An Educational Data Mining Tool to Browse Tutor-Student Interactions: Time Will Tell!

Jack Mostow, Joseph Beck, Hao Cen, Andrew Cuneo, Evandro Gouvea, and Cecily Heiner

Project LISTEN (www.cs.cmu.edu/~listen), Carnegie Mellon University
RI-NSH 4213, 5000 Forbes Avenue, Pittsburgh, PA, USA 15213-3890

mostow, joseph.beck, hcen, acuneo, egouvea, cecily @cs.cmu.edu

Abstract

A basic question in mining data from an intelligent tutoring system is, "What happened when…?" We identify requirements for a tool to help answer such questions by finding occurrences of specified phenomena and browsing them in human-understandable form. We describe an implemented tool and how it meets the requirements. The tool applies to MySQL databases whose representation of tutorial events includes student, computer, start time, and end time. It automatically computes and displays the temporal hierarchy implicit in this representation. We illustrate the use of this tool to mine data from Project LISTEN's automated Reading Tutor.

1. Introduction

Intelligent tutoring systems' ability to log their interactions with students poses both an opportunity and a challenge. Compared to human observation of live or videotaped tutoring, such logs can be more extensive in the number of students, more comprehensive in the number of sessions, and more exquisite in the level of detail. They avoid observer effects, cost less to obtain, and are easier to analyze. The resulting data is a potential gold mine (Beck, 2004) – but mining it requires the right tools to locate promising areas, obtain samples, and analyze them.

Educational data mining is an iterative cycle of hypothesis formation, testing, and refinement that alternates between two complementary types of activities. One type of activity involves aggregate quantitative analysis of many tutorial events. For example, knowledge tracing (Corbett & Anderson, 1995) analyzes growth curves by aggregating over successive opportunities to apply a skill. Embedded experiments (Aist, 2001; Mostow & Aist, 2001; Mostow, Beck, Bey et al., 2004; Mostow, Beck, & Heiner, 2004) compare alternative tutorial actions by selecting randomly among them and aggregating outcomes of many trials.

In contrast, qualitative analysis focuses in depth on understanding individual tutorial events. The research question it addresses is descriptive (Shavelson & Towne, 2002): "What happened when …?" Such case analyses serve any of several purposes, for example:

- Spot-check tutoring sessions to discover undesirable tutor-student interactions.
- Identify the most common types of cases in which a specified phenomenon occurs.
- Formulate hypotheses by identifying features that examples suggest are relevant.
- Sanity-check a hypothesis by checking that it covers the intended sorts of examples.

This paper describes an educational data mining tool to support such case analysis by exploiting three simple but powerful ideas. First, a student, computer, and time interval suffice to specify an event. Second, a containment relation between time intervals defines a hierarchical structure of tutorial interactions. Third, the first two ideas make it possible to implement a generic but flexible tool for mining tutor data with minimal dependency on tutor-specific details.

The paper is organized as follows. Section 2 discusses previous work and the requirements it suggests for such a tool. Section 3 describes an implemented tool and how it meets each requirement, using a real example based on interactions with Project LISTEN's Reading Tutor. Section 5 concludes by summarizing the papers' contributions.

2. Previous work

Intelligent tutoring systems commonly record their interactions in the form of log files. Log files are easy to record, flexible in what information they can capture, (sometimes) human-understandable to read, and useful in debugging. However, they are unwieldy to aggregate across multiple sessions and computers, and difficult to parse and analyze in ways not anticipated when they were designed (Mostow & Aist, 2001). Consequently, we have found that logging interactions directly to a database makes such analysis easier, more flexible, less bug-prone, and more powerful than analyzing conventional log files (Mostow, Beck, Chalasani, Cuneo, & Jia, 2002).

SentenceEncounter List					
StartTime	Duration	Num Actions	Num Utterances	Total Action	SentenceStr
04-05-2001 12:24:25.693	00:00:01.412	3	0	0	OVEN
04-05-2001 12:24:27.105	00:00:01.542	3	0	0	BATTER
04-05-2001 12:24:28.677	00:00:44.23	47	4	51	First get the batter
04-05-2001 12:25:12.700	00:00:24.886	20	4	24	Next put all the ingredients in
04-05-2001 12:25:37.857	00:00:33.908	3	2	5	Then put it in the oven
04-05-2001 12:26:11.765	00:00:40.539	3	3	6	Last eat them

Figure 1: Example from (Mostow et al., 2002a)

Figure 1 shows an earlier program (Mostow et al., 2002) intended to support case analysis by browsing interactions logged by Project LISTEN's Reading Tutor. The program displayed a table of tutorial interactions at a user-selected level of detail – computers, launches, students, sessions, stories, sentences, utterances, words, or clicks. It computed each table on demand using multiple queries requiring database joins. Hyperlinks in this table let the user drill down to a table at the next deeper level of detail.

However, this program suffered from several limitations. It was specific to a particular version of the Reading Tutor. It displayed a list of records as a standard HTML table, which was not necessarily human-understandable. Navigation was restricted to drilling down from the top-level list of students or tutors, with no convenient way to specify a particular type of interaction to explore, and no visible indication of context. Although tables are useful for comparing events of the same type, they are ill-suited to conveying the heterogeneous set of events that transpired during a given interaction, or the context in which they occurred.

2.1. Requirements for browsing tutorial interactions

How should researchers explore logged interactions with an intelligent tutor? Our previous experience suggests the following requirements for the content to display (1-2), the interface to display it (3-4), and the architecture to implement the tool (5):

1. Specify which phenomenon to explore.
2. Explore selected events and the context in which they occurred.
3. Dynamically drill down and adjust which details to display.
4. Summarize interactions in a human-understandable form.
5. Require minimal effort to adapt the tool to new versions, to new users, or to other tutors.

3. Approach, illustrated by running example

Project LISTEN's Reading Tutor listens to children read aloud, and helps them learn to read. A Reading Tutor session consists of reading a number of stories. The Reading Tutor displays a story one sentence at a time, and records the child's utterances for each sentence. The Reading Tutor logs each event (session, story, sentence, utterance, …) into a database table for that event type. Data from tutors at different schools flows into an aggregated database on our server. For example, our 2003-2004 database includes 54,138 sessions, 162,031 story readings, 1,634,660 sentences, 3,555,487 utterances, and 10,575,571 words.

Screenshots of the Reading Tutor itself appear elsewhere (Mostow, Beck, Bey et al., 2004); here we focus on the tool – a Java™ program that queries a MySQL database server (MySQL, 2004), both running on ordinary PCs. We now explain how this tool achieves the requirements listed in Section 2.1.

3.1. Specify which phenomenon to explore.

First, how can we specify events to explore? A deployed tutor collects too much data to look at, so the first step in mining it is to select a sample. A database query language provides the power and flexibility to describe and efficiently locate phenomena of interest. For example, the query "`select * from utterance order by rand() limit 10`" selects a random sample of 10 from the table of student utterances. Whether the task is to spot-check for bugs, identify common cases, formulate hypotheses, or check their sanity, our mantra is "check (at least) ten random examples." Random selection assures variety and avoids the sample bias of,

for example, picking the first ten examples in the database.

Although an arbitrary sample like this one is often informative, a query can focus on a particular phenomenon of interest, such as the set of questions that students took longest to answer, or steps where they got stuck long enough for the Reading Tutor to prompt them. Exploring examples of such phenomena can help the educational data miner spot common features and formulate causal hypotheses to test with statistical methods on aggregated data.

Our running example focuses 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_Thr
2004-10-20 ...	2004-10-20 ...	LISTEN01-30...	2004-10-20 ...	user_goe
2004-12-02 ...	2004-12-02 ...	LISTEN01-32...	2004-12-02 ...	user_goe
2005-01-07 ...	2005-01-07 ...	LISTEN01-27...	2005-01-07 ...	user_goe
2004-12-08 ...	2004-12-08 ...	LISTEN01-27...	2004-12-08 ...	user_goe
2005-01-07 ...	2005-01-07 ...	LISTEN01-30...	2005-01-07 ...	user_goe
2004-12-02 ...	2004-12-02 ...	LISTEN01-30...	2004-12-02 ...	user_goe
2004-11-10 ...	2004-11-10 ...	LISTEN01-33...	2004-11-10 ...	user_goe
2005-01-05 ...	2005-01-05 ...	LISTEN01-27...	2005-01-05 ...	user_goe
2004-11-18 ...	2004-11-18 ...	LISTEN01-28...	2004-11-18 ...	user_goe
2005-01-06 ...	2005-01-06 ...	LISTEN01-33...	2005-01-06 ...	user_goe

Figure 2: Query and its resulting table of events

The query in Figure 2 finds a random sample of 10 stories that students backed out of after spending more than a minute in the story. The columns of this table correspond to the fields for "story_encounter" (a reading of a story) 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.

Second, what information suffices to identify a tutorial interaction so a tool can explore it? A key insight here is that student, computer, and time interval are enough, because together they uniquely specify the student's interaction with the tutor during that time interval. (We include computer ID in case the student ID is not unique.) This "lowest common denominator" should apply universally to virtually any tutor, though the end time of the interval might be problematic for a web-based tutor that doesn't know when the student leaves a page.

Third, how can we translate the result of a query into a set of tutorial events? The tool scans the labels returned as part of the query, and finds the columns for student, computer, start time, and end time. The code assumes particular names for these columns, e.g. "user_id" for student, "machine_name" for computer, and "start_time" for start time. If necessary the user can enforce this naming convention, e.g., by inserting "as start_time" in the query to relabel the column. We require that the fields for student, computer, start time, and end time be keys in the database tables. Indexing tables on these fields enable fast response by the tool even for tables with millions of records.

3.2. Explore selected events and the context in which they occurred.

The tool lists field names and values for a selected record as shown in Figure 3. However, this information supports only limited understanding of the event, because it lacks context.

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 3: Attribute-value list for selected event

What is the context of an event? Our answer is: "its chain of ancestors." For example, the ancestors of the selected story encounter summarized in Figure 3 are the session in which it occurred, and the student in that session. Figure 4 summarizes this context.

Figure 4: Hierarchical ontext of selected event

How can we discern the hierarchical structure of student-tutor interaction? At first we computed this hierarchy using its hardwired schema for the Reading Tutor database to determine which events are part of which others. But then we had a key insight: exploit the natural hierarchical structure of nested time intervals.

If events A and B have the same student and computer, when is A an ancestor of B? We initially required that A contain all of B. But we relaxed the criterion to better handle occasional overlapping intervals in our data. We therefore define A as an ancestor of B if B starts during A.

The tool computes the event tree shown in Figure 4 (and in Figure 5 below) by partial-ordering the events according to the transitive closure of this ancestor relation. The parent of an event is defined as its minimal ancestor. Siblings are defined as sharing the same parent, and are ordered by their start times.

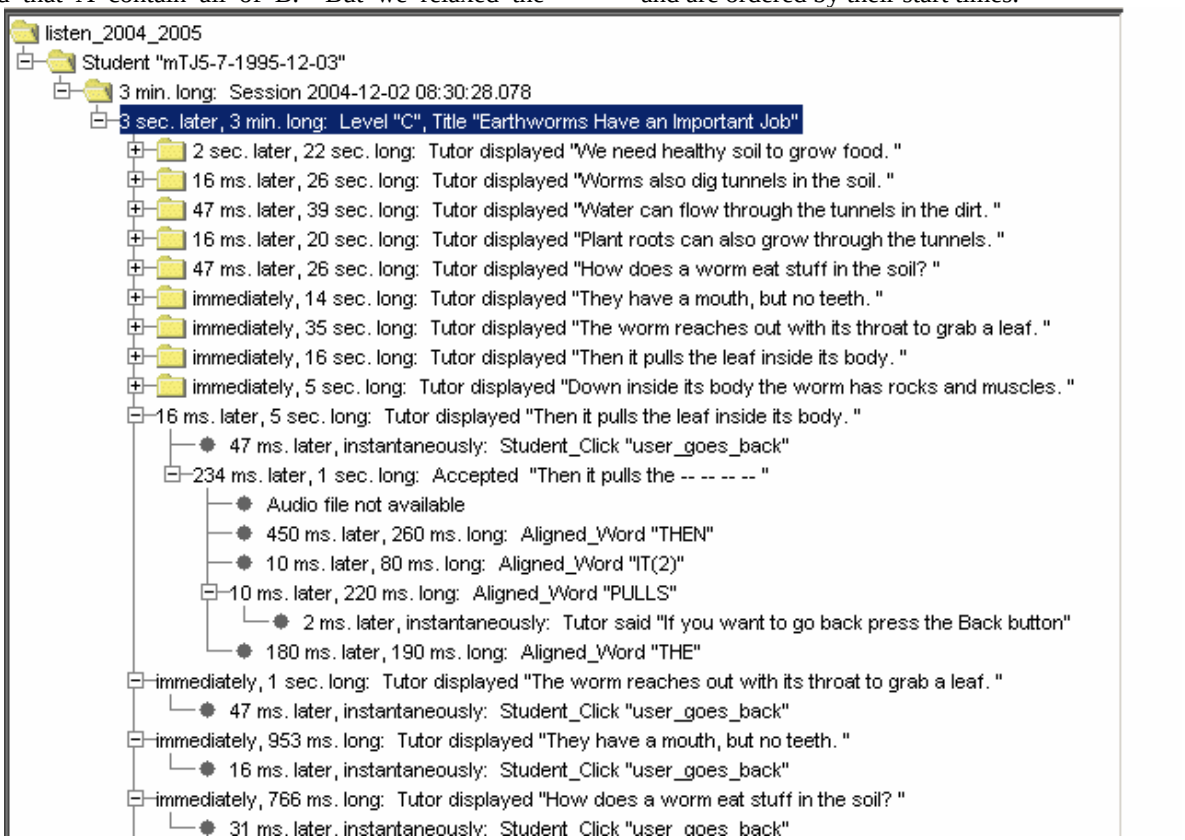

Figure 5: Hierarchical context and partially expanded details of a selected event

3.3. 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 5 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.

Expanding these details revealed a surprise: the Reading Tutor said to click *Back*. The student had clicked above the sentence. This event might mean a student wants to return to the previous sentence – or that he is trying to click on a word for help but missed the target. Due to this ambiguity, the Reading Tutor does 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!

What steps might subsequent data mining 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 3) 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 2) 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 return to the quantitative phase of data mining, but the tool can help test whether the query treats 10 randomly chosen cases correctly.

The focus of this paper is not this particular example but the tool, to which we now return.

Besides drilling down as above, we let the user specify more globally which types of events to display. Figure 6 shows the "Pick tables" tab. The left side shows which database is currently selected. We assume the database has a different table for each type of event. A checkbox for each table in the current 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."

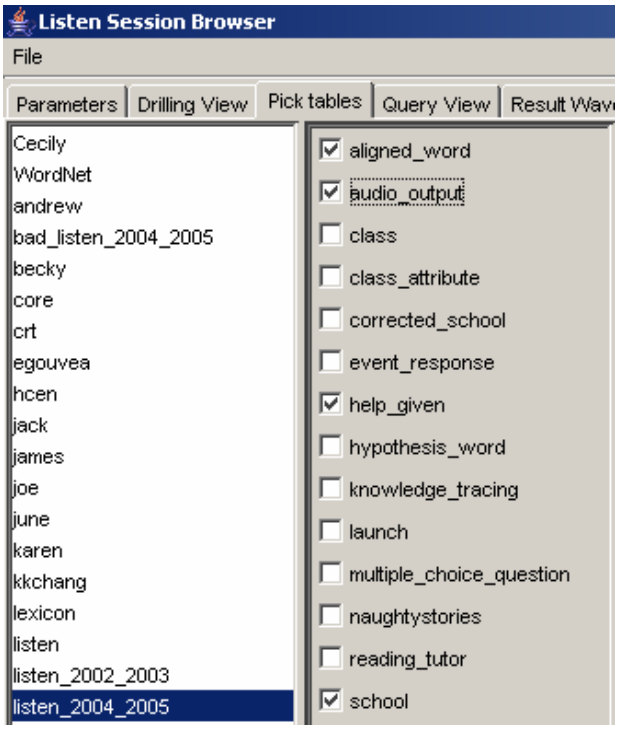

Figure 6: Select database and tables

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.

3.4. Summarize events in 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. The hiatus between a parent event A and its first offspring B is the start time of B minus the start time of A. The hiatus between two successive sibling events B and C is the start time of C minus the end 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 function 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 functions are simple enough for users (educational data miners) to modify to suit their own preferences. The default string for a table without such a function is just the name of the table, e.g., "Session" or "Story_encounter." Most functions just display one or more fields of the record for the event. For example, the function for a session just shows its start time. Some functions incorporate information from other tables. For example, the function 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).

3.5. Require minimal effort to adapt the tool to new versions, to new users, or to other tutors.

How can the tool obtain the information it needs about a database of tutor interactions? Its generic architecture enables it to make do with readily available meta-data, a few assumed conventions, and a little code. MySQL provides the required meta-data, namely the list of tables in the database, the fields in each table and event list, and their names and data types. We exploit the observation (or assumption) that the meaning of field names is consistent across database tables and over time. The code assumes particular field names for student, machine, and start and end times, but overrides this convention when necessary, as in the case of a particular table with a "Time" field instead of a "Start_time" field.

The method to compute the context of a selected target event is: First, extract its student, computer, and start time. Then query every table of the database for records for the same student and computer whose time interval contains the start of the target event. Finally, sort the retrieved records according to the ancestor relation, and display them accordingly by inserting them in the appropriate positions in the expandable tree widget.

The method to find the children of a given event fires only when needed to expand the event node. It finds descendants in much the same way as the method to find ancestors, but then winnows them down to the children (those that are not descendants of others).

A more knowledge-based method would know which types of Reading Tutor events can be parents of which others. However, this knowledge would be tutor- and possibly version-specific. In contrast, our brute force solution of querying all tables requires no such knowledge. Moreover, its extra computation is not a problem in practice. Our databases consist of a few dozen tables, the largest of which have tens of millions of records. Despite this table size, the tool typically computes the context of an event with little or no delay.

4. Future Work

Future work includes integrating the session browser with other educational data mining methods, and exploring its value to more users.

4.1. Integration with other methods

The Data Shop team at the Pittsburgh Science of Learning Center (www.learnlab.org) plans to incorporate the session browser in the suite of educational data mining tools it is developing to analyze data logged by tutors. The contextual inquiries performed by this team identified the need to understand context as an important aspect of analyzing data from tutors. The session browser should help researchers understand the context of data by exploring specific examples.

For example, one Data Shop tool will compute a learning curve for a given skill by aggregating over students' successive opportunities to apply that skill. A spike in a learning curve suggests a bug in the underlying cognitive model of skills. Integrating the session browser into the tool suite will facilitate analyzing such spikes by inspecting example steps for clues as to what made those steps harder than the model predicted.

Integrating the session browser with tools for analyzing and visualizing aggregated data will facilitate exploiting the synergy between quantitative and qualitative analysis. However, it is not clear that methods for visualizing aggregated data apply in any useful way to the individual cases explored by the session browser.

One reason is obvious: visualization methods apply to large sets of data and exploit their regularity. In contrast, the individual cases explored by the session browser are limited in scope and heterogeneous in structure (comprised of multiple event types).

Another reason is less obvious: visualization methods apply to quantitative features of data. But aside from temporal information, the features that provide useful context for tutorial events are largely qualitative rather than quantitative – what text the student was reading, what the tutor said, what the student did, and so forth. In order for visualization methods to apply usefully to such data, they will presumably need to be translated into quantitative form. Methods that map qualitative data into quantitative form by aggregation, such as histogramming, may have limited relevance to analyzing individual cases.

4.2. Expansion to other users

Who will find the session browser useful?

The first candidates are the researchers developing and evaluating Project LISTEN's Reading Tutor. It is not a foregone conclusion that we will find the session browser useful. After all, our previous attempt turned out to be too constraining, as Section 2 explained, motivating the requirements in Section 2.1.

The next candidates are researchers interested in analyzing data from other tutors. Incorporating the session browser into the Data Shop will test its utility for this purpose.

Will teachers find the session browser useful? Probably not – unless three challenges are overcome.

First, teachers are too busy to spend much if any time poring over detailed traces of their students' behavior. Data and analyses can be useless to them without prescriptions for what to do about it (Fuchs, Fuchs, Hamlett, & Ferguson, 1992). To warrant teachers spending time using the session browser, it would have to quickly deliver information they found useful.

Second, understanding data logged by a tutor tends to require intimate familiarity with how the tutor works. The Data Shop team's interviews with other researchers confirmed that this issue is not specific to our data. Unless this problem is solved, tools for mining data from a tutor may be useful only to its developers, or to a few intrepid researchers willing to invest the effort required to gain such familiarity. The session browser may help ameliorate this difficulty to the extent that it makes it easy to write tutor-specific methods to generate self-explanatory event summaries, but it is unrealistic to expect them to solve the whole problem.

Third, the session browser is designed more for power and flexibility than for simplicity. For instance, the mechanism to specify which events to explore is very general, but requires the ability to write a MySQL query. Accommodating less skilled users would require providing useful "pre-canned" queries or scaffolding the query formulation process.

5. Conclusion

This paper reports an implemented, efficient, generic solution to a major emerging problem in educational data mining: efficient exploration of vast student-tutor interaction logs. We identify several useful requirements for a tool to support such exploration. Our key conceptual contribution uses temporal relations to expose natural hierarchical structure. This is the sense in which "time will tell" many basic relationships among tutorial events.

The success of this approach suggests specific recommendations in designing databases of tutorial interactions: Log each distinct type of tutorial event in its own table. Include student ID, computer, start time, and end time as fields of each such table so as to identify its records as events. Name these fields consistently within and across databases created by successive versions of the tutor so as to make them easier to extract. Adding new tables and fields is fine, but keep the names of the old ones to reuse their display code.

Relevant criteria for evaluating this work include implementation cost, efficiency, generality, usability, and utility. Implementation cost was only several person-weeks for the tutor-specific prototype and about the same for its generalized 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; most tutors still log to files, not databases.

It is early to evaluate usability 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.

As for utility, the ultimate test of this tool is whether it leads to useful discoveries, or at least sufficiently facilitates the process of educational data mining that the miners find it helpful and keep using it. To repeat our subtitle in its more usual sense, "time will tell!"

Acknowledgements: This paper is an extended version of an AIED2005 poster and demonstration (Mostow, Beck, Cen, Gouvea, & Heiner, 2005; Mostow, Beck, Cuneo, Gouvea, & Heiner, 2005). 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.

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

Aist, G. (2001). Towards automatic glossarization: Automatically constructing and administering vocabulary assistance factoids and multiple-choice assessment. *International Journal of Artificial Intelligence in Education, 12*, 212-231.

Beck, J. (Ed.). (2004). *Proceedings of the ITS2004 Workshop on Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes*. Maceio, Brazil.

Corbett, A., & Anderson, J. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction, 4*, 253-278.

Fuchs, L. S., Fuchs, D., Hamlett, C. L., & Ferguson, C. (1992). Effects of expert system consultation within curriculum-based measurement using a reading maze task. *Exceptional Children, 58*(5), 436-450.

Mostow, J., & Aist, G. (2001). Evaluating tutors that listen: An overview of Project LISTEN. In K. Forbus & P. Feltovich (Eds.), *Smart Machines in Education* (pp. 169-234). Menlo Park, CA: MIT/AAAI Press.

Mostow, J., Beck, J., Bey, J., Cuneo, A., Sison, J., Tobin, B., & Valeri, J. (2004). Using automated questions to assess reading comprehension, vocabulary, and effects of tutorial interventions. *Technology, Instruction, Cognition and Learning, 2*, 97-134.

Mostow, J., Beck, J., Cen, H., Gouvea, E., & Heiner, C. (2005, July). *Interactive Demonstration of a Generic Tool to Browse Tutor-Student Interactions.* Supplemental Proceedings of the 12th International Conference on Artificial Intelligence in Education (AIED 2005), Amsterdam

Mostow, J., Beck, J., Chalasani, R., Cuneo, A., & Jia, P. (2002, June 4). *Viewing and Analyzing Multimodal Human-computer Tutorial Dialogue: A Database Approach.* Proceedings of the ITS 2002 Workshop on Empirical Methods for Tutorial Dialogue Systems, San Sebastian, Spain, 75-84.

Mostow, J., Beck, J., Cuneo, A., Gouvea, E., & Heiner, C. (2005, July). *A Generic Tool to Browse Tutor-Student Interactions: Time Will Tell!* Proceedings of the 12th International Conference on Artificial Intelligence in Education (AIED 2005), Amsterdam

Mostow, J., Beck, J. E., & Heiner, C. (2004, June 27-30). *Which Help Helps? Effects of Various Types of Help on Word Learning in an Automated Reading Tutor that Listens.* Eleventh Annual Meeting of the Society for the Scientific Study of Reading, Amsterdam, The Netherlands

MySQL. (2004). *Online MySQL Documentation.* Retrieved, from the World Wide Web: http://dev.mysql.com/doc/mysql

Shavelson, R. J., & Towne, L. (Eds.). (2002). *Scientific Research in Education*. National Research Council, Washington, D.C.: National Academy Press.