Why, What, and How to Log? Lessons from LISTEN

Jack Mostow[1] and Joseph E. Beck[2]
mostow@cs.cmu.edu, joseph.beck@EducationalDataMining.org
[1]Project LISTEN, Robotics Institute, Carnegie Mellon University
[2]Computer Science Department, Worcester Polytechnic Institute

The ability to log tutorial interactions in comprehensive, longitudinal, fine-grained detail offers great potential for educational data mining – but what data is logged, and how, can facilitate or impede the realization of that potential. We propose guidelines gleaned over 15 years of logging, exploring, and analyzing millions of events from Project LISTEN's Reading Tutor and its predecessors.

1 Introduction

There is presumably unanimous consensus in the EDM community that it is useful to log tutor data. However, there is far from a consensus about which data to log, or how. As a step toward this goal, this paper attempts to distill lessons from over fifteen years of experience in logging and mining data from Project LISTEN's Reading Tutor [1] and its predecessor [2]. We cite relevant publications about that work, but work on logging tutor data in other projects is outside the scope of this paper and hence not cited here. Rather we describe some guidelines we have developed along the way. We do not claim they are the best possible way to log tutor data, only that we have found them sufficiently helpful in our project to recommend them in considering why, what, and how to log.

The Reading Tutor uses speech recognition to listen to children read aloud, and helps them learn to read [3-8]. It logs its interactions with students throughout the school year in fine-grained, comprehensive detail, logging – and timestamping – every launch of the tutor, session with a student, story read in whole or part, text sentence displayed, multiple choice question and response, utterance by student or tutor, word recognized, keyboard input, and mouse click. The Reading Tutor logs this data directly to a relational database. The school server then forwards the data overnight to an aggregated database in our lab.

2 Why to log

Project LISTEN illustrates multiple purposes that logging tutor data can serve.

2.1 Tutoring

Besides logging data for later analysis, a tutor can query logged data itself at runtime. The Reading Tutor queries 12 of the 32 database tables that it logs:

- Enrollment and login use tables of classes and students.
- Usage tracking queries tables of launches and sessions, e.g. to track how long the student used the Reading Tutor so far today, which it displays on the screen.
- The database keeps the student's place even if the student logs out or the tutor exits or crashes [9]. The story choice mechanism uses the table of story

encounters to bookmark the student's position in a story, and to remember whose turn it is to pick a story next.

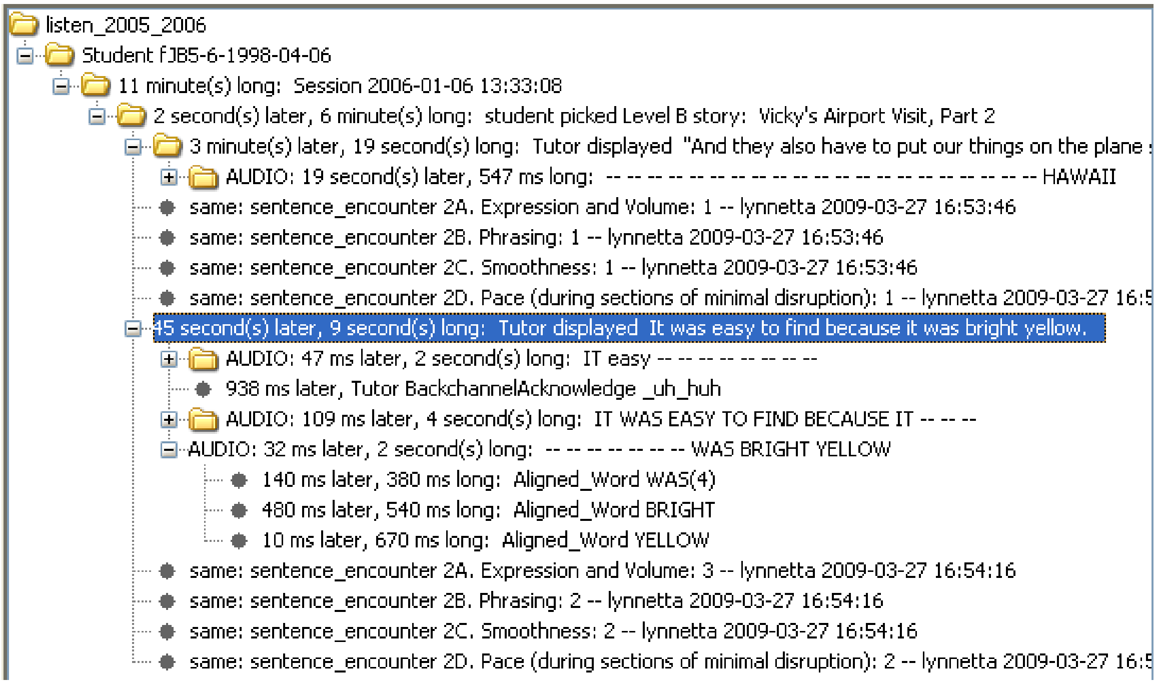
- The tables for the student model determine which level of stories to pick, when to promote or demote the student based on oral reading fluency, which letter-sound mappings to teach, and which words to practice.

2.2 Reporting

A separate program used the database to generate various types of password-protected, web-accessible reports on demand. A report for teachers summarized students' usage and progress [10]. A report for technical support staff computed usage and reliability at each school, and flagged possible technical problems indicated by falloff in usage.

2.3 Browsing

Project LISTEN's Session Browser [11, 12] enables researchers to find examples of tutorial interactions with specified characteristics, display them in human-understandable form, explore them in dynamically variable detail, and annotate them, as Figure 1 shows.



```
listen_2005_2006
  Student fJB5-6-1998-04-06
    11 minute(s) long:  Session 2006-01-06 13:33:08
      2 second(s) later, 6 minute(s) long:  student picked Level B story:  Vicky's Airport Visit, Part 2
        3 minute(s) later, 19 second(s) long:  Tutor displayed  "And they also have to put our things on the plane :
          AUDIO: 19 second(s) later, 547 ms long:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- HAWAII
        same: sentence_encounter 2A. Expression and Volume: 1 -- lynnetta 2009-03-27 16:53:46
        same: sentence_encounter 2B. Phrasing: 1 -- lynnetta 2009-03-27 16:53:46
        same: sentence_encounter 2C. Smoothness: 1 -- lynnetta 2009-03-27 16:53:46
        same: sentence_encounter 2D. Pace (during sections of minimal disruption): 1 -- lynnetta 2009-03-27 16:5
        45 second(s) later, 9 second(s) long:  Tutor displayed  It was easy to find because it was bright yellow.
          AUDIO: 47 ms later, 2 second(s) long:  IT easy -- -- -- -- -- -- -- --
            938 ms later, Tutor BackchannelAcknowledge _uh_huh
          AUDIO: 109 ms later, 4 second(s) long:  IT WAS EASY TO FIND BECAUSE IT -- -- --
          AUDIO: 32 ms later, 2 second(s) long:  -- -- -- -- -- -- -- -- WAS BRIGHT YELLOW
            140 ms later, 380 ms long:  Aligned_Word WAS(4)
            480 ms later, 540 ms long:  Aligned_Word BRIGHT
            10 ms later, 670 ms long:  Aligned_Word YELLOW
        same: sentence_encounter 2A. Expression and Volume: 3 -- lynnetta 2009-03-27 16:54:16
        same: sentence_encounter 2B. Phrasing: 2 -- lynnetta 2009-03-27 16:54:16
        same: sentence_encounter 2C. Smoothness: 2 -- lynnetta 2009-03-27 16:54:16
        same: sentence_encounter 2D. Pace (during sections of minimal disruption): 2 -- lynnetta 2009-03-27 16:5
```

Figure 1: Event tree displayed by Project LISTEN's Session Browser

The event highlighted in the tree is a 9-second-long encounter of the sentence *It was easy to find because it was bright yellow*, 45 seconds after the previous sentence encounter, which lasted 19 seconds and occurred 3 minutes into a 6-minute encounter of the Level B (grade 2) story "Vicky's Airport Visit, Part 2," which the student picked 2 seconds after starting an 11-minute-long session at 1:33pm on January 1, 2006. A human annotator later rated both sentences on a scale from 1 to 4 for four items of a fluency rubric [13]. The second sentence encounter has been expanded here to reveal three utterances, the last of which is expanded to show the text words *was bright yellow* aligned against the speech recognizer output. The Session Browser derives the tree structure from the temporal relationships among the events stored in the database.

2.4 Mining

Researchers query Reading Tutor databases from different school years for educational data to mine. These queries are shorter, faster to write, easier to understand, and less error-prone than the scripts used to analyze tutor log files [1]: "For example, analysis of the preemptive assistance experiment was hampered by the absence of a single uniform log entry for the generic event "give help on a word." Instead, an *ad hoc* script had to recover this implicit event by aggregating over the more specific events logged for each specific type of help, and for the Reading Tutor's spoken outputs. This script is sufficiently complex to doubt its correctness."

3 What to log

To put tutor data logging in perspective, it is helpful to think of the tutor and student as each sensing, thinking, and acting within the wider environment surrounding them – physically, socially, institutionally, culturally, and so forth. This environment includes the machine on which the tutor runs, the classroom and school where it's located, the student's classmates and teacher(s), and so on.

We can use this framework to characterize logged information. To illustrate with authentic concrete examples, we use it to categorize database tables logged by the 2007-2008 version of the Reading Tutor – as well as additional information that might be useful to log, but that we currently do not.

3.1 Environmental data

A "Candid Camera" analysis of video recorded by a camera mounted on a Reading Tutor monitor [14] showed that a tutor equipped with computer vision could potentially capture useful information about its physical and social context, such as whether a student is present or absent, looking at the screen or off-task, alone or with classmates, and working independently or interacting with a teacher. However, at present a typical tutor has limited if any ability to observe its environment directly.

In the absence of such observations, a tutor may input information about the student and context directly from the student, teacher, or other sources. Information logged about the environment may be a static fact (seldom or never updated), a maintained value (updated more or less frequently), or a timestamped event (appended to the ongoing historical record of the interaction). The 2007-2008 Reading Tutor's static environmental data included listed 11 schools, 37 classes, and 540 users (446 with at least one session). Its maintained data included 960 stories, updated when students added stories.

3.2 Raw input

A tutor's sensors are typically limited to keyboard, mouse, and microphone, enabling it to observe a subset of the student's actions. 2007-2008 raw input included 458,984 mouse clicks, 491,095 oral reading utterances, and 9,188 free-form spoken responses.

3.3 Interpreted input

A tutor's input observations go through layers of interpretation. For instance, the 2007-2008 Reading Tutor asked 105,223 multiple choice questions, of which 5,158 had a designated right answer. It interpreted 3,988 of these 5,158 responses as correct, 1,009 as incorrect, and 161 as null due to timing out, logging out, or crashing.

Similarly, the speech recognizer interpreted the 476,713 oral reading utterances as 1,747,259 spoken words, against which the Reading Tutor aligned 3,550,641 text words to interpret them as read, misread, or omitted.

Since the Reading Tutor's speech recognizer is imperfect, we rely on later manual transcription when we want to know what the student actually said. However, due to its cost, we transcribe only selectively. For our research on teaching reading strategies, we transcribe only students' free-form spoken responses to comprehension prompts [15].

Human interpretation of recorded speech is not restricted to transcribing it. For instance, an expert reading teacher annotated the transcribed student answers with how she would have responded to each answer, and why. Likewise, for an analysis of students' oral reading prosody, two annotators independently rated a sample of 200 recorded oral reading utterances on a 4-point fluency rubric [13].

3.4 Student model

A tutor cannot observe students' thinking directly, but may infer student models from observed student behavior. Fine-grained models represent estimates of individual skills. The 2007-2008 Reading Tutor tracked exposure to vocabulary by maintaining 265,579 counts of how many times 480 students saw or typed 16,932 distinct words. It tracked individual phonics skills with knowledge tracing of students' performance in word-building activities where they clicked on letter tiles to spell out a word spoken by the Reading Tutor. During 4,190 such activities, students built 26,562 words. The Reading Tutor interpreted the student's first attempt at each letter in a word as correct or incorrect. It accordingly made 78,069 updates to its estimates of students' phonic skills.

3.5 Tutor decisions

A tutor's observations guide layers of tutorial decision. For instance, after classifying words in the current utterance (if any) as read, misread, or omitted, the Reading Tutor decides whether to wait, backchannel, prompt the student, give praise, go on, read the sentence aloud, or give help on an individual word, and if so, what type of help. The 2007-2008 Reading Tutor logged 181,788 such decisions.

In analyzing such decisions, it is necessary not only to know the tutor's observable actions and the decisions that led to them, but the alternative choices for each decision. The 2007-2008 Reading Tutor logged this information for some types of decisions. For example, it logged not only its randomized decisions to backchannel or prompt the student, but also its decisions not to do so. Logging this additional information is crucial for subsequent analyses of the randomized controlled experiments embedded in the Reading Tutor.

Reading Tutor activities and interventions can set variables to record randomized decisions about which word to give which if any treatment, or outcomes in the form of what responses the student chose or typed. It logged 175,978 such timestamped variable assignment events. For example, it used this mechanism to log its randomized decisions about whether to explain a vocabulary word in depth, briefly, or not at all [16].

Without such logging the alternatives for each decision, it is necessary to reconstruct them. For example, the Reading Tutor's decisions about what type of decoding assistance to give on words are randomized, but are constrained by the types of help feasible and appropriate for a given word. The Reading Tutor cannot give a rhyming hint for the word *orange* because no words rhyme with it. It could sound words longer than four phonemes but refrains from doing so on grounds of inadvisability. In analyzing the relative efficacy of different types of help, a "level playing field" comparison requires knowing which types were possible on a given word. Otherwise, comparing the success of help based on the student's later performance on the word will be biased in favor of help such as rhyming hints, which tend to be available only on easier words. Proper analysis of help efficacy involved approximating this information based on lexical properties of the word and detailed knowledge of the Reading Tutor implementation [17]. In retrospect, it would have been better to log the actual set of choices for each decision.

Tutorial decisions go through layers of implementation to transform them into external behavior (acting). For instance, the 2007-2008 Reading Tutor logged 41,177 decisions about how to intervene before a story, before a sentence, or after a story. These interventions are specified in the same activity language used to represent stories as sequences of different types of steps, such as reading, listening, editing, and choosing.

3.6 Tutor actions

A tutor's external actions are typically limited to graphical display and audio output. The 2007-2008 Reading Tutor logged 1,964,620 records of .wav files played in full or in part.

We refrain from logging every individual graphical action because there are too many, nor does it seem useful to do so. However, the text displayed by the Reading Tutor is generally logged as part of the action that displays it. For instance, logging a sentence encounter includes the text of the sentence. Likewise, logging a multiple choice question includes the displayed prompt and menu of choices.

3.7 Events

Tutorial interaction can be described at different temporal *grain sizes*. The 2007-2008 Reading Tutor logged 3,184 launches, of which 71 ended in crashes with logged stack dumps, and 10,462 sessions, defined as starting when a student logged in, and ending by logging out, timing out, or crashing. It logged 193,101 executions of scripts for logging in or out, picking a story or other activity, performing an activity, editing a story, and updating information about a class, student, or story. The 21,580 story encounters logged included 357,630 sentences displayed. The 310,375 logged steps included, among others, 120,506 reading, 120,506 picking, 18,412 editing, 17,8891 timing out, 9,253 free-form speaking, 4,191 word building, 2,240 entering a password, and 861 "WordSwap" games comprising 11,728 sentences, each with one misread word to click on, and 19,581 clicks.

3.8 Observing the tutor

Reconstructing the exact appearance of the screen from logged tutor data is problematic. One reason is that the actual screen appearance at a given moment depends not only on what the tutor intends to display, but on what the operating system draws, and how promptly. One approach to this problem is the ability to replay logged tutor sessions. This feature can be very useful, but is infeasible to add after the fact to the Reading Tutor, and might be impractical anyway due to its multimodal, mixed-initiative, overlapping, stochastic, time-sensitive nature. Replay is much simpler for tutors limited to mouse and keyboard input and strict turn-taking, especially if their responses are deterministic and independent of student response time.

Another problem is that graphical design evolves across successive tutor versions. Old versions may not run as before (if at all), due to environmental changes such as updates to external content, upgrades to the operating system, and porting to faster computers.

We have found over the years that the most reliable way to document the appearance and behavior of a given version is to videotape children using it. However, this method is cumbersome and expensive to do well, and the result is inconvenient to use without the additional expense of indexing it to internally logged tutor data. Programmable screen capture software, fast PCs, and cheap disk space enable tutors to record video or at least occasional screenshots of their interactions, which we plan to do in the future.

We generally assume that tutor actions are observable by students. This assumption can be false. For instance, the display monitor may be turned off, broken, or maladjusted. Even likelier, the headset may be broken, disconnected, or plugged in to the wrong socket. Unlike humans, whose motor actions are accompanied by sensory feedback, computers do not actually see and hear the same displays and audio as their users. Screen capture is only a partial solution to this problem. A full solution will require additional devices to sense how tutor output actually looks and sounds in the environment.

4 How to log

Our experience has led us to develop the following 10 guidelines for logging interactions.

4.1 Log directly to a database.

Log files are easy to record, flexible in what information they can capture, (sometimes) human-understandable to read, and useful in debugging. However, we have found them unwieldy to aggregate across multiple sessions and computers, and difficult to parse and analyze in ways not anticipated when they were designed [1].

Logging tutorial interactions directly to a suitably designed and indexed database instead of to log files eliminates the need to parse them – a time sink and error source in our past. Starting with the 2003-2004 school year, the Reading Tutor has logged its interactions to a database, making their analysis easier, faster, more flexible, less bug-prone, and more powerful than analyzing conventional log files. This practice has enabled or facilitated nearly all the dozens of subsequent papers listed on Project LISTEN's Publications page.

Moreover, logging straight to a database supports immediate efficient access. This capability is essential for real-time use of logged information – in particular, by the tutor itself. The Reading Tutor's student model relies on information logged to its database. In contrast, although the Pittsburgh Science of Learning Center's DataShop (located at pslcdatashop.web.cmu.edu) uses a database, it generates the database only after the fact by parsing tutor logs in the form of XML files.

Each year's version of the Reading Tutor adds, drops, or modifies tables or fields in the database schema, and therefore has its own archival database. This practice facilitates running a query on one school year's data at a time. Also, each Project LISTEN member has his or her own database to modify freely without fear of altering archival data.

4.2 Include computer, student ID, and start time as standard fields.

A key insight is that student, computer, and time typically suffice to identify a unique tutorial interaction of a given type. Together they distinguish the interaction from those of another type, computer, or student. (We include computer ID in case the student ID is not unique, and also because some events, such as launching the Reading Tutor, do not involve a student ID.) There are two reasons this idea is powerful. First, these fields serve as a primary key for every table in the database, simplifying both access and the learning curve for working with the data. Second, nearly every tutor makes use of the concepts of students, computers, and time, so this recommendation is broadly applicable.

4.3 Index event tables by computer, student ID, and start time.

Database indices enable very fast retrieval even from tables of millions of events.

4.4 Log end time as well as start time.

This additional information makes it possible to compute the duration of a non-instantaneous event, measure the hiatus between the end of one event and the start of another, and determine if one event starts before, during, or after another event – capabilities essential for the Session Browser [11, 12] described in Section 2.3 above, in particular to infer hierarchies of events based on temporal relations among them. Logging the start and end time of an event in the same record requires the tutor to remember until the end of the event what information to log about it, such as when it began. Logging start and end times to different records merely replaces this requirement with the messier task of matching them up after the fact. Some tutors log start times but not end times. For instance, logging the end time of an event may be problematic for a web-based tutor that doesn't know when the student leaves a page or window.

4.5 Include a field for the parent event start time.

A field for an event's parent makes joins easier to write and faster to execute. For example, the sentence_encounter_start_time field of the record for a read word makes it easier and faster to find the sentence encounter containing the word than by querying for the sentence encounter that started before the word and ended after it.

4.6 Log each school year's data to a different database.

Each year's version of the Reading Tutor adds, drops, or modifies tables or fields in the database schema, and therefore has its own archival database. This practice also facilitates running a query on one school year's data at a time. Also, each team member has his or her own database to modify freely without fear of altering archival data. Making it easy for researchers to create new tables and views that are readily accessible to each other is key. This step enables "best practices" to propagate quickly, whereas if separate copies of the data are kept, "version skew" can become a problem.

4.7 Design databases to support aggregation across sites.

To merge separate databases from multiple schools into a single database, the MySQL server at each site sends each day's transactions to our lab server, which simply re-executes them on the aggregated database. To make this solution possible, each table must be able to combine records from different sites. Therefore the Reading Tutor uses student IDs unlikely to recur across different classes or schools, so as to distinguish data from different students. Similarly, it uses computer, ID, and start time to identify events, instead of numbering them from 1 separately at each site.

4.8 Name standard fields consistently within and across databases.

Naming fields consistently in successive versions of a tutor makes them easier for the Session Browser to extract. Thus most of the tables in a Reading Tutor database have fields named machine_name, user_id, start_time, and (for non-instantaneous events) end_time. Timestamps in MySQL (at least the version we used) are limited to 1-second resolution, so tables that require higher resolution encode the milliseconds portion of start and end times in fields named sms and ems, respectively. Adding new tables and fields is fine, but keeping the names of the old ones facilitates reuse of code to analyze tutor data.

4.9 Use a separate table for each type of tutorial event.

Using a separate table for each event type (e.g. story, sentence, utterance, click) lets us include fields for features specific to that event type, such as the title of a story, the text of a sentence, the audio recording of an utterance, or the word the student clicked on.

4.10 Logging the non-occurrence of an event is tricky.

"Subjunctive logging," or recording what could have happened but did not, is still an active topic of discussion in the EDM community. But tutor designers and data miners should be aware that recording some non-events can greatly simplify later analyses. For example, the Reading Tutor logs each word it hears the student read. The time at which a skipped word *wasn't* read is undefined, so the Reading Tutor logs its start and end time as null. However, logging its (non-null) sentence_encounter_start_time, and the position in the text sentence where it should have been read, lets analysis queries and the Session Browser retrieve words as ordered in the sentence, whether or not they were read.

5 Conclusion

The goal of this paper is to provide useful advice for logging tutor data. We discussed why to log tutor data, giving four examples from Project LISTEN – tutoring, reporting, browsing, and mining. We discussed what to log, with examples from the Reading Tutor. Finally, we proposed ten concrete guidelines for how to log tutor data.

Our guidelines exploit three simple but powerful ideas. First, logging tutorial interaction directly to a suitably designed and indexed database instead of to log files eliminates the need to parse them, and supports immediate efficient access. Second, student, computer, and time interval suffice to identify a tutorial event of a given type. Third, temporal relations among time intervals define a useful hierarchical structure of tutorial events.

Evidence for our guidelines includes the dozens of publications they have enabled. They helped us implement a flexible, efficient tool to browse tutor data in understandable form yet with minimal dependency on tutor-specific details. A short vignette may best illustrate their power. In a research talk, Dr. Vincent Aleven reported a partial correlation of students' learning gains in his tutor with the rate at which they asked for hints, controlling for pretest scores. The second author of this paper used the Reading Tutor database to replicate Dr. Aleven's result – before he finished his talk.

Acknowledgements

This work was supported in part by the National Science Foundation under ITR/IERI Grant No. REC-0326153 and in part by the Institute of Education Sciences, U.S. Department of Education, through Grants R305B070458, R305A080157, and R305A080628 to Carnegie Mellon University. The opinions expressed are those of the authors and do not necessarily represent the views of the Institute, the U.S. Department of Education, or the National Science Foundation. We thank the educators, students, and members of Project LISTEN who have helped generate, log, and analyze our data.

References (Project LISTEN publications at www.cs.cmu.edu/~listen)

[1] Mostow, J. and G. Aist. Evaluating tutors that listen: An overview of Project LISTEN. In K. Forbus and P. Feltovich, Editors, *Smart Machines in Education*, 169-234. MIT/AAAI Press: Menlo Park, CA, 2001.

[2] Mostow, J., S.F. Roth, A.G. Hauptmann, and M. Kane. A prototype reading coach that listens [AAAI-94 Outstanding Paper]. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 785-792. 1994. Seattle, WA: American Association for Artificial Intelligence.

[3] Mostow, J. and J. Beck. When the Rubber Meets the Road: Lessons from the In-School Adventures of an Automated Reading Tutor that Listens. In B. Schneider and S.-K. McDonald, Editors, *Scale-Up in Education*, 183-200. Rowman & Littlefield Publishers: Lanham, MD, 2007.

[4] Mostow, J. Experience from a Reading Tutor that listens: Evaluation purposes, excuses, and methods. In C.K. Kinzer and L. Verhoeven, Editors, *Interactive literacy education: facilitating literacy environments through technology*, 117-148. Lawrence Erlbaum Associates, Taylor & Francis Group: New York, 2008.

[5] Mostow, J., G. Aist, C. Huang, B. Junker, R. Kennedy, H. Lan, D. Latimer, R. O'Connor, R. Tassone, B. Tobin, and A. Wierman. 4-Month evaluation of a learner-controlled Reading Tutor that listens. In V.M. Holland and F.P. Fisher, Editors, *The Path of Speech Technologies in Computer Assisted Language Learning: From Research Toward Practice*, 201-219. Routledge: New York, 2008.

[6] 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.

[7] Mostow, J., G. Aist, J. Bey, P. Burkhead, A. Cuneo, B. Junker, S. Rossbach, B. Tobin, J. Valeri, and S. Wilson. Independent practice versus computer-guided oral reading: Equal-time comparison of sustained silent reading to an automated reading tutor that listens. *Ninth Annual Meeting of the Society for the Scientific Study of Reading* 2002. Chicago, Illinois.

[8] Beck, J.E., K.-m. Chang, J. Mostow, and A. Corbett. Does help help? Introducing the Bayesian Evaluation and Assessment methodology. *9th International Conference on Intelligent Tutoring Systems*, 383-394. ITS2008 Best Paper Award. 2008. Montreal.

[9] Aist, G. and J. Mostow. Faster, better task choice in a reading tutor that listens. In V.M. Holland and F.P. Fisher, Editors, *The Path of Speech Technologies in Computer Assisted Language Learning: From Research Toward Practice*, 220-240. Routledge: New York, 2008.

[10] Alpern, M., K. Minardo, M. O'Toole, A. Quinn, and S. Ritzie. Project LISTEN: Design Recommendations and Teacher Tool Prototype. In *Human Computer Interaction Institute*. 2001, Carnegie Mellon University: Pittsburgh, p. 85.

[11] 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)*, 884-886. 2005. Amsterdam.

[12] Mostow, J., J. Beck, and others. Lessons from Project LISTEN's Session Browser. In C.R. Morales, et al., Editors, *Handbook of Educational Data Mining*. Taylor & Francis Group: London, under review.

[13] Mostow, J. and M. Duong. Automated Assessment of Oral Reading Prosody. *Proceedings of the 14th International Conference on Artificial Intelligence in Education (AIED2009)* 2009. Brighton, UK.

[14] Kominek, J., G. Aist, and J. Mostow. When listening is not enough: potential uses of vision for a reading tutor that listens. *Working Notes of the AAAI Spring Symposium on Intelligent Environments*, 161-167. 1998. Stanford, CA.

[15] Zhang, X., J. Mostow, N.K. Duke, C. Trotochaud, J. Valeri, and A. Corbett. Mining Free-form Spoken Responses to Tutor Prompts. *Proceedings of the First International Conference on Educational Data Mining*, 234-241. 2008. Montreal.

[16] Heiner, C., J. Beck, and J. Mostow. Automated Vocabulary Instruction in a Reading Tutor. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, 741-743. 2006. Jhongli, Taiwan.

[17] Heiner, C., J.E. Beck, and J. Mostow. Improving the help selection policy in a Reading Tutor that listens. *Proceedings of the InSTIL/ICALL Symposium on NLP and Speech Technologies in Advanced Language Learning Systems*, 195-198. 2004. Venice, Italy.