# Adaptive Tutorial Dialogue Systems Using Deep NLP Techniques

**Myroslava O. Dzikovska, Charles B. Callaway, Elaine Farrow,**
**Manuel Marques-Pita, Colin Matheson and Johanna D. Moore**
ICCS-HCRC, School of Informatics
University of Edinburgh
Edinburgh, EH8 9LW, United Kingdom
(mdzikovs,ccallawa,efarrow,mmpita,colin,jmoore)@inf.ed.ac.uk *

## Abstract

We present tutorial dialogue systems in two different domains that demonstrate the use of dialogue management and deep natural language processing techniques. Generation techniques are used to produce natural sounding feedback adapted to student performance and the dialogue history, and context is used to interpret tentative answers phrased as questions.

## 1 Introduction

Intelligent tutoring systems help students improve learning compared to reading textbooks, though not quite as much as human tutors (Anderson et al., 1995). The specific properties of human-human dialogue that help students learn are still being studied, but the proposed features important for learning include allowing students to explain their actions (Chi et al., 1994), adapting tutorial feedback to the learner's level, and engagement/affect. Some tutorial dialogue systems use NLP techniques to analyze student responses to "why" questions. (Aleven et al., 2001; Jordan et al., 2006). However, for remediation they revert to scripted dialogue, relying on short-answer questions and canned feedback. The resulting dialogue may be redundant in ways detrimental to student understanding (Jordan et al., 2005) and allows for only limited adaptivity (Jordan, 2004).

We demonstrate two tutorial dialogue systems that use techniques from task-oriented dialogue systems to improve the interaction. The systems are built using the Information State Update approach (Larsson and Traum, 2000) for dialogue management and generic components for deep natural language understanding and generation. Tutorial feedback is generated adaptively based on the student model, and the interpretation is used to process explanations and to differentiate between student queries and hedged answers phrased as questions. The systems are intended for testing hypotheses about tutoring. By comparing student learning gains between versions of the same system using different tutoring strategies, as well as between the systems and human tutors, we can test hypotheses about the role of factors such as free natural language input, adaptivity and student affect.

## 2 The BEEDIFF Tutor

The BEEDIFF tutor helps students solve symbolic differentiation problems, a procedural task. Solution graphs generated by a domain reasoner are used to interpret student actions and to generate feedback.[1] Student input is relatively limited and consists mostly of mathematical formulas, but the system generates adaptive feedback based on the notion of student performance and on the dialogue history.

For example, if an average student asks for a hint on differentiating $sin(x^2)$, the first level of feedback may be "Think about which rule to apply", which

This work was supported under the 6th Framework Programme of the European Commission, Ref. IST-507826, and by a grant from The Office of Naval Research N000149910165.

[1] Solution graphs are generated automatically for arbitrary expressions, with no limit on the complexity of expressions except for possible efficiency considerations.



*NAACL HLT Demonstration Program*, pages 5–6,
Rochester, New York, USA, April 2007. ©2007 Association for Computational Linguistics

can then be specialized to "Use the chain rule" and then to giving away the complete answer. For students with low performance, more specific feedback can be given from the start. The same strategy (based on an initial corpus analysis) is used in producing feedback after incorrect answers, and we intend to use the system to evaluate its effectiveness.

The feedback is generated automatically from a single diagnosis and generation techniques are used to produce appropriate discourse cues. For example, when a student repeats the same mistake, the feedback may be "You've differentiated the inner layer correctly, but you're still missing the minus sign". The two clauses are joined by a contrast relationship, and the second indicates that an error was repeated by using the adverbial "still".

## 3 The BEETLE Tutor

The BEETLE tutor is designed to teach students basic electricity and electronics concepts. Unlike the BEEDIFF tutor, the BEETLE tutor is built around a pre-planned course where the students alternate reading with exercises involving answering "why" questions and interacting with a circuit simulator.

Since this is a conceptual domain, for most exercises there is no structured sequence of steps that the students should follow, but students need to name a correct set of objects and relationships in their response. We model the process of building an answer to an exercise as co-constructing a solution, where the student and tutor may contribute parts of the answer. For example, consider the question "For each circuit, which components are in a closed path". The solution can be built up gradually, with the student naming different components, and the system providing feedback until the list is complete. This generic process of gradually building up a solution is also applied to giving explanations. For example, in answer to the question "What is required for a light bulb to light" the student may say "The bulb must be in a closed path", which is correct but not complete. The system may then say "Correct, but is that everything?" to prompt the student towards mentioning the battery as well. The diagnosis of the student answer is represented as a set of correctly given objects or relationships, incorrect parts, and objects and relationships that have yet to be mentioned, and the system uses the same dialogue strategy of eliciting the missing parts for all types of questions.

Students often phrase their answers tentatively, for example "Is the bulb in a closed path?". In the context of a tutor question the interpretation process treats yes-no questions from the student as potentially hedged answers. The dialogue manager attempts to match the objects and relationships in the student input with those in the question. If a close match can be found, then the student utterance is interpreted as giving an answer rather than a true query. In contrast, if the student said "Is the bulb connected to the battery?", this would be interpreted as a proper query and the system would attempt to answer it.

**Conclusion** We demonstrate two tutorial dialogue systems in different domains built by adapting dialogue techniques from task-oriented dialogue systems. Improved interpretation and generation help support adaptivity and a wider range of inputs than possible in scripted dialogue.

## References


V. Aleven, O. Popescu, and K. R. Koedinger. 2001. Towards tutorial dialog to support self-explanation: Adding natural language understanding to a cognitive tutor. In *Proc. AI-ED 2001*.

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

M. T. H. Chi, N. de Leeuw, M.-H. Chiu, and C. LaVancher. 1994. Eliciting self-explanations improves understanding. *Cognitive Science*, 18(3):439–477.

P. Jordan, P. Albacete, and K. VanLehn. 2005. Taking control of redundancy in scripted tutorial dialogue. In *Proc. of AIED2005*, pages 314–321.

P. Jordan, M. Makatchev, U. Pappuswamy, K. VanLehn, and P. Albacete. 2006. A natural language tutorial dialogue system for physics. In *Proc. of FLAIRS-06*.

P. W. Jordan. 2004. Using student explanations as models for adapting tutorial dialogue. In V. Barr and Z. Markov, editors, *FLAIRS Conference*. AAAI Press.

S. Larsson and D. Traum. 2000. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, 6(3-4):323–340.