What's in a step? Toward general, abstract representations of tutoring system log data

Kurt VanLehn[1], Ken Koedinger[2], Alida Skogsholm[2], Adaeze Nwaigwe[2], Robert G.M. Hausmann[1], Anders Weinstein[1], Benjamin Billings[2]
[1] LRDC, University of Pittsburgh, Pittsburgh, PA, USA
[vanlehn, andersw,bobhaus]@pitt.edu
[2] HCII, Carnegie-Mellon University, Pittsburgh, PA, USA
[koedinger,alida,anwaigwe,bkb]@cmu.edu

Abstract. The Pittsburgh Science of Learning Center (PSLC) is developing a data storage and analysis facility, called DataShop. It currently handles log data from 6 full-year tutoring systems and dozens of smaller, experimental tutoring systems. DataShop requires a representation of log data that supports a variety of tutoring systems, atheoretical analyses and theoretical analyses. The theory-based analyses are strongly related to student modeling, so the lessons learned in developing the DataShop's representation may apply to student modeling in general. This report discusses the representation originally used by the DataShop, the problems encountered, and how the key concept of "step" evolved to meet these challenges.

Keywords: Student modeling, educational data mining, tutoring systems.

1 The Pittsburgh Science of Learning Center DataShop

The PSLC DataShop (http://learnlab.web.cmu.edu/datashop/) provides the following functions: (1) Data security with appropriate anonymity; (2) A standard, extensible representation; (3) Easy export to standard tools, such as spreadsheets and statistical packages; (4) Analytic tools specific to log data; and (5) Reification of the PSLC theoretical framework. This last goal is explained below.

The DataShop grew out of Ritter and Koedinger's [1] standard framework for representing log data. Its analysis tools, which are described below, evolved from Anderson and Koedinger's early work on learning curves [2].

The DataShop is part of the PSLC LearnLab—an internationally shared facility for doing *in vivo* experimentation (http://www.learnlab.org). Although the DataShop is in daily operation supporting thousands of students, teachers and researchers around the world, it is still developing in order to incorporate new kinds of student-tutor interactivity. We report on the representational challenges that have been faced.

2 Three levels of description

The log data are a chronological record of all the student's interactions with a tutoring system. These interactions are described at three levels: transactions, step histories and knowledge component applications. Each level is described below.

The lowest level is the *transaction* [1], which is a communication between the student and the system. For instance, the following is a sequence of transactions in an algebra tutor:

1. The tool displays "2x(3-4x)-13 =__x^2 + __x + __ = (__x + __)(__x + __)".
2. The student puts the cursor in the first blank and enters "8".
3. The tutor tells the student that the entry is incorrect.
4. The student asks for a hint.
5. The tutor tells the student "Check your signs."
6. The student replaces the "8" with "-8".
7. The tutor tells the student that the entry is correct.
8. The student puts the cursor in the next blank and enters "6".
9. The tutor tells the student that the entry is correct.

The next level represents the log data as a sequence of episodes, called *step-attempt histories* [3]. Each episode is terminated by a *step*, which is a user interface action that is correct and advances the solution of the problem. The *history* of that step consists of the student's incorrect attempts at entering that step, help requests, hints, and any other transactions that might aid the student to make the step. For instance, in the list above, transactions 2 through 7 comprise the first step-attempt history; transactions 8 and 9 comprise the second one. This level of description assumes that only some user interface actions are steps, and that the correct/incorrect distinction makes sense for them. Thus, this level of representation has some theoretical commitments, but fairly weak ones.

The third level of description is based on the PSLC theoretical framework, which assumes that domain knowledge can be usefully decomposed into *knowledge components* [4] This is intended to be a generic, neutral term that covers many kinds of knowledge: procedural, conceptual, perceptual, etc. For example, in learning Chinese as a second language, a single knowledge component (KC) might represent a word's phonological, orthographic, and semantic representations, as well as the associations between them. In physics, Newton's third law might be represented as a single knowledge component. Most PSLC tutoring systems represent domain knowledge as KCs, and they label every step with the KCs that must be applied to generate that step. For instance, the step entered at line 7 above results from applying two KCs: the Distributive Law and Simplification. Thus, at this level of description, the log data are viewed as a sequence of *knowledge component applications*.

3 The DataShop's analytical tools

We discuss only two tools, the Error Report and the Learning Curve generator, that illustrate the need for the three levels of log data description.

In its simplest usage, the Error Report is given a problem and prints a table that lists each step in the problem along with a summary of the students' step-attempt histories. For instance, the error report for filling in the first blank of "2x(-4x+3)-13= __x^2+…", might state that: 69% of the students entered the correct response on the first attempt, 12% asked for a hint, 10% entered "8" and got the hint "Check your signs," and 9% entered "-4" and got the hint "Hmm; not what I got. Please try again." Such error reports are useful for determining which common errors are not receiving pedagogically useful feedback. The Error Report uses only the step level of description. KC applications play no role in its reports, so a tutoring system that does not use KCs can still get Error Reports for its log data.

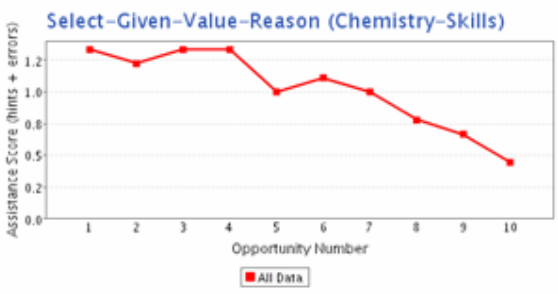

Fig. 1. Learning Curve for the KC Select-Given-Value-Reason

In our usage, a *learning curve* displays the students' increasing mastery of a knowledge component over time [2, 5]. As a simple illustration, suppose we want a learning curve for a particular KC for a particular student. The tool first locates all the step-attempt histories corresponding to applications of that KC. For each history, it calculates the *assistance score*, which is simply the number of help requests plus the number of errors in that episode. For instance, for the first step-attempt history mentioned above (transactions 2 through 7), the assistance score is 2; for the second step-attempt history, the assistance score is 0. Then the Learning Curve generator plots a graph (see Fig. 1) where the points correspond to step-attempt histories, the *y*-axis is the assistance score, and the *x*-axis is ordinal and chronological (i.e., the Nth KC application is at x = N on the graph.) Theory suggests that the learning curve should start with large amounts of assistance on the first KC application, less on the second, and so on. Often the learning curve for a single student is too noisy to see such a pattern, so it is common to aggregate over all the students. In Fig. 1, for instance, the point at $x = 1$ has a *y*-value that is the average over all students' assistance scores for their first application of the KC.

4 Representational lessons learned

This section discusses representational lessons learned while trying to accommodate an increasing set of tutoring systems. When log data from VLAB, a simulated

chemistry laboratory with a direct-manipulation interface (www.chemcollective.org), were added to the DataShop, we had to allow multiple transactions to be associated with a single step. For instance, a single step "heat beaker A" should be associated with the three transactions: (1) removing a Bunsen burner from storage, (2) placing it under the beaker and (3) turning on the flame.

More recently, we added the inverse capability: a single student transaction may be associated with multiple steps. When log data from Andes, a physics tutoring system (www.andes.pitt.edu) were first added to the DataShop, each correct equation entered by the student was treated as a step. However, this made the error reports nearly useless because few students entered the same steps. For instance, one student might enter "W_y = -W" as one step and "W = m*g" as another. A second student might enter their algebraic combination, "W_y = -m*g". Even if a problem needs only N primitive equations to solve it, most subsets of the set of N equations correspond to a possible compound equation. Thus, the error report for a problem with 10 primitive equations may have as many as N^2 = 1028 steps. Moreover, each would probably have just one or two step-attempt histories because only one or two students happened to enter exactly that algebraic combination of primitive equations. Fortunately, Andes decomposes a student equation into the primitive equations that comprise it. Each such primitive equation became a step in the DataShop representation. Thus, if the student entered "W_y = -m*g", then this student action is associated with two steps, "W_y = -W" and "W = m*g". That is, a single student transaction may be associated with multiple steps.

The third major issue involves partitioning the transactions into step-attempt histories. We implied earlier that all the errors, help requests, and other non-step transactions that occurred between two steps became the step-attempt history for the second step. That is, the partitions were chronological. This does not make sense in some cases. For instance, suppose the student makes the error mentioned earlier by entering "8" in the first blank of "2x(-4x+3)-13 = __x^2 + __x + __". The tutor gives the hint "Check your signs," but the student does not fix the error. Instead, the student puts the cursor in the second blank and enters "6" which is correct. If we used only the chronological scheme, the error and the hint would become part of the step-attempt history for "6." This is wrong because the student actually didn't have any trouble entering the "6." An Error Report that showed "-8" and "Check your signs" associated with the "6" step would be very confusing. On the other hand, a partition based on the location of the cursor at the time of the entry would assign the appropriate step-attempt histories to the steps of this problem.

Chronology and location are just two cues that can be used for deciding how to partition the log data into step-attempt history. The situation becomes more complex when dealing with natural language tutoring systems. A single transaction, such as a student saying "The block moves downward, speeding up," might be analyzed as two steps: "The block moves downward" and "The block speeds up". We are currently evaluating multiple heuristics by comparing their performance with human coders [6]. These explorations should be useful not only to the DataShop, but also to other applications that do student modeling (e.g., [3]).

6 Conclusions

The central concept in the DataShop log data representation has turned out to be the step. It connects the transaction-level representation to the theoretically-derived KC level. The step level also provides a way for tutoring systems that do not have KC-level analyses to still get some use of the DataShop.

However, the concepts of "step" and "step-attempt history" have evolved in subtle ways. Several years ago, "step" meant an actual student transaction that was a correct part of the solution to the problem, and "step-attempt history" meant all the non-step transactions that immediately preceded the step. Now there is no longer a one-to-one relationship between steps and transactions, and the transactions that comprise a step-attempt history need not immediately precede the step.

A *step* is now defined as the smallest possible correct entry that a student can make. By "smallest", we mean that the step cannot be re-expressed as two or more steps.

Although the KC applications required to solve a problem are determined solely by the problem and the KC-level analysis of the task domain, the steps required to solve a problem are also a function of the user interface. For instance, in a natural language interface, when the student enters "the baseball's velocity is 10 m/s at 30º," it corresponds to two steps: "the baseball's velocity is 10 m/s" and "the baseball's velocity is 30º" However, if the user interface were graphical instead, so that the student specifies the baseball's velocity by clicking and dragging out a vector, what was once a compound of two steps now becomes one, because in the graphical user interface, the vector drawing step cannot be decomposed.

References

1. Ritter, S. and K. Koedinger, *Towards lightweight tutoring agents*, in *Artificial Intelligence in Education, 1995*, J. Greer, Editor. 1995, Association for Advancement of Computers in Education: Charlottesville, NC. p. 91-98.
2. Anderson, J.R., et al., *Cognitive Tutors: Lessons Learned.* The Journal of the Learning Sciences, 1995. **4**(2): p. 167-207.
3. VanLehn, K., *Intelligent tutoring systems for continuous, embedded assessment*, in *The future of assessment: Shaping teaching and learning*, C.A. Dwyer, Editor. in press, Erbaum: Mahwah, NJ.
4. VanLehn, K., *The behavior of tutoring systems.* International Journal of Artificial Intelligence and Education, 2006. **16**.
5. Cen, H., K.R. Koedinger, and B. Junker, *Learning Factors Analysis -- A general method for cognitive model evaluation and improvement*, in *Intelligent Tutoring Systems: 8th International Conferenc*, M. Ikeda, K. Ashley, and T.-W. Chan, Editors. 2006, Springer: Berlin. p. 164-175.
6. Nwaigwe, A., et al., *Exploring alternative methods for error attribution in learning curve analysis in intelligent tutoring systems*, in *Proceedings of AI in Education, 2007*. in press, IOS Press: Amsterdam.