GnuTutor: An Open Source Intelligent Tutoring System Based on AutoTutor

Andrew M. Olney
Institute for Intelligent Systems
University of Memphis
Memphis, Tennessee 38152

Abstract

This paper presents GnuTutor, an open source intelligent tutoring system (ITS) inspired by the AutoTutor ITS. The goal of GnuTutor is to create a freely available, open source ITS platform that can be used by schools and researchers alike. To achieve this goal, significant departures from AutoTutor's current design were made so that GnuTutor would use a smaller, non-proprietary code base but have the major functionality of AutoTutor, including mixed-initiative dialogue, an animated agent, speech act classification, and natural language understanding using latent semantic analysis. This paper describes the GnuTutor system, its components, and the major differences between GnuTutor and AutoTutor.

Introduction

Just as human tutors arguably provide the most effective form of human instruction (Bloom 1984), intelligent tutoring systems (ITS) arguably provide the most effective form of e-learning (Dodds and Fletcher 2004; Wisher and Fletcher 2004). While ITS may tutor different topics or have different underlying implementations (cf. Falmagne et al. (2006) and VanLehn et al. (2002)), they share a common conceptual base: model the student and tailor instruction based on that model (VanLehn 1988).

But that is not the only thing ITS have in common: ITS are usually closed source systems that are available for a fee (e.g. Cognitive Tutor (Koedinger et al. 1997), Aleks (Falmagne et al. 2006), AutoTutor (Graesser et al. 2005a)) if they are available at all. While fee-based distribution may be successful for implementing curriculum change in primary and secondary education, it is less successful at creating an fertile environment for continued research on ITS (including teaching the design of ITS as well as academic research) and the adaptation of ITS by end-users to specific educational contexts. This paper addresses the need for a freely distributable ITS by presenting GnuTutor[1], an ITS inspired by AutoTutor. GnuTutor therefore shares the common conceptual base of its predecessors while differing in its approach to distribution: GnuTutor is available to all under the Gnu Public License[2] and is cross-platform. To the author's knowledge, GnuTutor is the only open-source release of an ITS.

The sections that follow describe how significant departures from AutoTutor's current design were made so that GnuTutor would use a smaller, non-proprietary code base but still have the major functionality of AutoTutor, including mixed-initiative dialogue, an animated agent, speech act classification, and natural language understanding using latent semantic analysis. This paper describes the GnuTutor system, its components, and the major differences between GnuTutor and AutoTutor.

AutoTutor

The AutoTutor system simulates a human tutor's natural language interaction with a student, so a student learns by having a conversation with AutoTutor (Graesser et al. 2001; VahnLehn et al., 2007). Pedagogical dialogue is therefore the core element of AutoTutor. In order to ensure that AutoTutor's pedagogical dialogue was appropriate and effective, the dialogue was modeled after novice human tutors at the University of Memphis (Graesser and Person, 1994; Graesser, Person, and Magliano, 1995; Person et al., 1994). Multiple subject domains of AutoTutor have been developed since its inception, including conceptual physics (Graesser et al. 2005a; VanLehn et al. 2007), computer literacy (Graesser et al. 2001b), and critical thinking (Cai et al. 2009), with corresponding learning gains in users of roughly .8 sigma, depending on the version and on the control conditions (e.g. read textbook, do-nothing, etc.).

Over the past ten years, versions of AutoTutor have been created not only for different subject matters but also for different platforms, including desktop (Graesser et al. 2001b), web browser (Olney et al. 2002), and an Internet version using .NET remoting (Graesser et al. 2005a). The target platform has become increasingly complex over time, from a standalone installation to one requiring multiple servers and databases. As a result, while the current architecture of AutoTutor is the most powerful (Graesser et al. 2005a; 2005b), it is also the most complex.

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

[2]http://www.gnu.org/licenses/gpl.html

Although the architecture has changed greatly over the years, the appearance and core interactivity of AutoTutor has remained much the same[3]. The interface contains at least one animated agent with synthesized speech, a text entry box for student input, a dialogue history that displays all the conversation between tutor and student, and a multimedia panel that displays images, movies, etc.

An AutoTutor session begins when the tutor presents a problem or asks a deep reasoning question, and the session concludes when the student gives a complete, essay-length answer. In between beginning and end, the tutor and the student engage in a conversation in which the tutor helps the student construct the best possible answer to the question or problem. In order to achieve this goal, AutoTutor must correctly interpret the student's input, compare the student's input to its internal representation of the correct answer, and plan what to say next. These functions map on to speech act classification, natural language understanding, and dialogue planning respectively. These three components, together with the user interface, are the common core of various versions of AutoTutor.

Therefore GnuTutor, as an open source reimplementation of AutoTutor, must have the functionality captured in these four components. The following sections describe GnuTutor's implementations in turn, noting in particular the differences between them and the corresponding implementations in AutoTutor.

Speech Act Classification

The speech act classifier used in GnuTutor is based on an early open source version of the AutoTutor speech act classifier (Olney et al. 2003). The primary objective of the speech act classifier is to detect initiative, i.e. is the student continuing the conversational topic of the tutor or introducing a new topic of discussion. Analysis of human tutoring sessions shows that students introduce a new topic by asking a question, and these questions can be categorized according to their content and the type of information sought (Graesser, Gordon, and Brainerd 1992). Table 1 presents 16 question categories from this analysis.

The speech act classifier determines the category of the student input by matching it against a set of regular expressions. Each category of input has one or more corresponding regular expressions, which may match keywords associated with that category, patterns of keywords, or syntactic patterns (Olney et al. 2003). Heuristics are used to select a category when multiple regular expressions match the student input. The current version of the AutoTutor speech act classifier uses the original categories (Graesser et al. 2005a; 2005b), but also uses a parser to analyze student input before classification.

In contrast, GnuTutor's speech act classifier uses the Brill part of speech tagger (Brill 1992) instead of a parser. Conveniently a GPL version of the Brill tagger exists for C# and meshes well with the .NET implementation of GnuTutor[4]. Finally, though no extensive evaluation of AutoTutor's current speech act classifier has been carried out, the version upon which the GnuTutor speech act classifier is based, Olney (2003), has a weighted F-measure performance of 97%, arguably sufficient for most purposes.

Natural Language Understanding

In GnuTutor, as in AutoTutor, the meaning of student input is analyzed after speech act classification occurs. If a student is continuing with the tutor's topic, i.e. gives a contribution, the tutor will analyze their response in terms of what the student was expected to say. These "expectations" of the tutor represent correct answers by the student. Thus the closer the student's answer is to an expectation, the more the student has mastered that part of the material and the more positive the tutor's feedback will be.

This notions of "closeness" can be captured using Latent Semantic Analysis (LSA), a machine learning technique capable of representing world knowledge (Deerwester et al. 1990; Dumais 1991; Landauer and Dumais 1997). The LSA process begins with a corpus, or collection of text, which is converted into a term document matrix by counting the number of times a $term_i$ is present in a $document_j$. This term document matrix is then decomposed into a lower dimensional approximation using singular value decomposition, such that text from the corpus may be projected into the lower dimensional space as vectors. Moreover, it can be shown that similar words in the corpus correspond to close vectors in the lower dimensional space; for more information on the mathematical details of LSA, see (Manning and Schütze 1999).

LSA has been shown to closely approximate vocabulary acquisition in children (Landauer and Dumais 1997), grade essays as reliably as experts in English composition (Foltz, Gilliam, and Kendall 2000), and understand student contributions in tutorial dialogue (Graesser et al. 2000; Olde et al. 2002). These results are particularly impressive considering that LSA creates its knowledge representation without human intervention.

GnuTutor's open source implementation of AutoTutor's LSA functionality has two major components, matrix creation and singular value decomposition (SVD). Unlike AutoTutor, which uses the closed source Bellcore LSI tools for these two steps, GnuTutor uses its own C# classes for matrix creation and manipulation and an outside program, currently an open source Matlab clone called Octave[5], to perform the SVD; however, any number of SVD programs could be used interchangeably, including GTP (Giles, Wo, and Berry 2003), PROPACK (Larsen 1998), SVDLIBC (Rohde 2007), or ARPACK (Lehoucq, Sorensen, and Yang 1998), amongst others. In terms of functionality, the LSA implementations of GnuTutor and AutoTutor are identical.

[3]Versions of AutoTutor have been reported that use speech recognition, question answering, interactive simulations, and emotion detection, amongst others. However, it is argued that the core approach of constructivist dialogues has held constant.

[4]Csharp_NLP_tagger_version_1.0. http://markwatson.com/opensource/

[5]http://www.gnu.org/software/octave/

Category	Example
Questions	
Verification	Does the pumpkin land in his hands?
Disjunctive	Is the pumpkin accelerating or decelerating?
Concept Completion	Where will the pumpkin land?
Feature Specification	What are the components of the forces acting on the pumpkin?
Quantification	How far will the pumpkin travel?
Definition	What is acceleration?
Example	What is an example of Newton's Third Law?
Comparison	What is the difference between speed and velocity?
Interpretation	What is happening in this situation with the runner and pumpkin?
Causal Antecedent	What caused the pumpkin to fall?
Causal Consequence	What happens when the runner speeds up?
Goal Orientation	Why did you ignore air resistance?
Instrumental/Procedural	How do you calculate force?
Enablement	What principle allows you to ignore the vertical component of the force?
Expectational	Why doesn't the pumpkin land behind the runner?
Judgmental	What do you think of my explanation?
Frozen Expressions	
Metacognitive	I don't understand.
Metacommunicative	Could you repeat that?
Contribution	The pumpkin will land in the runner's hands.

Table 1: AutoTutor's Speech Act Classification Scheme (Olney et al. 2003).

Dialogue Planning

Because AutoTutor is dialogue-centric, computational modeling of dialogue is critically important. There are a number of ways to model dialogue, and different versions of AutoTutor have incorporated aspects of the major methodologies: finite state, frame based, and plan based (McTear 2002). These methodologies and their AutoTutor implementations will be briefly described before progressing to GnuTutor's dialogue model.

Finite state approaches (Nuance 1996; McTear 1998) allow users to progress through the dialogue in a fixed order. Dialogues are represented by a set of states, and each state is connected to other states via a valid user input. Accordingly, finite state models can be easily visualized as a graph or flowchart. The common example is a voice menu, e.g. "Press zero to hear more options." Finite state systems have the advantage that user options are limited, thus the number of possible (valid) utterances the user could produce are very small. The smaller the number of possible utterances, the smaller the search space for the dialogue understander, and the more efficient and robust the system is.

Frame-based systems (Graesser et al. 2001a; Olney et al. 2002) are similar to finite state systems in that the number of valid options is limited, but the user can access these options in any order. A common example is that of an automated travel agent. The system must elicit the dates of travel, the departure and arrival cities, etc., but the speaker can produce this information in any order. The system matches user input to slots in a frame, or table, and then this frame is typically used to complete a database query.

Plan-based systems have been largely guided by the theories of speech acts (Austin 1962; Searle 1975). Amongst other things, speech act theory recognizes that conversational utterances can have an intended effect, or illocutionary force, independent of the literal meaning of the utterance. For example, "Can you please pass the salt" is not intended as a yes/no question but rather as a request. The intended effect of a utterance is largely synonymous with a goal, thus understanding an utterance can be considered as a problem of inferring the speaker's goal. This goal based perspective creates a bridge between utterance understanding and the artificial intelligence tasks of planning and theorem proving, and allows the methods of planning and theorem proving (Fikes and Nilsson 1971) to be applied to utterance understanding and dialogue modeling. Early systems (Cohen and Perrault 1979; Cohen 1984; Litman and Allen 1987) utilized AI planning methods both for understanding single utterances and for understanding utterances as part of complex dialogues.

In various versions, AutoTutor has used finite-state, frame-based, and plan based models of dialogue (Graesser et al. 2001a; Olney et al. 2002; Graesser et al. 2005b). Phases of tutoring, e.g. introduction and summary, have been modeled using both finite-state machines and plans, with plans giving the additional power to implement recursive mixed-initiative dialogues. Frames are used to represent progress towards AutoTutor's goals. In AutoTutor, the goal is to have the student provide a multi-part, essay-length answer to a deep-reasoning question. Each answer part fills a slot in a frame until the answer is complete.

AutoTutor's plans are hierarchically decomposable recipes for action, with associated termination conditions. For example, a plan might consist of the following sequence of actions: update coverage, feedback, hint. Or a plan might be an iteration, or loop, with some given termination condition, e.g. keep trying to cover expectations until they are all covered. These two examples illustrate how AutoTutor's plans, while properly called plans, are markedly different from the theorem-proving style planners mentioned earlier.

Each of these approaches to dialogue modeling have advantages and disadvantages with respect to ease of use, ease of implementation, and ability to scale up (for a general review of advantages and disadvantages, see McTear (2002)). Ideally a dialogue modeling approach should optimize all of these capabilities. It is worthwhile to consider the current dialogue management of AutoTutor in terms of these capabilities. First, this dialogue manager created a little language for describing dialogues together with a corresponding interpreter to execute dialogues in this language (Graesser et al. 2005b). This ability to author dialogues using dialogue "widgets" supported more rapid creation of tutorial dialogue patterns than previous finite-state approaches (Graesser et al. 2001a; McTear 1998). Secondly, AutoTutor's dialogue manager does not have backtracking, which sometimes results in poor dialogue. The following example occasionally occurred in question answering. The student would ask a question, AutoTutor would say "That's an interesting question, I'll try to answer it with this," and then AutoTutor would execute a question answering routine to generate the answer. However, sometimes AutoTutor could not answer the question, and so it would say"That's an interesting question, I'll try to answer it with this, I'm sorry I can't answer your question." To avoid this kind of problem, GnuTutor's dialogue manager needs to be able to backtrack to another plan when the current plan fails.

Both backtracking and a language describing dialogues can be accomplished by using Prolog, a declarative language that has been widely used for artificial intelligence research and dialogue systems (Bratko 1986; Larsson et al. 2000; Shoham 1994; Sterling and Shapiro 1994; Zinn, Moore, and Core 2002). In GnuTutor, an open source C# implementation of Prolog[6] has been used to replicate the dialogue functionality of AutoTutor in only 250 lines of Prolog. To duplicate the information state approach to dialogue modeling (Larsson and Traum 2000) currently used by AutoTutor, the GnuTutor dialogue manager uses a C# hash table as a global object for holding dialogue state across turns. This global object is created in C# and passed to the Prolog interpreter, which then can perform lookups on the table and also call methods in the C# code. This functionality allows declarative, or rule-based, aspects of the dialogue to be coded in Prolog, while delegating the procedural aspects, e.g. find the expectation with the lowest LSA score, to the C# code. A fragment of the GnuTutor dialogue manager is shown in Figure 1.

User Interface

The user interface of AutoTutor consists of a text entry box, a multimedia panel, an animated agent, and text-to-speech. For an open source, cross-platform implementation, the last two requirements are non trivial. AutoTutor has used several agent technologies, including Microsoft Agent and the Haptek Player (Graesser et al. 2005b; Olney et al. 2002). While both of these are essentially free (either pre-installed with Windows XP or freely downloadable), neither is open source nor cross-platform.

[6]tuProlog.NET http://tuprolog.alice.unibo.it/

```
%--------------
% CONTRIBUTION
%--------------
handlesac(PLAN,TEMP):-
state(sac,contribution),
update_coverage,
feedback(PLAN,T1),
tutor_initiative(T1,TEMP).
%--------------
% QUESTION
%--------------
handlesac(PLAN,TEMP):-
state(sac,question),
answer_question(PLAN,TEMP).
%-------------------
% METACOMMUNICATIVE
%-------------------
handlesac(PLAN,TEMP):-
state(sac,metacommunicative),
repeat(PLAN,TEMP).
```

Figure 1: A fragment of GnuTutor's dialogue rules in Prolog

Several open source agents and voices are currently available. XFace[7] is an open source, cross-platform 3D agent toolkit with emotional expression (Balcı 2005). Galatea[8] is similarly cross-platform, open source, and 3D, though it has a number of other features, including speech recognition and creation of 3D agents from human photo portraits (Kawamoto et al. 2002). Greta[9] is an advanced open source 3D agent toolkit for Windows (Mancini, Bresin, and Pelachaud 2007). iFace[10], like Greta, is an open source 3D animated agent package available for Windows (Arya et al. 2006). While all of the above are open-source, only the first two are cross-platform.

Given that both XFace and Galatea meet the constraints of GnuTutor, i.e. are open source and cross-platform, the next most meaningful selection criteria is simplicity and ease of use. XFace is significantly simpler to use than Galatea, since XFace uses a client/server architecture by which the server (a window containing the head) may be controlled by any program that can successfully connect to it and send properly formatted (XML) commands. Thus XFace was selected as the primary agent component of GnuTutor. However, to satisfy the additional needs of researchers wishing to explore more advanced agent capabilities in ITS, Greta was selected as the secondary agent component of GnuTutor.

XFace is designed to work with either Flite, an open source, cross-platform TTS (Black and Lenzo 2001), or with Microsoft's Speech Application Programming Interface (SAPI). This duality is advantageous: users wishing to remain open-source and cross-platform can choose Flite, while users wishing to use a higher quality speech engine

[7]http://xface.itc.it/

[8]http://hil.t.u-tokyo.ac.jp/ galatea/

[9]http://www.tsi.enst.fr/ pelachau/Greta/

[10]http://img.csit.carleton.ca/iface/

can make use of the SAPI interface for TTS. Greta can use the open source, cross-platform Mary TTS (Schrder and Hunecke 2007) which offers a variety of reasonably high quality voices.

Conclusion

GnuTutor preserves much of the original functionality of AutoTutor and includes mixed initiative dialogue, speech act classification, an animated agent, and natural language understanding using latent semantic analysis. However significant effort has been made to simplify the codebase to address the needs of different user communities. College instructors will find that GnuTutor is highly suitable for AI lab exercises. End users will find GnuTutor's authoring simple enough to create their own content. And researchers wishing to create derivative ITS will find that GnuTutor provides the core tools within a lightweight framework. GnuTutor is hosted on Sourceforge[11] and is cross-platform under the Mono and .NET runtimes. It is the first open source, conversational intelligent tutoring system.

Acknowledgments

The research reported here was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305A080594, and was supported by the National Science Foundation, through Grant 0826825, to the University of Memphis. The opinions expressed are those of the authors and do not represent views of the Institute or the U.S. Department of Education.

References

Arya, A.; DiPaola, S.; Jefferies, L.; and Enns, J. 2006. Socially communicative characters for interactive applications. In *Proceedings of the 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG06)*.

Austin, J. L. 1962. *How to do things with words*. Oxford: Oxford University Press.

Balcı, K. 2005. Xface: Open source toolkit for creating 3d faces of an embodied conversational agent. In *Smart Graphics*, 263–266. Munich, Germany: Springer.

Black, A., and Lenzo, K. 2001. Flite: a small fast runtime synthesis engine. In *4th ISCA Tutorial and Research Workshop (ITRW) on Speech Synthesis*, 20 – 24. ISCA.

Bloom, B. S. 1984. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher* 13(6):4–16.

Bratko, I. 1986. *Prolog programming for artificial intelligence*. Reading, Massachusetts: Addison Wesley Publishing Company.

Brill, E. 1992. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*. ACL.

[11]http://sourceforge.net/projects/gnututor/

Cai, Z.; Graesser, A. C.; Millis, K. K.; Halpern, D. F.; Wallace, P. S.; Moldovan, C.; and Forsyth, C. 2009. Interactive event: Aries.

Cohen, P. R., and Perrault, C. R. 1979. Elements of a plan-based theory of speech acts. *Cognitive Science* 3(3):177–212.

Cohen, P. R. 1984. The pragmatics of referring and the modality of communication. *Computational Linguistics* 10(2):97–146.

Deerwester, S. C.; Dumais, S. T.; Furnas, G. W.; Landauer, T. K.; and Harshman, R. A. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science* 41(6):391–407.

Dodds, P., and Fletcher, J. D. 2004. Opportunities for new "smart" learning environments enabled by next generation web capabilities. *Journal of Educational Multimedia and Hypermedia* 13(4):391–404.

Dumais, S. 1991. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers* 23(2):229–236.

Falmagne, J.; Cosyn, E.; Doignon, J.; and Thiery, N. 2006. The assessment of knowledge, in theory and in practice. In Missaoui, R., and Schmid, J., eds., *Formal Concept Analysis*, volume 3874, 61. Dresden, Germany: Fourth International Conference on Formal Concept Analysis.

Fikes, R., and Nilsson, N. J. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artif. Intell.* 2(3/4):189–208.

Foltz, P.; Gilliam, S.; and Kendall, S. 2000. Supporting Content-Based Feedback in On-Line Writing Evaluation with LSA. *Interactive Learning Environments* 8(2):111–127.

Giles, J.; Wo, L.; and Berry, M. 2003. GTP (General Text Parser) software for text mining. In Bozdogan, H., ed., *Statistical data mining and knowledge discovery*. Boca Raton: CRC Press. 455–471.

Graesser, A.; Wiemer-Hastings, P.; Wiemer-Hastings, K.; Harter, D.; Tutoring Research Group, T.; and Person, N. 2000. Using Latent Semantic Analysis to Evaluate the Contributions of Students in AutoTutor. *Interactive Learning Environments* 8(2):129–147.

Graesser, A. C.; VanLehn, K.; Rose, C.; Jordan, P.; and Harter, D. 2001a. Intelligent tutoring systems with conversational dialogue. *AI Magazine* 22:39–51.

Graesser, A.; Person, N.; Harter, D.; et al. 2001b. Teaching tactics and dialog in AutoTutor. *International Journal of Artificial Intelligence in Education* 12(3):257–279.

Graesser, A. C.; Chipman, P.; Haynes, B.; and Olney, A. 2005a. AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education* 48(4):612– 618.

Graesser, A. C.; Olney, A. M.; Haynes, B. C.; and Chipman, P. 2005b. AutoTutor: A cognitive system that simulates a tutor that facilitiates learning through mixed-initiaive dialogue. In Forsythe, C.; Bernard, M. L.; and

Goldsmith, T. E., eds., *Cognitive Systems: Human Cognitive Models in Systems Design*. Mahwah, NJ: Erlbaum.

Graesser, A. C.; Gordon, S. E.; and Brainerd, L. E. 1992. Quest: A model of question answering. *Computers and Mathematics with Applications* 23:733–745.

Kawamoto, S.; Shimodaira, H.; Nitta, T.; Nishimoto, T.; Nakamura, S.; Itou, K.; Morishima, S.; Yotsukura, T.; Kai, A.; Lee, A.; et al. 2002. Opensource software for developing anthropomorphic spoken dialog agent. In *Proc. of PRICAI-02, International Workshop on Lifelike Animated Agents*, 64–69.

Koedinger, K.; Anderson, J.; Hadley, W.; and Mark, M. 1997. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education* 8(1):30–43.

Landauer, T. K., and Dumais, S. T. 1997. A solution to plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review* 104:211–240.

Larsen, R. M. 1998. Lanczos bidiagonalization with partial reorthogonalization. Technical Report DAIMI PB-357, Department of Computer Science, Aarhus University.

Larsson, S., and Traum, D. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering* 6(3&4):323–340.

Larsson, S.; Ljunglöf, P.; Cooper, R.; Engdahl, E.; and Ericsson, S. 2000. GoDiS - an accomodating dialogue system. In *Proceedings of the ANLP/NAACL 2000 Workshop on Conversational Systems*, 7–10. Seattle: Association for Computational Linguistics.

Lehoucq, R.; Sorensen, D.; and Yang, C. 1998. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. Society for Industrial & Applied Mathematics.

Litman, D. J., and Allen, J. F. 1987. A plan recognition model for subdialogues in conversations. *Cognitive Science* 11(2):163–200.

Mancini, M.; Bresin, R.; and Pelachaud, C. 2007. A virtual head driven by music expressivity. *IEEE Transactions on Audio Speech and Language Processing* 15(6):1833.

Manning, C. D., and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

McTear, M. 1998. Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit. In *Fifth International Conference on Spoken Language Processing*. Sydney, Australia: ISCA.

McTear, M. 2002. Spoken dialogue technology: enabling the conversational user interface. *ACM Computing Surveys (CSUR)* 34(1):90–169.

Nuance. 1996. *Nuance Speech Recognition System, Version 5, Developers Manual.* Nuance Communications, Menlo Park, California.

Olde, B. A.; Franceschetti, D.; Karnavat, A.; and Graesser, A. C. 2002. The right stuff: Do you need to sanitize your corpus when using Latent Semantic Analysis? In *Proceedings of the 24th Annual Meeting of the Cognitive Science Society*, 708–713. Mahwah, NJ: Erlbaum.

Olney, A.; Person, N.; Louwerse, M.; and Graesser, A. 2002. AutoTutor: A conversational tutoring environment. In *Proceedings of the ACL-02 Demonstration Session*, 108–109. Philadelphia: Association for Computational Linguistics.

Olney, A.; Louwerse, M.; Mathews, E.; Marineau, J.; Hite-Mitchell, H.; and Graesser, A. 2003. Utterance classification in AutoTutor. In *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*, 1–8. Philadelphia: Association for Computational Linguistics.

Rohde, D. 2007. SVDLIBC.

Schrder, M., and Hunecke, A. 2007. Mary tts participation in the blizzard challenge 2007. In *Proc. Blizzard Challenge 2007*.

Searle, J. R. 1975. A taxonomy of illocutionary acts. In Gunderson, K., ed., *Language, Mind and Knowledge*. Minneapolis: University of Minnesota Press. 344–369.

Shoham, Y. 1994. *Artificial intelligence techniques in Prolog*. San Francisco: Morgan Kaufmann Publishers Inc.

Sterling, L., and Shapiro, E. 1994. *The art of Prolog: advanced programming techniques*. Cambridge, MA: MIT press.

VanLehn, K.; Jordan, P.; Rose, C.; Bhembe, D.; Bottner, M.; Gaydos, A.; Makatchev, M.; Pappuswamy, U.; Ringenberg, M.; Roque, A.; et al. 2002. The architecture of Why2-Atlas: A coach for qualitative physics essay writing. In *Proceedings of Intelligent Tutoring Systems Conference*, volume 2363, 158–167. Springer.

VanLehn, K.; Graesser, A. C.; Jackson, G. T.; Jordan, P.; Olney, A.; and Rose, C. 2007. When are tutorial dialogues more effective than reading? *Cognitive Science* 31:3–62.

VanLehn, K. 1988. *Student modeling*. Lawrence Erlbaum Associates. 55–78.

Wisher, R. A., and Fletcher, J. D. 2004. The case for advanced distributed learning. *Information & Security: An International Journal* 14:17–25.

Zinn, C.; Moore, J.; and Core, M. 2002. A 3-tier planning architecture for managing tutorial dialogue. In *ITS '02: Proceedings of the 6th International Conference on Intelligent Tutoring Systems*, 574–584. London: Springer.