Assistant Dashboard Plus – Enhancing an Existing Instructor Dashboard with Difficulty Detection and GPT-based Code Clustering

Samuel George
The University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
sdgeorge@cs.unc.edu

Tao Huang
The University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
thuang@unc.edu

Chandler Robinson
The University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
chandle@unc.edu

Gabriel Schell
The University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
gabrieli@ad.unc.edu

Wei Shan
The University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
wshan@unc.edu

Ziqian Zhao
The University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
ziqian@ad.unc.edu

Zeqi Zhou
The University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
zeqi@ad.unc.edu

Prasun Dewan
The University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
dewan@cs.unc.edu

ABSTRACT

As interest in programming as a major grows, instructors must accommodate more students in their programming courses. One particularly challenging aspect of this growth is providing quality assistance to students during in-class and out-of-class programming exercises. Prior work proposes using instructor dashboards to help instructors combat these challenges. Further, the introduction of ChatGPT represents an exciting avenue to assist instructors with programming exercises but needs a delivery method for this assistance. We propose a revision of a current instructor dashboard Assistant Dashboard Plus that extends an existing dashboard with two new features: (a) identifying students in dificulty so that instructors can effectively assist them, and (b) providing instructors with pedagogically relevant groupings of students' exercise solutions with similar implementations so that instructors can provide overlapping code style feedback to students within the same group. For dificulty detection, it uses a state-of-the-art algorithm for which a visualization has not been created. For code clustering, it uses GPT. We present a first-pass implementation of this dashboard.

IUI Companion '24, March 18–21, 2024, Greenville, SC, USA

ACM ISBN 979-8-4007-0509-0/24/03
https://doi.org/10.1145/3640544.3645231

CCS CONCEPTS

• Human-centered computing → Graphical user interfaces; User interface design; Visualization toolkits; Graphical user interfaces; • Applied computing → Computer-assisted instruction.

KEYWORDS

Computer programming, Dashboards, Learning at scale, ChatGPT, GPT

ACM Reference Format:
Samuel George, Tao Huang, Chandler Robinson, Gabriel Schell, Wei Shan, Ziqian Zhao, Zeqi Zhou, and Prasun Dewan. 2024. Assistant Dashboard Plus – Enhancing an Existing Instructor Dashboard with Dificulty Detection and GPT-based Code Clustering. In 29th International Conference on Intelligent User Interfaces - Companion (IUI Companion '24), March 18–21, 2024, Greenville, SC, USA. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3640544.3645231

1 INTRODUCTION & RELATED WORKS

Interest in computer science as a major continues to grow [1], and this increase in interest has naturally led to a rise in enrollment in computer science courses [11]. Large enrollment combined with the fact that programming is already a challenging subject to teach [12] has led to the development of several tools to help instructors with a particularly challenging aspect of handling large classes, which is assisting students with programming exercises.

Instructor dashboards are one such set of tools developed to address this challenge. These dashboards provide visualizations and interfaces with the goals of (a) helping instructors track students' progress on programming exercises so they can adequately gauge exercise and course pace [13] [10] [6] [4], (b) identifying students in dificulty during exercises to provide them help [4] [6] [3], and

(c) generating code clusters from similar code solution implementations to provide bulk code style feedback [5] [14] [7]. The most recent dashboard (to our knowledge) in the instructor dashboard space, VizProg, attempts to meet all three of these goals [15]. Aside from the pedagogical goals of these dashboards, they also target a wide variety of programming exercise types from those given in-class, which we call Synchronous exercises, to those given as take-home exercises, which we call Asynchronous exercises.

Many of these dashboards are intelligent dashboards – that is, they display AI inferences. These dashboards, thus, contribute to both AI and user interface (UI) design.

Not all AI-based class analytics research, however, has been visualized. Two such technologies involve dificulty detection [2] [9] and code-clustering capabilities of large language models, such as ChatGPT.

We have developed a system to provide a dashboard-based delivery method for these two kinds of technologies. Previous work on dashboards has proposed alternatives to existing UIs based on new UI elements. On the other hand, our system creates two new extensions to an existing dashboard UI that are consistent with the original dashboard's UI. The result is a less revolutionary UI than dashboard UIs designed from scratch – hence a demo submission. However, its advantage is that an instructor wishing to track the wide range of analytics the system provides can use a single UI rather than multiple UIs. The specific analytics provided by our dashboard UI are:

- An indication of class pace regardless of exercise type (synchronous or asynchronous).
- An identification of students in dificulty that works regardless of exercise type (synchronous or asynchronous).
- GPT-supported grouping of exercise solutions that represent pedagogically relevant groups that can share the same code style feedback.

We next discuss the UI and algorithms we use for a first-pass implementation to provide these analytics. We call this new dashboard Assistant Dashboard Plus. Our current implementation runs on a local computer with static data from our relevant algorithms. In the future, we will port this implementation to a website with the algorithms and dashboard running in real-time. Thus, this work only focuses on the UI design and the relevant algorithms to meet our analytic goals.

2 ASSISTANT DASHBOARD PLUS

We next show the original Assistant Dashboard and the current implementation of our Assistant Dashboard Plus.

2.1 Extending Assistant Dashboard UI

Figure 1 shows a side-by-side comparison of the original Assistant Dashboard (left) and our new Assistant Dashboard Plus (right).

We extended Assistant Dashboard to meet our analytic goals in several ways. The original Assistant Dashboard already contains a view of the class pace and students with dificulty in its original implementation (Features A-C), meeting part of our analytic goals. The original dashboard's dificulty detection was limited to synchronous exercises. However, we adapted the dashboard to support class pace (Features 1-2) and extended dificulty detection (Features 3-4) to asynchronous exercises. In addition, we added a tab (Feature 5) to display student solution groups. We leveraged the existing visual feature for dificulty detection (Feature C) to support a visual representation of these groupings (Feature 6). In addition, we allow the instructor to look at a student's current code like the original dashboard (Features D & 7), and we expanded the dashboard to work with multiple exercises (Feature 8). Finally, we removed all the original Assistant Dashboard visual elements for gauging model performance (Feature 9) since these are only used in the original implementation to evaluate their dificulty detection algorithm.

2.2 Assistant Dashboard Plus Scenario

The unique aspect of our system is the set of aspects of student progress it visualizes in a single user interface. Below, we present a scenario – a thought experiment – to motivate the user interface.

Imagine an instructor who has given a large class of students a take-home programming exercise. The instructor wants to see how many students have started and pulls up Assistant Dashboard Plus. Using the class pace features (Features 1-2), The instructor finds that only 7% of students have started the assignment and sends an email reminding students that the exercise is due in less than a week.

Two days before the assignment is due, the instructor holds ofice hours, and no one attends. Not wanting to waste ofice hours, the instructor opens our instructor dashboard and finds several students who need help based on their predicted time to finish (Feature 3). The instructor checks their current code (Feature 7), emails them asking if they need help, and gives them advice – personalized learning – based on their working code. One student responds that they are in dificulty, indicating the instructor was able to intervene early to help a struggling student, and the student decides to come to ofice hours. With our dashboard, the instructor gets a time-based visualization of students with dificulty, and the ofice hours are well-spent.

Finally, the assignment ends, and the instructor wants to understand if students correctly used the concepts taught in class to solve the exercise. Again, the instructor opens our dashboard and checks the GPT-based final solution code groupings (Features 5-6). The instructor finds a group of students who gave solutions that do not correctly apply the taught concepts and decides to discuss this set of solutions and how to improve them in class. Thus, the instructor can provide all those students with valuable code style feedback and optimize their teaching based on their real problems.

In this scenario, without our dashboard, the instructor would have to use multiple tools, such as a modified Assistant Dashboard that supports asynchronous exercises [4] and the ChatGPT interface [8] to discover solution groupings. Similarly, the instructor could modify a single existing dashboard, such as VizProg [15] to support GPT-based groupings to gain the benefits of our single dashboard. In the future, we intend to expand our dashboard to incorporate additional algorithms for dificulty detection (Feature 4) and allow its use for multiple assignments (Feature 8). We do not discuss these features in this work. We next examine the algorithms enabling this scenario.

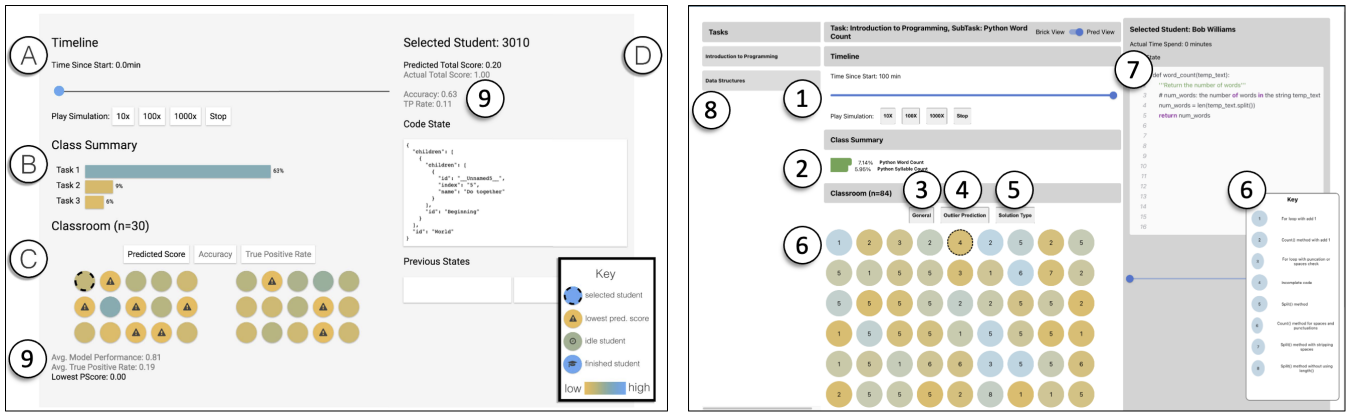

Figure 1: The left UI is the original Assistant Dashboard by Diana et al. [4]. It allows instructors to gauge class pace (Features A-B) and identify students in dificulty (Feature C). It also gives the instructor context to assist with that dificulty (Feature D). The right UI is our new Assistant Dashboard Plus. It provides the same view of class pace (Features 1-2) but includes a new dificulty detection algorithm based on time-to-finish (Feature 3) and a student's code history (Feature 4). It also shows student solution groupings (Feature 5) and leverages a visual feature similar to the original dashboard (Feature C) to show these groupings (Feature 6). Instructors can look at a student's current code as in the original dashboard (Features D & 7) and browse this view for multiple programming exercises (Feature 8). Assistant Dashboard Plus excludes all the in-dashboard metrics for gauging model performance in the original dashboard (Feature 9).

2.3 Assistant Dashboard Plus Algorithms

2.3.1 Class Pace. We implemented the same algorithm as the one used in Assistant Dashboard to determine class pace.

2.3.2 Difficulty Detection. The original implementation of Assistant Dashboard assumed that Predicted Score was a good indicator of students in dificulty. However, our experience from several real-world asynchronous assignments shows that most students finish asynchronous programming exercises and get perfect scores. This finding is intuitive since most students, given ample time and instructor help, will complete an assignment and get full credit. Therefore, we implemented an existing algorithm that lacked a visual vehicle to identify these students. This algorithm uses an implementation of Recent Temporal Patterning (RTP) proposed by Price et al. [9] to predict the time to finish.

2.3.3 Solution Grouping. Finally, we want to provide instructors with pedagogically meaningful code groupings. We chose to use GPT for our code groups because of the improved performance of GPT over time, which we expect to keep improving during the development of our implementation. The GPT-4-Turbo model we use for our current implementation showed a noticeable improvement over GPT-4, which we used initially. We asked GPT to give us pedagogically meaningful groups without giving the number of groups or examples – what we call Zero-shot Grouping.

We choose to use this method because it does not require effort on the instructor's part. In addition, GPT can automatically generate a simple summary of the solution group in natural language (Feature 6 legend) to help guide instructors to specific solution groups. To our knowledge, no other unsupervised grouping algorithm can produce such a summary.

3 SUMMARY, DISCUSSION & FUTURE WORKS

Here, we present a first-pass implementation of Assistant Dashboard Plus, an expansion of an original instructor dashboard by Diana et al. that supports gauging students' pace, identifying students in dificulty, and using GPT to create pedagogically meaningful groups of students' solutions to programming exercises.

While our dashboard is intended to support all these uses to benefit instructors, we also see some potential shortcomings. For instructors, our primary concern is that the AI-based inferences displayed, such as the students indicated to be in dificulty or the solution groups presented, lack accuracy. This lack of accuracy would harm instructors who might waste valuable time during class or ofice hours by addressing improper solution groups or engaging with students who are not having dificulty, respectively. In addition, students who are not in dificulty but are interrupted by the instructor might lose focus or be discouraged at being identified as in dificulty. Another potential limitation is that a single feature-full user interface may overwhelm an instructor who wishes to use a subset of the progress metrics supported in our user interface.

It is important to do real-time studies to validate the usefulness of our instructor dashboard and determine the severity of potential shortcomings. We intend to do studies that deploy the dashboard during synchronous and asynchronous programming exercises and refine the user interface and algorithms based on these studies.

ACKNOWLEDGMENTS

This work was funded in part by NSF awards OAC 1829752 and 1924059. The detailed and insightful comments on the paper improved the discussion of our technical contributions.

REFERENCES

[1] Nick Anderson. [n. d.]. College is remade as tech majors surge and humanities dwindle. The Washington Post ([n. d.]). https://www.washingtonpost.com/education/2023/05/19/college-majors-computer-science-humanities/ Accessed: 2024-01-06.

[2] Jason Carter and Prasun Dewan. 2009. Automatically Identifying That Distributed Programmers Are Stuck. In Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering (CHASE '09). IEEE Computer Society, USA, 12. https://doi.org/10.1109/CHASE.2009.5071403

[3] Jason Carter and Prasun Dewan. 2015. Mining Programming Activity to Promote Help. 23–42. https://doi.org/10.1007/978-3-319-20499-4_2

[4] Nicholas Diana, Michael Eagle, John Stamper, Shuchi Grover, Marie Bienkowski, and Satabdi Basu. 2017. An Instructor Dashboard for Real-Time Analytics in Interactive Programming Assignments. In Proceedings of the Seventh International Learning Analytics & Knowledge Conference (Vancouver, British Columbia, Canada) (LAK '17). Association for Computing Machinery, New York, NY, USA, 272–279. https://doi.org/10.1145/3027385.3027441

[5] Elena L. Glassman, Jeremy Scott, Rishabh Singh, Philip J. Guo, and Robert C. Miller. 2015. OverCode: Visualizing Variation in Student Solutions to Programming Problems at Scale. ACM Trans. Comput.-Hum. Interact. 22, 2, Article 7 (mar 2015), 35 pages. https://doi.org/10.1145/2699751

[6] Philip J. Guo. 2015. Codeopticon: Real-Time, One-To-Many Human Tutoring for Computer Programming. In Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (Charlotte, NC, USA) (UIST '15). Association for Computing Machinery, New York, NY, USA, 599–608. https://doi.org/10.1145/2807442.2807469

[7] Teemu Koivisto and Arto Hellas. 2022. Evaluating CodeClusters for Effectively Providing Feedback on Code Submissions. In 2022 IEEE Frontiers in Education Conference (FIE). 1–9. https://doi.org/10.1109/FIE56618.2022.9962751

[8] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, Zihao Wu, Dajiang Zhu, Xiang Li, Ning Qiang, Dingang Shen, Tianming Liu, and Bao Ge. 2023. Summary of ChatGPT/GPT-4 Research and Perspective Towards the Future of Large Language Models. arXiv:2304.01852 [cs.CL]

[9] Y. Mao. [n. d.]. One minute is enough: Early Prediction of Student Success and Event-level Dificulty during Novice Programming Tasks. In: Proceedings of the 12th International Conference on Educational Data Mining (EDM 2019) ([n. d.]). https://par.nsf.gov/biblio/10136495

[10] Christian Murphy, Gail Kaiser, Kristin Loveland, and Sahar Hasan. 2009. Retina: Helping Students and Instructors Based on Observed Programming Activities. SIGCSE Bull. 41, 1 (mar 2009), 178–182. https://doi.org/10.1145/1539024.1508929

[11] Natasha Singer. [n. d.]. The Hard Part of Computer Science? Getting Into Class. The New York Times ([n. d.]). https://www.nytimes.com/2019/01/24/technology/computer-science-courses-college.html Accessed: 2024-01-06.

[12] Aaron J. Smith, Kristy Elizabeth Boyer, Jeffrey Forbes, Sarah Heckman, and Ketan Mayer-Patel. 2017. My Digital Hand: A Tool for Scaling Up One-to-One Peer Teaching in Support of Computer Science Learning. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (Seattle, Washington, USA) (SIGCSE '17). Association for Computing Machinery, New York, NY, USA, 549–554. https://doi.org/10.1145/3017680.3017800

[13] David Stotts and Yu Ji. 2020. Bricks: Extreme Active Learning for Introductory Programming. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education (Portland, OR, USA) (SIGCSE '20). Association for Computing Machinery, New York, NY, USA, 1418. https://doi.org/10.1145/3328778.3372547

[14] Eliane S. Wiese, Michael Yen, Antares Chen, Lucas A. Santos, and Armando Fox. 2017. Teaching Students to Recognize and Implement Good Coding Style. In Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale (Cambridge, Massachusetts, USA) (L@S '17). Association for Computing Machinery, New York, NY, USA, 41–50. https://doi.org/10.1145/3051457.3051469

[15] Ashley Ge Zhang, Yan Chen, and Steve Oney. 2023. VizProg: Identifying Misunderstandings By Visualizing Students' Coding Progress. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (Hamburg, Germany) (CHI '23). Association for Computing Machinery, New York, NY, USA, Article 596, 16 pages. https://doi.org/10.1145/3544548.3581516