Chapter 19 Emerging Trends in Automated Authoring

Andrew M. Olney[1], Keith Brawner[2], Phillip Pavlik[1], Kenneth R. Koedinger[3]
[1] University of Memphis; [2] US Army Research Laboratory; [3] Carnegie Mellon University

Introduction

Traditional intelligent tutoring systems (ITS) are specialized feats of engineering: they are custom-made to implement a theory of learning, in a particular domain, within a specific computer environment. There are many ways to describe or categorize authoring tools used to make ITSs (Murray, 2004). This chapter considers authoring tools primarily in terms of intelligent tutor paradigms. Three popular ITS paradigms are dialogue-based tutors (Nye, Graesser & Hu, 2014), constraint-based tutors (Mitrovic, 2012), and model-tracing tutors (Anderson et al., 1995). These paradigms may be distinguished along two abstract axes, as shown in Figure 1. The axes reflect how the learning task is defined and how student progress in the task is measured.

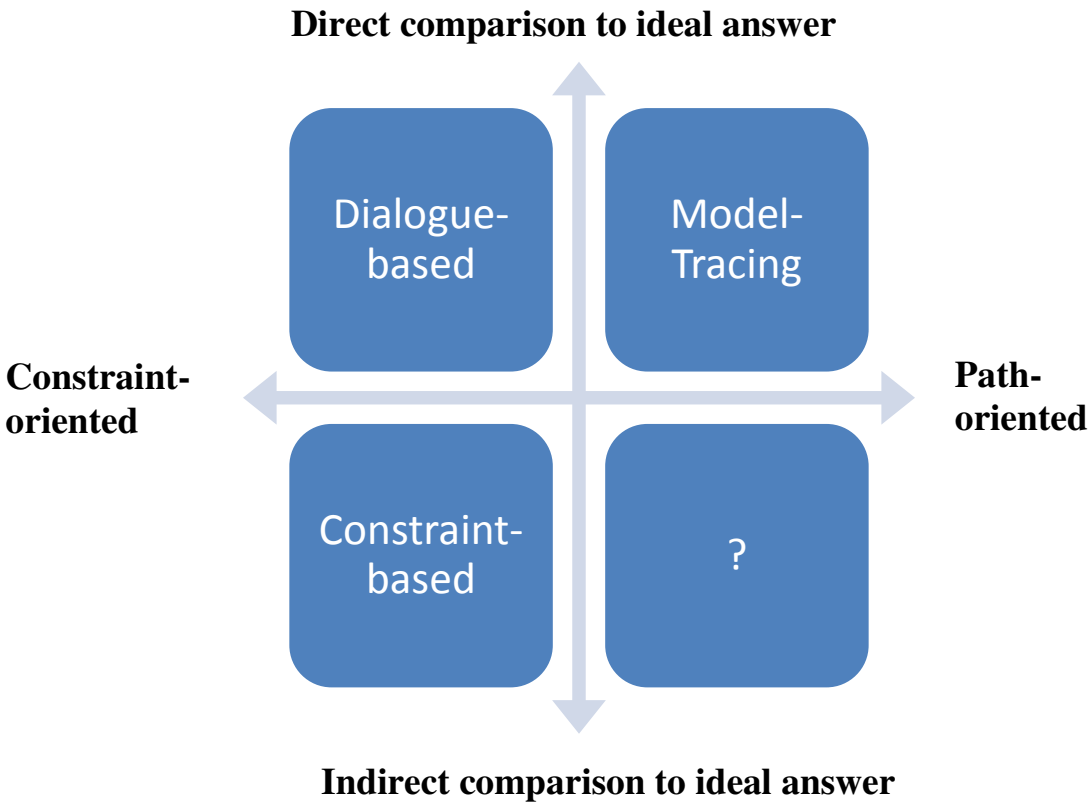

Figure 6. Tutoring paradigms arranged by orientation (path vs. constraint) and comparison to ideal answer (direct vs. indirect).

The horizontal axis indicates whether the paradigm is *primarily* path-oriented or constraint-oriented. A path-oriented paradigm conceives the learning task as a sequence of steps that lead to a solution. For example, the instructional theory behind model-tracing tutors can be expressed within the knowledge-

learning-instruction (KLI) framework (Koedinger, Perfetti & Corbett, 2012). Critical to the KLI framework are the ideas that (a) most of our knowledge in any area of expertise (e.g., grammar, algebra, design) is in the form of procedural skills, which are learned by induction from experience and feedback, and (b) two forms of instruction that best facilitate such learning are problem solving practice with as-needed feedback on student errors and as-needed examples of correct behavior (next step hints). The key is to engage the learner in the process of doing, that is, engaging in the target activity, and provide personalized tutoring support for the learner that adapts to their particular needs. Tutoring support is achieved by the use of a model of desired or correct performances and of particularly common undesired or incorrect performances. This *direct* comparison against a model (which includes ideal answers) also situates model-tracing on the vertical axis. Each student's actions are traced against this model such that feedback can be generated when undesired performance is observed and next-step hints can be generated when students are stuck. In both cases, the emphasis is on minimal intervention (Anderson et al., 1995) in order to maximize student active and constructive involvement in the thinking and learning process.

Conversely, a constraint-oriented paradigm conceives of the learning task as attaining a solution state irrespective of the path that led to it. Both dialogue-based and constraint-based paradigms share this property but they differ in many respects, most notably in how they represent knowledge and compare the student answer to an ideal answer. Dialogue-based tutors are typically frame-filling systems (McTear, 2002) that fill slots in a frame in any order. For example, given the physics question, "If a lightweight car and a massive truck have a head-on collision, upon which vehicle is the impact force greater, and why?", a dialogue-based system might have the slots [The magnitudes of the forces exerted by A and B on each other are equal] and [If A exerts a force on B, then B exerts a force on A in the opposite direction]. If a user says, "the forces are equal," the system would recognize that the first slot is filled and follow up with a question to fill the second slot like, "What can you say about the direction of the forces?" The slots are known as expectations, or expected components of the ideal answer (Graesser, D'Mello, et al., 2012), and the follow-up questions used to fill out the frame are aligned with models of naturalistic human tutoring (D'Mello, Olney & Person, 2010; Graesser & Person, 1994; Graesser, Person & Magliano, 1995; Person, Graesser, Magliano & Kreuz, 1994) plus ideal pedagogical strategies in some versions. Dialogue-based systems determine whether a slot, or expectation, is filled by *directly* comparing the student's answer to an ideal answer, typically using methods like latent semantic analysis (Landauer, McNamara, Dennis & Kintsch, 2007) and other semantic matching algorithms. Dialogue-based tutors can be considered as a very narrow form of constraint-based tutors where the constraints are defined by whether all slots are filled, i.e., all expectations are met.

Constraint-based tutors operationalize constraints as consisting of a relevance condition (R) and a satisfaction condition (S) (Ohlsson, 1992). The constraint is only applicable when the relevance condition is met, at which point the satisfaction condition defines what conditions the student's solution must meet in order to be correct. Only solutions that violate no constraints are correct. Constraints therefore do not specify a path or set of paths to a solution but rather define a space of correct solutions (Ohlsson & Mitrovic, 2007). For example, a constraint for fraction addition might have the relevance conditions *problem statement:* a/b + c/d and *student solution:* (a+c)/n with satisfaction condition b=d=n: the solution is correct only when the denominators of the problem statement and student solution are equal (Ohlsson, 1992). Constraints are well suited for design tasks and tasks that are ill defined precisely because they allow solutions to be recognized without requiring them to be enumerated by the author. With regard to the vertical axis, constraint-based tutors do not directly compare a solution to an ideal solution but instead compare *indirectly* via preserved and violated constraints.

The characterization presented in Figure 1 is undoubtedly an over-simplification of the differences between these tutoring paradigms because they differ on so many other dimensions. Moreover, they share more features than Figure 1 represents, because path-oriented tutors can relax ordering restrictions and constraint-oriented tutors can incorporate path-like ordering restrictions. However, from an authoring tool

standpoint, the above depiction highlights some of the key authoring problems faced by each paradigm. Model-tracing tutors require a model to trace, commonly in the form of sequences of steps and production rules that require next-step hints. Dialogue-based tutors require a set of expectations and associated follow-up questions (e.g., hints or prompts) when the expectations are unfulfilled. Constraint-based tutors require a set of constraints and associated feedback for their violations.

This chapter discusses several emerging approaches to ITS authoring that attempt to go beyond the typical human-created practice and automate more of the authoring process than has been previously attempted. Efforts are currently being undertaken in order to ease this burden from the authors in the form of programming by tutoring, automated concept map generation, metadata tagging, extensive content reuse, and continual refinement. With respect to Figure 1, the chapter emphasizes automated authoring in the model-tracing and dialogue-based traditions (see Mitrovic et al., 2006 and Mitrovic et al., 2009 for discussion of automated authoring of constraint-based tutors).

Related Research

Advanced Authoring for Model-Tracing Tutors

This section focuses on authoring tools for model-tracing tutors. The instructional approach in such tutors is to provide students with one-on-one tutoring support as they work on problem or activity scenarios of varying complexity. They do so within rich interface tools or simulation environments, for example, solving a physics problem using tools for drawing and annotating a free body diagram, and for writing and solving equations (e.g., VanLehn, 2006); solving a real-world quantitative reasoning problem (e.g., which cell phone plan to choose) using tools for creating tables, graphs, and equations (e.g., Koedinger et al., 1997); designing an efficient system using a thermodynamics simulation (see Fig. 26 in Aleven et al., 2009); and making an English grammar choice using a pop-menu (Wylie, Koedinger & Mitamura, 2010).

Effective and efficient authoring depends on how *completely, accurately, and quickly* an author can specify a sufficiently complete set of desired and common undesired student actions. This set of reasonable actions was traditionally specified in a general artificial intelligence (AI) rule-based system (cf., Anderson et al., 1995). For example, in a production system, each production rule is annotated with instructional messages, such as (a) next-step hints in the case of productions that represent desirable student actions and (b) error feedback messages in the case of productions that represent common student errors or underlying misconceptions. One successful alternative to production system authoring is to concretely enumerate, for each problem scenario, every action along all reasonable solution paths. This is the "example-tracing" approach taken in the Cognitive Tutor Authoring Tools (CTAT) (Aleven, Mclaren, Sewall & Koedinger, 2009), a complete tutor-authoring suite that has been used to create several dozen ITSs.

A second alternative to hand-authoring production systems is to have the author tutor a machine learning system that learns the production system (largely) from scratch. This is the approach taken by SimStudent (Matsuda, Cohen & Koedinger, 2015). SimStudent learns problem-solving skills from the two kinds of instruction that are arguably the most powerful in human skill acquisition: learning from examples and learning from (feedback on) doing (e.g., Gick & Holyoak, 1983; Roediger & Butler, 2011; Zhu & Simon, 1987). Figure 2 shows an example of SimStudent being tutored on algebra equation solving. An example of an acquired production rule (in the JESS language) from the first author demonstration is shown on the right. SimStudent has three online learning mechanisms that focus on learning (1) information retrieval paths (clauses of the IF-part of the production that identifies where in the interface relevant information may lie), (2) preconditions on actions (clauses of the IF-part that constrain when the production is appropriate), and (3) action plans (compositions of functions that compute appropriate actions). The

newest addition to SimStudent is a representation learning mechanism that learns the general structure of declarative memory structures, which are the basis for both the operation and learning of production rules (Li, Matsuda, Cohen & Koedinger, 2015).

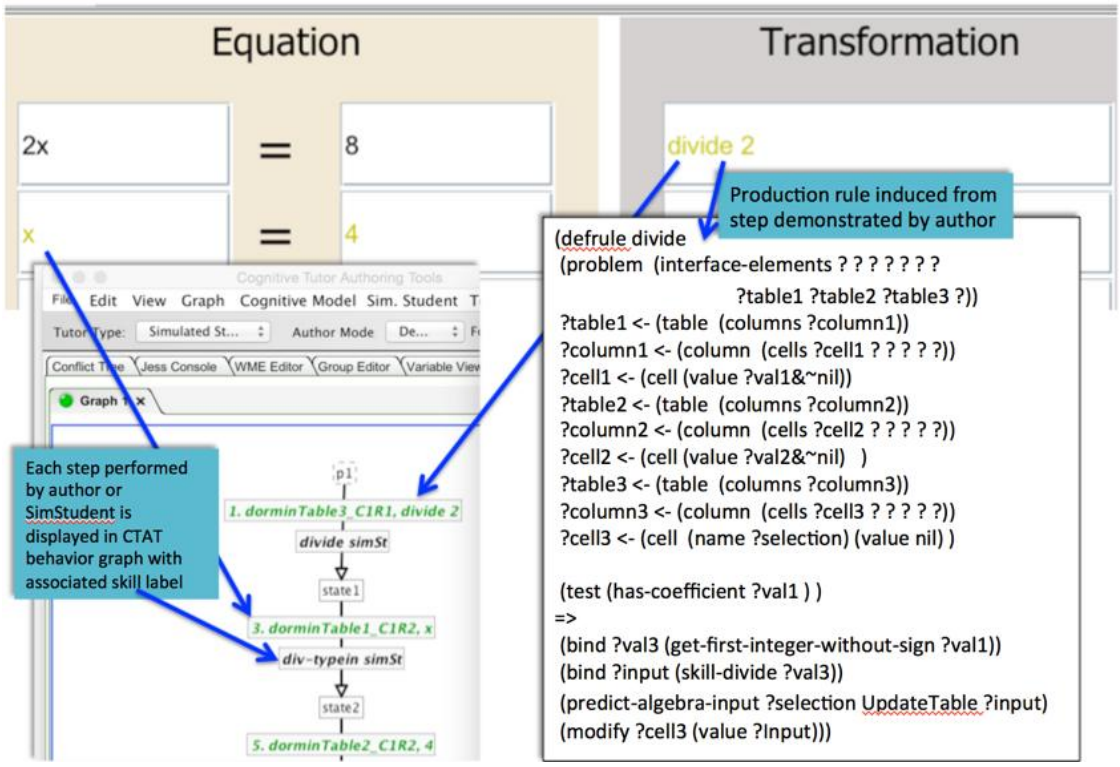

Figure 2. After using CTAT to create an interface (shown at top) and entering a problem ("2x=8"), the author begins teaching SimStudent either by giving yes-or-no feedback when SimStudent attempts a step or by demonstrating a correct step when SimStudent cannot (e.g., "divide 2"). SimStudent induces production rules from demonstrations (example shown on right) for each skill label (e.g., "divide" or "div-typein" shown on left). It refines productions based on subsequent positive (demo or yes feedback) or negative (no feedback) examples.

The use of SimStudent as authoring tool is still experimental, but there is evidence that it may accelerate the authoring process and that it may produce more accurate cognitive models. In one demonstration, Matsuda et al. (2015) explored the benefits of a traditional *programming by demonstration* approach to authoring in SimStudent versus a *programming by tutoring* approach, whereby SimStudent asks for demonstrations only at steps in a problem/activity where it has no relevant productions and otherwise it performs a step (firing a relevant production) and asks the author for feedback as to whether the step is correct/desirable or not. They found that programming by tutoring is much faster, 13 productions learned with 20 problems in 77 minutes versus 238 minutes in programming by demonstration. They also found that programming by tutoring produced a more accurate cognitive model whereby there were fewer productions that produced overgeneralization errors. Programming by tutoring is now the standard approach used in SimStudent and its improved efficiency and effectiveness over programming by demonstration follow from having SimStudent start performing its own demonstrations. Better efficiency is obtained because the author need only respond to each of SimStudent's step demonstrations with a single click, on a yes or no button, which is much faster than demonstrating that step. Better effectiveness is obtained because these demonstrations better expose overgeneralization errors to which the author

responds "no" and the system learns new IF-part preconditions to more appropriately narrow the generality of the modified production rule.

In a second demonstration of SimStudent as an authoring tool, MacLellan, Koedinger & Matsuda (2014) compared authoring in SimStudent (by tutoring) with authoring example-tracing tutors in CTAT. Tutoring SimStudent has considerable similarity with creating an example-tracing tutor except that SimStudent starts to perform actions for the author, which can be merely checked as desirable or not, saving the time it otherwise takes for an author to perform those demonstrations. That study reported a potential savings of 43% in authoring time by using SimStudent to aid in creating example-tracing tutors. A third demonstration by Li, Stampfer, Cohen, and Koedinger (2013) evaluated the empirical accuracy of the cognitive models that SimStudent learns as compared to hand authored cognitive models. The accuracy of a cognitive model in this demonstration was measured by the so-called "smooth learning curve" criteria (Martin, Mitrovic, Mathan & Koedinger, 2011; Stamper & Koedinger, 2011) that tests how well a cognitive model predicts student performance data over successive opportunities to practice and improve. Across four domains (algebra, fractions, chemistry, English grammar), Li et al.(2013) found that the cognitive model acquired by SimStudent produced cognitive models that typically produced better predictions of learning curve data (in 3 of 4 cases). More ambitious attempts to improve and evaluate SimStudent as a tutor authoring aid are underway. SimStudent and other means for AI-driven enhancement of ITSs, including data-driven hint generation and Markov decision process algorithms to optimize tutor action choices, are discussed in Koedinger et al. (2013).

Advanced Authoring for Dialogue-Based Tutors

In dialogue-based ITSs (Graesser et al., 2005; Olney et al., 2012, Rus et al., 2014), the computer attempts to tutor the student by having a conversation with them. These ITSs present similar challenges in ITS authoring as those without natural language dialogue, but there are greater dialogue-authoring demands than typical ITSs. The dialogue is typically authored by a subject matter expert (Graesser et al., 2004), though attempts have been made to semi-automate the process by automatically generating questions and representations that a subject matter can select or modify (Olney, Cade & Williams, 2011; Olney, Graesser & Person, 2012). However, both manual and semi-automated approaches have a common weakness: a shortage of motivated experts. In other words, experts are scarce, and it is uncommon for experts to volunteer their time to author ITS content. Without willing experts to use an authoring tool, an authoring tool will remain unused.

Our recent work addresses the shortage of motivated experts by considering expertise and motivation independently. Expertise may be approximated by allowing novices to do the authoring but then having other novices check the work to ensure quality. Motivation may be addressed by disguising the authoring task as another task in which novices are already engaged. We combine these two approaches in the BrainTrust system. In order to enhance motivation, BrainTrust leverages out-of-class reading activities, specifically online reading activities using eTextbooks through providers like CourseSmart[1], as opportunities for ITS authoring. As students read online, they work with a virtual student on a variety of educational tasks related to the reading. These educational tasks are designed to both improve reading comprehension and contribute to the creation of an ITS based on the material read. After reading a passage, the human student works with the virtual student to summarize, generate concept maps, reflect on the reading, and predict what will happen next. The tasks and interaction are inspired by reciprocal teaching (Palincsar and Brown, 1984), a well-known method of teaching reading comprehension strategies. Thus the key strategies to enhance motivation are leveraging a reading task to which the user has already committed, a teachable agent that enhances motivation (Chase et al., 2009), and a collaborative dialogue that increases arousal (D'Mello et al., 2010).

[1] http://www.coursesmart.com/

The virtual student's performance on these tasks is be a mixture of previous student answers and answers dynamically generated using AI and natural language processing techniques. As the human teaches and corrects the virtual student, they, in effect, improve the answers from previous sessions and author dialogues and a domain model for the underlying ITS. The process of presenting previously proposed solutions to a task for a new set of users to improve upon has been called "iterative improvement" in the human computation literature (von Ahn, 2005; Chklovski, 2005; Cycorp, 2005). These methods often use a simple heuristic that if the majority of evaluating users agrees a solution is correct, then the solution is correct, a process sometimes referred to as "majority voting." However, even simple tasks, such as determining if an image includes the sky, can have non-agreeing "schools of thought" who systematically respond in opposing ways (Tian & Zhu, 2012). Therefore it is preferable to use Bayesian models of agreement jointly to determine the ability of the user (and their trustworthiness as teachers) as well as the difficulty of the items they correct (Raykar et al., 2010). Although Bayesian approaches of this kind are an emerging research area, they are being actively pursued by the massive open online course (MOOC) community, because peer-grading is an important component of scaling MOOCs to many thousands of students (Piech et al., 2013).

Because BrainTrust activities are designed to facilitate learning (both of the content and of reading comprehension strategies) while preserving motivation, not all of the BrainTrust activities directly relate to the authoring of an ITS. In fact, the primary task that relates to ITS authoring is the construction of concept maps, as show in Figure 2. Concept maps can be used to generate exercises and questions in a dialogue-based ITS (Olney, Cade & Williams, 2011; Olney, Graesser & Person, 2012). They can also be used to generate rather trivially direct instruction, e.g., "attitudes are made of emotions and beliefs," as in Figure 2. With a small amount of additional information, such as the overall gist of a text passage, concept maps can also be used to generate larger summaries or "ideal answers" (Graesser et al., 2005). In the example given, the gist is "attitudes and attitude change," and using this gist, a concept-map driven summary can be topicalized so that "attitudes" are the key concept rather than another node. Topicalization is important because non-hierarchical concept maps can be read in any order, but a given text passage can only be read in the linear order in which it was written.

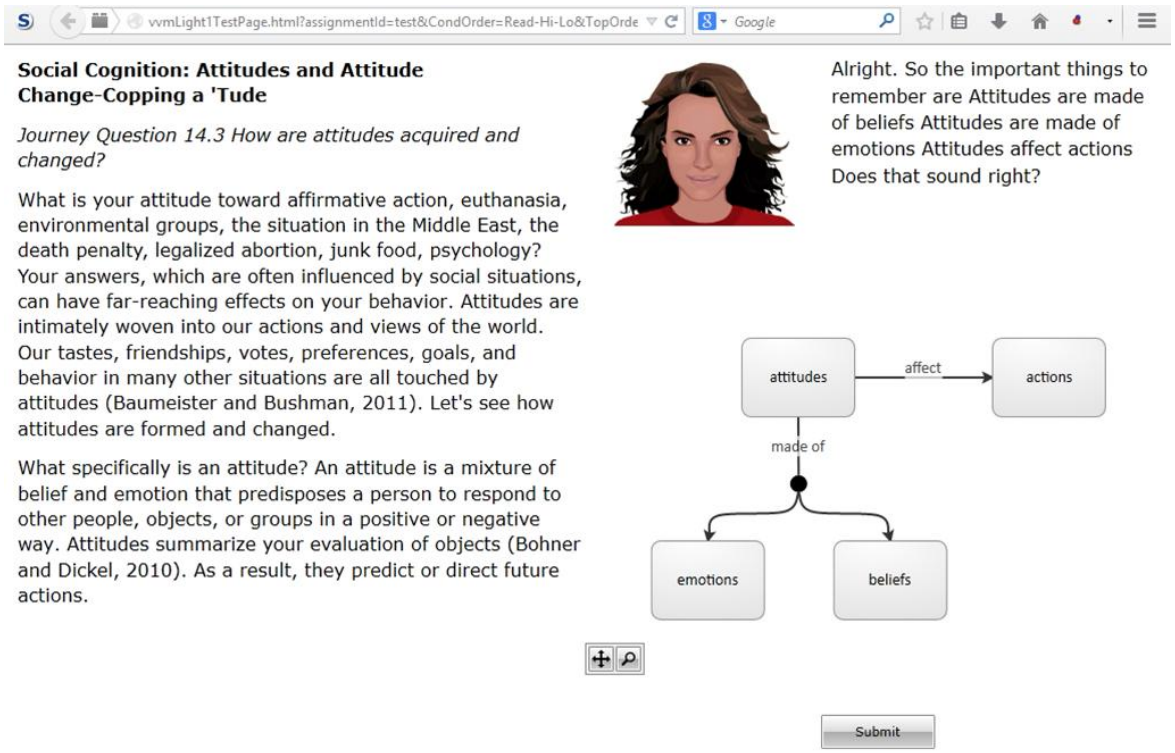

Figure 7. BrainTrust during a concept map activity

Advanced Component-Based Authoring

The previous sections describe efforts to automate authoring of a particular ITS component, such as the model in model-tracing tutors. However, there are also emerging technologies that facilitate the reuse and dynamic configuration of existing components, which allow for a different kind of automated authoring. Instead of authoring the components, these technologies attempt to dynamically assemble components for a particular learning objective. An outline of these technologies is presented in this section as a possible path forward to completely remove ITS expertise required in component-based authoring. In short, the steps to this process, addressed in further detail below, are the following:

(1) Gather content.

(2) Make the content discoverable.

(3) Make the content customizable.

(4) Generate additional tutoring-type information.

(5) Perform delivery for both information and practice sessions.

(6) Perform ITS-standard tasks (learner modeling, experience tracking, etc.); not discussed here.

(7) Repeat: perform pedagogical selection/adaption (steps 1–6).

With regards to gathering (1) of content, the Internet presents a wealth of information, but little of it is relevant to educational goals. There are a few such efforts that attempt to make learning-specific resources available: the Learning Registry (Jesukiewicz & Rehak, 2011), Gooru Learning (GooruLearning, 2014) and, to a lesser extent, the Soldier-Centered Army Learning Environment (Mangold, Beauchat, Long & Amburn, 2012). Fundamentally, each of these has faced the problem of indexing learning content for gathering purposes. The Learning Registry adopts a solution of maintaining separately developed, but interlinked, content repositories while making indexing information available. Gooru instead allows for a centrally managed cloud of content, indexed in the same fashion as a search engine.

Continuing with the Internet analogy, search engines make content discoverable through indexing and cross-referencing. Each of the two main learning architectures must make the content discoverable (2) for a given topic in order for it to be used. Gooru Learning takes a traditional web approach of using community-curated metadata tags, while the Learning Registry has a project to automate the generation of these tags using a project called "Data for Enabling Content in Adaptive Learning Systems (DECALS)" (Veden, 2014). Both approaches make use of metadata-based descriptions of the content in order to drive content selection, in approaches inspired by search engines. The content is made customizable (3) through the editing access from the Gooru platform, or a Sharable Contend Object Reference Model (SCORM) editing and packaging standard (Initiative, 2001) is made available through the Re-Usability Support System for eLearning (RUSSEL) for management of repurposing courses, documents, and multimedia (Eduworks Corporation, 2014). Such systems allow content to be found via search of metadata attributes (e.g., reading level, interactivity index, etc.) and customized for the user.

Generating tutoring-type information (4) is discussed in the previous sections, but simply involves the supplementation of a piece of content with learning-relevant information. One such example of a process involves the generation of a concept map of the key topics contained within the indexed material. Such a concept map can then be used grouping of learning content, with underlying content used for the supplementation of additional learning-relevant items. Examples of such machine-generated learning-relevant information include topic sequencing (Robson, Ray & Cai, 2013), question generation for learning assessment (Olney, Graesser & Person, 2012), hint generation for student help during learning, or other supplemental information.

Content delivery (5) is both an easy and a difficult problem. Both Gooru and the Generalized Intelligent Frameworks For Tutoring (GIFT) support delivery via a web browser, which can easily deliver the majority of modern content. Difficulty stems from more complex SCORM objects, executable programs, 3D simulations, or other items. RUSSEL makes use of human authoring of Gagne's 9 events (Gagné & Gagné, 1985), while GIFT automates the process of authoring through the Rule/Example/Recall/Practice quadrants of Merril's Component Display Theory (Wang-Costello, Tarr, Cintron, Jiang & Goldberg, 2013). The potential to automate the delivery of content based on searchable metadata parameters is one of the key services missing from most of the instructional architectures, but has great potential for content to reach a wide audience quickly.

With the difficult problem of content delivery, the user may be given a simulated environment to practice their obtained knowledge. The integration of intelligent tutoring technologies into systems of practice is not an easy problem, but it is one that is commonly addressed. This integration is a frequent and standard use of the GIFT and Cognitive Tutor systems (e.g., Aleven et al., 2009; Ritter & Koedinger, 1997). However, the current adaptive content (hints, prompts, pumps, etc.) is hand-generated. It may be possible to use the generated tutoring-style information from the previous sections of this work to assist within the practice environment, and eschew the need for expert authoring. As an example, the ordering-based hint "you should multiply before you add" can be generated from the content and used to populate the practice environment.

The above sequence of technologies has the potential to create an adaptive learning system without human intervention. Even substantially diminishing the human workload required would represent a significant savings of time. As part of this overall vision, learning content can be found on the Internet, indexed and sorted into repositories, tagged with searchable metadata information, supplemented with tutoring information, and delivered via browser. The combination of these technologies can allow an instructional system to use an instructional template (e.g., "Rule" content), define user characteristics (e.g., low motivation), match it with intended metadata (e.g., animated/interactable), query a learning system for the appropriate content on the subject (e.g., 4th grade history), and deliver it to the student. Such a combination averts the problem of authoring by reusing existing ITS components for a particular learning objective.

Closing the Authoring Loop: Continuous Feedback and Improvement

Many ITSs have a number of free parameters that must be fixed during the authoring process. For example, in dialogue-based systems, the author must decide how "correct" a student answer should be in order to be counted correct, e.g., must it be exactly the same as the ideal answer or can it be "close enough." Once fixed, these parameters usually remain fixed until the ITS is overhauled or re-parameterized with new data.

However, ideally, we would close the loop and an ITS would be "self-updating" such that the parameters of the theory of learning would be automatically adjusted to be more optimal as more students used the system. While automated authoring of content involves creating exercises for students to interact with, automated improvement in the pedagogical interactions means modifying the learner model used for pedagogical decision making. For example, a self-updating system may be able to make use of information on population dynamics to provide a "best guess" for model parameters of an unseen student. Such guesses could be updated based upon their effect on learning among groups, allowing broader applicability of the ITSs.

To do such continuous improvement will require a flexible model that characterizes the student learning in the domain. Flexible implies that the model will behave in multiple different ways, depending on how it is configured with parameters or mechanisms. For example, a model might characterize the different outcomes for the student from success and failure with a practice problem. This model can be flexible in its representation of the effect of the success and failure if the model allows this difference to vary, for example, by quantifying success and failure effects numerically. Similarly, a model might characterize forgetting, but again a flexible representation of forgetting would specify that it might range from none at all to very fast depending on some numerical parameter. Again, the model allows for continuous variation in the model space.

Given such a flexible model, one can configure a system with only preliminary settings for the different flexible mechanisms. Following this initial cold start, the system would be designed to be self-tuning, such that the model continuously improves both for groups of students and individual students connected in a server-client architecture. While the network communication and mathematical complexity of this proposal makes it challenging, the possibility for better effectiveness with students in ITSs may also be large. It should also be noted that similar, but conceptually much simpler, A/B tests are now commonly used in industry (e.g., deciding how many search results to put on a page). In the next few paragraphs, we sketch the outlines for such a system.

The system would be controlled by a central server that receives data from the individual clients in order for the server to reestimate parameters. These group estimates of parameters would then be offered to existing and new clients. This system would allow for all the students' data to be quickly analyzed by the

server to see if the default parameters resulted in a good fit or if they needed to be adjusted. Adjustments would be gradual. Default parameters would thus incrementally evolve on the server for the task, depending on the clients. These default parameters mean that the system will adapt to different contexts of use. For example, a poor performing school district might give rise to parameters that reflect higher forgetting than a better funded district. An adapted system would be expected to promote better learning, since the accuracy of the model effects the accuracy of pedagogical decision making.

In addition to this tracking at the server level, the individual student models would be adjusted at the client level as well. For example, a low performing student might find the task hard, since the system would have adapted to the average student. The client level tracking would correct this inaccuracy very quickly, since the client data would weigh heavily on the model parameters provided by the server. In fact, the server level tracking would function more as a seed for new students than having active effects on a student, minute to minute, supplied by the client level model.

Such capabilities are currently possible but have not been explored greatly. Some systems have been constructed which illustrate this client level tracking of the student model. For example, in the FaCT system experimental software (Pavlik Jr. et al., 2007), the model that controls student actions can be configured to automatically take optimization steps every N number of student practices. After N practices, any particular parameter can be optimized one step either up or down by a specific increment (the step size). This is accomplished by computing the log-likelihood of model fit for the parameter above, below and at the current value. The step size can be specified to determine how fast adaptation occurs. If adaptation occurs too quickly with too little data, pedagogical decisions may fluctuate too wildly.

One problem with this strategy is that often there is not enough variability in the data due to the consistency of the pedagogical decisions. For example, the FaCT system tries to balance correctness and spacing, and generally recommends practice at around 95% correct. Unfortunately, this means that there is little variability in the conditions of the data collection with which to improve the model. One solution to this is to embed small experiments in the tutored practice in order to better measure the parameters in the individual student's model. These embedded randomized trials might be delivered at much wider spacing than the tutor selected items, making them more difficult. Efforts like this to create varying conditions in the tutor data may be necessary to make sure that automated adjustment systems have some variability in the data in order to identify the parameters being optimized as being unique from the other parameters.

Discussion

The new approaches to authoring discussed in this chapter overlap in the problems they are trying to solve. Firstly, there must be content that the user will interact with, whether it is digital characters, simulated environments, or static webpages. This content is usually authored by a subject matter expert (SME) or an instructional design expert (ISD). SimStudent, BrainTrust, and component-based authoring all try to ease the burden of authoring content while still keeping a human "in the loop."

Secondly, adaptive tutoring systems must have something to adapt to, usually through the modeling of expert and learner knowledge. While this content has traditionally been authored by an SME, SimStudent and BrainTrust speed the authoring of both these models simultaneously, because they compare student actions to an ideal, or expert, answer. Component-based authoring can make use of diverse student models, and a system with continuous improvement and feedback re-parameterizes the student model making best use of the learner data collected to date.

Thirdly, adaptive tutoring systems must contain instruction and feedback to give to the student when diagnosed with a deficiency in a proficiency. These items consist of hints, scenario adaptations, texts, summaries, or other items in response to student actions. There are several efforts at authoring tools which attempt to automate this process. SimStudent uses author demonstrations and collects feedback from the author on the steps SimStudent performs to learn production rules that can be employed to check student solution progress and to generate next step hints when a student is stuck. BrainTrust uses the question-generation technology previously developed for Guru and uses concept maps to generate various kinds of question, e.g., hints, prompts, verification, as well as direct instruction. Similar reusable components are being developed for concept maps and question generation (Robson, Ray & Cai, 2013).

Naturally, all of the items of the ITS system must be delivered through an actual system, which is usually developed through the programming of a simulation or conversational interaction. Both SimStudent and BrainTrust assume specific systems, namely, they produce *expert models* to be used in existing model-tracing and dialogue-based systems. To create other modules (e.g., interface and tutoring modules), other tools (e.g., CTAT) or other system-level programming may be necessary. In contrast, component-based authoring addresses programming more comprehensively by attempting to dynamically assemble systems out of existing components with no additional programming.

The above discussion is summarized in Table 1. Both SimStudent and BrainTrust address the majority of authoring needs but do not squarely address system-level programming. If SimStudent is used as a module within CTAT, then systems-level programming support is provided through the remainder of the CTAT suite. CTAT provides non-programmer authoring tools for interface development and algorithms (model tracing and knowledge tracing) that provide adaptive tutoring support when given the production rules that SimStudent automatically learns. Component-based approaches and continuous improvement, as presented in this chapter, most directly address the authoring needs of programming and assessment, respectively. However, each approach is quite general and could be applied to other authoring needs.

Table 1 Authoring roles addressed by emerging approaches discussed in this chapter.

Authoring Need	Human role	SimStudent	BrainTrust	Component-based	Continuous-improvement
Content	SME & ISD	✓	✓		
Assessment	SME & ISD	✓	✓		✓
Instruction/Feedback	SME & ISD	✓	✓		
Programming	Programmer			✓	

Note. SME: Subject Matter Expert; ISD: Instructional Design Expert

As this chapter has focused on emerging areas of research, it is perhaps no surprise that these areas are operating somewhat in their own silos, motivated by authoring problems in their own ITS traditions. Perhaps a total integration of these approaches may not be possible, given the differences discussed in the introduction and depicted in Figure 1. To that end, it may be preferable for these emerging areas to continue to develop following their own needs, but also more broadly to the needs of their tutoring paradigm, namely, model-tracing, dialogue-based, and constraint-based. If general tools can be made for these quadrants, then in time it may be possible to assemble an integrated suite of tools that, once a paradigm has been selected, afford the greatest degree of automation possible so that ITS learning objectives may be authored completely, accurately, and quickly.

Finally, Figure 1 has an empty quadrant corresponding to path-oriented tutors that indirectly compare student activities to an ideal answer. Whether existing ITS research can properly be located in this quadrant is unclear, but there are several possibilities that may have implications for tutoring in ill-defined domains (Fournier-Viger, Nkambou & Nguifo, 2010; Lynch et al., 2006). In particular, it may be that tutors using case-based reasoning (CBR), such as those used for tutoring the law (Aleven, 2003), fall into this quadrant, because they are both path-oriented and only indirectly compare student input to an expert solution. CBR compares the current situation to previous situations (i.e., cases) and adapts solutions from previous situations to the current problem (Leake, 1996). From this standpoint, CBR may be viewed as representing solution paths in cases, but these paths are ultimately fragments that might be generalized or recombined in a new situation. Comparison to an ideal answer may be indirect because comparison may apply not only to the final solution (as in constraint-based tutoring), but also to whether the solution made use of the same cases and in the same way. If so, then CBR tutors may be an area of research that is currently underdeveloped and amenable to further research in automated authoring.

Recommendations and Future Research

Based on our findings, we can make several recommendations for GIFT and future ITSs. First, the four quadrants of ITS research described in Figure 1 should continue to be developed, with an end goal that the resulting authoring tools may ultimately form a suite of tools that could generally be applied to any problem in their respective tutoring paradigm. As discussed above, however, these approaches are largely building models and do not implement systems-level programming. To assemble new systems from scratch, GIFT should also encompass component-based authoring. This implies that tools operating in the four quadrants should output reusable components, but it further implies that these components must be discoverable and customizable. Finally, we argue that all future ITSs should implement continuous improvement so that the tutor can better adapt to an individual or specific population. As described in this chapter, continuous improvement best aligns with improving of learner models based on interaction data, but it is also conceivable to implement continuous improvement generally for content, assessment, and instruction.

Very impressive performance support tools for ITS authoring already exist (Aleven et al., 2009) and the research described in this chapter does not propose to replace these tools in the near future. Instead, we recommend that such tools continue to incorporate improvements in automated authoring in the research we describe, so that ITS learning objectives may be authored completely, accurately, and quickly. Indeed, it may be the case that some tasks supported by such performance support tools, such as drag-and-drop editors for building ITS graphical interfaces, may never be completely automated. We anticipate that the current generation of ITS authoring tools will instead continue to be enriched by new advances in automated authoring, which will ultimately lower the cost and increase the adoption of ITS.

References

Aleven, V. (2003). Using background knowledge in case-based legal reasoning: A computational model and an intelligent learning environment. *Artificial Intelligence, 150(1–2)*, 183–237. doi:10.1016/S0004-3702(03)00105-X

Aleven, V., Mclaren, B. M., Sewall, J. & Koedinger, K. R. (2009). A new paradigm for intelligent tutoring systems: Example-tracing tutors. *International Journal of Artificial Intelligence in Education*, *19*(2), 105-154.

Anderson, J. R., Corbett, A. T., Koedinger, K. R. & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4 (2), 167-207.

Chase, C.C., Chin, D.B., Oppezzo, M.A. & Schwartz, D.L. (2009). Teachable Agents and the Protégé Effect: Increasing the effort towards learning. *Journal of Science Education and Technology*, 18(4), 334-352.

Chklovski, T. (2005). Collecting Paraphrase Corpora from Volunteer Contributors. In *Proceedings of the 3rd International Conference on Knowledge Capture* (pp. 115–120). New York, NY, USA: ACM. doi:10.1145/1088622.1088644

Cycorp. (2005). Factory. http://game.cyc.com/. Accessed: 7/23/12.

D'Mello, S. K., Hays, P., Williams, C., Cade, W., Brown, J. & Olney, A. M. (2010). Collaborative Lecturing by Human and Computer Tutors. In *Intelligent Tutoring Systems* (pp. 178–187). Berlin: Springer.

Eduworks Corporation. (2014). Re-Usability Support System for eLearning (RUSSEL). from https://github.com/adlnet/RUSSEL

Fournier-Viger, P., Nkambou, R. & Nguifo, E. M. (2010). Building Intelligent Tutoring Systems for Ill-Defined Domains. In R. Nkambou, J. Bourdeau & R. Mizoguchi (Eds.), *Advances in Intelligent Tutoring Systems* (pp. 81–101). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-14363-2_5

Gagné, R. M. & Gagné, R. M. (1985). *Conditions of learning and theory of instruction*. New York: Holt, Rinehart and Winston.

Gick, M.L. & Holyoak, K.J. (1983). Schema induction and analogical transfer. *Cognitive Psychology, 15,* 1-38.

GooruLearning. (2014). http://www.goorulearning.org. Retrieved 10/6/2014, 2014

Graesser, A. C., Chipman, P., Haynes, B. & Olney, A. M. (2005). AutoTutor: An Intelligent Tutoring System with Mixed-Initiative Dialogue. *IEEE Transactions on Education*, 48(4), 612– 618.

Graesser, A. C., D'Mello, S. K., Hu, X., Cai, Z., Olney, A. & Morgan, B. (2012). AutoTutor. In P. McCarthy & C. Boonthum-Denecke (Eds.), *Applied Natural Language Processing: Identification, Investigation, and Resolution*. (pp. 169–187). Hershey, PA: IGI Global.

Graesser, A. C. & Person, N. K. (1994). Question Asking during Tutoring. *American Educational Research Journal*, 31, 104-137.

Graesser, A. C., Person, N. K. & Magliano, J. P. (1995). Collaborative dialogue patterns in naturalistic one-to-one tutoring. *Applied Cognitive Psychology*, 9, 1-28.

Initiative, A. D. L. (2001). Sharable Content Object Reference Model (SCORM™). *Advanced Distributed Learning, http://www. adlnet. org*.

Jesukiewicz, P. & Rehak, D. R. (2011). The Learning Registry: Sharing Federal Learning Resources. Paper presented at the Interservice/Industry Training, Simulation & Education Conference, Orlando, FL.

Koedinger, K. R., Anderson, J. R., Hadley, W. H. & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, *8,* 30-43.

Koedinger, K.R., Brunskill, E., Baker, R.S.J.d., McLaughlin, E.A., Stamper, J. (2013). New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine, 34*(3).

Koedinger, K.R., Corbett, A.C. & Perfetti, C. (2012). The Knowledge-Learning-Instruction (KLI) framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive Science, 36* (5)*,* 757-798.

Landauer, T. K., McNamara, D. S., Dennis, S. E. & Kintsch, W. E. (2007). *Handbook of latent semantic analysis*. Lawrence Erlbaum Associates Publishers.

Leake, D. B. (1996). *Case-Based Reasoning: Experiences, Lessons and Future Directions* (1st ed.). Cambridge, MA, USA: MIT Press.

Li, N., Matsuda, N., Cohen, W. & Koedinger, K.R. (2015). Integrating representation learning and skill learning in a human-like intelligent agent. *Artificial Intelligence*.

Li, N., Stampfer, E., Cohen, W. & Koedinger, K.R. (2013). General and efficient cognitive model discovery using a simulated student. In M. Knauff, N. Sebanz, M. Pauen, I. Wachsmuth (Eds.), *Proceedings of the 35th Annual Conference of the Cognitive Science Society*. (pp. 894-9) Austin, TX: Cognitive Science Society.

Lynch, C., Ashley, K., Aleven, V. & Pinkwart, N. (2006). Defining ill-defined domains; a literature survey. In *Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains at the 8th International Conference on Intelligent Tutoring Systems* (pp. 1–10). Retrieved from http://people.cs.pitt.edu/~collinl/Papers/Ill-DefinedProceedings.pdf#page=7

MacLellan, C.J., Koedinger, K.R., Matsuda, N. (2014) Authoring Tutors with SimStudent: An Evaluation of Efficiency and Model Quality. *Proceedings of the 12th International Conference on Intelligent Tutoring Systems*. Honolulu, HI. June 5-9, 2014.

Mangold, L. V., Beauchat, T., Long, R. & Amburn, C. (2012). An Architecture for a Soldier-Centered Learning Environment*.* Paper presented at the Simulation Interoperability Workshop.

Martin, B., Mitrovic, T., Mathan, S. & Koedinger, K.R. (2011). Evaluating and improving adaptive educational systems with learning curves. User Modeling and User-Adapted Interaction: The Journal of Personalization Research (UMUAI), 21(3), 249-283. [2011 James Chen Annual Award for Best UMUAI Paper]

Matsuda, N., Cohen, W. W. & Koedinger, K. R. (2015). Teaching the Teacher: Tutoring SimStudent leads to more Effective Cognitive Tutor Authoring. *International Journal of Artificial Intelligence in Education, 25*, 1-34.

McTear, M. F. (2002). Spoken dialogue technology: enabling the conversational user interface. ACM Computing Surveys (CSUR), 34, 90-169.

Mitrovic, A. (2012). Fifteen years of constraint-based tutors: what we have achieved and where we are going.User Modeling and User-Adapted Interaction, 22, 39-72.

Mitrovic, A., Suraweera, P., Martin, B., Zakharov, K., Milik, N., Holland, J. (2006). Authoring constraint-based tutors in ASPIRE. In Ikeda, M., Ashley, K., Chan, T.-W. (eds.), *Proceedings of ITS 2006*. LNCS, vol. 4053, pp. 41–50.

Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J., McGuigan, N. (2009). ASPIRE: an authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education*, 19(2), 155–188.

Nye, B. D., Graesser, A. C. & Hu, X. (2014). AutoTutor and Family: A Review of 17 Years of Natural Language Tutoring. *International Journal of Artificial Intelligence in Education*, 24(4), 427–469. doi:10.1007/s40593-014-0029-5

Ohlsson, S. (1992). Constraint-based student modelling. *International Journal of Artificial Intelligence in Education*, 3, 429-447.

Ohlsson, S. & Mitrovic, A. (2007). Fidelity and Efficiency of Knowledge Representations for Intelligent Tutoring Systems. *Technology, Instruction, Cognition and Learning (TICL)*, 5, 101-132.Olney, A. M., Graesser, A. C. & Person, N. K. (2012). Question generation from concept maps. *Dialogue & Discourse, 3*(2), 75-99.

Olney, A. M., Cade, W. & Williams, C. (2011). Generating Concept Map Exercises from Textbooks. In *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications* (pp. 111–119). Portland, Oregon: Association for Computational Linguistics. Retrieved from http://www.aclweb.org/anthology/W11-1414

Olney, A. M., D'Mello, S., Person, N., Cade, W., Hays, P., Williams, C., Graesser, A. (2012). Guru: A Computer Tutor That Models Expert Human Tutors. In S. Cerri, W. Clancey, G. Papadourakis & K. Panourgia (Eds.), *Intelligent Tutoring Systems* (Vol. 7315, pp. 256–261). Springer Berlin / Heidelberg.

Olney, A. M., Person, N. K. & Graesser, A. C. (2012). Guru: Designing a Conversational Expert Intelligent Tutoring System. In P. McCarthy, C. Boonthum-Denecke & T. Lamkin (Eds.), *Cross-Disciplinary Advances in Applied Natural Language Processing: Issues and Approaches* (pp. 156–171). Hershey, PA: IGI Global.

Palinscar, A. S. & Brown, A. L. (1984). Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and instruction*, 1(2), 117-175.

Pavlik Jr., P. I., Presson, N., Dozzi, G., Wu, S.-m., MacWhinney, B. & Koedinger, K. R. (2007). The FaCT (Fact and Concept Training) System: A new tool linking cognitive science with educators. In D. McNamara & G. Trafton (Eds.), *Proceedings of the Twenty-Ninth Annual Conference of the Cognitive Science Society* (pp. 1379–1384). Mahwah, NJ: Lawrence Erlbaum.

Person, N. K., Graesser, A. C., Magliano, J. P. & Kreuz, R. J. (1994). Inferring what the student knows in one-to-one tutoring: The role of student questions and answers. *Learning and Individual Differences*, 6, 205–229.

Piech, C., Huang, J., Chen, Z., Chuong Do, Andrew Ng & Daphne Koller. (2013). Tuned Models of Peer Assessment in MOOCs. In D'Mello, S. K., Calvo, R. A. & Olney, A. (Eds.), *Proceedings of the 6th International Conference on Educational Data Mining* (pp. 153–160).

Raykar, V. C., Yu, S., Zhao, L. H., Valadez, G. H., Florin, C., Bogoni, L. & Moy, L. (2010). Learning From Crowds. *Journal of Machine Learning Research*, 11, 1297–1322.

Ritter, S. & Koedinger, K. R. (1996). An architecture for plug-in tutoring agents. In *Journal of Artificial Intelligence in Education*, 7 (3/4), 315-347. Charlottesville, VA: Association for the Advancement of Computing in Education.

Robson, R., Ray, F. & Cai, Z. (2013). Transforming Content into Dialogue-based Intelligent Tutors. Paper presented at the The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC), Orlando, FL.

Roediger, H.L. & Butler A.C. (2011). The critical role of retrieval practice in long-term retention. *Trends in Cognitive Science 15*:20–27

Rus, V., Stefanescu, D., Niraula, N. & Graesser, A. C. (2014). DeepTutor: Towards Macro- and Micro-adaptive Conversational Intelligent Tutoring at Scale. In *Proceedings of the First ACM Conference on Learning @ Scale Conference* (pp. 209–210). New York, NY, USA: ACM. doi:10.1145/2556325.2567885

Stamper, J.C. & Koedinger, K.R. (2011). Human-machine student model discovery and improvement using data. In G. Biswas, S. Bull, J. Kay & A. Mitrovic (Eds.), *Proceedings of the 15th International Conference on Artificial Intelligence in Education*, pp. 353-360. Berlin: Springer.

Tian, Y. & Zhu, J. (2012). Learning from Crowds in the Presence of Schools of Thought. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 226–234). New York, NY, USA: ACM. doi:10.1145/2339530.2339571

VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education, 16*(3), 227-265.

Veden, A. (2014). Data for Enabling Content in Adaptive Learning Systems (DECALS). from https://github.com/adlnet/DECALS

Von Ahn, L. (2005). Human Computation (Doctoral thesis). Carnegie Mellon University.

Wang-Costello, J., Tarr, R. W., Cintron, L. M., Jiang, H. & Goldberg, B. (2013). *Creating an Advanced Pedagogical Model to Improve Intelligent Tutoring Technologies.* Paper presented at the The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC).

Wylie, R., Koedinger, K. R. & Mitamura, T. (2010). Analogies, explanations, practice: Examining how task types affect second language grammar learning. In V. Aleven, J. Kay & J. Mostow (Eds.), *Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 214-223). Heidelberg, Berlin: Springer.

Zhu, X. & Simon, H. A. (1987). Learning mathematics from examples and by doing. *Cognition and Instruction, 4*(3), 137-166.