# Recommendations For The Generalized Intelligent Framework for Tutoring Based On The Development Of The DeepTutor Tutoring Service

VASILE RUS, NOBAL NIRAULA, MIHAI LINTEAN, RAJENDRA BANJADE, DAN STEFANESCU, WILLIAM BAGGETT
The University of Memphis
Department of Computer Science/Institute for Intelligent Systems
Memphis, TN 38138
vrus@memphis.edu

**Abstract.** We present in this paper the design of DeepTutor, the first dialogue-based intelligent tutoring system based on Learning Progressions, and its implications for developing the Generalized Framework for Intelligent Tutoring. We also present the design of SEMILAR, a semantic similarity toolkit, that helps researchers investigate and author semantic similarity models for evaluating natural language student inputs in conversatioanl ITSs. DeepTutor has been developed as a web service while SEMILAR is a Java library. Based on our experience with developing DeepTutor and SEMILAR, we contrast three different models for developing a standardized architecture for intelligent tutoring systems: (1) a single-entry web service coupled with XML protocols for queries and data, (2) a bundle of web services, and (3) library-API. Based on the analysis of the three models, recommendations are provided.

**Keywords:** intelligent tutoring systems, computer based tutors, dialogue systems

## 1 Introduction

The General Framework for Intelligent Tutoring (GIFT; Sottilare et al, 2012) aims at creating a modular ITS/CBTS (intelligent tutoring systems/computer-based tutoring systems) framework and standards to foster "reuse, support authoring and optimization of CBTS strategies for learning, and lower the cost and skillset needed for users to adopt CBTS solutions for military training and education." GIFT has three primary functions: (1) to help with developing components for CBTS and whole tutoring systems; (2) to provide an instructional manager that integrates effective and exploratory tutoring principles and strategies for use in CBTS; and (3) to provide an experimental test bed to analyze the effectiveness and impact of CBTS components, tools, and methods. That is, GIFT is both a software environment and standardization effort. The availability of a GIFT software package suggests that for now the software environment has been given priority to standardization efforts. This paper intends to help make progress towards a GIFT standardization.

To that end, we present the design of DeepTutor (www.deeptutor.org; Rus et al., to appear), the first CBTS based on the emerging framework of Learning Progressions proposed by the science education research community (LPs; Corcoran, Mosher, & Rogat, 2009). LPs can be viewed as incrementally more sophisticated ways to think about an idea that emerge naturally while students move toward expert-level understanding of the idea (Duschl et al., 2007). That is, LPs capture the natural sequence of mental models and mental model shifts students go through while mastering a topic. It is this learner-centric view that differentiates LPs from previous attempts to reform science education. The LPs framework provides a promising way to organize and align content, instruction, and assessment strategies in order to give students the opportunity to develop deep and integrated understanding of science ideas.

DeepTutor is developed as a web service and a first prototype is fully accessible through a browser from any Internet-connected device, including regular desktop computers and mobile devices such as tablets. As of this writing, DeepTutor is designed as a bundle of two web services: (1) the tutoring service itself accessed by learners, and (2) the support service which includes everything else: authoring and content management, experiment management, user management, and instruction management. The latter service is viewed as a single service because there is a single-entry point to access all these functions. The tutoring service exports its functionality through an XML-based protocol. Third party developers can use their own development environments to design custom DeepTutor clients and integrate them with the DeepTutor tutoring service; all they need is to understand and generate an XML-like protocol, which is a query-language for accessing DeepTutor functionality.

We contrast the DeepTutor design with the design of another software environment, SEMILAR (www.semanticsimilarity.org; Rus et al., 2013). SEMILAR can be used to author semantic similarity methods for semantic processing tasks such as the task of assessing students' natural language inputs in dialogue-based CBTSs. SEMILAR, a SEMantic simILARity toolkit, has been designed as a Java library. Access to SEMILAR functionality is already available through a Java API (Application Programming Interface). Users can use the semantic similarity methods in SEMILAR as long as they link the SEMILAR library to their own Java programs. If a developer were to use SEMILAR from non-Java applications, a solution would be for the SEMILAR library to export its functionality through an XML-like protocol which is easily readable from any programming language. This latter integration solution is basically the export of functionality approach available in the DeepTutor tutoring service. SEMILAR has not been developed as a web service because it was initially developed for our own internal use. We have plans to make it available as a web service in the future. A GUI-based Java application has been developed and is currently tested to offer non-programmers easy access to the SEMILAR functionality.

The two designs, DeepTutor and SEMILAR, will help us discuss concretely three models for standardizing and implementing CBTS functionality to meet GIFT's goals: (1) a single-entry web service, e.g. the two DeepTutor services can be collated into one service (a one-stop-shop model); (2) a bundle of web services – the current DeepTutor design in which different functionality is accessed through different service points, and (3) a library of components accessed through an API. The three models share the common requirement of standardizing the communication between a client/user and provider of tutoring components/functions. While all three models have advantages and disadvantages, we favor the web services models for a Generalized Framework for Intelligent Tutoring as these models better suit the emerging world of mobile computing in which users access services in the cloud over the network as opposed to downloading full applications on their local, energy-sensitive mobile devices. Furthermore, the combination of a tutoring service and XML-based protocols for data and commands/queries fits very well with recent standards for representing knowledge proposed by the Semantic Web community, standards for authoring behavior of dialogue systems (see the FLORENCE dialogue manager framework; Fabbrizio & Lewis, 2004), or previous work in the intelligent tutoring community (see CircSim's mark-up language; Freedman et al., 1998).

The rest of the paper is organized as in the followings. The next section provides an overview of the DeepTutor web service. Then, we describe the design of the SEMILAR library. We conclude the paper with Discussion and Conclusions in which we make recommendations for GIFT based on the three models we discussed.

## 2 The Intelligent Tutoring Web Service DeepTutor

DeepTutor is a conversational ITS that is intended to increase the effectiveness of conversational ITSs beyond the interactivity plateau (VanLehn, 2011) by promoting deep learning of complex science topics through a combination of advanced domain modeling methods (based on LPs), deep language and discourse processing algorithms, and advanced tutorial strategies. DeepTutor currently targets the domain of conceptual Newtonian Physics but it is designed with scalability in mind (cross-topic, cross-domain).

DeepTutor is a problem solving coaching tutor. DeepTutor challenges students to solve problems, called tasks, and scaffolds their deep understanding of complex scientific topics through constructivist dialogue and other elements, e.g. multimedia items. DeepTutor uses the framework of Learning Progressions (LPs) to drive its scaffolding at macro- and micro-level (Rus et al, to appear). There is an interesting interplay among assessment, LPs, instructional tasks, and advanced tutoring strategies that is finely orchestrated by DeepTutor. The LPs are aligned with an initial, pre-tutoring assessment instrument (i.e., pretest) which students must complete before interacting with the system. Based on this first summative assessment, an initial map of students' knowledge level with respect to a topic LP is generated. The LPs encode both knowledge about the domain and knowledge about students' thinking in the form of models that students use to reason about the domain. The student models vary from naïve to weak to strong/mastery models. For each level of understanding in the LP a set of instructional tasks are triggered that are deemed to best help students make progress towards mastery, which coincides with the highest level of understanding modeled by the LP.

The task representation is completely separated from the executable code and therefore DeepTutor is compliant with the principles adopted by GIFT from Patil and Abraham (2010). Also, in accordance with GIFT principles (Sottilare et al., 2012), DeepTutor's pedagogical module interacts with the learner module (the Student) and adapts the scaffolding tasks and dialogue according to the learner's level of knowledge.

DeepTutor is an ongoing project. As of this writing, different modules are at different stages of maturity. For instance, our LP has been empirically validated based on data collected from 444 high-school student responses. Other components, e.g. the general knowledge module that can handle tasks related to general knowledge such as answering definitional questions ("What does *articulate* mean?"), is still in the works. The system as a whole will be fully validated in the next 6-12 months.

As already mentioned, DeepTutor has been designed as a web service accessible via HTML5-compatible clients, typically web browsers. The familiarity of users with web browsers and eliminating the need to install software packages (except the web browser) on each user's own computer environment makes it extremely convenient for users to access DeepTutor from any Internet-connected device and at the same time opens up unprecedented economies of scale for tutoring research. For instance, during Spring 2013 DeepTutor has been successfully used by more than 300 high-school students[7] from their Internet-device of choice (outside of traditional classroom instruction or experimental lab): home computer, tablet, mobile phones, or library computer.

All communication between the client and the DeepTutor server is handled through an XML-like protocol. The protocol specifies both commands and data that both client and server can interpret. The client communicates user actions and data to the server and the server replies with appropriate responses. Currently, the responses are in the form of display commands and values for various tutoring elements that are visible to the user on screen. That is, the client simply uses the information to update the corresponding interface elements, e.g. the client needs to update the dialogue history box with the most recent DeepTutor feedback response. The protocol contains sufficient information for learner software clients to display the elements of the standard DeepTutor interface. At the same time, the client uses the XML protocol to send the DeepTutor server important information about the user, e.g. user actions such as turning the talking head off, typed responses, time stamps, etc.

There are two major phases for learner clients to connect to the full DeepTutor system: the user authentication and initialization phase and the tutoring phase. In the authentication and initialization phase the user authenticates herself. A set of initialization parameters are sent to the DeepTutor system as well. Currently, the initialization parameters are set from the instructor view of the system, e.g. the researcher/experimenter or instructor/teacher can set a particular instructional strategy to be used by the system for a particular user or groups of learners. We can imagine in the future that these parameters are set dynamically based on the student model retrieved from a persistent database of learner information.

---

[7] This group of students is different from the 444 student group used for validating the LP.

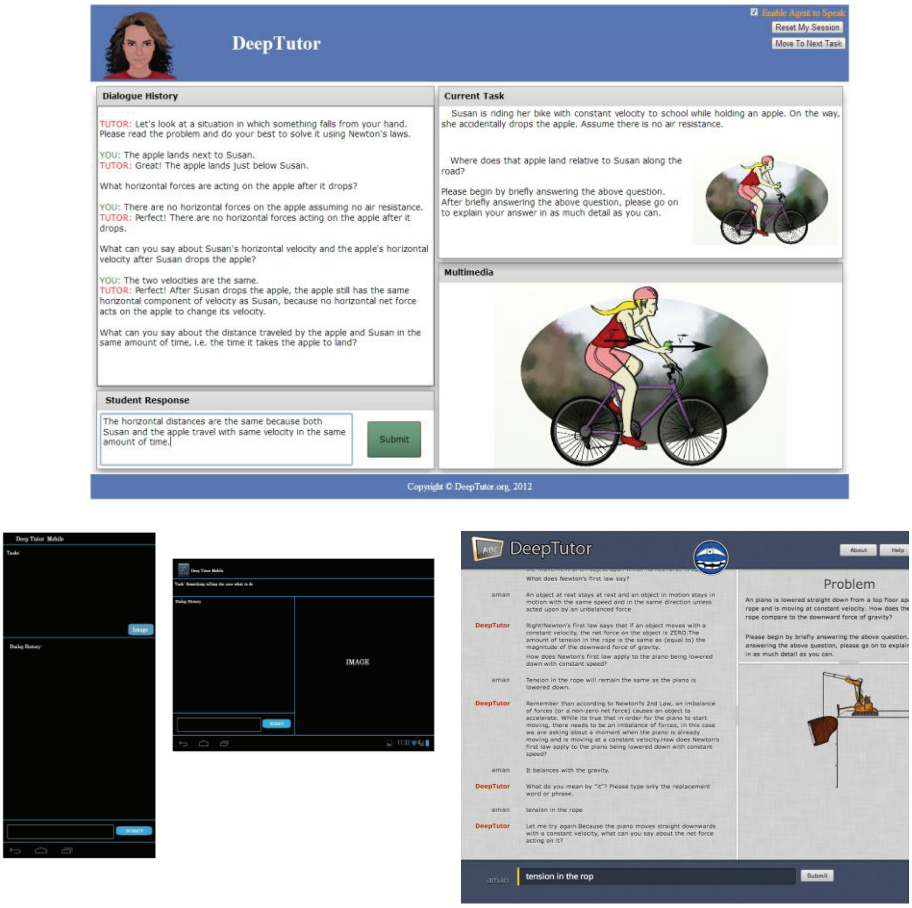


Figure 10. Three DeepTutor clients showing three different renderings of the learner-view of the DeepTutor Service: the currently official learner view in DeepTutor (top), an under-development Android app (bottom left) and a client developed for a Masters project (bottom right).

Client applications that access the full DeepTutor tutoring system (not individual components) can be designed quite easily. The main reason is the relatively simple but efficient current interface that allows the learner to focus on the interactive tutorial dialogue. Figure 1 bottom shows on the left-hand side an Android-based app client for DeepTutor designed by a small team of 5 Computer Science undergraduate students as a semester-long class project. The app has an interface design for a vertical versus horizontal positioning of the mobile device. The right-hand side of Figure 1 includes another DeepTutor client designed by a Masters student in Computer Science as his Masters project on Human-Computer Interaction.

It should be noted that more complex learner views are in the plans for DeepTutor. For instance, we plan to add several supplemental instructional aids and monitoring and informing elements such as how many tasks are left to cover in the current session or game-like features such as showing what percentage of a learner’s peers successfully finished the current task. The current interface of DeepTutor is as simple as it can be and it was intentionally kept this way. The goal was to reduce the number of on-screen distractors in order for the learner to focus on the tutorial dialogue. Adding more elements would make the interface richer which could distract the learners from the main tutorial interaction. It would be an interesting topic to investigate though.

We imagine that other users, e.g. developer of tutoring systems, may need to access specific functionality/components of DeepTutor according to the GIFT goals. As an example, we can imagine someone willing to access the output of the assessment module. As of this writing, the client-server protocol does not allow export of specific functionality. To allow export of functionality at a finer-grain level the current DeepTutor XML protocol must be extended such that the server provides developers/researcher clients output from specific modules, e.g. the assessment module. The exact format of the query and response must be clearly defined.

We believe that efforts to standardize access to GIFT-defined CBTS modules using XML protocols are best. The specification of these protocols needs to be done at different levels of abstractness such that the protocol is general enough to be applicable to all types of tutoring systems (at higher, more general levels of specification) and detailed enough for specific types of tutoring systems to be readily implementable by various groups. For instance, a general specification for querying the assessment module would include a general query element that indicates that an user input is needed together with a context variable which may contain other useful information for best assessing the student input (the context variable could be as simple as an user identifier and a session identifier or much more complex including a comprehensive list of factors that might impact assessment) and the format of the response from the assessment component of the tutoring service. This general specification can be further specified for *benchmark-based tutoring systems* (AutoTutor – Graesser et al., 2005, Guru – Olney et al. 2012; DeepTutor – Rus et al., to appear) as well as for *reasoning-based tutoring systems* (Why-Atlas; VanLehn et al., 2007). We use this broad categorization of tutoring systems to help us illustrate the need for further specifying general query formats. A *benchmark-tutoring system* is one that requires an expert-generated or benchmark response against which the student response is assessed (DeepTutor is such a system; Rus et al., to appear). For benchmark-tutoring systems the assessment query will need to pass (a pointer to) the benchmark response as one of the input parameters. *Reasoning-based systems* are able to infer the correct response automatically (Why-Atlas; VanLehn et al., 2007). For reasoning-based systems the benchmark response may not be needed but instead (a pointer to) a knowledge base.

In summary, a web service together with XML-based protocols may offer the best option for moving forward in GIFT. The advantage of using a web service solution with an XML-based protocol has the advantage of being easily extendable (new functionality can be added by simple adding new tags in the XML protocol). Another advantage is the decoupling the logical view from the actual implementation. The decoupling of functionality from actual implementation can be very useful. For example, the XML protocol can offer a GIFT-like view of the system with components so defined to meet GIFT standards while the actual, back-end implementation can be so designed to best fit particular types of ITSs. Sometimes refactoring and exporting functionality is conceptually challenging as for some tutoring systems there is a tight connection between components that GIFT suggest be separate. For instance, in LP-based ITSs such as DeepTutor, there is a tight relationship between learner models and the domain model because the domain is organized from a learner perspective (Rus et al., in press). Separating the learner model from the domain model is conceptually challenging and probably not recommended. The decoupling of functionality allows keeping the best implementation while offering differing views recommended by standards.

The combination of web service/XML protocol is also more advantageous when it comes to updates and extensions. There is no need to download and recompile a client application with the latest version of a component or the whole tutoring system.

We conclude this section by noting that the service model can further be refined into two types of service-based models: single service versus bundle of services. The current DeepTutor system is a **bundle of services**. In this model, the functionality of the various modules would be available as separate web services, e.g. the assessment module could be a separate web service. There are some interesting aspects of the **bundle of services model**. For instance, in DeepTutor some functionality is offered through a combination of the two DeepTutor services: debugging capabilities are offered through a combination of the tutoring and support services. That is, a developer polishing various components has to use both services.

All services can eventually be bundled together in a single, deep service (containing many subservices) in which case we have a **single-entry service model**. This model implements the concept of a one-stop-shop meaning users will use on access point for the components or the whole tutoring system.

## 3 The SEMILAR Library For Assessing Natural Language Student Inputs

Our SEMILAR (SEMantic similarity) toolkit, includes implementations and extensions of a number of algorithms proposed over the last decade to address the general problem of semantic similarity. SEMILAR includes algorithms based on Latent Semantic Analysis (LSA; Landauer et al., 2007), Latent Dirichlet Allocation (Blei, Ng, & Jordan, 2003), and lexico-syntactic optimization methods with negation handling (Rus & Lintean, 2012a; Rus et al., 2012b); Rus et al, in press). Due to space reasons, we do not present the set of methods available but rather discuss the design of SEMILAR as a Java library and its implications for using an akin design for GIFT.

The Java library design for SEMILAR has the advantage of being easily integrated as compiled code into Java applications which, at least in theory, should be platform independent. However, users have to download the whole package, install it, and then compile it with their tutoring systems. If these systems or components are written in a programming language different from Java, extra effort will be needed for integration. We call this **the library-API model** for a GIFT framework. Indeed, a GIFT framework based on the library-API model will require downloading and installing large software packages on various platforms by users of various technical backgrounds which may make the whole effort more challenging. For instance, the SEMILAR library and application is 300MB large (it includes large models for syntactic parsing among other things). SEMILAR can be regarded as a tutoring component for assessing students' natural language inputs. If ITS developers were to use SEMILAR as a library they have to download it and integrate it in their products. They have to install and update the API when updates become available. In fact, this is how SEMILAR is currently integrated in DeepTutor. Changes in implementation, e.g. bug fixes, would require a new download and reintegration of the systems that rely on the library. When SEMILAR will be available as a web service, all is needed is understanding the API, in the form of an XML-based communication protocol, and connect to the tutoring service. The need for a network connection are a potential risk for the service model in the form of network congestion which may make the service inaccessible or slow at times.

## 4 Discussion and Conclusions

We presented three models based on our experience with implementing a set of coherent functionalities related to intelligent tutoring systems and semantic processing. Each of the models has its own advantages and disadvantages. Ideally, all three models should be adopted by GIFT. However, if it were to choose we believe that the service-based models are the best solution for an emerging world of mobile devices in which accessing software services in the cloud is becoming the norm. The library-API and web service solutions are functionally equivalent with the former presenting more technical challenges for users with diverse backgrounds and computing environments and also being less suitable for a mobile computing world.

One apparent downside of the web service model is that potential developers cannot alter the code themselves in order to conduct research. This is just an apparent downside as a quick fix would be for each component to offer enough parameters, in the form of a switchboard, to allow potential users to alter behavior without the need to change the code. In fact, this solution should be preferred as users would not need to spend time to understand and alter the code, a tedious and error-prone activity.

Standardization efforts for XML-based protocols may start with previous efforts where available. For instance, the dialogue processing community has made attempts to standardize dialogue acts/speech acts, a major component in dialogue-based ITSs, for more than a decade. The resulting Dialogue Act Mark-Up in Several Layers (DAMSL) XML schema can be used as a start to standardize speech acts in dialogue ITSs.

In summary, we favor a **one-stop-shop service** model with **switchboard**-like facilities for implementing GIFT. Table 1 below illustrates the pros and cons of the three models discussed in this paper.

Table 6. Comparison of the three proposed model: single-entry service, bundle of services, and library.

| | **One-Stop-Shop/Single-Entry Service** | **Bundle of Services** | **Library** |
|---|---|---|---|
| **Programming Language Independent** | YES | YES | NO |
| **Install and update on local machine/ environment** | NO | NO | YES |
| **Fit for emerging mobile and cloud-computing fitness** | EXCELLENT | EXCELLENT | POOR |
| **Customization** | VERY GOOD | VERY GOOD | EXCELLENT |
| **Cost of Customization** | LOW | MEDIUM | HIGH (error prone and time to work with someone else' code) |
| **Extendible** | EXCELLENT | EXCELLENT | GOOD |

## 5 Acknowledgements

The authors would like to thank members of DeepTutor project (www.deeptutor.org). This research was supported by the Institute for Education Sciences (IES) under award R305A100875 by a grant from the Army Research Lab (ARL). Any opinions, findings and conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the IES or ARL.

## 6 References

1. Blei, D.M., Ng, A.Y., & Jordan, M.I. 2003. Latent dirichlet allocation, The Journal of Machine Learning Research 3, 993-1022.
2. Corcoran, T., Mosher, F.A., & Rogat, A. (2009). Learning progressions in science: An evidencebased approach to reform. Consortium for Policy Research in Education Report #RR-63. Philadelphia, PA: Consortium for Policy Research in Education.
3. Duschl, R.A., Schweingruber, H.A., & Shouse, A. (Eds.). (2007). Taking science to school: Learning and teaching science in grades K-8. Washington, DC: National Academy Press.
4. Graesser, A. C.; Olney, A.; Haynes, B. C.; and Chipman, P. 2005. Autotutor: A cognitive system that simulates a tutor that facilitates learning through mixed-initiative dialogue. In Cognitive Systems: Human Cognitive Models in Systems Design. Mahwah: Erlbaum.
5. Landauer, T.; McNamara, D. S.; Dennis, S.; and Kintsch, W. (2007). Handbook of Latent Semantic Analysis. Mahwah, NJ: Erlbaum.
6. Freedman, Reva, Yujian Zhou, Jung Hee Kim, Michael Glass, and Martha W. Evens.

7. SGML-Based Markup as a Step toward Improving Knowledge Acquisition for Text Generation AAAI 1998 Spring Symposium: Applying Machine Learning to Discourse Processing
8. VanLehn, K. (2011). The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems, Educational Psychologist, 46:4, 197-221.
9. Olney, A., D'Mello, A., Person, N., Cade, W., Hays, P., Williams, C., Lehman, B., & Graesser, A. (2012). Guru: A computer tutor that models expert human tutors. In S. Cerri, W. Clancey, G. Papadourakis & K. Panourgia (Eds.), Proceedings of the 11th International Conference on Intelligent Tutoring Systems (pp. 256-261). Springer-Verlag.
10. Patil, A. S., & Abraham, A. (2010). Intelligent and Interactive Web-Based Tutoring System in Engineering Education: Reviews, Perspectives and Development. In F. Xhafa, S. Caballe, A. Abraham, T. Daradoumis, & A. Juan Perez (Eds.), Computational Intelligence for Technology Enhanced Learning. Studies in Computational Intelligence (Vol 273, pp. 79-97). Berlin: Springer-Verlag.
11. Rus, V. & Lintean, M. (2012a). A Comparison of Greedy and Optimal Assessment of Natural Language Student Input Using Word-to-Word Similarity Metrics, Proceedings of the Seventh Workshop on Innovative Use of Natural Language Processing for Building Educational Applications, NAACL-HLT 2012, Montreal, Canada, June 7-8, 2012.
12. Rus, V., Lintean, M., Moldovan, C., Baggett, W., Niraula, N., Morgan, B. (2012b). The SIMILAR Corpus: A Resource to Foster the Qualitative Understanding of Semantic Similarity of Texts, In Semantic Relations II: Enhancing Resources and Applications, The 8th Language Resources and Evaluation Conference (LREC 2012), May 23-25, Instanbul, Turkey.
13. Rus, V.; Lintean, M.; Banjade, R.; Niraula, N.; Stefanescu, D. (2013). SEMILAR: The Semantic Similarity Toolkit, The 51st Annual Meeting of the Association for Computational Linguistics, System Demo Paper, August 4-9, 2013, Sofia, Bulgaria.
14. Rus, V., D'Mello, S., Hu, X., and Graesser, A.C. (to appear) .Recent Advances In Conversational Intelligent Tutoring Systems, AI Magazine.
15. VanLehn, K., Graesser, A. C., Jackson, G. T., Jordan, P., Olney, A., & Rose, C. P. (2007). When are tutorial dialogues more effective than reading? Cognitive Science, 31, 3-62.
16. VanLehn, K. (2011). The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems, Educational Psychologist, 46:4, 197-221.

## Authors

**Vasile Rus**: Dr. Vasile Rus is an Associate Professor of Computer Science with a joint appointment in the Institute for Intelligent Systems (IIS). Dr. Rus' areas of expertise are computational linguistics, artificial intelligence, software engineering, and computer science in general. His research areas of interest include question answering and asking, dialogue-based intelligent tutoring systems (ITSs), knowledge representation and reasoning, information retrieval, and machine learning. For the past 10 years, Dr. Rus has been heavily involved in various dialogue-based ITS projects including systems that tutor students on science topics (DeepTutor), reading strategies (iSTART), writing strategies (W-Pal), and metacognitive skills (MetaTutor). Currently, Dr. Rus leads the development of the first intelligent tutoring system based on learning progressions, DeepTutor (www.deeptutor.org). He has coedited three books, received several Best Paper Awards, and authored more than 90 publications in top, peer-reviewed international conferences and journals. He is currently Associate Editor of the International Journal on Artificial Intelligence Tools.

**Nobal Niraula**: Nobal B. Niraula received the B.E. in computer engineering from Pulchowk Campus, Tribhuvan University, Nepal, the M.E. in information and communication technology and the M.Sc. in communication networks and services from Asian Institute of Technology, Thailand and Telecom SudParis, France respectively. He was a research engineer at INRIA, Saclay, France where he worked in semantic web, database systems and P2P networks. Currently, he has been doing his PhD at The University of Memphis, USA. His research interests are primarily in Intelligent Tutoring Systems, Dialogue Systems, Information Extraction, Machine Learning, Data Mining, Semantic Web, P2P and Ad hoc networks. He has received the best paper and the best presentation awards. He also has intern experiences in leading research labs such as AT&T Labs Research.Mihai Lintean: Dr. Mihai Lintean is currently a research scientist at Carney Labs LLC and previous to that he was a Postdoctoral Research Fellow in the Computer Science Department at the University of Memphis, where he worked with Dr. Vasile Rus ondialogue based tutoring systems for teaching conceptual physics to high school students. Mihai's primary research interests are in Natural Language Processing (NLP), with focused applicability on educational technologies such as intelligent tutoring systems. Particularly he is interested in measuring semantic similarity between texts, representing knowledge through relational diagrams of concepts, automatic generation of questions, and using various machine learning techniques to solve other complex NLP problems. Mihai has published numerous papers and articles in reputable, peer-reviewed conferences and journals. He currently serves as co-chair of the Applied Natural Language Processing Special Track at the 25th International Conference of the Florida Artificial Intelligence Research Society (FLAIRS 2012).

**Rajendra Banjade**: Rajendra Banjade is a PhD student in Computer Science at The University of Memphis. He is a research assistant in the DeepTutor project (www.deeptutor.org) - a dialogue based tutoring system. Rajendra's research interests are in the area of Natural Language Processing, Information Retrieval, and Data Mining. Currently, he is focusing on measuring semantic similarity of short texts (word and sentence level) using knowledge based and corpus based methods and heading towards more human like inferencing techniques. His current research focus is on robust methods to evaluate student answers in conversational intelligent tutoring systems. He is keenly dedicated to enhancing the SEMILAR toolkit (www.semanticsimilarity.org) which is an off-the-shelf semantic similarity toolkit. Before joining The University of Memphis, he worked for five years as a Software Engineer (R&D) at Verisk Information Technologies CMMI III, Kathmandu (a subsidiary of Verisk Analytics inc.) where he got opportunities working on various healthcare data mining projects including DxCG Risk Solutions engine. He received an outstanding employee award at Verisk. Rajendra is a certified Scrum Master and Software Developer, and Certified HIPAA professional. He holds bachelor's degree in Computer Engineering.

**William B. Baggett**: William B. Baggett earned a PhD in Cognitive Psychology from The University of Memphis in 1998. He also holds an MS in Computer Science and an MBA in Management Information Systems. William is currently a Project Coordinator in the Computer Science Department at The University of Memphis, where he works on DeepTutor. DeepTutor is an intelligent tutoring system, implemented as a web application, which uses natural language dialogue to teach conceptual physics to high school and college students. Previously, William was a Professor and part-time Department Chair of Computer Information Systems at Strayer University and an adjunct Professor of Computer Science at The University of Memphis. In both positions, William taught graduate and undergraduate Computer Science courses, mentored, tutored, and advised students, and developed new curricula. He was also a Business Analyst at FedEx Express where he wrote software specifications for PowerPad, a mission-critical handheld computer carried by FedEx Express couriers. PowerPad software is designed to promote optimal courier behavior including the efficient pickup and delivery of FedEx shipments, package tracking, and conformance to policies and procedures for a wide variety of domestic and international services.

**Dan Ștefănescu**: Dr. Dan Ștefănescu is a Postdoctoral Research Fellow in the Department of Computer Science of the University of Memphis and the Institute for Intelligent Systems (IIS). As a member of DeepTutor team, his main research activity is dialogue-based Intelligent Tutoring Systems. Previously, Dr. Ștefănescu was a Senior Researcher at the Research Institute for Artificial Intelligence (RACAI) in Bucharest, Romania. He graduated from the Computer Science Faculty of ”A.I. Cuza” University of Iași in 2002 and obtained his MSc in Computational Linguistics from the same university in 2004. In 2010 he was awarded the PhD title (Magna Cum Laude) at the Romanian Academy for a thesis on Knowledge Extraction from Multilingual Corpora. He authored more than 50 papers in peer-reviewed journals and conference proceedings and successfully participated in various software competitions like the Question-Answering competitions organized by Conference and Labs of the Evaluation Forum (CLEF), Microsoft Imagine Cup and Microsoft Speller Challenge. His research work covers various Natural Language Processing topics like: Question Answering, Information Extraction, Word Sense Disambiguation, Connotation/Sentiment Analysis, Collocations/Terminology Identification, Machine Translation, or Query Alteration for Search Engines.