Interactive Demonstration of a Generic Tool to Browse Tutor-Student Interactions

Jack Mostow, Joseph Beck, Hao Cen, Evandro Gouvea, and Cecily Heiner
Project LISTEN (www.cs.cmu.edu/~listen, 412-268-1330), Carnegie Mellon University
RI-NSH 4213, 5000 Forbes Avenue, Pittsburgh, PA. USA 15213-3890

Abstract. Project LISTEN's Session Browser is a generic tool to browse a database of students' interactions with an automated tutor. Using databases logged by Project LISTEN's Reading Tutor, we illustrate how to specify phenomena to investigate, explore events and the context where they occurred, dynamically drill down and adjust which details to display, and summarize events in human-understandable form. The tool should apply to MySQL databases from other tutors as well.

1. Introduction

A basic question in mining data from an intelligent tutoring system is, "What happened when…?" We demonstrate a tool to help answer such questions. It lets users specify phenomena, find where they occur in the data, and browse them in human-understandable form. The tool applies to MySQL databases whose representation of tutorial events includes student, computer, start time, and end time [1]. It automatically computes and displays the temporal hierarchy implicit in this representation, as explained in [2]. Here we illustrate the use of this tool to mine data from Project LISTEN's Reading Tutor [3].

2. Select events to explore.

As an example, we focus on a particular student behavior: clicking *Back* out of stories. The Reading Tutor has *Go* and *Back* buttons to navigate to the next or previous sentence in a story. We had previously observed that students sometimes backed out of a story by clicking *Back* repeatedly even after they had invested considerable time in the story. We are interested in understanding what might precipitate this undesirable behavior.

```
select * from story_encounter
where Exit_through = 'user_goes_back'
and (unix_timestamp(End_time) - unix_timestamp(Start_time) > 60)
order by rand()
limit 10
```

Start_Time	End_Time	Machine_Na...	Session_St...	Exit_Through	Stor...	Us
2004-10-20 ...	2004-10-20 ...	LISTEN01-30...	2004-10-20 ...	user_goes_...	New...	fKD5-
2004-12-02 ...	2004-12-02 ...	LISTEN01-32...	2004-12-02 ...	user_goes_...	New...	mTJ5-
2005-01-07 ...	2005-01-07 ...	LISTEN01-27...	2005-01-07 ...	user_goes_...	New...	mMB4
2004-12-08 ...	2004-12-08 ...	LISTEN01-27...	2004-12-08 ...	user_goes_...	CMU...	fTS6-
2005-01-07 ...	2005-01-07 ...	LISTEN01-30...	2005-01-07 ...	user_goes_...	CLO...	mDY5
2004-12-02 ...	2004-12-02 ...	LISTEN01-30...	2004-12-02 ...	user_goes_...	New...	mDF6
2004-11-10 ...	2004-11-10 ...	LISTEN01-33...	2004-11-10 ...	user_goes_...	New...	fEF7-
2005-01-05 ...	2005-01-05 ...	LISTEN01-27...	2005-01-05 ...	user_goes_...	New...	fTW7-
2004-11-18 ...	2004-11-18 ...	LISTEN01-28...	2004-11-18 ...	user_goes_...	CMU...	fDA7-
2005-01-06 ...	2005-01-06 ...	LISTEN01-33...	2005-01-06 ...	user_goes_...	New...	fLR7-

Table: STORY_ENCOUNTER
Start_Time
2004-12-02 08:30:31.609
End_Time
2004-12-02 08:34:17.156
Machine_Name
LISTEN01-321-04
Session_Start_Time
2004-12-02 08:30:28.0
Exit_Through
user_goes_back
Story_Directory
New Story Fall 2003 - Earthworms
User_ID
mTJ5-7-1995-12-03
Student_Level
A
Initiative
student_initiative

Figure 1: a. Query and its resulting table of events. b. Attribute-value list for selected event.

The first step is to find examples of the phenomenon of interest. The query in Figure 1a finds a random sample of 10 stories that students backed out of after spending more than a minute in the story. The resulting table has a column for each field of "story_encounter" in the database, including the start and end times of the story, the name of the computer that recorded it, when the session started, how the user exited the story (by finishing it, backing out, etc.), the name of the story, the user ID of the student, and so on. When the user selects a record in this table, the tool lists its field names and values as shown in Figure 1b. However, this information supports only limited understanding of the event, because it lacks context. As a companion paper [2] explains, the tool computes the context of the event as its temporally enclosing events, in this case the session where the story was read, and the student, whom the tool treats as an infinite time interval when it computes context.

3. Dynamically drill down and adjust which details to include.

How can we generate a dynamic, adjustable-detail view of hierarchical structure in a human-understandable, easily controllable form? We adapted a standard widget for expandable trees. Given a target event, the tool at first displays only its context, i.e., its direct ancestors, omitting other students, sessions, and stories. To see the offspring of the selected story "Earthworms Have An Important Job" or any other event, the user expands it by clicking on the "+" icon to its left. The folder icon marks events not yet fully expanded. Collapsing and re-expanding a partially expanded event reveals its other offspring. Figure 2 shows the result of expanding some details of the event, in particular the sentence encounters preceding the end of the story encounter, back to where the student started backing out of the story, as the Student_Click "user_goes_back" events indicate.

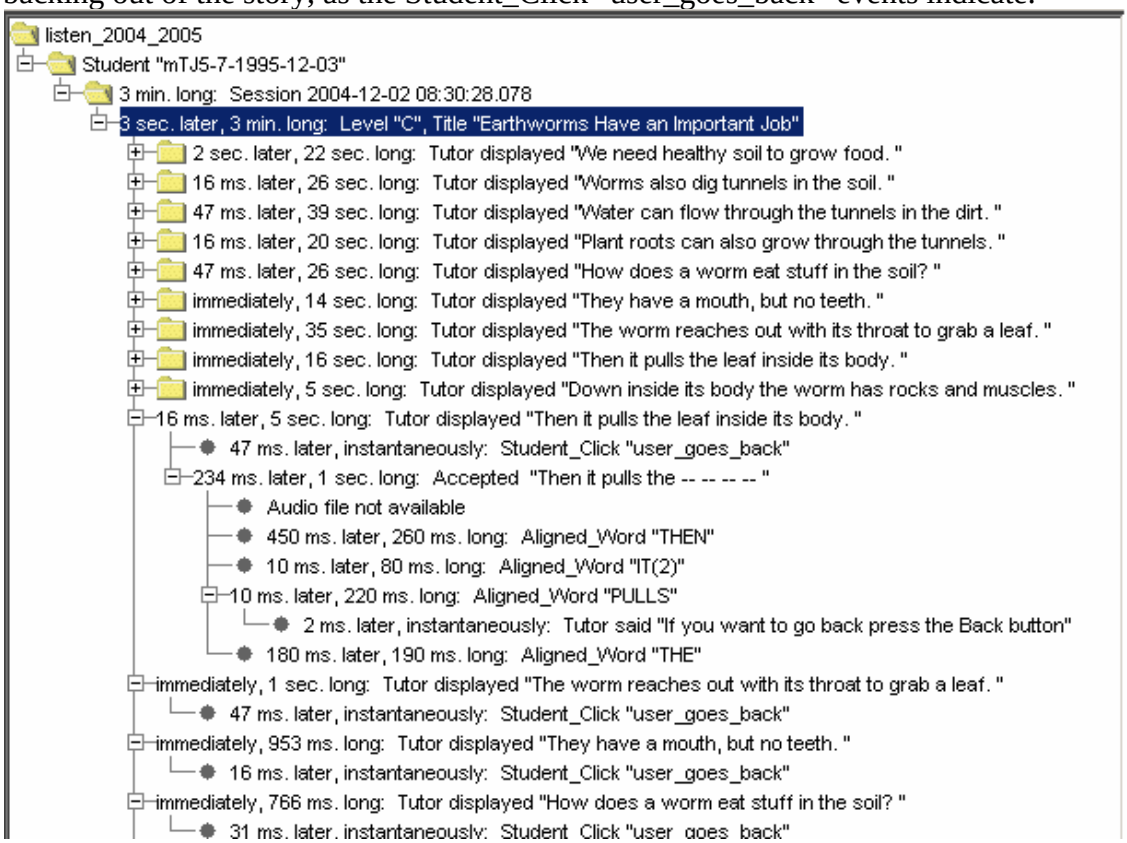

Figure 2: Hierarchical context and partially expanded details of a selected event

Expanding these details revealed a surprise: the Reading Tutor said to click *Back*. The student had clicked above the sentence. This event might mean the student wanted to return to the previous sentence – or that he was trying to click on a word for help but

missed the target. Due to this ambiguity, the Reading Tutor did not respond to "user_clicks_above_sentence" by backing up, but just by saying "If you want to go back, press the Back button." This example suggested the novel hypothesis that the Reading Tutor itself might unintentionally be prompting students to back out of stories!

Besides drilling down as above, the user can specify more globally which types of events to display. The "Pick tables" tab includes a menu that shows which database is currently selected. For instance, "listen_2004_2005" has data for the 2004-05 school year. We assume the database has a different table for each type of event. A checkbox for each table in the currently selected database specifies whether to include that type of event in the event tree. For example, turning on "audio_output" shows speech output by the Reading Tutor, such as "if you want to go back press the Back button."

The checkboxes do not distinguish among events of the same type. For instance, a user might want the event tree to include the tutor's spoken tutorial assistance but not its backchannelling (e.g., "mmm"). User-programmable filters would allow such finer-grained distinctions, but be harder than using check boxes to specify which event types to include.

What steps might subsequent exploration pursue, with what support by the tool?

- Continue browsing this case to try to identify other possible reasons for backing out.
- Check other instances of backing out (by clicking on other events from the table in Figure 1a) to see if the Reading Tutor suggested it.
- Retrieve cases of the same Reading Tutor prompt by formulating a suitable query to enter in the query box (see Figure 1a) to see if the student backed out then as well.
- Develop a query to count how often students back out with vs. without such a prompt. This step constitutes a more quantitative phase of data mining, but the tool lets us inspect 10 randomly chosen cases to check if the query treats them correctly.

4. Summarize events in a human-understandable form.

We have already described the event trees we use to convey the hierarchical structure of tutorial interaction. But how do we summarize individual events?

Temporal properties are common to all events, so we treat them uniformly. An event's absolute start and end times seldom matter except for time of day or time of year effects. Therefore we display them only in the event's attribute-value list, and for a session.

In contrast, the duration of an event is a simple but informative universal measure. For example, the fact that most of the sentence encounters before the student started backing out of the story lasted 14-39 seconds indicates a slow reader. The duration of an event is simply its end time minus its start time.

The hiatus between two events is informative because it reflects user effort, hesitation, confusion, or inactivity. For example, the fact that the hiatuses before *Back* clicks were less than 100 milliseconds long suggests that the student may have been clicking repeatedly as fast as possible. Consider a parent event A with offspring B and C: [A starts … [B starts … B ends] … [C starts … C ends] … A ends]. The hiatus between a parent A and its first offspring B is the start time of B minus the <u>start</u> time of A. The hiatus between successive sibling events B and C is the start time of C minus the <u>end</u> time of B.

Precise times seldom matter for a duration or hiatus, so for readability and brevity, we display only the largest non-zero units (days, hours, minutes, seconds, milliseconds).

The complete attribute-value list for an event occupies considerable screen space, and is displayed only for the currently selected event. In contrast, the tool displays all the one-line summaries for an event tree at once. What information should such summaries include? How should it be displayed? How should it be computed?

The answers depend on the type of event. We observed that although the Reading Tutor's database schema has evolved over time, the meaning of table names is nevertheless

consistent across successive versions and between databases created by different members of Project LISTEN. Therefore we wrote one method for each table to translate a record from that table into a one-line string that includes whatever we think is most informative. Ideally these methods are simple enough for users (educational data miners) to modify to suit their own preferences. The default string for a table without such a method is just the name of the table, e.g., "Session" or "Story_encounter." Most methods just display one or more fields of the record for the event. For example, the method for a session just shows its start time. Some methods incorporate information from other tables. For example, the method for a story encounter retrieves its title from a separate table. Special-purpose code adds a node the user can click to play back a recorded utterance, or displays "Audio not available" if its audio file has not yet been archived (as in the case of the December 2004 example shown here).

5. Evaluation

Relevant criteria for evaluating this work include implementation cost, efficiency, generality, usability, and utility. Implementation cost was only several person-weeks for a tutor-specific prototype, and about the same for its generic, time-interval-based successor.

Using ordinary PCs for the database server and the session browser to explore databases for hundreds of students, thousands of hours of interaction, and millions of words, the operations reported here usually update the display with no perceptible lag, though a complex query to find a specified set of events may take several seconds or more.

Structural evidence of generality includes the tool's predominantly tutor-independent design, reflected in the code's brevity and its scarcity of references to specific tables or fields of the database. Empirical evidence of generality includes successful use of the tool with databases from different years' versions of the Reading Tutor. We have not as yet tested it on databases from other groups. Typically ITSs still log to files, not databases.

It is early to evaluate usability or utility because the tool is still so new. We have not conducted formal usability tests on its initial target users, namely Project LISTEN researchers engaged in educational data mining. However, we can claim a ten- or hundred-fold reduction in keystrokes compared to obtaining the same information by querying the database directly. For example, clicking on an item in the event list displays its context as a chain of ancestor events. Identifying these ancestors by querying the database directly would require querying a separate table for each ancestor.

Acknowledgements: This work was supported in part by the National Science Foundation under ITR/IERI Grant No. REC-0326153. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation or the official policies, either expressed or implied, of the sponsors or of the United States Government. We thank the educators and students who generated our data, and our colleagues who contributed to developing the Session Browser.

References (see www.cs.cmu.edu/~listen)

1. Mostow, J., J. Beck, R. Chalasani, A. Cuneo, and P. Jia. Viewing and Analyzing Multimodal Human-computer Tutorial Dialogue: A Database Approach. *Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces (ICMI 2002)*, 129-134. 2002. Pittsburgh, PA: IEEE.
2. Mostow, J., J. Beck, A. Cuneo, E. Gouvea, and C. Heiner. A Generic Tool to Browse Tutor-Student Interactions: Time Will Tell! *Proceedings of the 12th International Conference on Artificial Intelligence in Education (AIED 2005)* 2005. Amsterdam.
3. Mostow, J., G. Aist, P. Burkhead, A. Corbett, A. Cuneo, S. Eitelman, C. Huang, B. Junker, M.B. Sklar, and B. Tobin. Evaluation of an automated Reading Tutor that listens: Comparison to human tutoring and classroom instruction. *Journal of Educational Computing Research*, 2003. *29*(1): p. 61-117.