A Tool for Integrating Log and Video Data for Analysis and Model Generation

Abstract. Log information is an essential part of many studies, and its importance has recently been growing with the advent of the web and the considerable amount of data it brings. Therefore, ongoing research is exploring how data mining techniques can take advantage of such a massive repository of data. One of its applications is on student model generation. Another important data source is video. However, information video information is usually not used on mining due to its qualitative nature. In this paper, we present a tool that seeks to use both types of data on the creation of a student model.

Keywords: log analysis tool, cognitive modeling, intelligent tutor

1 Introduction

Much research on educational systems and user behavior relies on log information for analysis. Recently, with the advent of the web, the amount of log data has greatly increased [1], creating a gold mine for educational data [10]. To handle this growing amount of information, Data Mining techniques are being employed on these datasets to create student models [11].

Many studies, however, also gather valuable data in the format of video. This type of data can capture events that log files many times cannot. Among many of the advantages of video over qualitative data, there is the potential to capture unexpected behavior, the existence of many dimensions of physical and verbal behavior, and the possibility of reuse of its data for different focuses of analysis [9]. However, it is difficult to compress the video data in a way that can be meaningful for analysis [6]. While there are ways of automatically processing videos, one of the most important need for video analysis is to be able to code the information contained in it [7].

The challenge this paper seeks to address is that of integrating both log and video data into the creation of a model through a single tool. While computers can automatically generate useful models with little to no expert intervention based on log files, there may be valuable information contained on a video recording of the user. This data needs to be coded in a machine-readable format so that it can also be used in the model generation. In this paper, we present our implementation of such tool in the context of an educational system in the domain of geometry called X. We then present a proof of concept, in which we demonstrate the applicability of the generated model

both with and without the video information. Finally, we conclude presenting ways of expanding this tool in ways that would provide more flexibility and efficiency for analysis.

1.1 The X System

The X system was first introduced in [8]. It is a system that aims to teach geometry concepts through an embodied environment using a teachable agent framework. In X, students are told that they need to help Quinn, a teachable agent, to solve simple geometry problems (such as "Plot point (3,1)"). They can do so by giving Quinn commands such as Move *N* units, or Turn *M* degrees.

Every action performed by Quinn is logged to a CSV file. The actions that are logged are (1) Move *n* units, (2) Turn *n* degrees, (3) Turn in a direction (North, South, East, West), (4) Reset problem, (5) Load new problem, (6) Solution is Correct/Incorrect, (7) Change user. Each action is associated to a timestamp.

2 The Analysis Tool

In order to facilitate a deeper analysis of this data, we built a tool for integrating the video and log information, along with a computer-generated model. It integrates data from log and video files, coupling them together for analysis. Furthermore, it displays an automatically generated behavior graph, for both aggregated and individual student data. This tool was built in Java, using *Swing* for its GUI. Video playback is done through the *vlcj* library, while the graphs are generated using the *Jung* library. Figure 1 shows the main interface of the application.

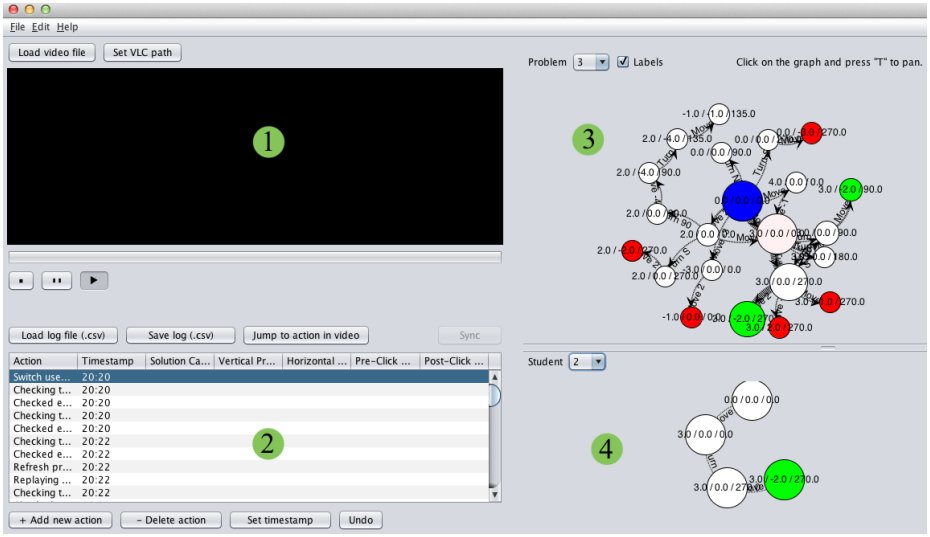

Figure 1: The tool's main interface. (1) Video viewport. (2) Logs table. (3) Aggregate graph. (4) Individual student's graph.

2.1 Video and Log Information

The first aspect of tool is the integration between video and log data. The goal is to allow a seamless two-way navigation of the video and logs. This enables the visualization of the video recording of any action on the logs. Furthermore, while the video is played, the log visualization highlights the action that is currently being executed at the video.

Talk about how the video is synced with the logs. Explain the Add action, delete action, set timestamp and undo buttons.

2.2 The Behavior Graph

The graph is built using the information from the log files. The graph consists of states (where the robot is at a given point) and actions (how did the robot get from state A to state B). A state consists of a triad of information from the robot: its x and y location, and its current orientation (0 – 359 degrees). An example of a state is position x = 2, y = 3 and rotation = 45, which could be synthesized in the form (2,3,45).

The tool iterates over each instruction in the log files and updates the graph accordingly. Since the logs do not contain information about the state, the tool has to infer it from the actions. For example, whenever the action "Refresh problem", or "Current problem index: 1" (log message for when a new problem is loaded), the tool knows that the robot will be at position 0,0 with a rotation angle of 0. If the robot is at the state 0,0 and rotation 0, and an action "Move 2" is parsed, the graph will add a transition called "Move 2" from the current state (0,0,0) to the state (2,0,0).

Some statistic information is encoded visually on the graph. The nodes' diameter is proportional to how many students passed through that state. A big node indicates that many students have passed by there, while a small one indicated the opposite. The same is valid for transitions, with the difference that this information is encoded on the thickness of it. A thicker transition indicates more students. Color is also a major factor. A blue node indicates the starting state (0,0,0). Different intensities of red determine the nodes where students have checked a wrong answer for correctness (a light red indicates that of all students who passed by that state, only a few have incorrectly submitted their solutions), and different intensities of green show where students have submitted a correct answer (a light green indicates that of all students who passed by that state, only a few have correctly submitted their solutions).

There are two graphs in the main interface. The top one displays an aggregate graph that was built using the information from all students in a given problem. The bottom one shows the graph for the same problem, but using the information from a single student. Interactivity also plays a big part on the behavior graphs. Users are able to change between problems and students, enable or disable labels, pan, zoom, and move nodes around. By clicking on a node or transition, the user is able to view detailed information on it, such as which students have passed through a state or transition, or which ones submitted a correct or incorrect solution in any given state.

3 Proof of Concept

3.1 Data Collection

Data comes from a study with 19 5th and 6th grade students (8 female) from a middle school in a large southwestern city. Subjects were trained on how to use X, and were given 45 minutes to teach Quinn how to solve problems. During this time, they could refer to cheat sheets (for both domain and X-specific instructions) and solution cards. The student had to help Quinn with two different kinds of problems: plotting points and translating points, of which the former will be the focus of this analysis. The point plotting problems are: (1) problem 1: Plot point (3,1) (2) problem 2: Plot point (-2,-3) and (3) problem 3: Plot point (3,-2). Each session was videotaped and generated log files. These two data sources were then inputted into the analysis tool, which generated the graph behavior graph and set up the tool for the manual coding.

3.2 Behavior Graph Analysis

Metacognitive Strategies. The first type of information that can be derived from the behavior graphs is on the metacognitive strategies that students use to solve problems. The goal is to classify a students' behavior, and if necessary, enable the system to address students whose strategies are not leading him or her towards correctly solving the problem. Figure 2 shows graph images for the different strategies.

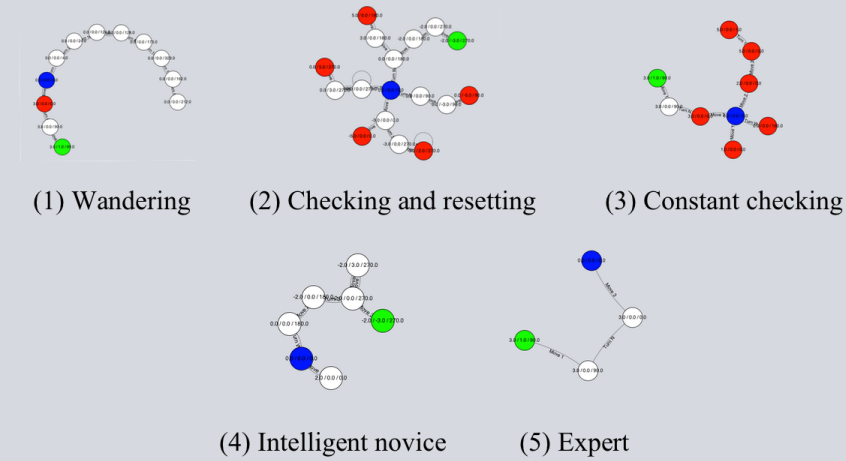

Figure 2: Visualization of the metacognitive strategies taken by students while solving problems in the X system.

The first and most obvious strategy to be identified was *wandering*. In this strategy, the student commands the agent in an apparently senseless way. This pattern is usually seen at the first problem, when students are still trying to understand the system,

and is a good indicator of confusion. Visually, it is characterized by a long sequence of commands that do not move towards a solution.

Another strategy was *checking and resetting*. In it, a student would follow a path, check it, and if incorrect, he would restart the problem, trying a different approach this time. This process would be repeated until the correct solution was found. Visually, this can be identified as several different paths away from the origin that ended in a red node of zero outdegree. A variation of this is when the user, without checking for correctness, gives up on a path and decides to restart the problem. The only visual difference is that the final nodes in a path are not red, but white.

The strategy of *constant checking*, or *trial and error* consists of performing an action and checking for correctness. Without restarting, the student would perform another action and check it again. Visually, this strategy is seen as several paths full of red nodes close to one another.

The next strategy was of *intelligent novice*, which corresponds to students who took a longer path to the correct solution. Despite taking steps that could be considered unnecessary, they usually did not check for correctness until they considered having reached the correct position. If they did check for a wrong solution, they would not restart the problem. Instead, they would proceed from that point towards a correct solution. Visually, this corresponds to a longer path to the correct solution, containing none or very few red nodes.

Finally, the *expert* strategy occurs when a student knows how to solve a problem, and moves directly towards its solution node. Visually, this is seen as a completely white path between the origin (blue) and the solution (green) nodes.

The most used strategy was *Intelligent Novice*, used by 13 participants. The *Expert* and *Check and Reset* strategies, being used by 12 and 11 respectively, followed it. The least used strategies were *Wandering* and *Constant Checking*, both being used by 3 participants. For the first problem, the most used strategies were *Check and Reset* (used by 5 participants) and *Expert* (5). For the second problem, the most used strategy was *Intelligent Novice* (11). For the last problem, it was *Expert* (10). It is also interesting to note that despite having already solved one problem before, only one student used the *Expert* strategy on problem 2, compared to five on problem 1. One possible explanation is that problem 1 deals only with positive numbers, while problem 2 deals with negatives. This may indicate that students may be struggling with negative numbers, thus needing further support from the system.

By categorizing the way that different students use to system to solve its problems, it will be possible to respond more adequately to their behavior, with the potential of scaffolding their learning. For example, prompts could be designed to target specifically those users who are wandering about, in order to reduce their path and lead them to success. And by relating those strategies with the problems, it is possible to identify where students may be struggling the most.

Bug Taxonomy. Through the behavior graph, it was possible to identify the nodes in which students submitted an incorrect response and code their wrong solution for common patterns. We have identified 5 common bugs, or misconceptions, across students. They are presented in Table 1, with the percentage of students who dis-

played each misconception at the rightmost column. A student could perform one or more of these mistakes at a time. Some mistakes could not be classified, and where therefore not included in this table. They are most likely the result of the student's exploration of the system.

Misconception	Description	Problems	% Students
Miscount	The student counts the origin as if it were (1,1).	P1	15.79
Sum coordinates	The student sums the two numbers in the coordinate and moves that amount in one arbitrary axis	P1, P2	47.37
Switch x and y	The first point in the coordinate is considered to be the y-axis, while the second is considered the x-axis.	P1, P2, P3	31.58
Invert axis	The student moves negatively when he should move positively and vice-versa	P1, P2, P3	36.84
Move only in one dimension	The student moves the correct distance in either x or y, but remains on zero for the other dimension.	P1, P2, P3	42.10

Table 1: Common misconceptions derived from the behavior graph. These can occur independently of each other or in groups.

As an example of this analysis, in problem 1, students were asked to plot the point (3,1). Some students plotted a point at (2,0). In the bug taxonomy, this would be classified as *Miscount*, since the student counted the origin as being point (1,1). Another example is of students who plotted the point (-1,3). In this case, this was classified as both *Switch x and y* (since he moved 3 units in y and 1 in x instead of doing the opposite) and *invert axis*, since he moved one negative unit instead of a positive one.

Another interesting point from this analysis is that not all mistakes appear in all problems. In fact, the occurrence of the types of misconceptions fades as students progress through the problems. For example, the misconception "*miscount*" appears only on the first problem, and the misconception "*sum coordinates*" appears only on problems 1 and 2, but not on 3. Looking at the graphs, it's possible to see a decrease in its size, which is another indication that the students could be building proficiency as they complete the problems, resulting in less exploration and fewer mistakes.

Multiple Paths to a Solution. Lastly, the graph enables a visualization of the different paths traced by students to get to the correct answer. Figure 3 shows a part of the graph that contains some of the paths traced by the students that led to correct answers.

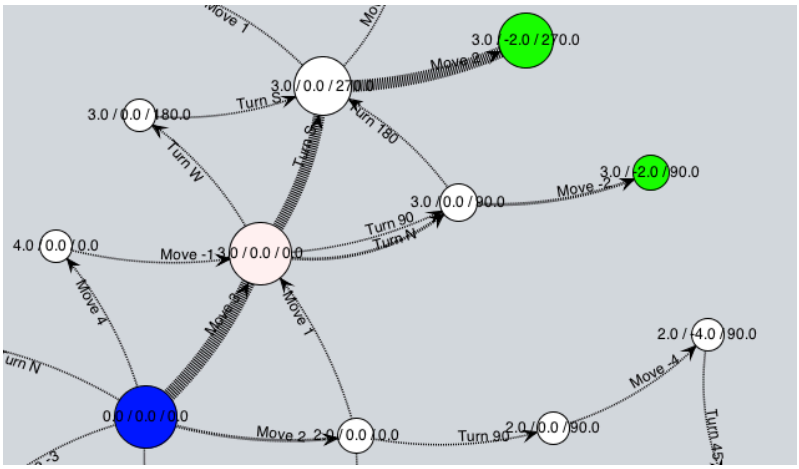

Figure 3: Part of the behavior graph for problem 3, showing different paths taken by students to get to the solution. The most used path is shown as having the thickest edges and largest vertices.

From this image, it's possible to see the many different paths that students take to solve this problem (Problem 3: plot point (3,-2)). The graph shows that there is one path that students chose the most: *move 3, turn S, move 2*. This is clearly seen by the thicker line and bigger nodes. A similar path to this one is move 3, turn N (or 90), move -2. One important aspect of this new path is the predominance of turning by a cardinal point (turn N) over turning by an angle (turn 90), and of moving positive distance over negative ones. This pattern can also be identified on the previous two graphs.

One possible conclusion is that students opt to follow what they are usually more comfortable with: positive directions instead of negative ones, and cardinal points instead of angles. The system could explore this information by prompting the user to explore alternative ways of solving a given problem, making students use commands that they would not normally use.

3.3 Analysis Using Video Data and Annotations

4 Discussion

The tool presented was built with the goal of simplifying the process of analyzing log and video data together with the purpose of generating models and understanding user behavior. Through seamless log and video interaction, and the visualizations contained within the behavior graph, we have demonstrated the several conclusions and their applicabilities that could be drawn from the graph before user annotations are inserted, and the extra insight that could be achieved through annotating the graph with video information, and visualizing those annotations on the graph.

However, this was but an initial attempt to tackle such problems. There are ways in which we envision that this software could grow in usefulness to the X project, to others that may need similar features to support their research.

One such way would be to improve how *generalizable* the system is. Currently, it works with X's log structure and assumptions about the states and transitions. This could be fixed by allowing the user to customize the information about states and actions. Suppose that instead of solving geometry problems, a student was required to solve algebra problems. In this case, states could be the current state of an equation, and actions could be algebraic operations. This way, the same benefits obtained in the geometry domain – model creation, identification of common misconceptions, multiple paths to a solution, etc. – could be derived to the algebra domain.

Since the graph represents a sequence of actions made by a student in a given time, allowing a temporal visualization of the graph construction could be meaningful in contributing to analysis. Such visualization would allow the user to go back and forth in time, watching how the graph evolves in time.

Leveraging clustering algorithms could also prove beneficial to this system. Since the tool generates a graph, it would be a natural step to explore this versatile structure and the many automatic ways of analyzing it that have been developed. Some tasks (such as identifying bugs or strategies) could be delegated mostly or entirely to the machine, thus improving the speed through which conclusions could be drawn from the graph.

5 Conclusion

In this paper, we have presented a tool that facilitates analysis by integration data from logs and video in the context of the X system. The features of this tool were presented and demonstrated in a proof of concept using data from a study ran with middle school children. Using the graph generated by the tool, many conclusions and observations could be made, such as identifying strategies, common misconceptions, and opportunities for identifying alternative paths to a solution. Furthermore, the tool provided an environment to encode video information into the log messages, making it richer with information thus allowing more observations to be made from an updated graph. This could prove to be even more useful by making it more customizable to different problems, adding new forms of visualization, and by leveraging the powerful of existing algorithms to automatically extract information.

6 Acknowledgments

This research was funded by NSF 1249406: EAGER: A Teachable Robot for Mathematics Learning in Middle School Classrooms and by the CAPES Foundation, Ministry of Education of Brazil, Brasília - DF 70040-020, Brazil.

References

1. Romero, C., Ventura, S., & García, E. (2008). Data mining in course management systems: Moodle case study and tutorial. *Computers & Education*, *51*(1), 368-384.
2. Aleven, V., Mclaren, B., Roll, I., & Koedinger, K. (2006). Toward meta-cognitive tutoring: A model of help seeking with a Cognitive Tutor. *International Journal of Artificial Intelligence in Education*, *16*(2), 101-128.
3. Arroyo, I., & Woolf, B. P. (2005, May). Inferring learning and attitudes from a Bayesian Network of log file data. In *AIED* (pp. 33-40).
4. Lau, T., & Horvitz, E. (1999). Patterns of search: analyzing and modeling Web query refinement. *COURSES AND LECTURES-INTERNATIONAL CENTRE FOR MECHANICAL SCIENCES*, 119-128.
5. McLaren, B. M., Koedinger, K. R., Schneider, M., Harrer, A., & Bollen, L. (2004). Bootstrapping Novice Data: Semi-automated tutor authoring using student log files.
6. Mackay, W. E. (1989). EVA: An experimental video annotator for symbolic analysis of video data. *Acm Sigchi Bulletin*, *21*(2), 68-71.
7. Harrison, B. L., & Baecker, R. M. (1992, September). Designing video annotation and analysis systems. In *Graphics Interface* (Vol. 92, pp. 157-166).
8. X
9. Jacobs, J. K., Kawanaka, T., & Stigler, J. W. (1999). Integrating qualitative and quantitative approaches to the analysis of video data on classroom teaching.*International Journal of Educational Research*, *31*(8), 717-724.
10. Mostow, J., & Beck, J. (2006). Some useful tactics to modify, map and mine data from intelligent tutors. *Natural Language Engineering*, *12*(2), 195-208.
11. Romero, C., & Ventura, S. (2007). Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications*, *33*(1), 135-146.