Student Progress Modeling, Hinting, and Affect Detection in SAGE

Sambhav Anand and Allison Sawyer

COMS W3998, Section 14

February 14, 2017

Table of Contents

1 Introduction

This project aims to enhance the capabilities of the existing Social Addictive Gameful Engineering (SAGE) project [**1**] by building on the Scratch Analyzer and Scratch Editor to automate the collection of additional data, and then using machine learning algorithms to gain insights from these data. The major focuses of this machine learning approach are threefold: first, producing a student progression model that may help identify groups of students who learn similarly and trace individuals' progress over time; second, automating the generation of useful and situationally appropriate hints when students need assistance; and third, experimentally evaluating how certain data may correlate with relevant student affective states.

SAGE is designed to immerse students in an enjoyable, game-like learning experience. It offers adaptable features and feedback in order to maximize engagement and minimize the risk of negative emotions including boredom, frustration, and confusion. Research attests to the usefulness of maintaining a sense of "flow" for students playing a learning game [**2**] [**3**], which entails a high level of focus, a sense that time is passing more quickly than it is, and immediate feedback about one's performance. This project will enhance the "Addictive" component of SAGE, which includes the concept of flow, by seeking to gain insights about students' emotional states and by improving the quality of the real-time feedback provided. Additionally, this project will contribute to the "Social" dimension of SAGE through student modeling, which may establish connections between students and enable teachers to direct their attention toward the students who need the most help. While we propose to work on some applications of the data we will

collect during this semester, the potential applications of these data are limitless. The final section of this report imagines some of these possible future applications.

2 Related Work

2.1 Machine Learning

2.1.1 "Modeling How Students Learn to Program" [4]

Piech et al. developed a graphical model of how university students progressed through a programming assignment by taking snapshots of each student's progress at various points and then modeling progress with a Hidden Markov Model (HMM). The researchers used the k-means clustering algorithm to find clusters of paths taken through the HMM. Furthermore, they showed that these clusters had implications for student success, as they were predictive as of students' midterm grades.

2.1.2 "A Framework for Capturing Distinguishing User Interaction Behaviours in Novel Interfaces" [5]

Kardan and Conati developed a user modeling framework based on two steps: behavior discovery and user classification. Behavior discovery involved extracting user log data, transforming it into feature vectors, and running the k-means clustering algorithm on those feature vectors to classify users into two clusters. They performed association rule mining to identify the behaviors that characterized each cluster. In the user classification phase, this classifier was used on-line to classify new users based on their interactions with the program, an artificial intelligence algorithm visualization tool. By comparing tests taken by

users both before and after the study, the researchers found a significant difference in learning gains between clusters.

2.1.3 DT Tutor [6]

DT Tutor is an intelligent approach to hinting that uses a probabilistic model to assesses how students might react to a variety of possible hints that could be issued by an intelligent tutoring system (ITS). It then evaluates the utility of the possible resulting student states and takes the action with the highest expected utility. This approach allows the ITS to tune its hinting strategy to fit student needs at a particular time.

2.1.4 Andes Physics Tutor [7]

Andes is an ITS designed to help students learn college-level physics by working through problems in an environment with helpful visualizations and immediate feedback. Its granular hinting capability allows it to give feedback to the user at any step in his/her problem-solving process. When Andes identifies errors likely resulting from specific, common misconceptions, it provides error-specific feedback that aims to help the learner avoid making that same mistake in the future.

3 Proposal

This section will discuss data collection and machine learning strategies to be explored during the course of this project, along with proposed applications.

3.1 Data Collection and Feature Extraction

The main goal of this portion of the project is to modify Scratch Analyzer [**8**] and the Scratch Editor to include additional metrics that will aid in data analysis by machine learning algorithms (to be discussed in the following section).

This objective can be separated into two parts: first, altering the Scratch Analyzer to get the output data in the format desired for the machine learning algorithms, and second, modifying the Scratch Editor to include keystroke-tracking and other related data in the output .sb2 files.

3.1.1 Scratch Analyzer

This portion of the project aims to modify Scratch Analyzer to track potential indicators of students' affective states, which will be useful as input to machine learning algorithms. Using the mean and standard deviation of per-block actions and latency [**5**] as features may be a low-cost and effective way to gain information. Additional data that Scratch Analyzer will be modified to collect include:

- Mean and standard deviation of time elapsed since the student's last action (This will include devising an algorithm to determine how different snapshots must be to constitute a significant, intentional action; trivial actions like fixing a misspelling should not be counted.)
- Mean and standard deviation of the number of times an individual block is moved
- Mean and standard deviation of the number of times an individual block's parameters are changed

- Number of times a block is removed from the scripts area

As mentioned above, computing latency will require devising a method that determines the level of difference between two project snapshots. One potential method is Bag of Words Difference: this was used by Piech et al. to create histograms of keyword frequency and compute the Euclidean distance between the histograms "as a naïve measure of the dissimilarity," "similar to distance measures of text commonly used in information retrieval systems" [**4**].

Another method that might be used, with more accuracy than Bag of Words Difference, is Application Program Interface (API) Call Dissimilarity. As Piech et al. describe it, "the sequence of API calls made in the program" serve as "the 'DNA' of that program, and we are comparing the DNA of two programs to determine their dissimilarity" [**4**]. As all Scratch projects use a common set of blocks, which function like API calls, difference between Scratch programs could be measured using this method.

3.1.2 Scratch Editor

The Scratch Editor will be tooled to enable additional data collection. These changes will include:

- Maintaining an awareness of which blocks the student should be using for a particular assignment in order to output a count of the number of times the student attempts to use an irrelevant block (We will explore ways to automatically generate this list of relevant blocks, as ideally the instructor should not need to hardcode it.)

- Tracking mouse activity data, such as the total number of clicks, total quantity of movement, average amount of movements every 2 seconds, etc., and including these data in the output .sb2 file
- Gathering keystroke data and including these data in the output .sb2 file

3.2 Machine Learning

This portion of the project aims to use machine learning techniques to gain insights about students' different progressions through an assignment as well as their affective states, and explore how these data might be leveraged to intelligently, automatically provide hints.

3.2.1 Modeling student progress to intelligently generate hints

This section will discuss two machine learning algorithms that are likely to be useful in modeling student progression through an assignment and using this data to intelligently generate hints. Hint generation is an inner-loop function, in the terminology proposed by VanLehn [**9**], meaning it has the potential to occur between any pair of steps taken by the student. We will create dummy data by manually completing puzzles, if necessary, to tune parameters and test these approaches for their effectiveness. The need for student-generated data will be reevaluated as the project progresses.

3.2.1.1 k-means clustering of project snapshots

A number of researchers doing similar work, including Piech et al. [**4**] and Kardan and Conati [**5**], reported k-means working well as a clustering algorithm. Due to the unlabeled nature of existing Scratch output and other

Scratch data, and the time-consuming and imprecise nature of manual labeling, an unsupervised learning algorithm like k-means clustering is an optimal use of resources. The algorithm should be run on the ensemble of all students' latest project snapshots at various points in the course of the assignment.

The resulting clusters and their development over time should help reveal how most students progress through an assignment and help identify struggling students. Struggling clusters may be identified due to their slower progress, lower assessment scores, and/or teacher intuition; then, these clusters can be more closely analyzed to reveal how struggling students' approaches to an assignment differ from those of their peers. The clusters may help identify groups that need additional instruction in certain areas, as well as groups that are more advanced and in need of more challenging work. In general, students could be matched to others using a similar approach to work together, or to students using a different approach so that they can explain their approaches to one another.

An ensemble of high-level states of progress will constitute the basis of the model, so snapshots will need to be classified by the state they likely represent. A Bag-of-Words approach can be used to extract features from project snapshots. Counts of different blocks used and bigrams or trigrams of blocks in order may also be useful as features, to preserve important sequencing information. Comparisons of different feature sets will have to be performed in order to identify which is the most effective.

3.2.1.2 Sequential pattern mining

Once students have been clustered, as described above, this project will explore the application of sequential pattern mining algorithms within the clusters. Sequential pattern mining algorithms aim to discover patterns of association between items in a dataset, with attention paid to the order in which these items occur. According to Romero et al. [**10**], the GSP and PrefixSpan algorithms worked slightly better than AprioriAll for their purposes. Writing a new *Operator.operate* function for Scratch Traverser, a part of Scratch Analyzer [**8**]**,** may be useful for extracting data related to proper block sequence.

The patterns identified by sequential pattern mining will then be used to predict what the student should do next—in other words, to generate a situation-specific hint. Two hinting strategies worth implementing are shaking a certain block to encourage a student to select or move it, and issuing a text hint if the student remains stuck or if the next block the student should try is ambiguous.

Implementing hinting will require answering the question of how often to provide hints, which will likely require fine-tuning as we receive feedback from SAGE users. Hints should probably be provided not after every wrong move but only after x moves that do not advance the student toward the solution, with x to be determined experimentally as users try out the system. Hinting could be linked to the detection of certain affective states, as described in the next section, if this detection proves to be accurate.

3.2.2 Detecting student affective states

Affective states that would be relevant and useful to identify in students include boredom, frustration, confusion, and distractedness. Once additional data related to student actions is being collected, as discussed above, algorithms can be developed to detect intuitive hallmarks of certain emotions: several repeated manipulations of the same block, for example, may indicate frustration, confusion, or guessing; repeated manipulation of blocks unrelated to those needed for the assignment may indicate boredom; a long period with little or no interaction with the correct blocks may indicate distractedness or confusion. When a larger pool of students begins using SAGE, a clustering algorithm such as k-means clustering can be applied to these data to identify groups of learners with tendencies toward common behaviors.

Once an undesirable student affective state is detected, it may be useful to notify the teacher so that, if s/he is available, s/he can personally intervene. Thus, this project will also aim to implement a mechanism for notifying a teacher in real-time when a student is likely struggling. The detection of certain affective states could also serve as a trigger for automated hinting.

Due to the uncertain and experimental nature of affect detection algorithms in new environments, such as SAGE, and the time-intensive and difficult nature of building these systems, we hope to gain modest theoretical insights and set up the SAGE platform for this type of analysis. We hope this will provide a foundation to be built on in future work.

4 Timeline

Milestone	Estimated completion date	Party responsible
Modify Scratch Editor to include new behavioral information in .sb2 files	February 28, 2017	Sambhav
Tune and apply k-means clustering algorithm to Scratch Extractor data	March 3, 2017	Allison
Run sequential pattern mining algorithms on clusters	March 24, 2017	Allison
Devise algorithms to parse Scratch Analyzer data and quantify the level of difference between project snapshots	March 28, 2017	Sambhav
Midterm report and presentation	**March 31, 2017**	**team**
Automatically generate hints based on sequential pattern rules	April 14, 2017	Allison
Further develop features of the software suite (implement displaying of hints, implement teacher notifications, gather keystroke data, etc.)	April 28, 2017	Sambhav
Attempt clustering for affective states and/or develop experimental algorithms to detect frustration/guessing, and distractedness	April 28, 2017	Allison
Final report and presentation	**May 5, 2017**	**team**

5 Future Work

Much future related work is possible, contingent on the success of the project outlined here. We will start working on these objectives during the current project in the optimistic event that other objectives are completed ahead of schedule.

5.1 Multi-puzzle models

Modeling student progress through one puzzle at a time is an important step that could lead to broader, aggregate models over multiple puzzles of how different clusters of students approach puzzles in general. Aggregate models could reveal to teachers which concepts students in general understand best and which will require more attention. A model accounting for an entire collection of puzzles, rather than just one puzzle at a time, could also enable analysis of student performance on one assignment in order to generate an outer-loop recommendation of which puzzles s/he should work on next or which concepts s/he should practice further. Furthermore, the features extracted for each student for each puzzle s/he completes can be saved in individual student profiles in order to better track his/her progress over time.

5.2 Further hinting improvements

Eventual improvements to hinting could include gradually diminishing the quantity and specificity of the hints shown to students, so that they receive less feedback as they acquire mastery in a topic. This could include providing less detailed hints for competences the student presumably already has than those for concepts they have not seen before, which involves maintaining a model of the student's knowledge at any given time. Analogously, it might be useful to offer a more extensive hint if a student takes several misguided steps early on in the assignment.

Another improvement would be generating more granular error-specific feedback, as described by VanLehn [**9**]. This involves determining precisely which misunderstanding led the student to take an incorrect action and correcting that misunderstanding in order to help the student avoid making the same mistake in the

future. This would only be useful in situations in which different types of student mistakes should influence the type of hint given.

5.3 Studies with real student users

Eventually, it will be necessary to have a larger sample of student users in the target sixth- to eighth-grade range try SAGE. This student data would strengthen models of how different clusters of students approach puzzles. It would also allow for empirical evaluations of the usefulness and appropriateness of the hinting system.

6 References

[**1**] J. Bender, "Tooling Scratch: Designing a Collaborative Game-Based Learning System to Infuse Computational Thinking within Grade 6-8 Curricula," Columbia University, New York, NY, 2015.

[**2**] K.H. Koh, A. Basawapatna, H. Nickerson, and A. Repenning, "Real Time Assessment of Computational Thinking," in *2014 IEEE Symposium on Visual Languages and Human-Centric Computing*, Melbourne, Australia, 2014, pp. 49-52.

[**3**] M.D. Kickmeier-Rust and D. Albert, "Micro-adaptivity: protecting immersion in didactically adaptive digital educational games," in *Journal of Computer Assisted Learning*, 26. University of Graz, Graz, Austria: Blackwell Publishing Ltd., 2010, pp. 95-105.

[**4**] C. Piech, M. Sahami, D. Koller, S. Cooper, and P. Blikstein, "Modeling How Students Learn to Program," Stanford University, Stanford, CA, ACM, 2012.

[**5**] S. Kardan and C. Conati, "A Framework for Capturing Distinguishing User Interaction Behaviours in Novel Interfaces," in *Proceedings of the 4th International Conference on Educational Data Mining*, Eindhoven, the Netherlands, 2011, pp. 159-168.

[**6**] R.C. Murray, K. Vanlehn, and J. Mostow, "Looking Ahead to Select Tutorial Actions: A Decision-Theoretic Approach," in *International Journal of Artificial Intelligence in Education*, 14. 2004, pp. 235-278.

[**7**] K. VanLehn, C. Lynch, K. Schulze, J.A. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill, "The Andes Physics Tutoring System: Lessons Learned," in *International Journal of Artificial Intelligence in Education*, 15(3). 2005.

[**8**] J. Bender, "Scratch Analyzer: Transforming Scratch Projects into Inputs Fit for Educational Data Mining and Learning Analytics," Columbia University, New York, NY, 2014.

[**9**] K. VanLehn, "The Behavior of Tutoring Systems," University of Pittsburgh, Pittsburgh, PA, 2006.

[**10**] C. Romero, S. Ventura, A. Zafra, and P. de Bra, "Applying Web usage mining for personalizing hyperlinks in Web-based adaptive educational systems," in *Computers & Education*, 53. Elsevier Ltd., 2009, pp. 828-840.

[**11**] R.S.J.d. Baker, "Data Mining for Education," Carnegie Mellon University, Pittsburgh, PA.