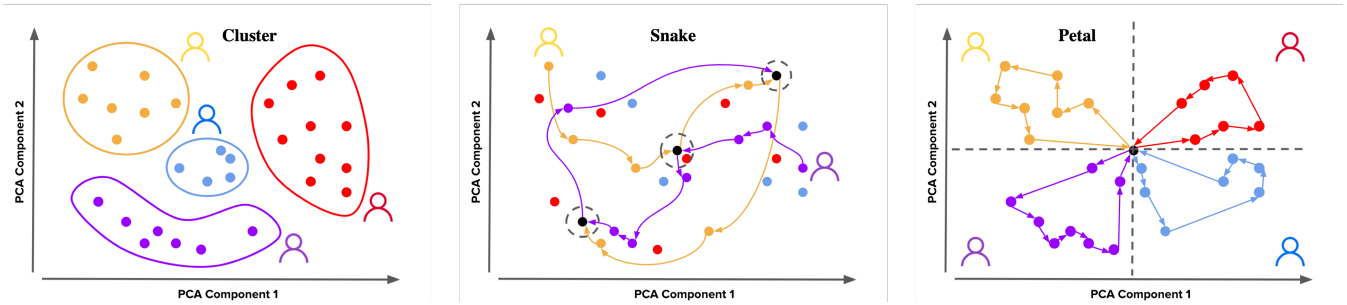


Figure 2: Submission assignment via three algorithms: Cluster, Snake, Petal

method of encoding semantics in human language [10, 13, 3, 3], and these same techniques applied to code accomplish similar results. Algorithms for generating embeddings are constantly evolving and improving; to avoid over-optimization at the embedding generation stage, we chose to employ the simple baseline Word2Vec algorithm. We then demonstrated that our embeddings are semantically significant using zero-shot rubric sampling [18]. For details, see the Appendix[1].

3.2 Does similarity influence grader accuracy?

We hypothesize that graders score submissions more accurately when they have recently seen a submission similar to the current submission. To test this hypothesis, we analyze grading data for four exams. First, for each grader, we generate a "percentage grading error," which is an average of their absolute percent deviation from the correct answer on all validation submissions that they graded. Then, for each of the validation submissions that a grader evaluated, we sort their personal grading logs by time and look at the window of three submissions leading up to each validation submission they graded. To quantify similarity of the validation submission to recently graded submissions, we take the maximum of the cosine similarity between the current validation submission and the three previous submissions. We plot the maximum similarity between a validation submission and the previous submissions against a grader's percentage grading error in order to identify the relationship between a grader's history and accuracy. Then we can infer a formula that approximates the relationship between previous submission similarity and percentage grading error.

3.3 Algorithms to assist human grading

We compare four algorithms for assigning submissions to graders: (1) Random, in which submissions are randomly assigned to graders, with five "validation" submissions interspersed for assessing grader bias. This is the status quo and serves as the baseline. (2) Cluster, in which each grader is assigned to a cluster of highly similar submissions. (3) Snake, in which each grader is randomly assigned a set of submissions and is shown the submissions greedily by nearest neighbor. (4) Petal, in which the dataset is divided into "petals" and all graders begin in the same place. Figure 2 provides a visualization of (2), (3), and (4). Detailed explanations of the algorithms are in the Appendix[1].

[1]https://compedu.stanford.edu/papers/appendices/SimGradeAppendix.pdf

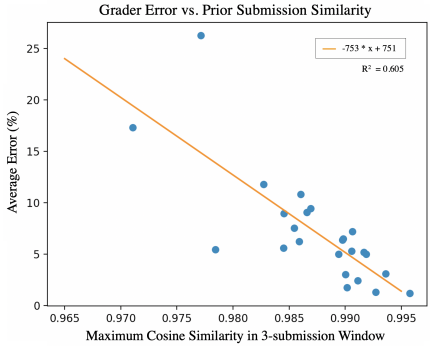


Figure 3: Relationship between grader accuracy and similarity in 3-submission window prior to validation submission

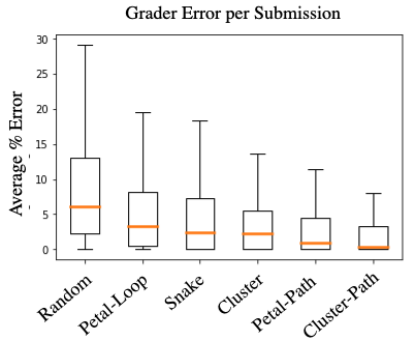
3.4 Algorithm evaluation

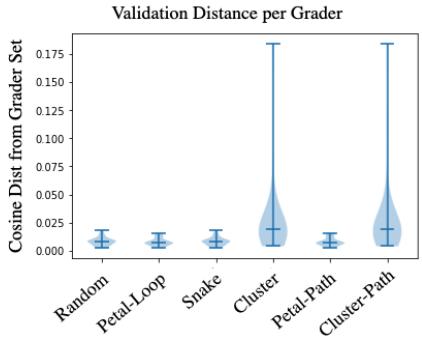
To evaluate the performance of the different algorithms, we simulate grading for a 444-person six-problem exam and ten graders, using real student programs from an actual exam. Details about the selection of validation submissions are in the Appendix[1]. When running the simulation, we infer percentage grading error by examining the similarity of the previous three submissions to the current submission. While we emphasize grader error as the most important metric for assessing an algorithm, a secondary consideration is how naturally validation submissions integrate with the rest of a grader's assigned submissions. Ideally, a validation submission is not "out-of-distribution" with respect to the other submissions that a grader is assigned. Otherwise, a grader will be able to tell when they are being evaluated for grading accuracy. To assess how "out-of-distribution" the validation submissions are, we examine how dissimilar the validation submissions are from the non-validation submissions assigned to a grader. Specifically, for each validation submission, we measure the distance between the validation submission and the nearest non-validation submission assigned to that grader. We average over the five validation submissions in order to get the mean minimum distance from validation to non-validation for a grader, which will be higher if one of the validation submissions is out-of-distribution.

4. EXPERIMENTAL RESULTS

4.1 Similarity scores are meaningful

Embeddings are semantically significant because similarity between embeddings corresponds to similarity between sub-





Algorithm	Grading Error (%)	Validation Dist (cos)	P-Value
Random	10.2	0.0084	-
Petal-Loop	6.1	0.0080	0.07
Snake	6.2	**0.0068**	0.10
Cluster	3.9	0.028	< 0.005
Petal-Path	3.9	0.0080	< 0.01
Cluster-Path	**2.7**	0.028	< 0.005

Figure 4: Left: Average per-submission grading error for each algorithm, Center: Distance of validation submissions from normally assigned submissions, Right: Summary performance statistics, including comparison to random baseline.

mission feedback labels, as described in the Appendix[1].

4.2 Similarity influences grading

Graders score assignments more accurately when they have recently seen a submission similar to the current submission they are grading. From our analysis of historical data, we find that when there is a high similarity between the current submission and at least one of the previous three submissions, the percentage grading error is low. Conversely, when the similarity between previous submissions is low, the percentage grading error is high. We find a linear relationship between the maximum similarity of the previous three submissions and the percentage grading error as shown in Fig. 3, with $R^2 = 0.605$. Given that the grading process involves the numerous uncertainties that come along with human involvement, we believe this correlation coefficient shows a statistically significant relationship between historical similarity and grader accuracy. While the linear relationship between historical submission similarity and percentage grading error is a simplifying assumption, it is the best assumption we can make given evidence provided in Fig. 3.

4.3 Improved accuracy by algorithm

We compare six algorithms for assigning submissions to graders and selecting an order in which a grader will view a submission in Figure 4. We apply the equation of the linear relationship shown in Figure 3 to the similarity of submissions as ordered for evaluation by different algorithms in our experiments. This equation allows us to predict grader accuracy when using the orderings provided by different algorithms. We find that implementing a path ordering on a clustered assignment of graders to submissions yields the lowest mean error of 2.7% (bold-ed in Fig. 4), while the other algorithms all show an improvement over the baseline 10.2% grading error. We utilize bootstrapping [4] over 100,000 trials in order to get the p-values that indicate the significance of the difference in means between the baseline algorithm and the other algorithms (see table in Fig. 4).

4.4 Validation viability by algorithm

When comparing the cluster, snake, and petal algorithms, we observe that the cluster-based algorithms are most likely to have validation submissions that are "out-of-distribution," with a mean validation distance of 0.0277. All other algorithms have substantially lower mean minimum distances.

5. DISCUSSION

Overall, we saw that all of our novel proposed algorithms for assignment of submissions to graders provided improvements over the random baseline in simulation. In general, we saw that path-based algorithms (petal-path and cluster-path) had lower grading error than their non-path counterparts because they are designed to optimize for maximum similarity between consecutive submissions that a grader grades. In particular, the cluster-path algorithm yielded the lowest grader error in simulation due to its strong tendency to assign very similar submissions to graders. On the other hand, the snake algorithm provided the most optimal average distance to validation submissions, which may be important for a smooth experience for a real-life grader. Finally, we saw that the petal algorithm offered a balanced trade-off between these two extremes – while not optimal in either metric, it can be a good choice when both metrics (grading error and validation submission distance) are equally important for designing a grading experience. For a more in-depth discussion of our observed results, see the Appendix[1].

6. CONCLUSION

Through analysis of historical exams, we demonstrated that there is inconsistency between true scores and grader-assigned scores. In doing so, we introduce a new task and associated measure, *grading correctness*. Moreover, we found experimental support for our hypothesis that graders are able to assign scores to exam problems more accurately when they have previously seen similar submissions. In turn, we proposed the use of code embeddings to capture semantic information about the structure and output of programs and identify similarity between submissions. Using similarity of code embeddings in conjunction with historical grading data, we demonstrate in simulation that graders are indeed able to score submissions more accurately when they have previously seen another submission similar to it. We propose and compare several algorithms for this task, showing that it is possible to achieve a significant increase in grading accuracy over simple random assignment of submissions. Future extensions of this work include (i) improvements on code embeddings and (ii) deployment of the grading algorithms in an operational system to allow more direct experimental comparison of grading accuracy. The use of such algorithms show promise for improving accuracy, and in turn fairness, in evaluations of student performance.

7. REFERENCES

[1] U. Alon, M. Zilberstein, O. Levy, and E. Yahav. code2vec: Learning distributed representations of code. *CoRR*, abs/1803.09473, 2018.

[2] S. Basu, C. Jacobs, and L. Vanderwende. Powergrading: a clustering approach to amplify human effort for short answer grading. *Transactions of the ACL*, October 2013.

[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[4] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, Jan. 1979.

[5] J. Henkel, S. Lahiri, B. Liblit, and T. W. Reps. Code vectors: Understanding programs through embedded abstracted symbolic traces. *CoRR*, abs/1803.06686, 2018.

[6] M. Joy, N. Griffiths, and R. Boyatt. The boss online submission and assessment system. *J. Educ. Resour. Comput.*, 5(3):2–es, Sept. 2005.

[7] A. Kanade, P. Maniatis, G. Balakrishnan, and K. Shi. Learning and evaluating contextual embedding of source code, 2019.

[8] A. S. Lan, D. Vats, A. E. Waters, and R. G. Baraniuk. Mathematical language processing: Automatic grading and feedback for open response mathematical questions. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale*, L@S '15, pages 167–176, New York, NY, USA, 2015. ACM.

[9] A. Merceron and K. Yucef. Clustering students to help evaluate learning. *Technology Enhanced Learning*, 2004.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[11] A. Nguyen, C. Piech, J. Huang, and L. Guibas. Codewebs: scalable homework search for massive open online programming courses. In *Proceedings of the 23rd international conference on World wide web*, pages 491–502, 2014.

[12] S. Parihar, Z. Dadachanji, P. K. Singh, R. Das, A. Karkare, and A. Bhattacharya. Automatic grading and feedback using program repair for introductory programming courses. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pages 92–97, 2017.

[13] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[14] C. Piech, J. Huang, A. Nguyen, M. Phulsuksombati, M. Sahami, and L. Guibas. Learning program embeddings to propagate feedback on student code, 2015.

[15] A. Singh, S. Karayev, K. Gutowski, and P. Abbeel. Gradescope: A fast, flexible, and fair system for scalable assessment of handwritten work. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, L@S '17, pages 81–88, New York, NY, USA, 2017. ACM.

[16] D. Thissen, H. Wainer, and X.-B. Wang. Are tests comprising both multiple-choice and free-response items necessarily less unidimensional than multiple-choice tests?an analysis of two tests. *Journal of Educational Measurement*, 31(2):113–123, 1994.

[17] K. Wang, B. Lin, B. Rettig, P. Pardi, and R. Singh. Data-driven feedback generator for online programing courses. In *Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale*, pages 257–260, 2017.

[18] M. Wu, M. Mosse, N. Goodman, and C. Piech. Zero shot learning for code education: Rubric sampling with deep learning inference, 2018.