# ABSTRACT

BOYER, KRISTY ELIZABETH. Structural and Dialogue Act Modeling in Task-Oriented Tutorial Dialogue. (Under the direction of James C. Lester and Mladen A. Vouk.)

Creating intelligent systems that bring the benefits of one-on-one human tutoring to a broad population of learners is a grand challenge for the field of computing. Tutorial dialogue systems, which engage learners in rich natural language dialogue in support of a learning task, hold great promise for addressing this challenge. A particularly important research direction involves utilizing data-driven approaches for defining the behavior of tutorial dialogue systems based on corpora of effective human tutoring. These data-driven approaches may facilitate rapid dialogue system development, give rise to flexible dialogue management policies, and ultimately result in a more effective learning experience for students. The goal of the research reported in this dissertation is to develop computational models of effective human tutorial dialogue. The work includes two phases of data-driven investigation. The first phase involves collecting, annotating, and exploring corpora, while the second phase involves developing and evaluating computational models of hidden dialogue state, student dialogue act classification, and tutor move prediction.

Three tutorial dialogue corpora in the domain of introductory computer programming were collected through human tutoring studies. Exploring these corpora revealed some important aspects of the structure of dialogue in this complex task-oriented domain. First, human tutors appear to adapt to learner characteristics such as incoming knowledge level, self-efficacy, and gender. Second, tutors undertake a variety of cognitive and motivational remediation; sometimes these cognitive and motivational concerns appear to be at odds with each other, but it may be possible to reconcile the two goals by selecting appropriate feedback. Finally, compared to a highly proactive tutoring approach in which the tutor maintains control of the dialogue, offering more autonomy may better support students' motivation.

To construct computational models of the tutorial dialogue, a hidden Markov modeling framework was selected because hidden Markov models (HMMs) explicitly represent a stochastic layer of hidden structure. One of the hypotheses of this dissertation states that this hidden structure corresponds to tutoring modes from the literature. The utility of HMM-based modeling was examined through qualitative analysis and quantitative comparison of classification and prediction accuracy with other types of models. Qualitative examination of learned HMMs indicated that their structure bears a resemblance to tutoring modes from the literature. Analysis also revealed that the frequency of occurrence of a subset of the automatically extracted tutoring modes significantly correlates with student learning, suggesting that HMMs can probably discover meaningful hidden dialogue structure.

Based on these encouraging results, HMMs were also utilized to produce feature vectors that were used within a larger set of attributes for vector-based maximum likelihood classification of student dialogue acts. However, in the presence of automatically extracted lexical (word-based) features, the HMM's features did not improve the classification accuracy of the model. On the other hand, when the HMM approach was extended to predict tutorial moves within the corpus, the hidden dialogue state significantly increased prediction accuracy. Furthermore, explicitly modeling task structure within a hierarchical HMM provided a significant improvement in performance accuracy compared to a flat HMM.

While the Intelligent Tutoring Systems field is only just beginning to understand the impact of natural language dialogue, affect, collaboration, and other phenomena on students' learning, the Natural Language Dialogue research community is also beginning to embrace complex task-oriented domains as a focus for system development. This work takes an important step toward creating fully data-driven tutorial dialogue management models that may address the high development cost and barriers to effectiveness that are associated with the current generation of dialogue systems.

Structural and Dialogue Act Modeling in
Task-Oriented Tutorial Dialogue

by
Kristy Elizabeth Boyer

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2010

APPROVED BY:

| | |
|---|---|
| Dr. Laurie Williams | Dr. Tiffany Barnes |
| Dr. Mladen Vouk<br>Co-Chair of Advisory Committee | Dr. James Lester<br>Chair of Advisory Committee |

# DEDICATION

To Mildred Elizabeth Newberry (1927–2008), my grandmother and best friend. Grammies, I wish you were still here so you could see that you were right all along.

I *did* do it.

# BIOGRAPHY

Kristy Elizabeth Bland was born in Valdosta, Georgia on February 2, 1979. She graduated as valedictorian of the class of 1997 from Lowndes High School. She obtained a Bachelor of Science degree in Mathematics and Computer Science with a minor in Spanish from Valdosta State University in 1999. She completed a Master of Science degree in Applied Statistics from the Georgia Institute of Technology in 2000. She was employed as a statistician and software developer in the Operations Research department of Delta Air Lines during 2000 and 2001. She then joined the faculty at Valdosta State University as an Instructor of Mathematics and Computer Science until 2005, when she began Ph.D. studies at North Carolina State University. Kristy married Tom Boyer in 2004, and their son Rowan was born in 2007.

# ACKNOWLEDGMENTS

My husband Tom and my son Rowan have given me love and encouragement during very long working hours and when I had to be away from home. Tom has taken on a huge share of household duties, and his hard work, patience, and sense of humor have made my years as a Ph.D. student truly wonderful. My mother, Beth Bland, has made loving sacrifices throughout my life that have made this achievement possible. Her ongoing generosity of time and energy has also made the first years of my son's life the best they could possibly be. My father, Russell Bland, always encouraged me—through word and deed—to take on a challenge. My sister, Julie Doner, provided unconditional love and a listening ear. My dear friend Tim Brown made frequent, extraordinary sacrifices on behalf of my family and me personally. My extended family, Bob Doner, David and Erica Boyer, Rob Boyer, Mike and Sue Boyer, Jill Boyer, Doris Boyer, Jack and Betty Rickard, and Leo and Darline Ramer provided love and support. My close friends and church family at Triangle Vineyard Christian Fellowship made sure I never felt alone. Mike Stockstill enthusiastically and precisely answered many statistics questions.

James Lester, my advisor, has set an impeccable example of scholarly professionalism. Mladen Vouk, my advisor, has provided amazing breadth of insight and staunch support. Laurie Williams and Tiffany Barnes, members of my committee, have set a high scholarly bar for which I am very thankful, and they have mentored me both personally and professionally. Kera Bell-Watkins and Teresa Dahlberg have provided friendship and mentoring. Brian Dorn, Ed Gehringer, Mark Guzdial, Orit Hazzan, Chad Lane, Diane Litman, Andy Meneely, Carol Miller, Tanja Mitrovic, Rodney Nielsen, Amanda Stent, and Allison Tew helped shape this work with their knowledgeable advice. Thanks to all my friends and family not mentioned by name for countless hours of insightful conversations, practical help, and encouragement.

My colleagues in the Intellimedia Group have provided vital support and collaboration. Eunyoung Ha implemented the base HMM training software and extracted linguistic features. She has also engaged in challenging and stimulating technical discussions with me that have shaped the direction of my work. Rob Phillips has also kindly applied his vast technical knowledge to our collaboration, assisting with model interpretation, designing dialogue act annotation schemes, and formulating the task annotation schemes. He has also spent many long hours annotating corpora. Michael Wallis provided implementation support for software and many hours of labor during tutoring studies. Colleagues including Rachael Dwight, Julius Goth, Joe Grafsgaard, Seung Lee, Scott McQuiggan, Bradford Mott, Jennifer Robison, Jonathan Rowe, and Lucy Shores assisted with many aspects of this work.

I have had the pleasure of collaborating with several undergraduate researchers who have contributed in essential ways to this dissertation research. August Dwight and Taylor Fondren applied their considerable talents to developing the collaborative tutoring environment software. They showed great patience with my inexperience at overseeing such projects, and they achieved productivity well beyond what is normally expected of undergraduate researchers. Amy Ingram collaborated on refining and applying the dialogue act annotation scheme to the main tutoring corpus. Even after her summer research position had ended, she spent many tireless hours collaboratively refining the task annotation scheme and then applying it to more than sixty hours of tutorial dialogue. Her excellent work ethic and natural intuition regarding tutorial dialogue phenomena were very valuable to this project.

The IT staff in the Department of Computer Science have provided frequent support. Marhn Fullmer stayed late on many occasions to help set up computers for the tutoring studies. His dedication to furthering the research in our department, and to serving the faculty and students as an IT professional, is well above the call of duty. Carlos Benavente, Jason Corley, Trey Murdoch, and Sarah Williams always provided needed help despite their heavy workloads. Thanks to the Computer Science departmental staff including Barbara Adams, Ginny Adams, Carol Allen, Ron Hartis, Linda Honeycutt, Ann Hunt, Margery Page, Susan Peaslee, Missy Seate, and Monica Watkins.

Members of the Realsearch group have provided software development project support. Dright Ho spent hours assisting with the extension of his pair programming Eclipse plug-in. Lucas Layman offered his Eclipse development expertise and very welcome sense of humor. Mark Sherriff provided technical support for the RIPPLE development repository.

This material is based in part upon work supported by the National Science Foundation under a Graduate Research Fellowship and grants CNS-0540523, CNS-0739216, ITWF-0204222, REC-0632450, IIS-0812291. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation. Support has also been provided by North Carolina State University through the Department of Computer Science and the Office of the Dean of the College of Engineering.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1

# Introduction

One-on-one human tutoring is a highly effective mode of instruction that generally results in significantly higher student learning than group classroom instruction (Bloom, 1984; Chi *et al.*, 2001; Cohen *et al.*, 1982; VanLehn *et al.*, 2007). Bringing this highly effective instruction to learners is a primary focus of research on intelligent tutoring systems (ITSs). This research has adapted and created methods and technologies applicable to education, often in the form of intelligent tutoring systems (Wenger, 1987). One of the grand challenges for the field of computing today has been identified as providing a *teacher for every learner*,[1] and today's ITSs have made great strides toward that goal. However, the field has not yet seen ITSs that meet or exceed the learning gains achieved with expert human tutors (VanLehn, 2008). One hypothesis is that the greater effectiveness of human tutoring lies with the natural language dialogue exchanged between tutor and student (Graesser *et al.*, 1995). This hypothesis states that intelligent systems will not achieve human-like effectiveness as tutors until the systems can engage in rich natural language dialogue with students. This hypothesis has spurred the emergence of natural language intelligent tutoring systems, known as *tutorial dialogue systems*. The work reported in this dissertation makes contributions to tutorial dialogue systems research by exploring corpora of tutorial dialogue and modeling effective human-human tutoring strategies.

[1] Computing Research Association's *Grand Challenges in Information Systems*, http://www.cra.org/uploads/documents/resources/rissues/gc.systems_.pdf

In addition to drawing heavily on ITS research, tutorial dialogue systems draw on natural language dialogue systems research, which constitutes an active line of investigation within the field of computational linguistics. For all dialogue systems, including tutorial dialogue systems, two central challenges are interpreting user (student) utterances and selecting system (tutor) dialogue moves.

Interpreting user utterances involves a variety of speech and language understanding steps. One important aspect of the process is to identify the *dialogue act*, or communicative purpose, of each utterance (Austin, 1962). Dialogue acts provide a valuable intermediate representation that can be used for dialogue management because they summarize the action represented by a dialogue move, *e.g.*, asking a question or giving a command.

The need for automatic dialogue act interpretation has led to machine learning approaches that take into account a variety of features for data-driven dialogue act tagging (Bangalore *et al.*, 2008; Hardy *et al.*, 2006; Sridhar *et al.*, 2009; Stolcke *et al.*, 2000). This work on dialogue act interpretation has generally focused on conversational speech or on simple task-oriented dialogue. Complex task-oriented dialogue, such as tutorial dialogue for introductory computer programming, has not been extensively studied within dialogue systems research. A contribution of this dissertation is a machine-learned model of user dialogue act classification in a complex task-oriented domain.

A complementary task to user utterance interpretation is selecting a system dialogue move. Machine learning techniques for this task are also receiving increasing attention (Bangalore *et al.*, 2008; Chotimongkol, 2008; Levin *et al.*, 2000; Singh *et al.*, 2002; Stolcke *et al.*, 2000; Toney *et al.*, 2006; Young, 2000; Young *et al.*, 2009). These approaches leverage the growing set of available human-human dialogue corpora to directly author new computer-based dialogue system behavior. In contrast, historically the behavior of tutorial dialogue systems has been informed by observational studies of human tutoring followed by manual identification of phenomena of interest and desirable tutoring behaviors (Cade *et al.*, 2008; Graesser *et al.*, 1995; Lepper *et al.*, 1993). Recently, the use of machine learning techniques has begun to make its way into tutorial dialogue research (Ai *et al.*, 2007; Chi, M. *et al.*, 2009; Tetreault & Litman, 2008; Chi, M. *et al.*, 2010), but adapting these techniques to a complex task-oriented domain such as introductory computer programming is a central challenge. A contribution of the current research is a data-driven approach to extracting a tutorial dialogue management model from a corpus of effective human-human tutoring with hidden Markov models. These models infer the hidden dialogue state and leverage knowledge of the hierarchical task/subtask structure.

This work utilizes tutorial dialogue corpora from introductory computer programming. Improving the student experience in an introductory computing course is important. Research suggests some ways to achieve that goal include encouraging students to pair program (Nagappan *et al.*, 2003), providing a variety of course formats such as one in digital media computation (Guzdial & Tew, 2006), and maintaining smaller class sizes (Boyer, Dwight *et al.*, 2007). Another contribution of the current work is a model of tutoring effectiveness that reveals insights into the cognitive and affective mechanisms by which students learn computing.

This project contributes to the author's longer-term goal of creating a data-driven tutorial dialogue system for introductory computer science that is as effective as an expert human tutor. As the remainder of this chapter describes, this endeavor poses significant challenges, and the current work addresses some of those challenges by creating computational models of effective human tutoring.

## 1.1 Challenges

A number of challenges are posed by creating a tutorial dialogue system that is as effective as the most effective human tutors. Decades of intelligent tutoring systems research have held such effectiveness as their ultimate goal, but none have achieved it (VanLehn, 2008).

Developing tutoring systems is costly, often requiring hundreds of development hours per hour of tutoring instruction, and tutorial dialogue management systems have limited generalizability across domains (Aleven *et al.*, 2009). The central tasks of user utterance interpretation and system response selection, along with how to deal with issues such as ill-formed user input, identifying the user's goals within the task, and selecting a system move are all open problems (Bangalore *et al.*, 2008; Tetreault & Litman, 2008). These problems have been less explored in the context of complex task-oriented domains than in other dialogue areas such as conversational speech (*e.g.*, people talking socially on the telephone) or simpler task-oriented dialogue (*e.g.*, booking an airline ticket through a telephone reservation system). A task focus of solving an introductory computer programming problem provides some structure that is absent from conversational speech, but also introduces a more complex and open-ended task than most of the other task-oriented domains that have been studied within the dialogue systems community. To address these challenges, the research questions and associated hypotheses in this dissertation focus on how to learn computational models of complex task-oriented tutorial dialogue.

## 1.2 Research Questions and Hypotheses

This project contributes to the goal of creating a data-driven tutorial dialogue system. Toward that end, the primary goal of this work is *to discover, investigate and construct computational models of human tutoring in the domain of introductory computer programming*. This project can be viewed in two phases. The first phase consists of data collection, annotation, and exploration to understand the structure of tutorial dialogue in introductory computer programming, a domain for which no sizeable corpora were previously available. This phase corresponds to Research Question 1 below and its associated hypotheses. The second phase, which involves discovering and evaluating computational models on the collected corpora, corresponds to Research Question 2 below and its associated hypotheses.

***Research Question 1 (Exploratory).*** *How do human tutors help students learn introductory computer programming?*

**Hypothesis 1.1.** Because human tutors adapt their behavior based on student characteristics including skill level, self-efficacy, and gender, *the distribution of dialogue acts within human-to-human tutoring sessions are dependent on these student characteristics* (Section 5.1).

**Hypothesis 1.2.** Because some tutoring approaches are more effective than others, given a tutoring context, *the frequency of some tutor moves is positively correlated with student learning and motivational outcomes while other moves are negatively correlated with these outcomes* (Section 5.2).

**Hypothesis 1.3.** Because autonomy is an important aspect of the learning process that may impact cognitive and motivational outcomes differently, *the level of autonomy given to students during tutoring is correlated with learning and motivational outcomes* (Section 5.3).

***Research Question 2.*** *How can a tutorial dialogue management model be machine learned directly from a corpus of human tutoring?*

**Hypothesis 2.1.** Hidden Markov models (HMMs) are able to discover tutoring modes, or *hidden dialogue states*, that *i) qualitatively correspond to tutoring modes from the literature,* and *ii) whose frequencies of occurrence correlate with student learning* (Sections 6.1-6.3).

**Hypothesis 2.2.** The structural components of hidden dialogue state and task/subtask features are predictive of student dialogue acts (Chapter 7).

Specifically, let $\boldsymbol{B_1}$, a baseline model, be a first-order Markov (bigram) model over dialogue act sequences. Let model $\boldsymbol{M_1}$ be a classifier that uses lexical features (words, parts of speech, syntax) and structural features (speaker history, dialogue act history) associated with student utterances to classify the dialogue act of those utterances. Let $\boldsymbol{M_1'}$ be a classifier that extends $\boldsymbol{M_1}$ by additionally utilizing hidden dialogue state features as learned by an HMM, and manually annotated task/subtask structure. Then, in ten-fold stratified cross-validation, in which 90% of the data are used for training and 10% are used for testing, the following result will emerge:[2]

$$\boldsymbol{accuracy(B_1) < accuracy(M_1) < accuracy(M_1')}$$

**Hypothesis 2.3.** The structural components of hidden dialogue state and task/subtask features are predictive of tutor dialogue acts (Chapter 8). Specifically, let $\boldsymbol{B_2}$, a baseline model, be a first-order Markov (bigram) model over dialogue act sequences. Let model $\boldsymbol{M_2}$ be a hidden Markov model that predicts tutor dialogue acts based on sequences of dialogue acts and task events. Let $\boldsymbol{M_2'}$ be a hierarchical hidden Markov model whose structure extends that of $\boldsymbol{M_2}$ by explicitly capturing the hierarchical nesting of tasks and subtasks. Then, cross-validation on the corpus will reveal the following result:

$$\boldsymbol{accuracy(B_2) < accuracy(M_2) < accuracy(M_2')}$$

## 1.3 Approach

This project utilizes a corpus-based research methodology (Figure 1). First, a corpus of human task-oriented tutorial dialogue was collected. The domain is introductory computer

[2] Accuracy is calculated as number of correctly predicted instances divided by total number of predicted instances.

programming in Java. Next, the dialogue corpus was manually annotated with dialogue acts designed to capture the cognitive, motivational, and affective purposes of each utterance. The annotation also included a separate annotation for task/subtask structure and problem-solving correctness. In both dialogue and task annotation, inter-rater agreement studies were conducted to ensure sufficient reliability of the tagging scheme. With the corpora annotated, exploratory analyses included Pearson's correlation analysis, logistic regression, and Chi-square tests for independence of factors (De Veaux *et al.*, 2005). These analyses address Hypotheses 1.1, 1.2, and 1.3 above. HMMs (Rabiner, 1989) address Hypothesis 2.1, with subsequent correlation analysis between the components of the HMM and the student learning outcomes. HMMs were also utilized, along with a vector-based logistic regression classifier, for the student utterance classification task of Hypothesis 2.2. Finally, HMMs and hierarchical HMMs (Fine *et al.*, 1998) were learned to address the tutor move prediction task of Hypothesis 2.3. The models were evaluated based on their performance for the tasks of interest. For exploratory models this measure involves level of statistical significance. For classification and prediction models the accuracy is used, where the manually annotated dialogue act is treated as the true answer. Accuracy is calculated via stratified *n*-fold cross-validation.

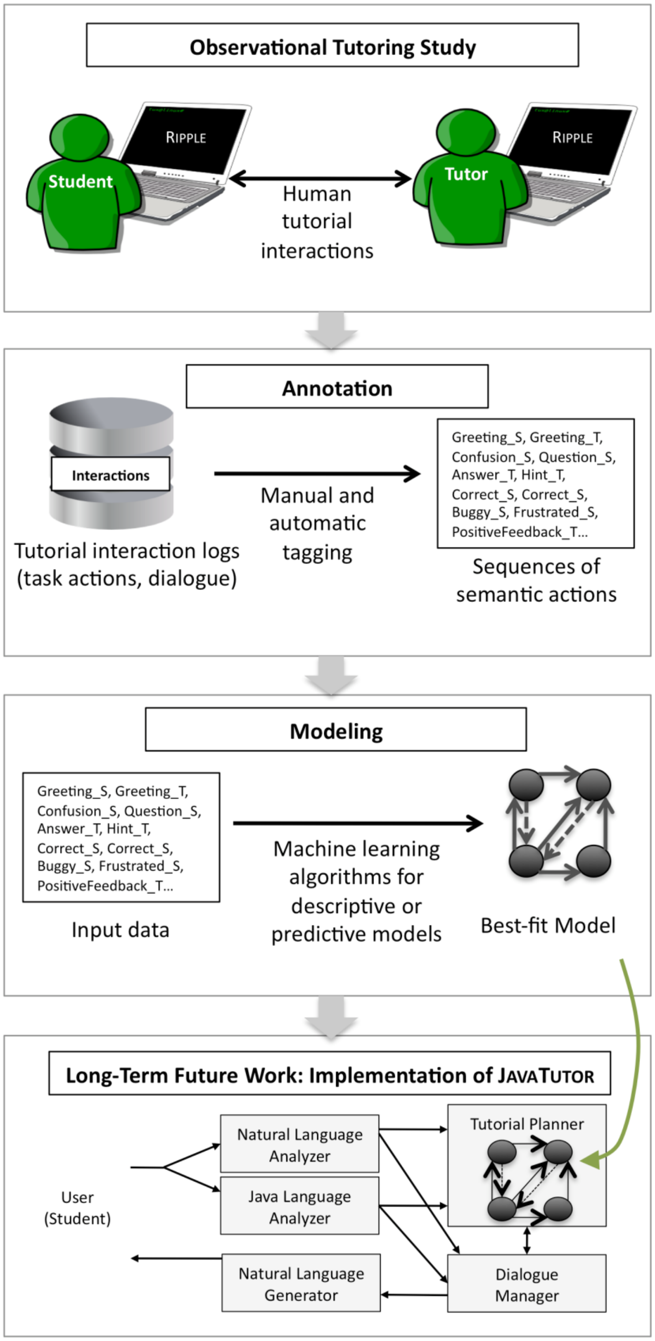

Observational Tutoring Study
Student
Ripple
Human tutorial interactions
Tutor
Ripple
Annotation
Interactions
Tutorial interaction logs (task actions, dialogue)
Manual and automatic tagging
Greeting_S, Greeting_T, Confusion_S, Question_S, Answer_T, Hint_T, Correct_S, Correct_S, Buggy_S, Frustrated_S, PositiveFeedback_T...
Sequences of semantic actions
Modeling
Greeting_S, Greeting_T, Confusion_S, Question_S, Answer_T, Hint_T, Correct_S, Correct_S, Buggy_S, Frustrated_S, PositiveFeedback_T...
Input data
Machine learning algorithms for descriptive or predictive models
Best-fit Model
Long-Term Future Work: Implementation of JavaTutor
User (Student)
Natural Language Analyzer
Java Language Analyzer
Natural Language Generator
Tutorial Planner
Dialogue Manager

*Figure 1. Corpus-based approach*

## 1.4 Terminology

The pre- and post-measures taken during tutoring studies utilize a variety of cognitive and motivational outcomes. *Cognitive* measures deal with students' knowledge or understanding of concepts or applied skills. In this context, cognitive metrics include *pre-test score*, which measures to what extent students are familiar with the target introductory computer science course material prior to the tutoring session, and *post-test score*, which measures the corresponding familiarity after the tutoring session. *Learning gain* is calculated as post-test score minus pre-test score, and is used to measure how much a student learns during the tutoring session. The full learning gain instruments from the tutoring studies are included in Appendices A and B.

The motivational measure utilized in this work deals with *self-efficacy*, which is defined as one's own belief in his or her capability to produce given levels of attainment on a particular task (Bandura, 1997). In the context of the current research, self-efficacy questions asked students to rate how certain they were, on a scale of 0-100, that they could complete various programming tasks. This method of measuring domain-specific self-efficacy is adapted directly from Bandura's (2006) domain-specific self-efficacy scale. This dissertation uses the term "self-efficacy" rather than "confidence," in keeping with the important distinction between these terms within the Educational Psychology literature. As explained by Bandura (1997), "Confidence is a nonspecific term that refers to strength of belief but does not necessarily specify what the certainty is about. I can be supremely confident that I will fail at an endeavor. Perceived self-efficacy refers to belief in one's agentive capabilities, that one can produce given levels of attainment."

The dialogue and problem-solving traces collected during each tutoring study constitute a *corpus*, a collection of written material. The discrete data points within this corpus may be either dialogue moves or task actions. *Dialogue moves,* also called *utterances*, are turns taken within the conversation. In the current work, these utterances were sent in textual instant message format between tutors and students. While “dialogue move” and “utterance” refer to the actual content of the message sent during dialogue, *dialogue act* refers to the communicative purpose of the utterance. For example, the utterance, “How do I declare the array?” would have a dialogue act label of QUESTION.

*Task actions* refer to the computer programming actions that students took in pursuit of solving the given programming exercise, also called the *task*. This terminology is chosen in keeping with Natural Language Dialogue community’s tradition of referring to dialogue as “task-oriented” when the dialogue focuses on a task that is being undertaken. Task-oriented dialogues are often undertaken collaboratively, *e.g.*, problem-solving actions in the pedagogical context of tutorial dialogue. The same community also uses the term “task” to refer to particular problems that must be addressed by a computer-based dialogue system, such as interpreting user input. In this dissertation, the term “task” will be used to refer to both of the following: 1) the computer programming problem that students solved during the tutoring studies, as in “task/subtask structure,” and 2) the natural language dialogue tasks of user utterance interpretation and system move selection, as in “the task of interpreting user input.” This distinction will be made explicitly when it is not clear from the context.

Please see the glossary for a summary of the terms above, as well as additional definitions.

## 1.5 Contributions

The work reported in this dissertation has made the following novel contributions:[3]

- **Tutorial dialogue corpus.** Two pilot corpora and one main corpus of textual human tutorial dialogue were collected in the domain of introductory computer programming (Boyer, Vouk *et al.*, 2007; Boyer, Phillips *et al.*, 2008a; Boyer, Phillips, Wallis *et al.*, 2009a). All corpora were manually annotated with dialogue act labels. The main corpus consists of approximately 60 hours of tutoring and has been manually annotated with dialogue act labels, hierarchical task/subtask structure, and correctness of students' problem-solving actions.
- **Dialogue act annotation scheme.** The dialogue act annotation scheme was inspired by schemes for conversational speech, task-oriented dialogue, and tutoring (Core & Allen, 1997; Forbes-Riley & Litman, 2005; Marineau *et al.*, 2000; Stolcke *et al.*, 2000) and was adapted specifically to capture the communicative purposes of utterances in task-oriented tutorial dialogue. Inter-rater reliability studies have established the reliability of this dialogue act annotation scheme.
- **Task annotation scheme.** The task annotation scheme was inspired by other work on tutoring programming (Johnson & Soloway, 1985; Lane & VanLehn, 2004), and was adapted to capture the hierarchical structure of the computer programming exercise around which the dialogue is centered. Like the dialogue

[3] Portions of this work have been conducted in collaboration with colleagues. Rob Phillips proposed an initial dialogue act tagging scheme that heavily influenced the final scheme; Michael Wallis, Amy Ingram and William Lahti collaborated during the refining process. Rob Phillips and Amy Ingram were also instrumental in devising the task annotation scheme. Eunyoung Ha implemented the base HMM learning software. August Dwight and Taylor Fondren developed much of the RIPPLE system.

act tagging schemes, inter-rater reliability studies with the task annotation scheme indicate that it is sufficiently reliable.

- **Software for synchronous remote tutoring of Java programming.** RIPPLE (Remote Interactive Pair Programming and Learning Environment) was developed as part of the current work to support the tutoring studies (Boyer, Dwight *et al.*, 2008). RIPPLE extends Sangam, an existing pair programming environment for Java (Ho *et al.*, 2004) with a textual dialogue interface and real-time database capture of all interactions.
- **Results on the Structure of Tutoring in Introductory Computer Programming.** The exploratory results from the current work suggest ways in which tutors adapt to student characteristics (Boyer, Vouk *et al.*, 2007),[4] select strategies with cognitive and motivational factors in mind (Boyer, Phillips *et al.*, 2008a), give and take initiative (Boyer, Phillips, Wallis *et al.*, 2009a), and ask questions (Boyer, Lahti *et al.*, 2010).
- **HMM Framework for Learning Hidden Dialogue State.** The hidden Markov model (HMM) and hierarchical hidden Markov model (HHMM) learning approach discovers tutoring *modes*, or *hidden dialogue state* by utilizing a framework that combines sequential representation of dialogue acts with unsupervised discovery of adjacency pairs and a hierarchical task/subtask structure (Boyer, Phillips, Wallis *et al.*, 2009a). These tutoring modes are of intrinsic pedagogical interest (Boyer, Phillips, Wallis *et al.*, 2009b) and have been shown to correlate with student learning (Boyer, Phillips, Ingram *et al.*, 2010)[5]

[4] Recipient of the **Best Student Paper Award** at the International Conference on Artificial Intelligence in Education, 2007.

[5] **Best Paper Award Nominee** at the International Conference on Intelligent Tutoring Systems, 2010.

and to aid in the classification of student dialogue acts and prediction of human tutor moves (Boyer, Phillips, Ha *et al.*, 2010a; Boyer, Phillips, Ha *et al.*, 2010b).

- **Statistical Dialogue Act Model for Student Utterance Classification.** The models produced by this work are designed to address the complex task-oriented domain of introductory computer programming. This domain has characteristics and challenges different from those in conversational speech and from most of the other task-oriented domains that have been studied in dialogue systems research. Leveraging lexical, syntactic, dialogue history, task history, and hidden dialogue state features, the classifier performs comparably well to state-of-the-art classifiers in less complex domains (Boyer *et al.*, In press). The automatic classification of dialogue acts is an important step toward data-driven automatic extraction of a dialogue management model (Bangalore *et al.*, 2008).
- **HMMs and HHMMs for Tutorial Dialogue Act Prediction.** The HHMM framework for predicting tutorial dialogue acts is a step toward data-driven tutorial planning (Boyer, Phillips, Ha *et al.*, 2010a; Boyer, Phillips, Ha *et al.*, 2010b). The results demonstrate that hidden dialogue state, or tutoring mode, is an important structural element that improves the predictive power of the learned models on the corpus, and that explicitly representing hierarchical task/subtask structure within a hierarchical HMM yields a significant improvement over the prediction accuracy of a flat HMM.

Together, these contributions further the field's understanding of the effectiveness of human tutoring and advance the tools and techniques available for the collection and modeling of tutorial dialogue.

## 1.6 Organization

The remainder of this document is structured as follows. Chapter 2 presents background and related work on the effectiveness of tutorial dialogue, a historical view of tutorial dialogue systems, and an overview of dialogue modeling for user dialogue act classification and system act selection. Chapter 3 describes the tutorial dialogue studies that were conducted during the data collection phase of the project. Chapter 4 presents the annotation schemes for dialogue acts and task/subtask structure within the corpora. Chapter 5 describes the exploratory analyses that provided insights into the structure of tutorial dialogue in the complex task-oriented domain of introductory computer programming. These results speak to Hypotheses 1.1, 1.2, and 1.3. Chapter 6 presents machine-learned hidden Markov models of hidden dialogue states, discusses their structure, and examines their correlation with student learning (Hypothesis 2.1). Chapter 7 presents a learned model for user dialogue act classification (Hypothesis 2.2), while Chapter 8 presents a learned model for tutor dialogue move prediction (Hypothesis 2.3). Chapter 9 revisits the hypotheses and presents conclusions and directions for future work.

# CHAPTER 2

# Background and Related Work

This dissertation project falls at the intersection of two research fields: Intelligent Tutoring Systems and Natural Language Dialogue. Intelligent Tutoring Systems research is concerned with the design of intelligent systems to support learners, which often involves investigating fundamental learning processes that influence the way students interact with tutors or with systems. Section 2.1 presents background from this literature on the effectiveness of human tutorial dialogue, which holds important design implications for effective tutorial dialogue systems. Section 2.2 presents an overview of existing tutorial dialogue systems to highlight the novelty of this dissertation's data-driven approach to authoring a tutorial dialogue management model.

The second field in which this dissertation is positioned is that of Natural Language Dialogue, which is concerned with, among other things, the implementation of natural language dialogue systems. Background from this literature involving data-driven statistical approaches to the two central tasks of user utterance interpretation and system dialogue move selection is presented in Sections 2.3 and 2.4, respectively. These sections highlight existing techniques that can be adapted and extended to meet the needs of tutorial dialogue in a complex task-oriented domain.

## 2.1 Effectiveness of Human-Human Tutorial Dialogue

The field of educational psychology gave rise to seminal work establishing that one-on-one tutoring is significantly more effective than classroom instruction (Bloom, 1984; Cohen *et al.*, 1982). Spurred by those findings, a rich body of research has explored how students learn through tutoring. Results regarding what makes tutoring so effective are diverse and far-reaching; yet, the field does not fully understand which mechanisms are responsible for the full effectiveness of human tutoring. However, one important feature that has been emphasized in myriad studies is the *interaction* that takes place in human tutoring.

### *2.1.1 Importance of interactivity*

Human-human tutorial dialogue is a highly interactive process. Studies have revealed that this interaction often takes a structured form that includes the tutor and student collaboratively constructing and refining the solution to a problem (Fox, 1993; Graesser *et al.*, 1995). Controlled experiments have aimed to isolate this interactivity, providing empirical evidence that the effectiveness of tutoring is not due solely to the quality of the student's work nor to the quality of the tutor's moves, but rather, to the "interaction effect" between the participants (Chi, M.T.H. *et al.*, 2001). A further experiment has shown that tutorial dialogue is so powerful that it may even be effective when viewed vicariously by another student (Chi, M.T.H. *et al.*, 2008). Highly interactive natural language dialogue facilitates important cognitive moves on the part of the student such as deep questions (Graesser & Person, 1994) and self-explanation (Chi, M.T.H. *et al.*, 1994), and tutorial dialogue is particularly effective when the student's knowledge level is not well matched to existing learning materials (VanLehn *et al.*, 2007). Collectively, this research indicates that the highly interactive nature of natural language tutoring is an important contributor to its effectiveness.

In addition to its high level of interactivity, human tutorial dialogue exhibits other features that have been hypothesized to contribute to its effectiveness; among these are motivational and affective considerations. Expert human tutors have been found to pay close attention to student motivation to improve the student's motivational or emotional state, for example by occasionally presenting an easy problem that does not challenge the student, in order to increase the student's confidence (Lepper *et al.*, 1993). Further work suggests that even inexperienced human tutors respond based partly on the perceived affective state of the student, either due to mostly-subconscious adherence to universal rules of politeness (Wang *et al.*, 2005), or due to conscious strategy choice (Forbes-Riley & Litman, 2005; Porayska-Pomsta & Pain, 2004). Another contributor to the effectiveness of tutoring is the individualized instruction made possible when a tutor considers the student's knowledge and tailors questions or feedback based on the student's knowledge gaps (Glass *et al.*, 1999; Holt *et al.*, 1994; Ohlsson, 1994; Zhou & Evens, 1999).

### *2.1.2 Cognitive and motivational goals in tutoring*

Much of the research on motivation conducted in the ITS community is theoretically grounded in frameworks developed in the cognitive science community over the past several decades (Cameron & Pierce, 1994; Deci *et al.*, 2001; Keller, 1983). These theories state that student motivation plays a key role in the learning process. Studies of expert tutors have found that the most effective tutors give equal attention to both the motivational and cognitive concerns of students (Lepper *et al.*, 1993). This work refined previous models of motivation by postulating that motivation is comprised of confidence, challenge, control, and curiosity. It further identifies the two strategies of *praise* and *reassurance* as direct means of bolstering student confidence. These strategies are a form of "verbal persuasion," also identified by Bandura (1997), as one way of increasing *self-efficacy*, or people's beliefs about their capabilities to accomplish a particular task.

An increasingly active area of investigation is the search for tutorial dialogue policies that address the complementary cognitive and affective concerns that shape the tutoring process. Porayska-Pomsta and Pain (2004) use dialogue analysis to classify cognitive and affective feedback[6] in terms of the degree to which each addresses a student's need for both autonomy and approval. Forbes-Riley and Litman (2005, 2009) employ bigram analysis at the dialogue act level to extract tutorial strategies for responding to student uncertainty. Corpus analysis techniques have also informed work on the automatic classification of tutorial dialogue acts (Marineau *et al.*, 2000), though with respect to a much more limited set of dialogue acts than is considered in this paper. Corpora have also been used to compare the effectiveness of tutorial strategies in terms of learning outcomes (Ohlsson *et al.*, 2007; Rosé *et al.*, 2003; Rosé *et al.*, 2001).

Developing a clear understanding of the tradeoffs between cognitive and affective feedback is an important next step in tutorial dialogue research. Prior investigations of tutorial feedback have established a foundational understanding of cognitive feedback in terms of how and when it is delivered (Koedinger *et al.*, 1997). Jackson and Graesser (2007) found the presence of cognitive feedback, as opposed to motivational "progress" feedback, was responsible for higher learning gains in experimental versions of AutoTutor; on the other hand, the presence of cognitive feedback lowered students' motivational ratings. A consistent finding observed by Tan and Biswas (2006) was that students working with modified versions of the Betty's Brain tutoring system were able to learn better when given cognitive rather than affective feedback. Kelly and Weibelzahl (2006) investigated a motivational strategy in which a student was progressively shown another piece of a hidden image after each successful step through the learning task. Students in the motivational treatment group showed significantly larger increases in confidence levels compared with those in the control group, while there was no significant difference in learning gain. Finally, Wang *et al.* (2005)

[6] We use *feedback* to refer to "information communicated to the learner that is intended to modify the learner's thinking or behavior for the purpose of improving learning" (Shute, 2007).

found that tutors who gave polite feedback facilitated higher student self-efficacy gains, while learning was nearly unaffected.

Beyond these broadly observable tradeoffs, investigators have also found that tutorial strategies may impact student subgroups (*e.g.*, low ability vs. high ability students) in different ways. Rebolledo-Mendez *et al.* (2006) explored the effect of enhancing a tutoring system with motivational scaffolding. In M-Ecolab, initially unmotivated students were found to perform better with motivational adaptation and feedback, while students who were already motivated did not benefit from the motivational support. In a study of perceived politeness (a motivational aspect of tutorial utterances), Wang *et al.* (2005) found that students who were experienced with computers were less bothered by direct commands from a machine, while inexperienced students were more apt to appreciate politeness.

### *2.1.3 Student motivation in computer science education*

The overwhelming majority of computer science education literature has focused on the purely cognitive aspect of learning (Machanick, 2007). This trend is not surprising given the alluring parallels between cognitive learning models and the basic functions of computing that are fundamental to the discipline. For instance, the theoretical framework known as *constructivism* has been embraced for its insights into CS learning processes (Ben-Ari, 1998), and direct analogies are sometimes made between the constructivist view, in which students build and "debug" knowledge, and the activities involved in computer programming. Constructivism and other purely cognitive models of learning (*e.g.* Bloom, 1956) are valuable in understanding many phenomena surrounding the teaching and learning of computing. However, these models may not capture some important facets of the computer science learning process.

As Machanick (2007) observes, there are phenomena in CS education that are not readily explained by current purely cognitive frameworks. He proposes that *social* *constructivism,* a theoretical framework that is gaining acceptance in the broader education community, might offer explanations as to the observed effectiveness of some approaches such as peer assessment and apprenticeship-style teaching (Guzdial & Tew, 2006). The potential insights afforded by social constructivism stem from the theory's foundational tenets that learning has important social roles, and that communication is key to defining the knowledge of a learner. Evidence of the importance of communication in computer science learning environments has been noted by Barker and Garvin-Doxas (2004), who observe that the type of discourse that occurs in a computing classroom has far-reaching effects on learners. Further results on the importance of communication and the social role of learning have emerged from research in the contexts of pair programming (Slaten *et al.*, 2005) and non-majors learning to program (Wiedenbeck, 2005).

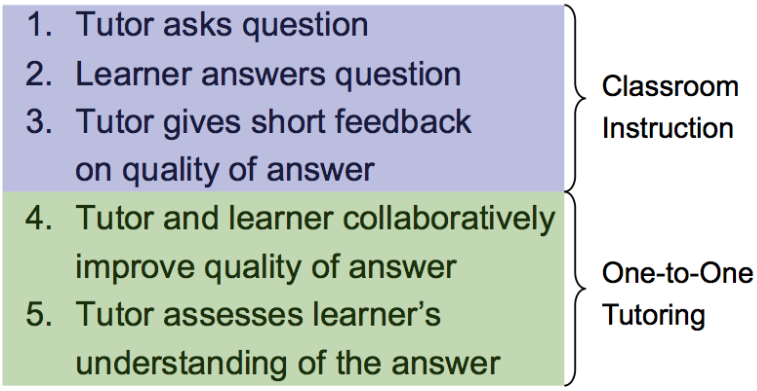
Tutoring is an instructional setting that has been proven effective in building knowledge and that is rich in communication. Long studied as an exemplary way to facilitate mastery of a subject, tutoring has been the setting for recent work in CS education research, for example, in investigating how students plan the solution to a programming problem (Lane & VanLehn, 2005). Because of the completeness of the instructional record created by controlled tutorial dialogue studies, it is possible to observe and make inferences on the fine details of learner activities.

Motivation, which refers to a learner's impetus for engaging in learning activities, has received attention in the general education research community for at least two decades (Cameron & Pierce, 1994; Deci *et al.*, 2001; Keller, 1983). Recently, motivating the learner has also been identified as a component of a complete conception of teaching computer science (Lister *et al.*, 2007). Learner motivation has also been considered in several recent empirical studies in computer science education. For example, Soh *et al.* (2007) included attitudinal variables for student self-efficacy and motivation as part of a data collection effort to assess the effectiveness of a redesigned computer science curriculum. Additionally, pair programming researchers recognize motivation as an important facet when measuring the impact of pair programming in educational settings (Williams *et al.*, 2002). These studies show an increased awareness of the importance of motivation in the computer science learning process.

In much of the existing computer science education research, motivational measures are taken at the beginning and end of an academic term to assess the impact of the instructional approach utilized during the term. Tracking changes at this granularity has proven a useful research approach. However, studying learner motivation at a finer granularity, for instance, over the course of a single programming assignment, can complement the coarser granularity approach generally undertaken to date. For example, Wolfe (2004) considers learner motivation at the level of a single programming assignment, observing that the rhetoric used in problem descriptions influences students' motivation. Wolfe suggests that programming assignments should emphasize real-world purpose and human factors. This kind of contribution is made possible by studying learner motivation at a finer granularity than over entire academic terms.

## 2.2 Tutorial Dialogue Systems

Motivated in part by the demonstrated effectiveness of one-on-one human tutoring compared to classroom instruction, the first intelligent tutoring systems (ITSs) emerged more than two decades ago (Wenger, 1987). Despite great strides, ITSs have fallen consistently short of the highest learning outcomes achieved by human tutors (VanLehn, 2008). One hypothesized explanation for this discrepancy is the systems' lack of natural language interaction with students. Observational studies of human tutoring have revealed a common pattern referred to as the *tutoring frame* (Graesser *et al.*, 1995). The first three elements of this frame are present in classroom instruction and traditional ITSs. However, the last two elements, which involve interacting in natural language to improve students' responses, are present only in tutorial dialogue (Figure 2).



1. Tutor asks question
2. Learner answers question
3. Tutor gives short feedback on quality of answer
4. Tutor and learner collaboratively improve quality of answer
5. Tutor assesses learner's understanding of the answer

*Figure 2. The 5-step tutoring frame*

In response to the hypothesis that natural language interaction constitutes a sort of "missing link" for achieving the effectiveness of expert human tutors with intelligent systems, recent years have seen the rise of *tutorial dialogue systems* that interact with learners through natural language dialogue.

### *2.2.1 Behavior of existing tutorial dialogue systems*

This section presents several tutorial dialogue systems and discusses the extent to which the behavior of each was informed by the study of tutoring corpora.

**CIRCSIM-Tutor.** CIRCSIM-Tutor, which supports students in developing an understanding of the human circulatory system, was the result of a lengthy collaboration between researchers at Illinois Institute of Technology and Rush Medical College (Evens & Michael, 2006). The system presents a scenario and asks the student to predict the directionality of change in several parameters pertaining to the cardiovascular system. Students enter those predictions in a predictions table, and the tutor provides feedback on correctness by marking through incorrect predictions. Subsequently the system engages the student in a tutorial dialogue to correct any misconceptions displayed by the students' response.

The tutoring strategies in CIRCSIM-Tutor are comprised of a combination of two tutorial moves: *elicit* and *inform*. *Elicit* involves making a request or asking a question. *Inform* involves presenting facts or explanations. Rather than providing immediate feedback on mistakes, CIRCSIM-Tutor waits for a student to complete a subset of the predictions required by a problem before providing feedback. This strategy allows the tutor to observe a pattern in the student's prediction to more accurately diagnose the potential student misconception.

Numerous human tutoring studies were conducted throughout the CIRCSIM project, most with two expert tutors who were university professors. Some studies were also conducted with unskilled tutors. The dialogues were conducted with a remote textual dialogue system and were annotated using Standardized General Markup Language. Some rules for the behavior of the system were extracted directly from these annotated transcripts using decision trees. Other tutorial strategies were extracted by manually clustering tutoring transcripts into similar groups and then noting the patterns that emerged. The project produced particularly influential findings regarding tutorial dialogue, especially with respect to the differences between expert and novice tutors (Evens & Michael, 2006).

**AutoTutor.** Developed at the University of Memphis, AutoTutor is an intelligent tutoring system for qualitative physics and computer literacy (Graesser *et al.*, 1999; Graesser *et al.*, 2004; Graesser *et al.*, 2005). AutoTutor asks questions and assesses each student answer using a statistical approach that matches words in the student's response to words in expected correct or incorrect answers. AutoTutor then engages in remedial dialogue intended to address misconceptions or fill gaps that were indicated in the student's response. The system's utterances to the student are delivered in speech through a "talking head" that uses gestures and facial expressions as well as intonation. The tutor utterances are also recorded in a textual dialogue history in the interface. Student utterances are delivered textually.

AutoTutor employs a dialogue strategy that involves providing brief feedback to assess the correctness of each student turn. Then the system moves on to another dialogue move by selecting from one of several possible actions: *pumping, prompting*, *elaborating*, *correcting*, and *hinting*. *Pumping*, such as “Uh huh,” and “What else?” is used near the beginning of each dialogue to encourage the student to continue constructing utterances. *Prompting* is a more content-rich version of pumping in which the tutor begins a sentence and then pauses with vocal tone or gesture inviting the student to type the phrase that completes this sentence; this scaffolded type of pumping is used primarily when material is believed to be unfamiliar to students. *Elaborating* involves the tutor stating information that the student has not supplied. *Hints* are used when the student is struggling with a question. Finally, AutoTutor employs the strategy of direct *correction* when the student makes an utterance that the system is confident contains shallow errors. This strategy is not used if the system has low confidence in its assessment or if the student’s utterance is judged to display deep misconceptions that should be reasoned out rather than directly corrected.

The behavior of AutoTutor was informed by extensive studies of unskilled human tutors; that is, the tutors were knowledgeable about the subject matter but had no training in formal tutoring methods. Over several years, researchers videotaped and transcribed approximately 100 hours of tutoring in domains such as undergraduate psychology and middle school algebra. AutoTutor’s short feedback immediately after student turns was based on observing this behavior consistently with human tutors. AutoTutor’s omission of an explicit student model was also based on the tutoring studies; researchers noticed that tutors did not appear to longitudinally model the students’ knowledge in a sophisticated way, but rather, that tutors responded on a turn-by-turn basis to the knowledge (or lack thereof) displayed by the most recent student turn. The rich body of qualitative observations from tutoring studies served as the primary source of design decisions for AutoTutor; in fact, the system designers often refer to it as a "simulation of a human tutor." However, as is the case with most systems presented in the remainder of this section, the corpus analysis did not involve constructing generative models of dialogue.

**The Geometry Explanation Tutor.** The Geometry Explanation Tutor was developed at Carnegie Mellon University and extended with dialogue capabilities a previously existing cognitive tutor (Aleven *et al.*, 2001; Aleven *et al.*, 2004). The goal of this research was to support students' self-explanations of their actions (Chi, M.T.H. *et al.*, 1994). As students take problem-solving steps in the problem-solving pane, they must fill in a box with an explanation for that problem-solving step. For incomplete or incorrect explanations, Geometry Explanation Tutor engages students in restricted dialogues designed to elicit explanations that are reasonably mathematically precise.

Following the cognitive tutor architecture, the Geometry Explanation Tutor traces the students' solutions to a problem and provides hints and feedback depending on whether the student's actions and explanations match correct or "buggy" rules. Unlike Autotutor which uses latent semantic analysis to process student input statistically, Geometry Explanation Tutor parses student input into knowledge structures corresponding to rules of the geometry domain. The tutorial strategy involves accepting student input that was correct and complete. For incomplete or incorrect explanations, the tutor randomly selects from the list of violated or missing rules and engages in remedial dialogue that involves asking the student a follow-up question or giving advice on how to provide a better explanation. The student then enters a new explanation and the process repeats.

During the design process of the Geometry Explanation Tutor, researchers collected a data set of written student explanations on a paper test to reveal some of the language processing challenges facing the system. These data revealed that the system would face error-ridden and very short student utterance input. Based on these data, researchers manually identified a hierarchy of 149 explanation categories and designed tutor utterance responses to the categories. The dialogue is shallow, involving only one tutor feedback turn indicating correctness or the presence of errors for each student explanation, and the tutor responses were not based directly on any human tutoring study.

**Why2-Atlas.** Researchers at the University of Pittsburgh have developed Why2-Atlas, a tutorial dialogue system for qualitative physics (Jordan *et al.*, 2006; VanLehn *et al.*, 2002). In this system, students are asked a qualitative physics problem and must write an essay to answer the problem. After assessing the essay, the system engages the student in a dialogue intended to give feedback, address misconceptions, and discuss missing explanations in the student's original essay. After this dialogue, the student is asked to write a revised essay. The process continues until the student's essay is judged acceptable.

Why2-Atlas parses student essays into a set of propositions in first-order logic. Explicit misconceptions create a tutor goal of remedying that misconception, while expected propositions that were missing create corresponding tutor goals of eliciting the required content. The tutorial dialogue for addressing misconceptions and eliciting required content are conducted through Knowledge Construction Dialogues (KCDs), scripts that elicit lines of reasoning from students by asking questions.

Publications on the development of Why2-Atlas do not explicitly describe tutoring corpora that informed the system's design. Instead, requirements were gathered from physics tutors regarding the propositions required in each essay and the KCDs were implemented accordingly to address erroneous and missing concepts.

**CycleTalk.** CycleTalk (Rosé, Aleven *et al.*, 2004) is a dialogue system from Carnegie Mellon University that tutors college-level thermodynamics. The tutoring strategy involves negotiating the problem-solving goals between tutor and student and allowing students to pursue these goals within an exploratory learning environment, CyclePad, in which students construct thermodynamic cycles and perform efficiency analyses. A goal of the project was to determine the desired behavior of the CycleTalk system by analyzing corpora of tutoring collected through a Wizard-of-Oz study with a human tutor. Qualitative analysis of the Wizard-of-Oz corpus yielded several general system desiderata (Rosé, Torrey *et al.*, 2004). For example, it was observed that the system should be able to engage in activities including supporting students' functional analysis of their designs and weighing tradeoffs between alternate design choices.

A tutorial dialogue system was implemented using KCDs as in Why2-Atlas. These scripts were modeled after the human tutors from the Wizard-of-Oz study (Kumar *et al.*, 2006). The CycleTalk system was used not only to support individual learners but also to provide adaptive support for pairs of students, who were found to learn significantly more than students in the individual condition (Kumar *et al.*, 2007).

**ITSPOKE.** ITSPOKE, developed at the University of Pittsburgh, engages students in spoken dialogue for tutoring qualitative physics (Litman & Silliman, 2004). It is a speech-enabled version of the Why2-Atlas text-based tutoring system. In ITSPOKE, students first type responses to a qualitative physics problem, and ITSPOKE engages the students in spoken dialogue to refine the original answer. During this spoken dialogue the system takes actions such as providing feedback and prompting the student. When the dialogue has completed the system asks the student to edit the original written essay. Rounds of spoken tutoring continue until the system judges the student's essay to be acceptable.

The tutoring behavior of ITSPOKE was not originally based on empirical corpora, but rather, was provided by the Why2-Atlas tutoring system (Jordan *et al.*, 2006; VanLehn *et al.*, 2002) described earlier in this section. However, in recent years extensive empirical investigations have been undertaken in the context of ITSPOKE to refine its tutorial strategies. It has been found that detecting and responding to student uncertainty increases the effectiveness of ITSPOKE (Forbes-Riley & Litman, 2009). Machine learning approaches such as reinforcement learning have also been applied to corpora collected with human students using ITSPOKE. These studies suggest that micro-level tutorial decisions, such as whether to elicit a piece of information from a student or tell it directly, impact the effectiveness of tutoring (Chi, M. *et al.*, 2010). Additional empirical results demonstrate how surface-level language features are associated with learning in both spoken and textual dialogue (*e.g.*, Litman *et al.*, 2006; Purandare & Litman, 2008). Ongoing work with the ITSPOKE system aims to explore ways in which the system can respond to student affect to improve student learning.

**ProPL.** Developed at the University of Pittsburgh, ProPL is a tutorial dialogue system that supports novice computer science students as they write pseudocode for a solution to an introductory programming problem (Lane, 2004; Lane & VanLehn, 2005). Students make design notes and write pseudocode in the problem-solving pane and carry on textual dialogue with the system in the dialogue pane.

ProPL considers student solutions as consisting of goals and plans, and for each of these units that is recognized in the problem-solving pane, the system can employ a Knowledge Construction Dialogue (KCD), a script that elicits a line of reasoning from the student by asking a series of questions. If the system does not recognize a correct student answer through identification of keyword phrases, another KCD for remediation is entered; alternately, the system can make a bottom-out move, an utterance that provides the complete answer to the question. The tutorial moves that comprise KCDs include *pumping*, *pointing* to a relevant piece of information in the original problem statement, *rephrasing* a previously asked question, or *elicit*ing an observation from the student about the condition of the current solution. Tutorial strategies encoded as KCDs also include *hypotheticals*, in which the tutor asks the student to consider possible scenarios that might cause the solution to fail, *eliciting abstractions*, intended to request a more general answer to a previously asked question, and *concrete examples* designed to set up tutor elaboration of a topic.

The corpus study that informed ProPL's design consisted of 27 tutoring sessions across 2 problems conducted with a single tutor. These corpora were used to identify approaches and misconceptions of the students and to analyze the behavior of the tutor. The discourse was manually segmented at the location of each top-level "what" question with which the tutor aimed to elicit the goal of the student; these segments were further decomposed based on the location of second-level "how" questions designed to elicit plans. Facilitated by the fact that the system designer served as the tutor during corpus collection, manual qualitative analysis of the tutoring corpora led to the extraction of tutoring rules, which were then implemented as system behaviors.

**BeeDiff.** Researchers at the University of Edinburgh have developed the BeeDiff tutor, a tutorial dialogue system that helps students solve symbolic differentiation problems (Callaway *et al.*, 2007; Dzikovska *et al.*, 2006). Its predecessors were BEETLE and BEETLE2 (Zinn *et al.*, 2002), tutoring systems for the domain of basic electricity and electronics. BeeDiff displays a differentiation problem to the student, who is then able to work out a solution sequentially or input an answer immediately using an equation-editing pane; in addition, students can ask questions in the textual dialogue pane. Unlike AutoTutor, which uses latent semantic analysis, BeeDiff uses the TRIPS dialogue parser (Allen *et al.*, 2001), which extracts a domain-independent representation of the student's utterance. This representation is mapped onto the domain of differentiation and is then passed to a domain reasoner that assesses whether differentiation rules were correctly represented.

When the student makes a mistake, BeeDiff employs an adaptive feedback strategy that depends on the student's performance. High performing students get vague feedback such as “Not quite,” while weaker students receive more specific help that includes a hint or, in the bottom-out case, the complete answer. Four levels of feedback are included, which range from a complete answer to a content-free hint as illustrated above.

BeeDiff’s tutoring strategy is informed by a corpus of nineteen human-human tutorial dialogues. These tutoring sessions were carried out with tutors and students in separate rooms and communicating through textual dialogue; tutors viewed a synchronized version of the students’ problem-solving workspace. These dialogues were used to focus the system design, such as how verbose the tutor should be and how prevalent was the use of inline equations in the textual utterances. The dialogues were also manually annotated for “task segments” such as *State Problem*, *Solve*, and *Tidy Up*. Finally, the corpus was annotated with dialogue acts. Binary transition diagrams were used to indicate the presence (or absence) of adjacent pairs of dialogue acts. This analysis is comparable to the bigram analysis utilized in this dissertation as a baseline model for predicting tutor moves.

**iList.** The iList tutorial dialogue system supports students in learning and applying basic data structures and algorithms content in computer science (Fossati *et al.*, 2008). The system provides students with a problem and a data structure drawing, which the student may modify to help solve the problem. Students write a computer program to solve the data structure problem in the problem-solving pane. The system provides feedback within a feedback pane, with hints produced by constraint-based modeling which can identify student mistakes.

A major goal of the iList project was to define the system’s behavior in an empirically grounded way. One corpus analysis revealed that the frequency of positive tutorial feedback was correlated with learning. This finding was used to modify the approach of iList, which originally provided mostly negative feedback because this feedback is more straightforward to implement with the chosen approach of constraint-based modeling (Fossati *et al.*, 2009).

The constraint-based model was learned as a Markov chain based on the corpus of past student interactions (Fossati *et al.*, 2010).

All of the tutorial dialogue systems discussed in this section aim to engage students in rich natural language dialogue in support of a learning task. As a group, the systems have important limitations with respect to development time and effectiveness, as discussed below. The work in this dissertation aims to address those limitations with data-driven techniques.

Today's tutorial dialogue systems generally required a large amount of development time: several hundred hours per hour of tutoring instruction (Aleven *et al.*, 2009). As such, these systems provide tutoring for only a small number of topics. Due in part to this limitation, tutorial dialogue systems have never achieved the effect sizes observed with expert human tutors over the course of an academic term (Bloom, 1984). Instead, the effect sizes that have been observed with tutorial dialogue systems are often on par with ordinary (not expert) human tutors (Van Lehn *et al.*, 2008).

As discussed above, the extent to which corpora of human dialogue have historically been used to inform the behavior of each system is limited. The disconnect between effective human tutoring and the implemented behavior of tutorial dialogue systems may be partly responsible for the systems' less-than-optimal effectiveness. Recent research within the contexts of projects such as CycleTalk, ITSPOKE, and iList described above made a step toward data-driven tutorial dialogue system authoring by applying more extensive corpus-based techniques than had previously been utilized within the Intelligent Tutoring Systems community. The work in this dissertation goes one step farther by inducing a model of tutorial strategies at the dialogue act level directly from a corpus. Furthermore, the current models leverage hidden dialogue state, a novel way to potentially capture tutorial dialogue modes automatically.

New dialogue structure modeling techniques that facilitate data-driven authoring of tutorial dialogue system behavior may allow tutoring systems to more closely reflect the behavior of the most effective human tutors, increasing the systems' flexibility and robustness. Data-driven system development may also allow systems to cover more topics and therefore provide longitudinal, effective support to students. Sections 2.3 and 2.4 discuss these data-driven dialogue management techniques as they have been reported in the Natural Language Dialogue literature.

## 2.3 User Utterance Interpretation in Dialogue Systems

For natural language dialogue systems, including tutorial dialogue systems, a central challenge is interpreting users' input. This interpretation involves numerous levels of natural language understanding; for example, in spoken dialogue systems, automatic speech recognition is a challenging problem in itself. It focuses on simply identifying the words that were spoken. Beyond low-level word understanding, interpreting the user's input in dialogue involves identifying the *dialogue act*, or communicative purpose (*e.g.*, asking a question, giving a command) of each utterance. Dialogue acts (Austin, 1962; Jurafsky & Martin, 2008) provide a valuable intermediate representation that can be used for dialogue management.

A variety of dialogue act classification approaches have been investigated in the Natural Language Dialogue literature. These techniques have utilized both sequential approaches and vector-based classifiers. Sequential approaches (Stolcke *et al.*, 2000) often treat dialogue as a discrete-time Markov chain, in which an observation depends on a single preceding observation (Jurafsky & Martin, 2008; Levin *et al.*, 2000). It is known that the first-order Markov assumption does not strictly hold in natural language dialogue because of the potential dependence of each observation on the full dialogue history. However, because of the strong local dependence that has been observed in dialogue and because of the models' computational tractability, Markov models such as first-order observed Markov models (also known as *bigram* models) and Markov decision processes have proven useful in a wide variety of dialogue applications (Bangalore *et al.*, 2008; Forbes-Riley & Litman, 2005; Forbes-Riley *et al.*, 2007; Levin *et al.*, 2000; Young *et al.*, 2009).[7] HMMs are an example of these sequence-based models, and they model uncertainty within a doubly stochastic framework (Rabiner, 1989). An introduction to HMMs is provided in Section 6.1.

Vector-based approaches to dialogue act classification, such as maximum entropy modeling, frequently take into account a variety of lexical and syntactic features of local utterance context.[8] Many vector-based classifiers also leverage structural features such as dialogue act history and task/subtask history. Work by Bangalore *et al.* (2008) on learning the structure of human-human dialogue in a catalogue-ordering domain (also extended to the Maptask and Switchboard corpora) utilizes features including words, part of speech (POS) tags, supertags, and named entities, and structural features including dialogue acts and task/subtask labels when applicable. To perform incremental decoding of both dialogue acts and task/subtask structure, they take a greedy approach that does not require the search of complete dialogue sequences. The models reported in this dissertation also perform left-to-right incremental interpretation and prediction with a greedy approach. For student dialogue act classification (Chapter 7) the feature vectors differ from the aforementioned work slightly with respect to lexical and syntactic features and notably in the addition of a set of hidden dialogue state features generated by a separately trained HMM.

[7] A dialogue act classification model is only one component within a fully functional dialogue system. While the dialogue act classification model may make a first-order Markov assumption, the dialogue system usually does not; it stores and makes use of many aspects of the full dialogue history.

[8] Although vector-based approaches and sequence-based approaches make use of different techniques, both approaches make an *n*-th order (often first-order) Markov assumption regarding the process being modeled. Sequential Markov models make this assumption explicitly through conditional probability distributions. Vector-based models, by requiring feature vectors to be finite and of fixed size, make the same assumption implicitly.

Recent work by Sridhar *et al.* (2009) has explored the use of lexical, syntactic, and prosodic features for online dialogue act tagging; this work explores the notion that other structural features could be omitted altogether from incremental left-to-right decoding, resulting in computationally inexpensive and robust dialogue act classification. Although textual dialogue does not feature acoustic or prosodic cues, this dissertation reports on the use of lexical/syntactic features alone to perform dialogue act classification.

Like Bangalore *et al.* (2008), the work reported in this dissertation treats task structure as an integral part of the dialogue model. Other work that has taken this approach includes the Amitiés project, in which a dialogue manager for a financial domain was derived entirely from a human-human corpus (Hardy *et al.*, 2006). The TRIPS dialogue system also closely integrated task and dialogue models, for example, by utilizing the task model to facilitate indirect speech act interpretation (Allen, Ferguson *et al.*, 2001). Work on the Maptask corpus has modeled task structure in the form of conversational games (Wright Hastie *et al.*, 2002). Recent work in task-oriented domains has focused on learning task structure from corpora with supervised (Bangalore *et al.*, 2008) and unsupervised (Chotimongkol, 2008) approaches. Emerging unsupervised methods, such as for detecting actions in multi-party discourse, also implicitly capture a task structure (Purver *et al.*, 2006).

The domain of tutoring introductory programming differs from all the task-oriented domains discussed above in that the dialogues center on the user's creation of a standalone artifact through a separate, synchronous stream of user-driven task actions. To illustrate, consider a catalogue-ordering task (Bangalore *et al.*, 2008) in which one subtask is to obtain the customer's name. The fulfillment of this subtask occurs entirely through the dialogue, and the resulting artifact (a completed order) is produced by the system. In contrast, our task involves the user constructing a solution to a computer programming problem. The fulfillment of this task occurs partially in the dialogue through tutoring, and partially in a separate synchronous stream of user-driven task actions about which the system must reason.

To deal with this complexity, task actions and dialogue acts are integrated into a shared sequential representation. Additionally, task events and hidden dialogue state features are encoded in feature vectors for classification.

## 2.4 Move Selection in Dialogue Systems

Like the dialogue act classification work mentioned in Section 2.3, much of the data-driven research on selecting dialogue system dialogue moves relies on a Markov assumption (*e.g.*, Levin *et al.*, 2000; Chi, M. *et al.*, 2010). Although this assumption is not strictly true in real world dialogue data because a long history of dialogue moves may influence the current observation, the Markov assumption has proven useful in numerous dialogue modeling applications due to its computational tractability and the relatively large amount of variation captured within a short window of dialogue history. When a first-order Markov model is used, each observation is assumed to depend only on the preceding observation; models that consider pairs of dialogue moves in this way are also called *bigram* models.

Dialogue is often treated as a Markov decision process (MDP) or partially observable Markov decision process (POMDP) and then reinforcement learning (RL) is applied to derive optimal dialogue policies (Frampton & Lemon, 2009). Sparse data and large state spaces can pose serious obstacles to RL, and recent work aims to address these issues with novel approaches to user simulations (Ai *et al.*, 2007), combining supervised and reinforcement learning techniques (Henderson *et al.*, 2008), and constraining the state space with information-state update rules (Heeman, 2007). Another approach involves partitioning the state space to form equivalence classes when the data do not support further distinctions (Young *et al.*, 2009).

For tutorial dialogue, RL has been applied for feature selection with the goal of selecting a state space representation that best facilitates learning an optimal dialogue policy (Tetreault & Litman, 2008). RL has also been used for comparing specific tutorial dialogue tactic choices, such as whether to tell a student the next step during tutoring, or whether to elicit the student's hypothesis (Chi, M. *et al.*, 2008).

While RL learns a dialogue policy through exploration, the work in this dissertation assumes that a flexible, good (though possibly not *optimal*) dialogue policy is realized in successful human-human dialogues. This policy can be extracted by learning a model that predicts human tutor actions within a corpus. Using human dialogues directly in this way has been the focus of work in other task-oriented domains. For example, in the Amitiés project, a dialogue manager for a financial domain was derived entirely from a human-human corpus (Hardy *et al.*, 2006). Work by Bangalore *et al.* (2008) on learning the structure of human-human dialogue in a catalogue-ordering domain (also extended to the Maptask and Switchboard corpora, which involve giving directions and conversational speech, respectively) utilizes a variety of lexical, syntactic, and task features to perform incremental decoding of the dialogues, including predicting system (customer service agent) dialogue moves. Like the parse-based models of Bangalore *et al.*, the hierarchical hidden Markov models (HHMM) in this dissertation explicitly capture the hierarchical nesting of tasks and subtasks in the domain (Chapter 8). While their parse-based model constitutes a learned model of task structure, the HHMM in this work treats task/subtask structure as given. However, in contrast to Bangalore *et al.*'s results, the hierarchical models described here outperformed flat models in terms of accuracy for predicting system dialogue acts.

For tutorial dialogue, there is compelling evidence that human tutoring is a valuable model for extracting dialogue system behaviors. Over the course of its rich history, the CIRCSIM-TUTOR project utilized corpora of both expert and novice tutoring to inform the system's behavior, and machine learning rule extraction approaches were used for identifying tutor strategies (Evens & Michael, 2006). Using bigram models, Forbes-Riley *et al.* utilized a corpus of human tutoring to derive a model for responding to student uncertainty, and this model has subsequently been shown effective for improving tutoring outcomes (Forbes-Riley *et al.*, 2007; Forbes-Riley & Litman, 2009). This dissertation builds on the bigram work of Forbes-Riley *et al.* by moving beyond first-order Markov models to consider HMMs and HHMMs that capture hidden dialogue state to increase the predictive and classification power of the models.

Other ongoing work that aims to develop tutorial dialogue system behavior in a data-driven way is being conducted in the context of KSC-PAL, an intelligent tutoring system that supports peer collaboration (Kersey *et al.*, 2009). Like the introductory computer science task involved in this dissertation, the computer science task that KSC-PAL supports involves the creation of a separate learning artifact, in their case drawings of computer science data structures. Events related to modifying this learning artifact are considered as unified components of the dialogue management model.

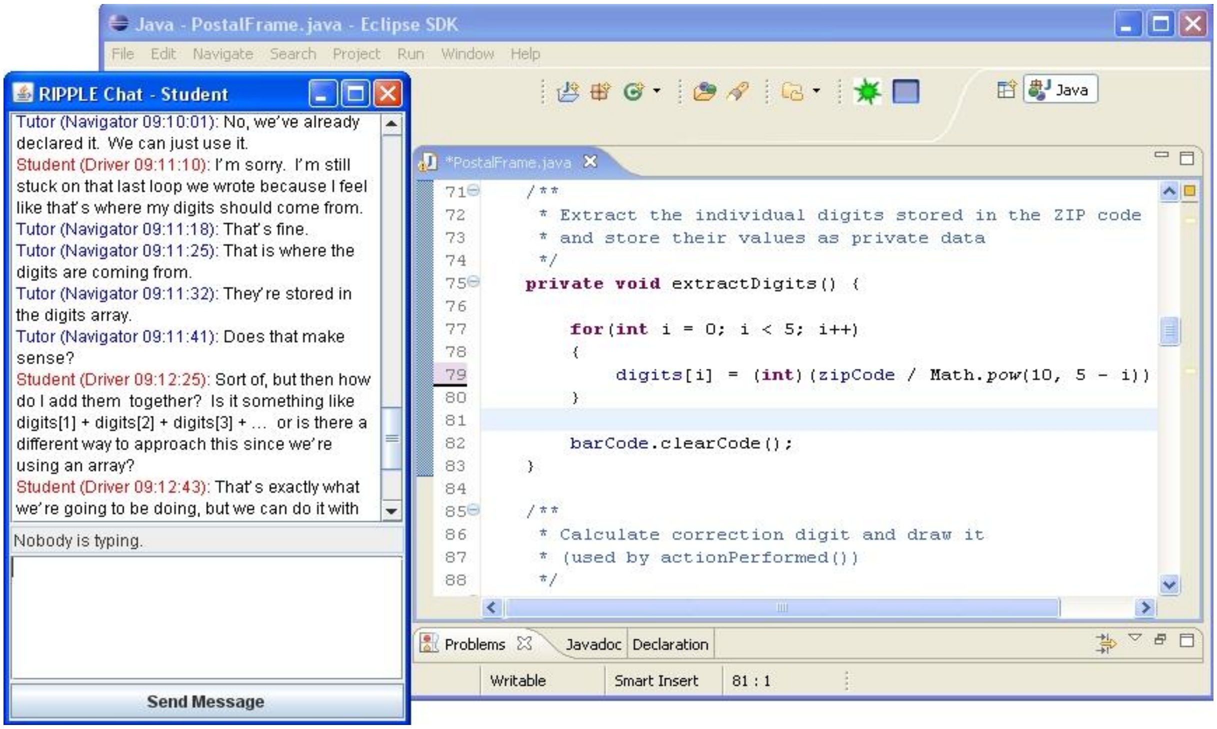
# CHAPTER 3

# Human Tutoring Studies

This project, which adopts a corpus-based methodology (Section 1.3), utilizes records of human-human tutorial dialogue as the basis for exploratory analysis and machine learning of predictive models. It was desirable for the corpora to consist of rich, naturalistic human dialogue that centered on a course-embedded student learning task within an introductory computer programming course. Three observational tutorial dialogue studies, two pilot studies and one primary study, were conducted with human tutors and introductory computer programming students. The studies yielded three textual dialogue corpora. All three of these studies were exploratory in nature; that is, they were not designed experiments, but were controlled studies conducted with the goal of producing a corpus of rich, natural language tutorial dialogue in the domain of introductory computer programming. Another important goal was for the tutoring to produce positive student learning gains as measured from pre-test to post-test, an outcome that was confirmed in learning gain analysis after the studies were complete (Sections 5.1-5.3). The study process and materials underwent iterative refinement. Software and instruments to measure motivation and learning were piloted in the first two studies, and then utilized in the final and largest study. Study I was conducted in the Fall semester of 2006, Study II was conducted in the Spring semester of 2007, and Study III, the main study, was conducted in the Spring semester of 2008. The remainder of this chapter describes the three tutoring studies.

## 3.1 Software

Tutors and students collaborated remotely through textual dialogue from separate rooms. This remote collaboration was made possible by RIPPLE, a software tool designed to facilitate real time remote collaboration on programming projects (Boyer, Dwight *et al.*, 2008).[9] RIPPLE features a synchronized problem-solving pane and a textual dialogue pane (Figure 3). All programming and dialogue events are logged to a database.



Tutor (Navigator 09:10:01): No, we've already declared it. We can just use it.
Student (Driver 09:11:10): I'm sorry. I'm still stuck on that last loop we wrote because I feel like that's where my digits should come from.
Tutor (Navigator 09:11:18): That's fine.
Tutor (Navigator 09:11:25): That is where the digits are coming from.
Tutor (Navigator 09:11:32): They're stored in the digits array.
Tutor (Navigator 09:11:41): Does that make sense?
Student (Driver 09:12:25): Sort of, but then how do I add them together? Is it something like digits[1] + digits[2] + digits[3] + … or is there a different way to approach this since we're using an array?
Student (Driver 09:12:43): That's exactly what we're going to be doing, but we can do it with

```
    /**
     * Extract the individual digits stored in the ZIP code
     * and store their values as private data
     */
    private void extractDigits() {

        for(int i = 0; i < 5; i++)
        {
            digits[i] = (int)(zipCode / Math.pow(10, 5 - i))
        }

        barCode.clearCode();
    }

    /**
     * Calculate correction digit and draw it
     * (used by actionPerformed())
     */
```

*Figure 3. Remote collaborative tutoring interface*

[9] The RIPPLE system was implemented collaboratively with August Dwight and Taylor Fondren, whose undergraduate research projects were funded by the STARS Alliance (www.starsalliance.org) and the NC State University Department of Computer Science. Dright Ho, Lucas Layman, and Andy Sherriff provided valuable technical support during the development project.

The majority of RIPPLE's functionality for supporting synchronous views of a project is provided by Sangam (Ho *et al.*, 2004), an Eclipse plug-in for distributed pair programming. Sangam relies on Eclipse[10] for compilation and execution of students' source code. When the student performs a single action, an event is generated and transmitted to the tutors' workspace. The set of supported actions includes file system manipulation (*e.g.*, creating and deleting files), editor operations (*e.g.*, typing and highlighting) and program launch (execution). To ensure robust operation over the network, RIPPLE regularly performs integrity checks to assure the editor contents are identical for both users.

## 3.2 Student Participants

Student participants were volunteers who were enrolled in CSC 116, an introductory computer science course titled "Introduction to Computing – Java" at North Carolina State University. Study I involved 35 participants, Study II involved 43 participants, and Study III involved 61 participants.[11] Students were compensated for participation through a small amount of class credit that varied according to instructor preference. Data to establish the representativeness of the students in terms of class grades are not available; however, over half of the enrolled students each semester participated in the study. Participants included students whose declared majors were mechanical, electrical, and computer engineering, along with students majoring in computer science.

## 3.3 Tutors

Study I used six volunteer tutors: four graduate students, one female and three male, and two advanced undergraduate students, both male. Study II used fourteen volunteer tutors: twelve

[10] http://www.eclipse.org

[11] Approved Human Subjects Research studies at North Carolina State University. Study I IRB #134-06-4; Studies II –III IRB #344-07-10

graduate students, two female and ten male, and two advanced undergraduates, both male. Study III involved the two most effective tutors from the prior studies, that is, the tutors who had the highest average student learning gains across Studies I and II. The tutors in Study III, one female graduate student and one male upper-division undergraduate student, were paid for their time. All tutors across the studies were between the ages of 19 and 30 and had a minimum of one semester of experience as a tutor or teaching assistant. Two tutors in Study II also had experience as classroom instructors. None of the tutors were involved as instructors or teaching assistants with the course from which the participants were drawn. Students' and tutors' identities were not revealed to each other before, during, or after the tutoring sessions.

The tutor orientation consisted of a problem-solving session in which all the tutors met to work through alternate solutions to the programming problem. In addition, tutors were shown the student instruction video (described in Section 3.5) to familiarize them with the starting knowledge of the student regarding the software being used. The student instruction video also served as the tutor orientation to the software. Tutors were not instructed to use any specific instructional approaches or tutorial strategies because the intent was for each tutor to use his or her own strategies to accomplish the goal of helping students complete a programming exercise while ensuring that students developed an understanding of the general concepts used in the solution. In this way, the data represent a sampling of naturalistic human tutoring for introductory computer science.

## 3.4 Problem-Solving Task

The studies began in the eighth week of each semester, and the problem-solving task given to students was designed to be commensurate with their classroom and laboratory exposure at the time. Studies I and II used a programming exercise taken from the standard laboratory manual for the course (Appendix A). The programming exercise focused on using array data structures along with loop constructs. Students were provided a partial solution that included an (initially empty) graphical display of the generated results. Students were required to complete three code modules to solve the programming problem. Based on tutor feedback from the previous two studies, which indicated the programming problem was unnecessarily confusing for students, Study III used a slightly simplified programming exercise (Appendix B). The latter was designed with social relevance in mind, a property thought to be implicitly motivational to students (Layman *et al.*, 2007). As in the previous studies, the programming exercise focused on using array data structures and loop constructs to complete three modules. Because this programming exercise was handcrafted for the tutoring study, it was tested in the semester prior to Study III with a small group of volunteers who were enrolled in CSC 116 (the same course used for all three tutoring studies) or CSC 216 (the next course in the programming sequence). Anomalies in the problem description, code scaffolding, and the corresponding pre-test/post-test were corrected prior to deploying the exercise in Study III.

## 3.5 Procedure and Instruments

Upon arrival, students completed a pair of written instruments consisting of survey items on the student's motivation to study computer science, including the student's self-efficacy (Bandura, 2006). In Study I, these data were the first to be collected for each participant; in Studies II and III, participants were also asked to complete an electronic survey containing several demographic and psychometric instruments prior to arriving for the study. The demographic instrument collected students' ethnicity, expected graduation date, and major. Psychometric instruments included the Achievement Goals Questionnaire (Elliot & McGregor, 2001) and the Interpersonal Reactivity Index (Davis, 1983).

Student learning was assessed using pre- and post-tests. The pre-tests and post-tests were developed expressly for the purposes of the project. The tests were iteratively refined between studies in an effort to make the questions more sensitive to differences in learning that occurred over the course of the tutoring sessions. The multiple choice pre- and post-tests for Studies I and II underwent no external evaluation (Appendix A); for Study III, the pre/post-test consisting of free response questions (Appendix B) underwent formal review by a panel of three independent subject matter experts with experience in teaching introductory computer science.[12]

Upon completing the written instruments (including pre-test), students were seated at a computer where they watched a short (3 minutes) instructional video describing RIPPLE, illustrating how to create and run programs, and instructing the students to greet their tutors through the textual dialogue interface immediately after the video ended. The students and tutors interacted remotely while the students planned and implemented the solution to the programming exercise. The choice to have students and tutors interact through remote typed dialogue was to ensure that students and tutors remained anonymous to each other and that all interactions were captured. The alternative formats would have been in-person tutoring or remote spoken tutoring. In-person tutoring permits a wider bandwidth of communication (*e.g.*, facial expressions and gestures), but creating transcripts of video and capturing the nonverbal communication would not have been feasible given the time frame of the projects. Spoken remote tutoring would have restricted the use of nonverbal communication, but might have compromised the anonymity of tutors or students and would also have required extensive time in creating written transcripts for further analysis. The textual dialogue platform was designed to behave similarly to mainstream instant messaging clients likely to be familiar to the participants.

[12] The three experts were Dan Longo and Carol Miller (NC State University), and Dr. Chris Eason (Mercer University).

In Studies I and II, sessions were time-controlled at 50 and 55 minutes, respectively. In Study III, students were permitted to work until completion of the programming exercise or until one hour had elapsed. When the tutoring session reached its conclusion, a paper-based post-survey and post-test with items analogous to the pre-survey and pre-test were administered.

## 3.6 Tutorial Interaction

Tutors and students were not aware of each other's identity. No individual characteristics including gender, ethnicity, age, or level of preparedness were disclosed to the tutor or the student. This restriction was communicated to all participants ahead of time. In the rare event that students inquired as to the tutor's identity, tutors were instructed to redirect the student with a response such as, "Sorry, we're supposed to talk only about the programming exercise." The need for this redirection arose infrequently in the studies, but was necessary to ensure that student and tutor assumptions would be controlled to the fullest extent possible.

In Study I, there were no restrictions placed on the construction of dialogue messages; that is, while one user actively constructed a textual message, the other user was also permitted to construct and send messages. This design choice was made because of its consistency with the interface design of commercial instant messaging platforms familiar to the student population. In these instant messaging platforms, if one user completes a new message (possibly starting a new topic) while the other user is typing a response to the previous topic, the chronological record of dialogue can appear inconsistent with respect to the conversational structure. Human users deal with this phenomenon readily as the textual dialogue unfolds in real time; however, the situation gives rise to analysis challenges because researchers must "untangle" the logs manually before analysis. To address this issue, the dialogue interface was modified for Studies II and III to enforce strict turn taking. When a user was actively constructing an utterance in the textual dialogue interface, the other user was not permitted to construct an utterance.[13] However, the student was permitted to continue working in the problem-solving window regardless of the status of the textual dialogue interface.

## 3.7 Structure of Tutoring Corpora

As students and tutors interacted during the tutoring sessions, all dialogue and problem-solving actions were recorded in a database. The complete record of tutoring therefore involves both textual dialogue and task events. Students generated textual dialogue utterances and task events; tutors generated only textual dialogue utterances. The structure of the interleaved dialogue and task events within the database is shown in Table 1.

In separate rounds of annotation, the dialogue and task events were manually tagged. The individual keystrokes of task events as shown in Table 1 were aggregated to form semantic problem-solving actions, and the dialogue utterances were labeled with dialogue acts. The annotation is described in detail in the next chapter.

[13] An alternative approach, which involved time-stamping a textual dialogue message with the start of typing rather than with the time of sending, was explored. However, this solution would still have allowed topics to be introduced in an asynchronous way, which would have posed significant challenges to automatically modeling the structure of the dialogues. In the free-response area of post-surveys, some students commented that they found the strict turn-taking behavior to be restrictive; therefore, future work should explore alternatives to this restriction.

*Table 1. Excerpt of corpus as structured in database*

| | | | |
|---|---|---|---|
| 2007-03-28 17:47:45 | Tutor | Dialogue | So, basically, if table[correction][i] == 0, we need to draw a small bar |
| 2007-03-28 17:47:50 | Tutor | Dialogue | and if it’s 1 we draw a full bar. |
| 2007-03-28 17:47:53 | Tutor | Dialogue | make any sense? |
| 2007-03-28 17:48:15 | Student | Dialogue | ah,yes, I see. |
| 2007-03-28 17:48:18 | Student | Task | i |
| 2007-03-28 17:48:19 | Student | Task | f |
| 2007-03-28 17:48:19 | Student | Task | |
| 2007-03-28 17:48:20 | Student | Task | |
| 2007-03-28 17:48:22 | Student | Task | |
| 2007-03-28 17:48:23 | Student | Task | {} |
| 2007-03-28 17:48:23 | Student | Dialogue | Typing table is acceptable here? |
| 2007-03-28 17:48:52 | Student | Task | t |
| 2007-03-28 17:48:54 | Student | Task | a |
| 2007-03-28 17:48:54 | Student | Task | b |
| 2007-03-28 17:48:55 | Student | Task | l |
| 2007-03-28 17:48:55 | Student | Task | e |
| 2007-03-28 17:48:55 | Student | Task | [] |
| 2007-03-28 17:48:57 | Tutor | Dialogue | That’s what I would do |

# CHAPTER 4

# Dialogue and Task Annotation

The observational tutoring studies described in Chapter 3 produced three corpora of task-oriented tutorial dialogue. The RIPPLE tutoring environment (Section 3.1) recorded the parallel textual dialogue and task event streams chronologically within a database, fully capturing the interplay between the dialogue utterances and the students' task actions. Each of the studies yielded a corpus with over five thousand dialogue moves and tens of thousands of student programming actions.[14] To perform exploratory analysis on, and subsequently induce models over these corpora, they were annotated with dialogue acts and task/subtask structure. An excerpt from Corpus III that illustrates the interleaved nature of the dialogue and task, along with the type of annotation applied, is depicted in Figure 4. The remainder of this chapter describes the dialogue act and task annotation schemes and processes.[15]

[14] The task events were logged at the keystroke level because each keystroke, including deletions and corrections, were considered relevant problem-solving events that should be considered in manual annotation. Additionally, while a mechanism was available for dividing dialogue utterances into discrete events (specifically, the users chose to press the "Send" button), no such mechanism was available or desirable for dividing student task actions automatically into semantic events.

[15] The dialogue act annotation schemes for the second pilot study and the main study drew heavily on a scheme originally proposed by NC State graduate student Rob Phillips and refined in collaboration with NC State graduate students Michael Wallis and William Lahti and Meredith College undergraduate student Amy Ingram. The task annotation scheme was devised in collaboration with Rob Phillips and Amy Ingram.

| Time Stamp | Dialogue Stream [Dialogue Act Tag] | Task | Task Tag |
|---|---|---|---|
| 2008-04-11 18:23:45 | Student: so do i have to manipulate the array this time? [Q] | | |
| 2008-04-11 18:23:53 | Tutor: This time, we need to do two things [S] | | |
| 2008-04-11 18:24:02 | Tutor: first, we need to create a new array to hold the changed values [S] | | |
| 2008-04-11 18:24:28 | | i | **1-a-i**<br>**BUGGY** |
| 2008-04-11 18:24:28 | | n | |
| 2008-04-11 18:24:28 | | t | |
| 2008-04-11 18:24:28 | | \sp | |
| 2008-04-11 18:24:35 | | \del | |
| 2008-04-11 18:24:36 | | \sp | |
| 2008-04-11 18:24:36 | | d | **1-a-i**<br>**CORRECT** |
| 2008-04-11 18:24:36 | | o | |
| 2008-04-11 18:24:36 | | u | |
| 2008-04-11 18:24:36 | | b | |
| 2008-04-11 18:24:37 | | l | |
| 2008-04-11 18:24:37 | | e | |
| 2008-04-11 18:24:37 | | \sp | |
| 2008-04-11 18:24:39 | | [] | |
| 2008-04-11 18:24:40 | | \sp | |
| 2008-04-11 18:24:42 | | n | **1-a-ii**<br>**CORRECT** |
| 2008-04-11 18:24:42 | | e | |
| 2008-04-11 18:24:42 | | w | |
| 2008-04-11 18:24:43 | | \sp | |
| 2008-04-11 18:24:44 | | \del | |
| 2008-04-11 18:24:45 | | T | |
| 2008-04-11 18:24:46 | | \del | |
| 2008-04-11 18:24:54 | | T | |
| 2008-04-11 18:24:54 | | i | |
| 2008-04-11 18:24:54 | | m | |
| 2008-04-11 18:24:54 | | e | |
| 2008-04-11 18:24:54 | | s | |
| 2008-04-11 18:24:55 | | 3 | |
| 2008-04-11 18:24:57 | | ; | |
| 2008-04-11 18:25:11 | Student: good? [RF] | | |
| 2008-04-11 18:25:14 | Tutor: good so far, yes [PF] | | |
| 2008-04-11 18:25:29 | Student: so now i have to change parts of the times array right? [Q] | | |
| 2008-04-11 18:25:34 | Tutor: not quite [LF] | | |
| 2008-04-11 18:25:57 | Tutor: So, when you create a new object, like a String for example, you'd say something like String s = new String() [S] | | |

*Figure 4. Excerpt from Corpus III illustrating dialogue and task annotations*

## 4.1 Dialogue Act Annotation

Dialogue act annotation involves marking each dialogue move with a tag summarizing the utterance's purpose (*e.g.*, greeting, questioning, answering, disagreeing). For example, in tutorial dialogue, common dialogue acts include asking questions (*e.g.*, "What kind of variable should I use?"), making assessments of knowledge (*e.g.*, "I don't know how to declare an array"), and acknowledging a previous statement (*e.g.*, "Okay"). Because there is no gold standard for annotating tutorial dialogue, the set of dialogue act tags was adapted from annotation schemes from the dialogue analysis literature to capture the salient characteristics of the corpora. Some dialogue acts were taken directly from a set applied in the domain of qualitative physics (Forbes-Riley *et al.*, 2005), while other tags were inspired by a more expansive set of tags created for conversational telephone speech (Stolcke *et al.*, 2000) and another comprehensive, hierarchical scheme for task-oriented dialogue (Core & Allen, 1997).

### *4.1.1 Establishing inter-annotator agreement for tagging schemes*

Inter-annotator agreement studies are used to establish whether an annotation scheme can be consistently applied by more than one human. These agreement studies usually involve a primary annotator tagging an entire corpus, while a subset of the corpus is annotated independently by a second tagger (Carletta, 1996; Litman & Forbes-Riley, 2006; Ohlsson *et al.*, 2007). The agreement between the two taggers is often measured using a Kappa agreement statistic, which adjusts the absolute percentage agreement to account for the agreement that would be expected by chance (Cohen, 1960). The Kappa statistic ranges from -1 (no agreement) to 1 (perfect agreement). Negative values indicate that the annotator agreement was worse than would be expected if both taggers guessed randomly, and positive values indicate that the agreement was better than would be expected with random guessing.

A widely used framework for interpreting the Kappa statistic is depicted in Figure 5 (Landis & Koch, 1977).

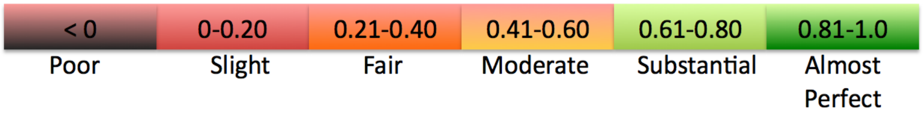


*Figure 5. Kappa statistic interpretation scheme*

For each of the annotation schemes reported in this dissertation, the inter-annotator agreement study proceeded as follows. First, the tagging scheme and its accompanying tagging protocol were developed. Two annotators then applied the scheme collaboratively to a small subset of the data during a first training round until they believed that they would be able to apply the scheme consistently during independent tagging.[16] If necessary, this training round included refinement of the tagging scheme. The two annotators then applied the scheme independently of each other to another small subset of the corpus during a testing round, and their agreement was assessed by calculating the Kappa statistic. If the agreement was not acceptably high (>*0.70* for dialogue act tagging) then a second training and testing round was conducted. Once a training and testing round produced an acceptably high agreement statistic, the primary tagger proceeded with annotating the entire remainder of the corpus and the secondary tagger annotated a previously unseen subset of 10%-30% of the corpus. The final Kappa statistic for agreement was calculated using the subset of the corpus that was tagged independently (not as part of a training and testing round) by both annotators.

[16] The author of this dissertation served as the primary dialogue act annotator for the two pilot corpora, and as the secondary annotator for the main corpus. August Dwight and Rob Phillips performed secondary dialogue act annotation for the pilot corpora, and Amy Ingram performed the primary dialogue act annotation for the main corpus.

### *4.1.2 Corpus I dialogue act annotation*

The first pilot study produced the smallest of the three corpora, consisting of 5,034 dialogue acts: 3,075 tutor turns and 1,959 student turns. The dialogue act tagging scheme drew from two tutoring tag-sets (Forbes-Riley *et al.*, 2005; Marineau *et al.*, 2000) and one annotation scheme for conversational speech (Stolcke *et al.*, 2000). The set of dialogue act tags is depicted in Table 2. The entire corpus was manually annotated by the author of this dissertation, with a second researcher (a Computer Science undergraduate student) annotating a subset of 19%, or 969 utterances selected with random stratified sampling by tutor of complete tutoring sessions.[17] This agreement study found that the inter-annotator agreement was substantial, at $\kappa$=0.75.

### *4.1.3 Corpus II dialogue act annotation*

The second pilot study, conducted in Spring 2007, produced Corpus II consisting of 4,864 dialogue moves: 1,528 student moves and 3,336 tutor moves. The set of dialogue act tags was augmented for the second pilot study with division into two channels: a *cognitive* channel and an *affective/motivational* channel. The cognitive dialogue acts refined the set used for Corpus I by drawing heavily on a comprehensive hierarchical tagging scheme for task-oriented dialogue in a travel domain (Core & Allen, 1997). The affective/motivational dialogue acts drew from extensive studies of the motivational tactics of human tutors (Lepper *et al.*, 1993) and from a tagging scheme for student certainty (Forbes-Riley *et al.*, 2007). The tagging scheme applied to Corpus II is depicted in Table 3.

[17] While the two annotators had different levels of expertise with respect to dialogue analysis, this discrepancy does not hinder the findings regarding the ability of two individuals to reliably apply the tagging scheme. In fact, greater individual differences between annotators strengthen the favorable agreement statistics by suggesting that a particular level of prior expertise is not needed to apply the tagging scheme.

All dialogue moves in the corpus were annotated by a single researcher, a Computer Science graduate student, with a second researcher, the author of this dissertation, annotating 29%, or 1,418 moves, which were selected with random stratified sampling by tutor of complete tutoring sessions. The resulting agreement statistics were $\kappa$=0.76 for the cognitive channel and $\kappa$=0.64 for the motivational/affective channel, both indicating substantial inter-annotator agreement.

*Table 2. Corpus I dialogue act annotation scheme*

| | Act | Description | Examples |
|---|---|---|---|
| Student or Tutor | Task Question (TQ) | Questions about goals, plans, and their ordering. | *Where should we start?*<br>*Should we use an array?* |
| | Concept Question (CQ) | Questions about domain elements, concepts, or facts that are not problem-specific. | *How do I declare an array?*<br>*I don't know how to write a loop.* |
| | Answer (A) | Answers to task or concept questions. | *Yes/No.*<br>*We need to give it an index.* |
| | Acknowledgement (ACK) | Positive acknowledgement of a previous statement. | *Okay.*<br>*Alright.* |
| | Extra-Domain (EX) | Not related to the computer science discussion. | *Sorry.*<br>*Nice working with you.* |
| Student | Request Feedback (RF) | Request for evaluative feedback on completed or proposed problem-solving steps. | *Should I do array[0]=1?*<br>*Does that look good?* |
| | Signal Non-Understanding (SNU) | An indication that a previous statement is unclear. | *Kind of makes sense.*<br>*Not really.* |
| | Statement (S) | Assertion of fact. | *I am going to use a for loop.*<br>*We need to initialize that variable.* |
| Tutor | (Un)Prompted Positive Feedback | Positive feedback. | *Good job.*<br>*Looks great.* |
| | (Un)Prompted Lukewarm Feedback | Partly positive, partly negative feedback. | *You're close.*<br>*The first part is right, but…* |
| | Hint/Advice (HA) | Problem solving or conceptual hint or advice not in answer to a direct question. | *Each digit is represented by 5 bars.*<br>*Let's move on.* |
| | Request to Confirm Understanding (RCU) | Request to confirm the student's understanding. | *Does that make sense?*<br>*Are you with me?* |

*Table 3. Corpus II dialogue act annotation scheme*

| | Act | Description | Examples |
|---|---|---|---|
| Cognitive Channel | Question (Q) | Questions about goals, plans, and domain concepts. | *Where should we start?*<br>*How do I declare an array?* |
| | Evaluating Question (EQ) | Questions that explicitly inquire about the student knowledge state or correctness of problem-solving actions. | *Do you know how to declare an array?*<br>*Is that right?* |
| | Statement (S) | Declarative assertion. | *You need a closing bracket there.*<br>*I am looking for where this method is declared.* |
| | Grounding (G) | Acknowledgement, thanks, other conversational grounding. | *Okay.*<br>*Alright.* |
| | Extra-Domain (EX) | Not related to the computer science discussion. | *Sorry.*<br>*Nice working with you.* |
| | Positive Feedback (PF) | Unmitigated positive feedback regarding problem-solving action or student knowledge state. | *Yes, I know how to declare an array.*<br>*That is right.* |
| | Lukewarm Feedback (LF) | Partly positive, partly negative feedback. | *Sort of.*<br>*Almost.* |
| | Negative Feedback (NF) | Totally negative feedback. | *No.*<br>*Actually, that won't work.* |
| Motivational/Affective Channel | Confusion (C) | Indicates disorientation beyond that indicated by negative feedback. | *I have no idea what to do.*<br>*I'm lost.* |
| | Frustration (F) | Explicit expression of frustration. | *Grr!*<br>*This is so frustrating.* |
| | Excitement (E) | Explicit expression of excitement. | *Sweet!*<br>*Cool!* |
| | Praise (P) | Emphasizes a student's success. Goes beyond positive feedback. | *Great job on that part!*<br>*That's perfect.* |
| | Reassurance (R) | Intended to minimize a student's failure. | *That part was hard.*<br>*Don't worry about it.* |
| | Other Emotion (O) | Affective or motivational utterance for which there is no pre-defined tag. | *Haha.*<br>*I'm sorry.* |

### *4.1.4 Corpus III (main corpus) dialogue act annotation*

For Study III, lessons learned in Studies I and II were used to create a modified dialogue act tagging scheme designed to capture the aspects of the dialogue that were deemed most useful from a dialogue system implementation perspective. For example, the low occurrence of affective/motivational dialogue moves in Corpus II and the even lower occurrence of these moves in Corpus III (based on preliminary manual analysis for the presence of affective/motivational content) led to a cognitive-only dialogue act tagging scheme for this corpus. This tagging scheme is depicted in Table 4. Inter-rater agreement for this tagging scheme on 10% of the corpus was $\kappa$=0.80, indicating substantial agreement between the primary annotator, an undergraduate Computer Science student, and the secondary annotator, the author of this dissertation.

## 4.2 Task Annotation

The previous section focused on textual dialogue utterances sent between tutors and students. In addition to dialogue, the tutoring sessions also involved students writing computer programming code in Java to solve an introductory programming exercise, the *task*. The students' programming keystrokes, including typing or deleting elements of their computer program along with other actions as described in Section 3.1, comprise the *task action event stream*. In the main corpus, the task event stream included 97,509 keystroke-level student task events. The keystroke-level task events were manually aggregated into 3,793 task/subtask event clusters that were annotated for subtask structure and then annotated for correctness as described below.

The task annotation scheme is hierarchical and reflects the nested nature of the subtasks with the programming task (this programming exercise is provided in Appendix B). A subset of the task annotation scheme is depicted in Figure 6. Each group of task events that occurred between dialogue utterances was tagged for all its constituent subtask labels.

*Table 4. Corpus III dialogue act annotation scheme*

| Dialogue Act | Description | Student Relative Freq. | Tutor Relative Freq. |
|---|---|---|---|
| ASSESSING QUESTION (AQ) | Request for feedback on task or conceptual utterance. | . 204 | . 107 |
| EXTRA-DOMAIN (EX) | Asides not relevant to the tutoring task. | . 076 | . 040 |
| GROUNDING (G) | Acknowledgement/thanks. | . 261 | . 057 |
| LUKEWARM ELABORATED FEEDBACK (LCF) | Lukewarm assessment with explanation. | . 011 | . 031 |
| LUKEWARM FEEDBACK (LF) | Lukewarm assessment of task action or conceptual utterance. | . 019 | . 026 |
| NEGATIVE ELABORATED FEEDBACK (NEF) | Negative assessment with explanation. | . 014 | . 097 |
| NEGATIVE FEEDBACK (NF) | Negative assessment of task action or conceptual utterance. | . 045 | . 017 |
| POSITIVE ELABORATED FEEDBACK (PEF) | Positive assessment with explanation. | . 024 | . 028 |
| POSITIVE FEEDBACK (PF) | Positive assessment of task action or conceptual utterance. | . 093 | . 158 |
| QUESTION (Q) | Task or conceptual question. | . 094 | . 027 |
| STATEMENT (S) | Task or conceptual assertion. | . 160 | . 411 |

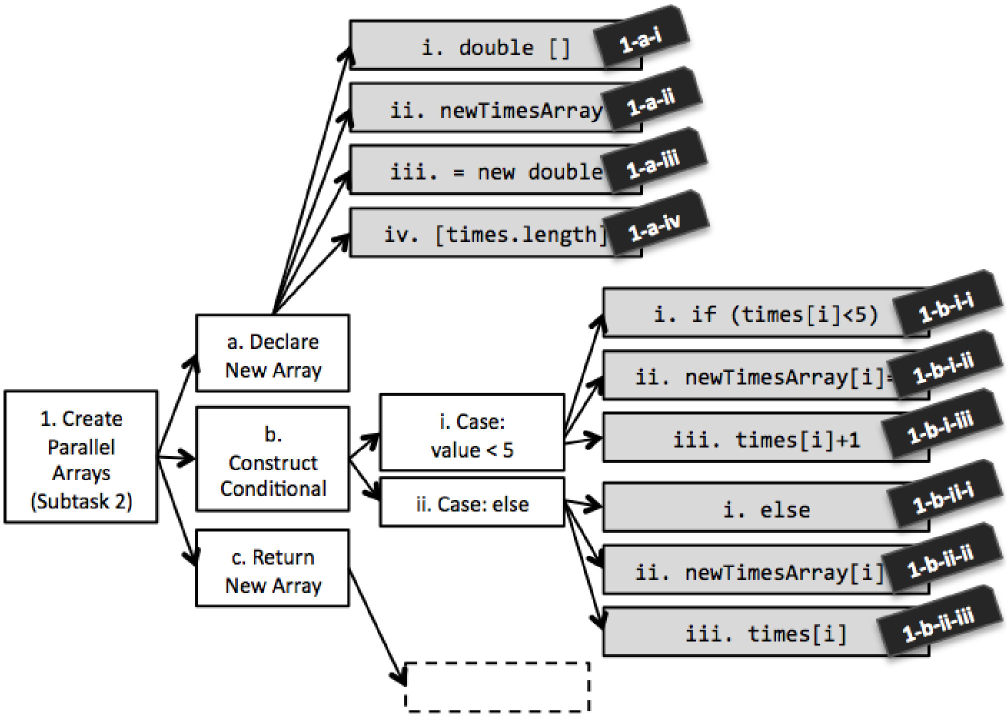


*Figure 6. Portion of hierarchical task annotation scheme*

The human annotator aggregated the students' raw task (programming) keystrokes and tagged the task/subtask hierarchy for each cluster. A second annotator tagged 20% of the corpus in a reliability study for which one-to-one subtask identification was not enforced (giving the annotators maximum flexibility to apply the tags). All unmatched subtask tags were treated as disagreements. The resulting unweighted Kappa statistic at the leaves was $\kappa$=0.58, indicating moderate agreement. However, we also observe that the sequential nature of the subtasks within the larger task produces an ordinal relationship between subtasks. For example, in Figure 6, the "distance" between subtasks *1-a* and *1-b* can be thought of as "less than" the distance between subtasks *1-a* vs. *3-d* because those subtasks are farther from each other within the larger task. The weighted Kappa statistic (Artstein & Poesio, 2008) takes into account such an ordinal relationship and its implicit distance function. The weighted Kappa is $\kappa_{weighted}$=0.86, which indicates acceptable inter-rater reliability on the task/subtask annotation.

Along with its tag for hierarchical subtask structure, each task event was also judged for correctness according to the requirements of the task. Correctness categories are shown in Table 5. The agreement statistic for correctness was calculated for task events on which the two annotators agreed on the subtask tag. The resulting statistic for correctness was $\kappa$=0.80, indicating substantial inter-annotator agreement.

*Table 5. Task correctness annotation scheme*

| Task Correctness Category | Description |
|---|---|
| CORRECT | Fully satisfying the requirements of the learning task. Does not require tutorial remediation. |
| BUGGY | Violating the requirements of the learning task. Usually requires tutorial remediation. |
| INCOMPLETE | Not violating, but not yet fully satisfying, the requirements of the learning task. May require tutorial remediation. |
| DISPREFERRED | Technically satisfying the requirements of the learning task, but not adhering to its pedagogical intentions. Tutors often choose to remediate. |

## 4.3 Other Types of Annotation

The dialogue act and task annotation described in Sections 4.1 and 4.2 serve as the basis for the modeling contributions of this dissertation, which involve machine-learned models for both user dialogue act classification and tutorial move selection. However, during the exploratory phases of the work, other annotations were applied either automatically or manually. No inter-annotator agreement studies were conducted for these annotation projects.

### *4.3.1 Automatic heuristic annotation for student problem-solving correctness*

An automatic heuristic annotation for correctness was applied to Corpus II and was used in the exploratory analysis reported in Section 5.2. In this annotation, events were automatically tagged using the following rule:

i) if a problem-solving action was a programming keystroke that survived until the end of the session, then this event was tagged promising to indicate it was probably correct;

ii) if a problem-solving act was a programming keystroke that did not survive until the end of the session, then the problem-solving act was tagged questionable.[18]

This heuristic is based on the process used in this work, *i.e.*, in this tutoring context, students solved the problem in a linear fashion and tutors did not allow students to proceed past a step that had incorrect code in place. Finally, periods of consecutive scrolling were also marked questionable because in a problem where the entire solution fits on one printed page, scrolling was usually conducted in irrelevant source files included to support graphical output of the programming exercise (See Appendix A). Because the student's solution did not interface directly with these source files, scrolling through them was generally not a productive problem-solving step. This heuristic may not hold in all cases of student scrolling through peripheral files, and as such, is one limitation of the preliminary automatic task annotation. This limitation does not extend to the manual annotation employed for the main corpus.

[18] The automatic problem-solving action tagger was implemented collaboratively with Michael Wallis.

### *4.3.2 Annotation for initiative*

While dialogue act annotation involves marking a corpus at the level of dialogue turns, another useful type of annotation entails marking the higher-level structure of the dialogue with regard to which collaborator holds the *initiative* at a given point, according to the following criteria. We distinguish STUDENT-INITIATIVE and TUTOR-INITIATIVE modes.

In STUDENT-INITIATIVE mode the student maintains control and direction over the problem-solving effort. STUDENT-INITIATIVE mode is characterized by the following activities:

- The student states his/her plan and (optionally) asks the tutor for feedback,
- The student reads the problem description or constructs a portion of the actual solution independently, as indicated by no dialogue exchanged while the student is conducting these problem-solving activities,[19]
- The student asks content-based questions (*e.g.*, “I should start this index at 0, right?”) as opposed to content-free questions (*e.g.*, “What do I do now?”).

In TUTOR-INITIATIVE mode, the tutor directs the problem-solving effort. Because the user interface does not allow tutors to edit the students’ solutions, TUTOR-INITIATIVE mode does not involve the tutor actively constructing the problem solution. However, the tutor often used the textual dialogue interface to actively guide and direct the student to take very specific problem-solving actions. TUTOR-INITIATIVE mode includes the following activities:

- The tutor offers unsolicited advice or correction
- The tutor lectures on a concept

[19] Researchers were only present periodically during the studies, and therefore, during times that the student appears to be reading the problem description, he or she may actually be engaged in off-task or other behavior. This limitation should be addressed in future work by use of technology such as eye-trackers to more clearly identify the focus of students’ attention.

- The tutor explicitly suggests the next step in problem solving, or
- The tutor poses questions to the student.

To illustrate the initiative modes, two excerpts are presented (Figure 7). The first excerpt illustrates STUDENT-INITIATIVE mode. In this excerpt, the student asks a content-based question indicating he/she knows the problem lies in a *return* statement. The tutor provides an answer that the student acknowledges. Finally, the student spends five uninterrupted minutes coding part of the problem solution. Lengthy periods of independent student work are common in STUDENT-INITIATIVE mode. The second excerpt illustrates TUTOR-INITIATIVE mode. In this excerpt, the tutor gives unsolicited advice and asks questions of the student. The student spends a brief time repairing the problem solution, and the tutor once more provides unsolicited feedback. As illustrated in this excerpt, brief periods of student work interspersed with frequent dialogue are common in TUTOR-INITIATIVE mode.

Tags at the level of initiative can span many individual dialogue acts. Since the corpora consist of dialogue turns interleaved chronologically with student problem-solving actions, tags for initiative can also span contiguous sections of textual dialogue and student problem-solving. The two tags of STUDENT-INITIATIVE and TUTOR-INITIATIVE were manually applied in an exploratory study that did not feature inter-rater reliability tagging.

STUDENT-INITIATIVE Mode

**Student:** What am I not typing right in the *return* statement?
**Tutor:** You only need to return the identifier.
**Tutor:** In other words, you just need to return *newTimes*.
**Student:** Ok.

*[Student works independently for 5 minutes. ]*

TUTOR-INITIATIVE Mode

**Tutor:** Hmm, that doesn't look quite right.
**Tutor:** Do you see the projected array output?
**Student:** Yes.
**Tutor:** It looks like it's only getting the first value.
**Tutor:** So your loop must be stopping before it's done with its work.
**Tutor:** Do you see what might be causing that?

*[Tutor-led conversation continues. ]*

**Tutor:** But it's coming out 1. 0 instead of 4. 3.
**Tutor:** Anything else look wrong on the graph, compared with the instructions?
**Student:** The second bar is not right.
**Tutor:** I think fixing the length might be the only thing you need to change.

*[Student works for 10 seconds. ]*

**Tutor:** Much better.
**Student:** Yeah!!

*Figure 7. Excerpt from STUDENT-INITIATIVE and TUTOR-INITIATIVE modes*

# CHAPTER 5

# Exploratory Analysis of Tutorial Dialogue Corpora

To work toward the goal of creating a highly effective, data-driven tutorial dialogue system, it was essential to explore how tutorial phenomena from the literature manifested within the domain of introductory computer programming. This chapter describes the exploratory analyses that were conducted prior to utilizing the corpora to machine learn statistical dialogue act classifiers and predictive models of tutor moves. In addition to informing subsequent modeling approaches, these exploratory analyses hold intrinsic value for illuminating pedagogical phenomena in task-oriented tutoring with respect to cognitive, motivational, and affective outcomes, and for furthering the state of knowledge regarding how students come to understand computing.

## 5.1 Tutorial Adaptation to Student Characteristics

Results from Corpus I, collected during the first pilot study, suggest that tutors adapt in specific ways to student characteristics (Boyer, Vouk *et al.*, 2007). The student characteristics considered in this section include 1) incoming knowledge level as measured by pre-test, 2) self-efficacy as measured on the pre-survey (Appendix A), and 3) gender. Although no student characteristics were explicitly revealed to the tutors, the tutorial dialogues with low pre-test students differ from those of students with high pre-test, and significant differences in dialogue profile also emerge between low and high self-efficacy students and between students of different genders.

Overall, the tutoring sessions in Study I were effective: on average, students scored 13 percentage points higher on the post-test than the pre-test. This average learning gain is statistically significant ($p$<0.0001 using a $t$-test with 34 DF, SD=0.12) and the effect size is 1.08.[20] The self-efficacy measure was obtained from a pre-survey item in which students were asked to rate how certain they are, on a scale of 0-100, that they could complete a simple programming exercise on their own. The items used to measure computer science self-efficacy are based on Bandura's widely utilized domain-specific self-efficacy scale (Bandura, 1997).

For each student dialogue session, the relative frequency of each dialogue act was computed as the ratio of the number of occurrences of that dialogue act to the total number of dialogue acts in the session. The relative frequency of each dialogue act was then computed for the following three partitions of students: high pre-test and low pre-test students, high self-efficacy and low self-efficacy students, and female and male students. Figure 8 presents two sample annotated dialogue excerpts from the corpus.

In Dialogue Excerpt A, the tutor interacts with a low pre-test student, Student A, whose pre-test score was well below the median. The structure of Dialogue A illustrates many features commonly seen with low pre-test students. Student A responds to the tutor's first question with an unsure answer. After receiving a hint from the tutor, Student A types a proposed problem-solving step into the dialogue interface before implementing it in the problem-solving environment. This pattern of receiving a hint and then requesting feedback repeats. It appears that Student A, who also happens to be in the low self-efficacy group, in addition to being in the low pre-test group, seeks to establish confirmation of his proposed plan before proceeding to implementation.

In contrast to Dialogue A, Dialogue B illustrates some common characteristics of dialogues with high pre-test students. Student B asks a specific question and after receiving

[20] This effect size is computed as difference in pre-test mean and post-test mean divided by standard deviation of pre-test.

tutorial advice begins problem-solving work. Student B does not type the proposed problem-solving step into the dialogue interface to obtain feedback from the tutor; rather, the student proceeds directly to implementation.

Dialogue Excerpt A

**Tutor:** Do you know how to do that? [TQ]
**Student:** Not really. [A]
**Tutor:** Well we first need a new String that will hold zipCode’s string value. [HA]
**Student:** So String z = zipCode? [RF]
**Tutor:** Close. [PLF]
**Tutor:** Then you can set that string equal to “”+zipcode. [HA]
**Student:** Ok so String z = “”+zipCode [RF]
**Tutor:** Yeah. [PPF]
**Student:** Then what? [TQ]
**Tutor:** Ok, so now we need somewhere to keep the individual digits. [A]

Dialogue Excerpt B

**Student:** So I need an if for each digit? [TQ]
**Tutor:** One if should suffice, since it will be called in each iteration. [A]
**Tutor:** You just need to know which element to reference. [A]
**Tutor:** This would be done in the inner loop. [HA]
Student: Ok. [ACK]
*(Student works for 2.5 minutes. )*
**Tutor:** You’ve got the right idea. [UPF]
**Student:** Yeah, I had programmer’s block. [EX]
*(Student works for 3 minutes. )*
**Tutor:** Perfect. [UPF]

*Figure 8. Excerpts from Corpus I illustrating low vs. high pre-test student dialogue*

To determine whether inter-group differences in means were significant, *t*-tests were performed. The relative frequencies and statistically significant differences (bold) are given in Figure 9. It should be noted that the partitions are not independent; for example, high pre-test students were more often in the high self-efficacy group. However, sample sizes do not support multi-level analysis, so the following analysis examines each learner characteristic individually.

| | | Relative Frequencies | | | | | |
|---|---|---|---|---|---|---|---|
| | | Pre-test Performance | | Self-efficacy Level | | Gender | |
| | Dialogue Act | $n_{high}$=17, $n_{low}$=18 | | $n_{high}$=19, $n_{low}$=16 | | $n_{female}$=7, $n_{male}$=28 | |
| Tutor and Student | Student | High | 11.2% | High | 11.1% | Female | 12.8% |
| | Task Question (TQ) | Low | 12.1% | Low | 12.3% | Male | 11.4% |
| | Tutor | High | 6.7% | High | 6.2% | Female | 6.5% |
| | Task Question (TQ) | Low | 6.4% | Low | 6.9% | Male | 6.5% |
| | Student | High | 0.7% | High | 0.9% | Female | 0.6% |
| | Concept Question (CQ) | Low | 0.9% | Low | 0.6% | Male | 0.8% |
| | Tutor | High | 1.1% | High | 0.8% | Female | 1.1% |
| | Concept Question (CQ) | Low | 1.0% | Low | 1.3% | Male | 1.0% |
| | Student Answer (A) | High | 6.9% | High | 5.8% | Female | 6.7% |
| | | Low | 5.7% | Low | 6.8% | Male | 6.2% |
| | Tutor Answer (A) | High | 13.0% | High | 13.2% | Female | 14.8% |
| | | Low | 14.5% | Low | 14.4% | Male | 13.5% |
| | Student | **High** | **10.3%** | High | 9.1% | Female | 8.3% |
| | Acknowledgement (ACK) | **Low** | **6.3%** | Low | 7.3% | Male | 8.2% |
| | Tutor | High | 2.3% | **High** | **2.5%** | Female | 1.2% |
| | Acknowledgement (ACK) | Low | 1.5% | **Low** | **1.2%** | Male | 2.1% |
| | Student | High | 8.8% | High | 8.6% | Female | 6.0% |
| | Extra Domain (EX) | Low | 6.1% | Low | 6.0% | Male | 7.8% |
| | Tutor | **High** | **6.9%** | High | 8.5% | Female | 8.4% |
| | Extra Domain (EX) | **Low** | **9.4%** | Low | 7.8% | Male | 8.2% |
| Student | Request Feedback (RF) | **High** | **1.2%** | High | 1.6% | **Female** | **2.8%** |
| | | **Low** | **2.2%** | Low | 2.0% | **Male** | **1.5%** |
| | Statement of Non- | High | 0.1% | High | 0.2% | Female | 0.3% |
| | Understanding (SNU) | Low | 0.3% | Low | 0.2% | Male | 0.1% |
| | Statement (S) | **High** | **3.5%** | **High** | **3.5%** | **Female** | **1.2%** |
| | | **Low** | **1.6%** | **Low** | **1.4%** | **Male** | **2.8%** |
| Tutor | Unprompted Positive | High | 4.6% | High | 4.5% | Female | 3.8% |
| | Feedback (UPF) | Low | 4.3% | Low | 4.4% | Male | 4.6% |
| | Prompted Positive | **High** | **0.5%** | High | 0.9% | **Female** | **1.7%** |
| | Feedback (PPF) | **Low** | **1.5%** | Low | 1.3% | **Male** | **0.9%** |
| | Unprompted Lukewarm | High | 0.8% | High | 0.7% | Female | 0.7% |
| | Feedback (ULF) | Low | 0.6% | Low | 0.6% | Male | 0.7% |
| | Prompted Lukewarm | **High** | **0.1%** | High | 0.2% | Female | 0.6% |
| | Feedback (PLF) | **Low** | **0.6%** | Low | 0.5% | Male | 0.3% |
| | Unprompted Negative | High | 0.6% | High | 0.7% | Female | 0.5% |
| | Feedback (UNF) | Low | 0.5% | Low | 0.4% | Male | 0.5% |
| | Prompted Negative | **High** | **0.1%** | **High** | **0.0%** | Female | 0.3% |
| | Feedback (PNF) | **Low** | **0.3%** | **Low** | **0.4%** | Male | 0.1% |
| | Hint/Advice (HA) | High | 20.5% | High | 20.8% | Female | 20.5% |
| | | Low | 23.5% | Low | 23.5% | Male | 22.4% |
| | Request Confirmation | **High** | **0.1%** | High | 0.3% | **Female** | **1.4%** |
| | of Understanding (RCU) | **Low** | **0.9%** | Low | 0.8% | **Male** | **0.3%** |

*Figure 9. Dialogue profiles with statistically significant differences (p<0.05) in bold*

Students were divided into low pre-test and high pre-test groups, and into low self-efficacy and high self-efficacy groups, based on whether the student's score fell below or above the median incoming score of all participants on the pre-test or pre-survey, respectively. Analyses yielded the following findings (Boyer, Vouk *et al.*, 2007; Boyer, Phillips *et al.*, 2008b; Boyer, Phillips, Wallis *et al.*, 2009b) (full quantitative results are shown in Figure 9):

- High pre-test students made more acknowledgements, requested feedback less often, and made more declarative statements than low pre-test students.
- Tutors paired with low pre-test students made more extra-domain statements, gave more prompted feedback, and made more requests for confirmation of understanding than tutors paired with high pre-test students.
- Students in the high self-efficacy group made more declarative statements, or assertions, than students in the low self-efficacy group.
- Tutors paired with low self-efficacy students gave more negative feedback and made fewer acknowledgements than tutors paired with high self-efficacy students.
- Women made more requests for feedback and fewer declarative statements than men.
- Tutors paired with women gave more positive feedback and made more requests to confirm understanding than tutors paired with men.

The results provide support for Hypothesis 1.1, which stated that "Because human tutors adapt their behavior based on student characteristics including skill level, self-efficacy, and gender, *the distribution of dialogue acts within tutoring sessions will be significantly different when compared based on these characteristics*." Some tutor and student dialogue acts did occur with significantly different frequencies across those student characteristics. For example, tutors more often engaged in extra-domain conversation, provided additional feedback, and more frequently engaged in discussions to gauge students' level of understanding when conversing with low pre-test or low self-efficacy students. These same groups of students tended to request more feedback, make fewer declarative statements, and make fewer acknowledgements.

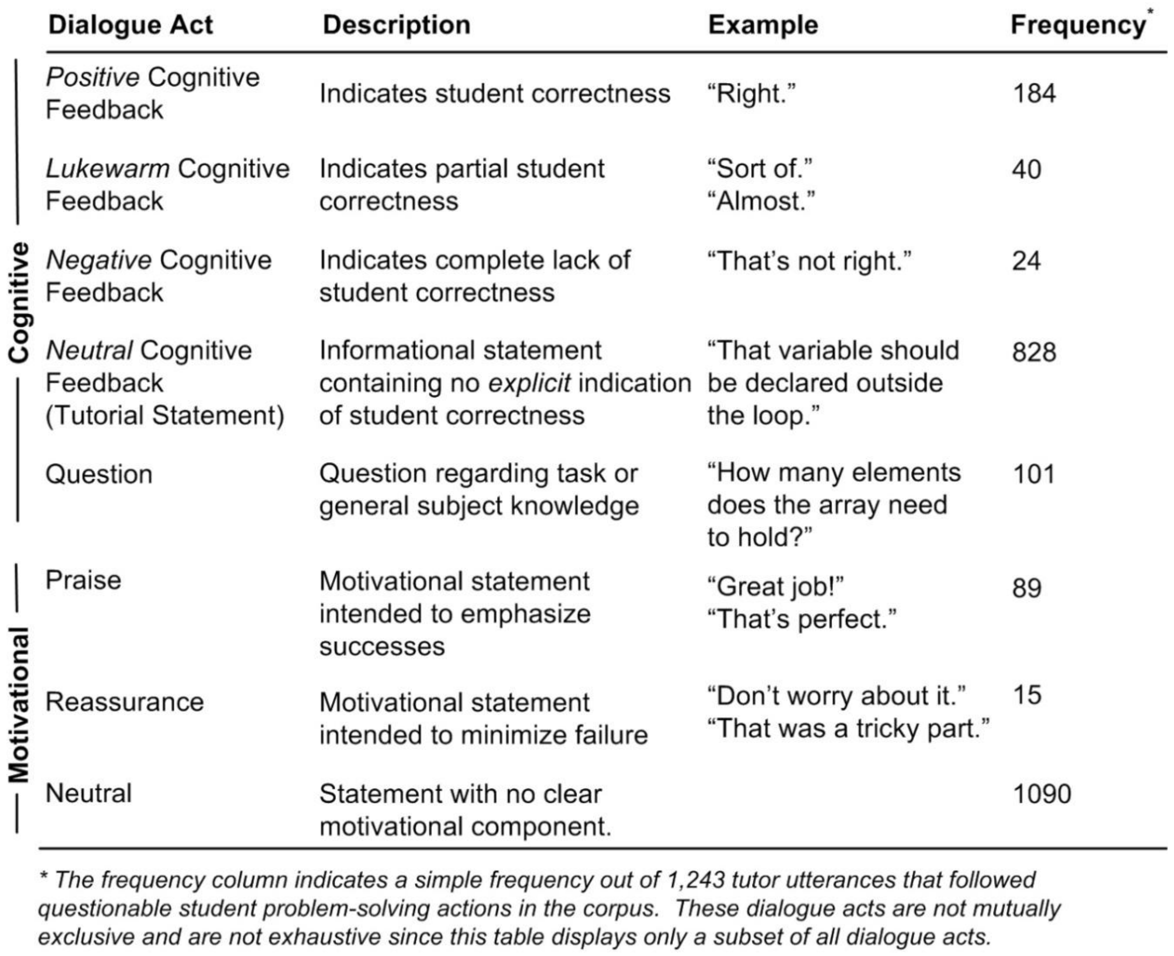
## 5.2 Impact of Corrective Feedback

Although the learning gain results, as measured by difference in post-test and pre-test, indicate that the tutoring sessions overall were effective at increasing students' knowledge about the computing constructs involved in the programming exercise, the findings from the first pilot study led naturally to the question of which tutorial adaptations were more or less effective from either a cognitive or a motivational perspective. However, sample size, along with limitations on the learning gain instruments of Study I, did not allow such fine-grained analysis. The second pilot study was refined to address these issues.[21]

The second pilot study generated Corpus II, which was tagged with a dialogue act annotation scheme that included a cognitive and a motivational/affective channel (Section 4.1). The primary exploratory analysis conducted on this corpus examined the impact of certain cognitive and motivational corrective strategies by focusing on dialogue acts utilized by tutors immediately following plausibly incorrect student problem-solving action (according to the heuristic correctness annotation described in Section 4.3). The motivational strategies of *praise* and *reassurance* were compared with several types of cognitive feedback to identify relationships with student cognitive and motivational outcomes. Of the 3,336 tutor utterances, 1,243 occurred directly after a student problem-solving action that had been

[21] Recall that all three studies, including the two pilot studies and the main study, were exploratory in nature, not confirmatory. There were no control groups involved. Future work should include control groups to gauge the effectiveness of the tutoring treatment compared to another meaningful condition such as classroom instruction or reading edited texts (VanLehn *et al.*, 2007).

tagged questionable. Because these utterances immediately followed student action that presumably warranted correction, this subset of tutorial utterances served as the basis for comparing corrective tutorial strategies. The frequency of these tutorial acts of interest is given in Figure 10. The dialogue acts depicted in Figure 10 constitute the second component of bigrams whose first component is the incorrect student action.



| | Dialogue Act | Description | Example | Frequency* |
|---|---|---|---|---|
| Cognitive | *Positive* Cognitive Feedback | Indicates student correctness | "Right." | 184 |
| | *Lukewarm* Cognitive Feedback | Indicates partial student correctness | "Sort of."<br>"Almost." | 40 |
| | *Negative* Cognitive Feedback | Indicates complete lack of student correctness | "That's not right." | 24 |
| | *Neutral* Cognitive Feedback (Tutorial Statement) | Informational statement containing no *explicit* indication of student correctness | "That variable should be declared outside the loop." | 828 |
| | Question | Question regarding task or general subject knowledge | "How many elements does the array need to hold?" | 101 |
| Motivational | Praise | Motivational statement intended to emphasize successes | "Great job!"<br>"That's perfect." | 89 |
| | Reassurance | Motivational statement intended to minimize failure | "Don't worry about it."<br>"That was a tricky part." | 15 |
| | Neutral | Statement with no clear motivational component. | | 1090 |

** The frequency column indicates a simple frequency out of 1,243 tutor utterances that followed questionable student problem-solving actions in the corpus. These dialogue acts are not mutually exclusive and are not exhaustive since this table displays only a subset of all dialogue acts.*

*Figure 10. Dialogue acts that follow incorrect student task action*

Overall, according to difference in pre-test and post-test, the forty-three tutoring sessions were effective at increasing students' knowledge of computing constructs related to the programming task. The mean learning gain from pre-test to post-test was 5.9%, a statistically significant difference ($p$=0.038, $t$-test with pooled variance, 42 DF, SD=0.18), though displaying a modest effect size of 0.33. For this study, cognitive benefit and motivational benefit were considered. Students rated their own self-efficacy regarding the subject matter significantly higher, 12.1% on average, after the tutoring session than before ($p$=0.0021, $t$-test with pooled variance, 42 DF, SD=0.24) with effect size 0.5.

As in the first pilot study, the student outcomes of learning gain and self-efficacy gain for each participant were partitioned into binary categories of High and Low based on the median learning gain across all participants of 10.0%. Multiple logistic regression was then applied, with the outcome category (High learning gain vs. Low learning gain and High self-efficacy gain vs. Low self-efficacy gain) as the predicted variable, to determine whether a relationship existed between corrective tutorial strategy and student outcomes. In these logistic regression models, the number of occurrences of particular tutoring strategies were treated as predictors; in addition, the students' incoming values of pre-test score and initial self-efficacy rating were treated as predictors to control for their effects on the outcomes of the session. All of the findings in the remainder of this section are related to Hypothesis 1.2, which states that the frequency of some tutorial moves will be positively correlated with motivational outcomes and negatively correlated with learning.

### *5.2.1 Presence of tutorial encouragement*

Two categories of corrective tutorial utterances are first considered: those *with* and those *without* explicit encouragement (*i.e.*, praise or reassurance). Both these categories may, but need not, contain cognitive feedback components. We restrict the analysis to only cognitive feedback in the next subsection, and later omit all such feedback to consider standalone tutorial encouragement. A logistic regression model quantified the significant relationships between tutorial encouragement and learning gain, revealing that after accounting for the effects of pre-test score and incoming self-efficacy rating (both of which were significant in the model with $p<0.001$), observations containing tutorial encouragement were 56%[22] less likely to result in high learning gain than observations without explicit tutorial encouragement ($p=0.001$). On the other hand, tutorial encouragement was weakly linked to self-efficacy gains, with explicit encouragement being 57% more likely to result in high self-efficacy gain than tutorial responses that had no explicit praise or reassurance ($p=0.054$). These models suggest that *the presence of tutorial encouragement in response to questionable student problem-solving action is weakly linked to self-efficacy gain but may be associated with lower learning gain.*

### *5.2.2 Adding encouragement to positive feedback*

Corrective tutorial acts that were tagged as cognitive feedback were compared for relative impact of those *with* and *without* explicit tutorial praise or reassurance. Because the co-occurrence of cognitive feedback with reassurance was very low ($n=2$), we omit this strategy from consideration and compare the two strategies of *purely cognitive feedback* and *cognitive feedback plus praise.* A logistic regression model built as described above revealed that observations in which the tutor used cognitive feedback plus praise were associated with 40% lower odds of high learning gain than observations in which the tutor used purely cognitive feedback. No significant impact was observed on self-efficacy gain. These results suggest that in response to questionable student problem-solving action, *to achieve learning*

[22] This value and its counterparts throughout this section represent logistic regression point estimates of odds ratio (analogous to the regression coefficient in multiple linear regression). The accompanying $p$-value indicates the level at which the predictor variable was significant in the model.

*gains*, *purely cognitive feedback is preferred over cognitive feedback plus praise, while self-efficacy gain does not appear to be impacted either way*.

### *5.2.3 Standalone tutorial encouragement*

In this corpus, tutorial encouragement is sometimes encountered with no cognitive feedback component; that is, the tutorial utterance is in no way aimed at giving substantive task-related feedback, but instead, is aimed at the student's motivational or affective state through explicit praise or reassurance. We now consider this tutorial strategy of *standalone motivational acts*. Unlike the previous results that had a consistent (or no statistically significant) impact on student sub-groups, and were therefore reported only for the general student population, purely motivational statements appear to affect low and high self-efficacy students differently. A separate logistic regression was run for the low initial self-efficacy and high initial self-efficacy student groups. Among students with low incoming self-efficacy, observations in which the tutor employed a standalone motivational act were 300% as likely to be in the higher self-efficacy gain group as observations in which the tutor employed a purely cognitive statement or a cognitive statement combined with encouragement ($p$=0.039). In contrast, among students with high initial self-efficacy, a purely motivational tactic resulted in 90% lower likelihood of being in the high self-efficacy gain group. Standalone motivational acts showed no statistically different impact on learning gain compared to other tutorial acts ($p$=0.268). This relationship held for both the low self-efficacy ($p$=0.216) and high self-efficacy subgroups ($p$=0.441) with regard to impact on learning gain. These results suggest that *standalone praise or reassurance may be useful for increasing self-efficacy gain among low initial self-efficacy students, but may be associated with lower self-efficacy gain in high initial self-efficacy students*. In addition, *standalone praise or reassurance may not be associated with higher learning gains*.

### *5.2.4 Superiority of positive cognitive feedback*

We have seen evidence thus far that explicit tutor encouragement in the form of praise or reassurance has mixed effects on learning and self-efficacy gains. We now consider the class of purely cognitive tutorial moves, *i.e.*, all tutorial acts that have no explicit encouragement attached. The strategies under consideration here are *positive*, *lukewarm*, *negative*, and *neutral* cognitive feedback plus *tutorial questions*. Because positive cognitive feedback related similarly to each of the other types of cognitive moves, we forego pairwise comparisons and instead contrast positive cognitive feedback against the group of all other purely cognitive strategies. Chi-square analysis reveals positive cognitive feedback had a significantly different impact on self-efficacy than other strategies (*p*=0.0028). A logistic regression refined the relationship, revealing positive feedback resulted in 190% increased odds of high student self-efficacy gain compared to the other cognitive strategies (*p*=0.0057). Positive cognitive feedback did not differ significantly from other types of cognitive strategies in a Chi-square comparison with respect to learning gains (*p*=0.390). This result suggests that when dealing with questionable student problem-solving action, *positive cognitive feedback is preferable to other types of cognitive feedback for eliciting self-efficacy gains, but this type of feedback is not found to be significantly associated with higher or lower learning gains.*

### *5.2.5 Discussion*

This section has examined corrective feedback as defined by bigrams of plausibly incorrect student actions and the subsequent tutorial move. The results provide evidence regarding Hypothesis 1.2, which states that the frequency of some tutorial moves will be positively correlated with motivational outcomes and negatively correlated with learning. The presence of explicit tutorial encouragement was associated with the outcomes in this way, highlighting the tension that sometimes exists between cognitive and affective goals in tutoring (Section 2.1.2).

## 5.3 Tutor Initiative

The two pilot studies utilized a number of tutors; therefore, no single tutor interacted with a large number of student participants. The third study was conducted using only the two tutors who had been identified as highly effective in previous studies. In addition, these two tutors had been observed in previous studies to have very different tutoring approaches. The primary exploratory analysis of Corpus III examines the differences between these tutors with respect to the conversational initiative, indicating which participant was directing the dialogue at a given moment (Walker & Whittaker, 1990).

Because the level of student autonomy is thought to support increased motivation (Dickinson, 1995), it was hypothesized that different levels of tutor initiative (that is, taking control of the problem solving and dialogue)[3] might be associated with differing student outcomes. Results presented in this section explore whether there was a difference in learning gains (measured by post-test score minus pre-test score) or self-efficacy gains (as measured by self-efficacy post-survey score minus pre-survey score) between groups of students paired with tutors who naturally took significantly different levels of initiative. There were sixty-one tutoring sessions distributed approximately equally between the tutors. From these sessions, fifteen were randomly selected for each tutor yielding a total of thirty sessions to be annotated for initiative using the annotation scheme described in Section 4. 3.

Each STUDENT-INITIATIVE and TUTOR-INITIATIVE tag was associated with a period of time over which that instance of the tutoring mode occurred. The sum (in minutes) of all

[3] The software permitted a synchronized view of the student's problem solving, but tutors were not able to edit the solution. Nonetheless, tutor initiative was present when the student took only programming actions that the tutor had specifically directed, as described in Section 4.3.

TUTOR-INITIATIVE periods in a given tutorial session divided by the total time elapsed during the session yielded the percentage of the tutoring session that was spent in TUTOR-INITIATIVE mode. One tutor took initiative 55% of the time on average, while the other tutor took initiative 73% of the time. We refer to the tutor who took initiative 55% of the time as the *moderate* tutor, while the other tutor is referred to as the *proactive* tutor. This difference in approach is significantly different (*p*=0.029, *t*-test with pooled variances, 28 DF, SD=0.21). One possible explanation for this difference could be that, despite the randomized assignment of students to tutors, the moderate tutor may have been assigned a group of students with a different level of preparedness than the proactive tutor. However, analysis of pre-test scores do not suggest that this confounding factor was present. Average student pre-test scores were 79.5% for the moderate tutor and 78.9% for the proactive tutor, yielding no evidence of a difference in student preparedness between the two treatment groups for the subset of students considered in the initiative annotation (*p*=0.764, *t*-test with pooled variances, 28 DF, SD=0.19).

For each participant, the cognitive outcome of learning gain was calculated as post-test score minus pre-test score. The mean learning gain across each set of fifteen annotated student sessions was 6.9% for the moderate tutor and 6.0% for the proactive tutor, yielding no evidence of improved learning gains associated with a particular level of student control (*p*=0.895, *t*-test with pooled variances, 27 DF, SD=0.09).

Therefore, it will be assumed that in the present context the thirty sessions were representative of the larger data set in terms of tutor initiative because the subset was selected at random. Also, it is meaningful to consider all learning gains and assume each tutor took a sufficiently uniform approach across all tutoring sessions. The mean learning gain for all students tutored with the moderate approach was 6.9%, while the mean learning gain for the proactive tutor was 8.6%. In this larger set of learning gains, there is still no evidence that one tutor was more or less effective than the other ($p$=0.569, $t$-test with pooled variances, 58 DF, SD=0.11).

Student self-efficacy[23] gain was measured as the difference between post-survey and pre-survey score on an item that asked students to rate their certainty, on a scale of 0-100, that they have the capability to learn the necessary course material for their introductory computer science class. A significantly different average self-efficacy gain was found between student groups paired with the two tutors. Students who worked with the proactive tutor had an average self-efficacy gain of less than one point from pre-survey to post-survey. On the other hand, students paired with the moderate tutor had an average self-efficacy gain of more than six points, which is significantly higher ($p$=0.047, $t$-test with pooled variance, 28 DF, SD=6. 5). This finding suggests that within the two levels of tutor initiative considered here, affording the student more control may yield motivational benefit without sacrificing cognitive outcomes. The results speak to Hypothesis 1.3, which states that the level of autonomy given to students during tutoring will be correlated with learning and motivational outcomes. There is no evidence that the level of autonomy is statistically significantly correlated with learning, but there is a significant correlation between giving the student more autonomy and facilitating gains in self-efficacy.

## 5.4 Discussion of Exploratory Findings

The exploratory findings suggest that computer science tutors naturally adapt to student characteristics, and that particular tutorial strategies may be associated with higher student learning or self-efficacy gains. These patterns complement research results from other tutoring domains. For example, student utterances exhibiting reasoning and reasoning-oriented questions posed by the tutor have been shown to be positively correlated with

[23] Recall that the term *self-efficacy* differs from *confidence*, as described in Section 1.5.

learning in a human-computer corpus for qualitative physics, as has the introduction of new concepts in the dialogue by students in a human-human corpus (Forbes-Riley et al., 2005). Student deep reasoning questions (as opposed to questions that ask students to respond with a simple fact) have also been associated with improved learning (Graesser *et al.*, 2008), as has the dialogue property of lexical cohesion, especially for low-performing students (Ward & Litman, 2006).

The need for balancing cognitive and motivational strategies has also been recognized in other domains; for example, the presence of cognitive feedback, as opposed to motivational "progress" feedback, was responsible for higher learning gains in experimental versions of AutoTutor (Jackson & Graesser, 2007). On the other hand, the presence of cognitive feedback lowered students' motivational ratings. Students working with modified versions of a natural science tutor learned better when given cognitive rather than affective feedback (Tan & Biswas, 2006). Finally, in a tutoring system for ecology, initially unmotivated students were found to perform better with motivational adaptation and feedback, while students who were already motivated did not benefit from the motivational support (Rebolledo-Mendez *et al.*, 2006).

These findings are valuable for informing the behavior of tutoring systems as well as giving insight into the cognitive and motivational processes at work as students learn through tutoring. However, the findings raise additional questions regarding specific tutoring phenomena, such as the benefits of positive cognitive feedback, that can be explored further through experimental investigation. This is a promising direction for future work. Additionally, these exploratory analyses have provided a basic understanding of the structure of task-oriented dialogue for introductory computer science, a foundation for the models described Chapters 6 and 7.

# CHAPTER 6

# Modeling Hidden Tutorial Dialogue State with Hidden Markov Models

The notion that dialogue has an underlying unobservable structure that influences the observed activity is widely accepted. A major goal of this dissertation is to explore whether this *hidden dialogue state* can be discovered automatically with hidden Markov models (HMMs), as evidenced by whether the HMM-learned structure correlates with student outcomes and proves useful for the dialogue management tasks of user utterance interpretation and system dialogue move selection. Before embarking on those tasks, it was desirable to investigate whether the hidden dialogue states discovered by HMMs from the tutoring corpora qualitatively resemble tutoring *modes* from the literature (Cade *et al.*, 2008) and whether these automatically extracted states are correlated with student learning. This chapter describes that investigation.

Section 6.1 provides an introduction to Markov models (MMs) and HMMs. Section 6.2 describes preliminary application of hidden Markov modeling to the corpus, qualitative analysis of which suggests that HMMs can discover tutoring modes in an unsupervised fashion (that is, with no labeled modes present in the training data). Section 6.3 examines an enhancement to these basic HMMs that involves leveraging information about adjacency pairs, which are dialogue acts that tend to co-occur because the first act establishes an expectation for the second act to follow (Schegloff & Sacks, 1973). Finally, Section 6.4

presents findings regarding the correlation between student learning and the relative frequency of the hidden states of the learned HMMs.[24]

## 6.1 Introduction to Hidden Markov Models

To introduce hidden Markov models (HMMs) it is useful to first present an overview of first-order Markov models (MMs). In the context of dialogue, a first-order Markov model is also referred to as a *bigram* model (Forbes-Riley & Litman, 2005), and it serves as a useful baseline for comparing the performance of more complex modeling approaches.

A Markov model that generates observation (state) sequence $o_1o_2...o_t$ is defined in the following way. The observation symbols are drawn from the alphabet $\sum=\{\sigma_1, \sigma_2, ..., \sigma_M\}$, and the initial probability distribution is $\Pi=[\pi_i]$ where $\pi_i$ is the probability of a sequence beginning with observation symbol $\sigma_i$. The transition probability distribution is $A=[a_{ij}]$, where $a_{ij}$ is the probability of observing state $j$ immediately after state $i$. Figure 11 illustrates the time-slice topology of an MM in the context of task-oriented dialogue modeling, where the observations consist of dialogue acts and labeled subtask actions. Under the first-order MM assumptions, each observation depends only on the immediately preceding observation.

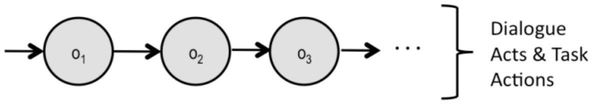


*Figure 11. Time-slice topology of first-order Markov model*

While an MM assumes that the entire process is observable, a HMM explicitly models unobservable, or *hidden* states, within a doubly stochastic structure (Rabiner, 1989). For a first-order HMM, the observation symbol alphabet $\sum=\{\sigma_1, \sigma_2, ..., \sigma_M\}$ is defined, along

[24] Sections 6.1 and 6.2 are based on HMM analysis of Corpus II from the second pilot study. Section 6.3 is based on analysis of Corpus III from the main tutoring study.

with a set of hidden states $S=\{s_1,s_2,\dots,s_N\}$. The transition and initial probability distributions are defined analogously to MMs, except that they operate on hidden states rather than on observation symbols. That is, $\Pi=[\pi_i]$ where $\pi_i$ is the probability of a sequence beginning in hidden state $s_i$ in S. The transition matrix is $A=[a_{ij}]$, where $a_{ij}$ is the probability of the model transitioning from hidden state $i$ to hidden state $j$. This framework constitutes the first stochastic layer of the model, which can be thought of as modeling hidden, or unobservable, structure. The second stochastic layer of the model governs the production of observation symbols: the emission probability distribution is $B=[b_{ik}]$ where $b_{ik}$ is the probability of state $i$ emitting observation symbol $k$. The time-slice topology of the HMMs is depicted in Figure 12, where each $q_t$ is in $S$ and each $o_t$ is in $\sum$. Each transition emits one and only one symbol.

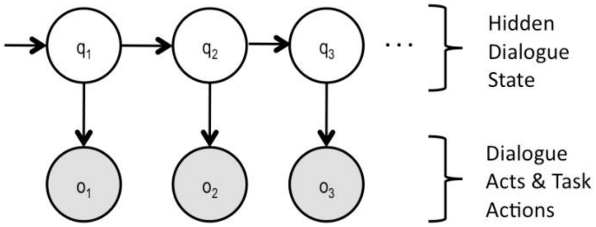


*Figure 12. Time-slice topology of first-order hidden Markov model*

The standard machine learning algorithm for acquiring HMM parameters from a set of observation sequences is the Baum-Welch algorithm (Rabiner, 1989; Bishop, 2006). Baum-Welch is an instance of the general machine learning algorithm Expectation Maximization, in which parameters are iteratively estimated and then refined until convergence or until a stopping criterion is met. In the current work, this algorithm and the Viterbi algorithm described below were implemented in Java.[25]

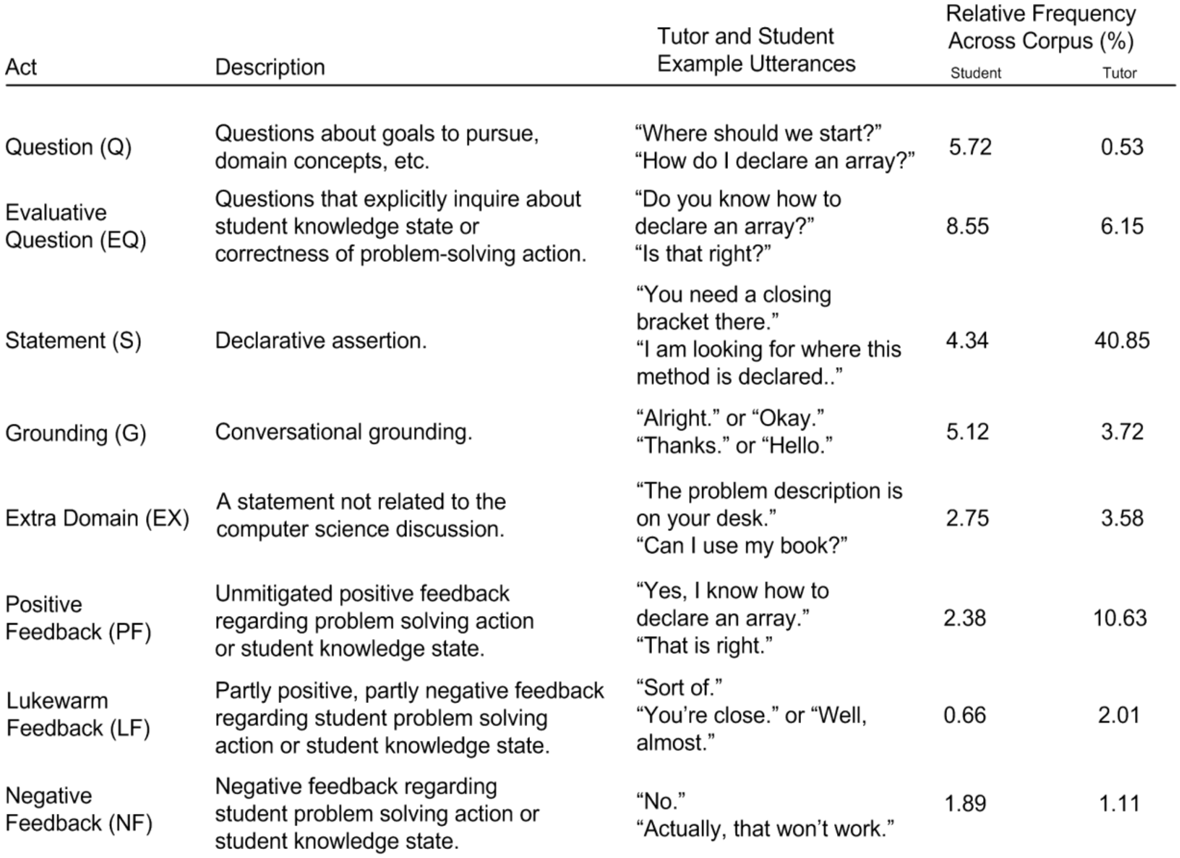
[25] Eunyoung Ha implemented this software.

In addition to fitting the HMM parameters, an important HMM problem involves identifying the best-fit sequence of hidden states that corresponds to a given observation sequence. The Viterbi algorithm (Rabiner, 1989; Bishop, 2006; Jurafsky & Martin, 2008) is used for this task. This algorithm operates over a lattice that includes all possible hidden states at each possible time step in the observation sequence. Rather than explicitly computing probabilities over the exponentially large space of paths through the lattice, the algorithm leverages the Markov property of the model and retains only the path at a given time step *t* and particular state *q* that had the highest probability until that point. At the final time step *T*, one state will correspond to the most probable complete path. By backtracking through the lattice from that most probable ending state, the algorithm identifies the most probable hidden state at each time step.

## 6.2 Identifying Hidden Tutorial Dialogue States with HMMs

The primary impetus for selecting HMMs as the modeling framework was the notion that the HMMs' hidden layer can explicitly capture tutorial dialogue *modes*, sometimes referred to as *strategies*. The work presented in this section utilizes Corpus II and its set of cognitive channel dialogue acts. The cognitive tags and their relative frequencies are shown in Table 6.

*Table 6. Modified dialogue act tagset for training HMMs on Corpus II*



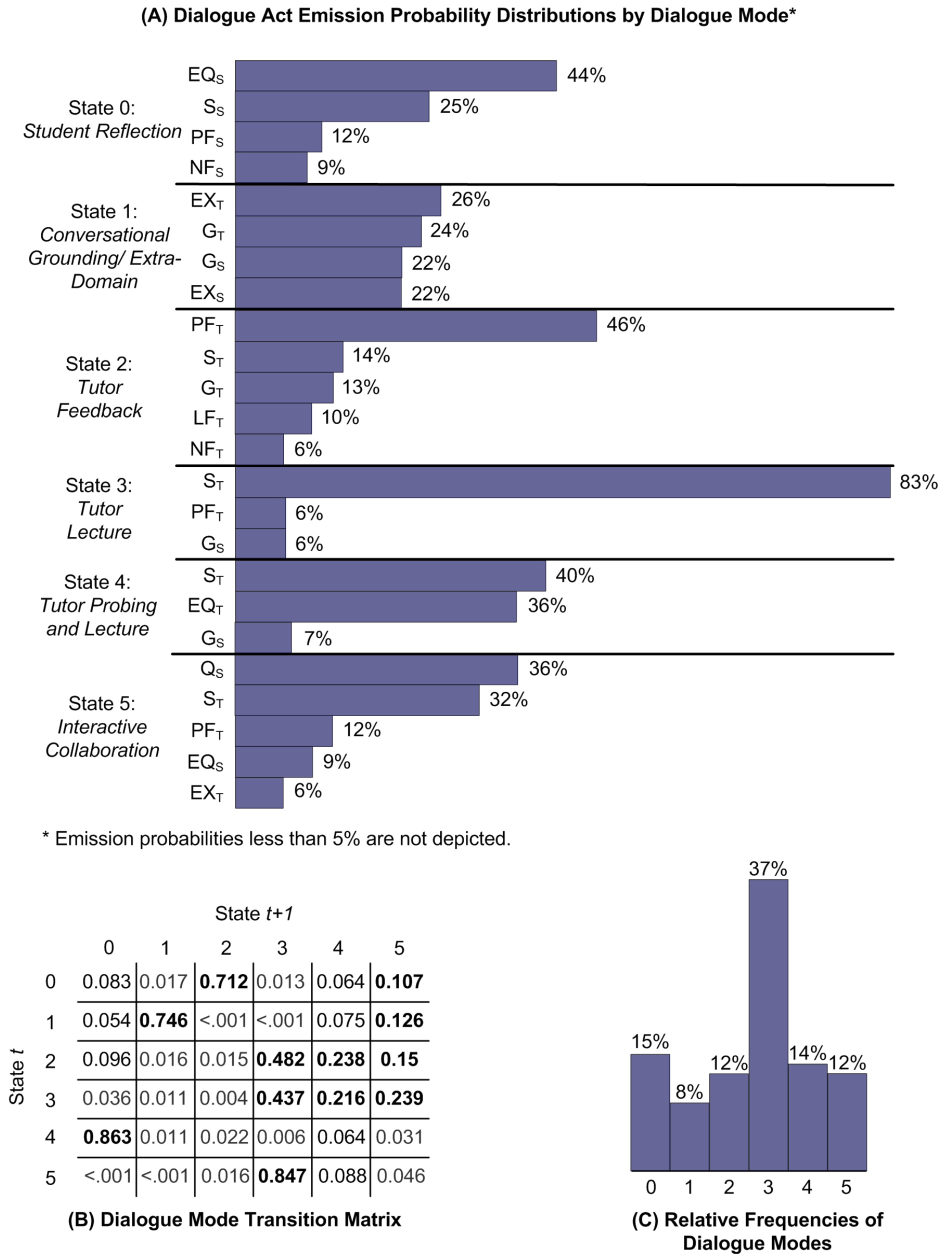
| Act | Description | Tutor and Student Example Utterances | Relative Frequency Across Corpus (%) | |
|---|---|---|---|---|
| | | | Student | Tutor |
| Question (Q) | Questions about goals to pursue, domain concepts, etc. | "Where should we start?"<br>"How do I declare an array?" | 5.72 | 0.53 |
| Evaluative Question (EQ) | Questions that explicitly inquire about student knowledge state or correctness of problem-solving action. | "Do you know how to declare an array?"<br>"Is that right?" | 8.55 | 6.15 |
| Statement (S) | Declarative assertion. | "You need a closing bracket there."<br>"I am looking for where this method is declared.." | 4.34 | 40.85 |
| Grounding (G) | Conversational grounding. | "Alright." or "Okay."<br>"Thanks." or "Hello." | 5.12 | 3.72 |
| Extra Domain (EX) | A statement not related to the computer science discussion. | "The problem description is on your desk."<br>"Can I use my book?" | 2.75 | 3.58 |
| Positive Feedback (PF) | Unmitigated positive feedback regarding problem solving action or student knowledge state. | "Yes, I know how to declare an array."<br>"That is right." | 2.38 | 10.63 |
| Lukewarm Feedback (LF) | Partly positive, partly negative feedback regarding student problem solving action or student knowledge state. | "Sort of."<br>"You're close." or "Well, almost." | 0.66 | 2.01 |
| Negative Feedback (NF) | Negative feedback regarding student problem solving action or student knowledge state. | "No."<br>"Actually, that won't work." | 1.89 | 1.11 |

In this application of HMMs to the tutorial dialogue from Corpus II, the input sequences were comprised of unigram tutorial dialogue acts augmented with tags indicating the speaker. An example of such a sequence is, (GROUNDING$_{S}$, GROUNDING$_{T}$, QUESTION$_{S}$, STATEMENT$_{T}$, STATEMENT$_{S}$, POSITIVEFEEDBACK$_{T}$). These observed symbols are provided, without any additional context regarding their meaning, as the input sequence for training HMMs. The hidden variable is interpreted as the dialogue mode. Rather than specifying *a priori* the number of dialogue modes, the best-fit number $N$ of hidden states was learned from the observed sequences during model training. The measure of fit is *log-likelihood*, which indicates how likely the current model would be to generate the observed sequences. The log of the original likelihood value is taken to avoid numerical underflow. For each value of *N*, seven models were randomly initialized and fine-tuned through ten-fold cross-validation on the corpus to obtain an average log-likelihood value.[4] A model containing *N*=6 hidden states produced the best log-likelihood fit for the current corpus.[5] Figure *13*(A) presents the most important components of the emission probability distribution $B=[b_{ik}]$ for each hidden state in the best-fit model. Probability values that are less than 5% are not shown in the diagram.

We interpret each state as a dialogue mode and assign intuitive state names by examining the emission probability distribution of dialogue acts that occur in that state. Because State 0 is dominated by student evaluation questions, statements, and feedback, this state is interpreted as *Student Reflection* mode. State 1 is dominated by extra-domain talk and conversational grounding by both the student and tutor, so this state is interpreted as *Conversational Grounding/Extra-Domain* mode. State 2 consists primarily of feedback from the tutor, with some statements and tutor grounding, so this state is interpreted as *Tutor Feedback* mode. State 3 is strongly dominated by tutorial statements, so this state is interpreted as *Tutor Lecture* mode. State 4 emits primarily tutor statements and tutor evaluation questions, so this state is interpreted as *Tutor Probing and Lecture*. Finally, State 5 is dominated by a mixture of student questions with tutor statements and feedback, so this state is interpreted as *Interactive Collaboration* mode.

[4] Model parameters were learned with the Baum-Welch expectation maximization algorithm (Rabiner, 1989), beginning with randomly-initialized parameters and then iterating until convergence. Training between five and ten models is in keeping with standard practice when this random initialization approach is used. Ten-fold cross-validation involves repeated systematic sampling of the data to partition into a 90% training set and a 10% testing set.

[5] Log-likelihood fit is a measure of how likely the observed sequences would be under a proposed model. The number of hidden states, *N*, was allowed to range from 2 to 20, with the best fit produced by *N*=6.



* Emission probabilities less than 5% are not depicted.

| State *t* \ State *t+1* | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0.083 | 0.017 | **0.712** | 0.013 | 0.064 | **0.107** |
| 1 | 0.054 | **0.746** | <.001 | <.001 | 0.075 | **0.126** |
| 2 | 0.096 | 0.016 | 0.015 | **0.482** | **0.238** | **0.15** |
| 3 | 0.036 | 0.011 | 0.004 | **0.437** | **0.216** | **0.239** |
| 4 | **0.863** | 0.011 | 0.022 | 0.006 | 0.064 | 0.031 |
| 5 | <.001 | <.001 | 0.016 | **0.847** | 0.088 | 0.046 |

**(B) Dialogue Mode Transition Matrix**

*Figure* 13. *Unigram HMM*

The transition matrix $A$=[$a_{ij}$] in Figure *13*(B) shows the probability of transitioning from one hidden state to the next. This transition matrix represents the higher-level flow of dialogue. For example, from State 0 (*Student Reflection*), the dialogue transitions with probability 0.712 to State 2 (*Tutor Feedback*) and with probability 0.107 to State 5 (*Interactive Collaboration*). From State 2 (*Tutor Feedback*), the dialogue is most likely to transition to State 3 (*Tutor Lecture*), with State 4 (*Tutor Probing and Lecture*) or State 5 (*Interactive Collaboration*) also likely candidates for the next mode.

Because the learned HMM implies a best-fit sequence of hidden states for each observed sequence of dialogue acts,[6] it is possible to summarize the frequency of each dialogue mode across the corpus as depicted in Figure *13*(C). Not surprisingly, State 3 (*Tutor Lecture*) occurs most frequently. This result is expected because in the current corpus, tutor statements account for 40% of all dialogue acts.

While a mapping between sets of tutoring modes is difficult to achieve (Litman *et al.*, 2009), qualitative inspection of the learned HMM demonstrates that the hidden states do resemble some tutoring modes or strategies from the literature. For example, State 0 (*Student Reflection)* consists of the student's own feedback, statements, and evaluation questions. Prior work has shown that eliciting this type of student reflection is a challenging task, even with the one-on-one attention provided during tutoring (Graesser & Person, 1994). Such a finding is consistent with State 0 accounting for only 8% of dialogue moves in the corpus. The structures of State 1 (*Conversational Grounding/Extra-Domain)* and State 3 (*Tutor Lecture)* do occur in a set of handcrafted tutoring modes for expert tutoring (Cade *et al.*, 2008). When comparing the experience level of tutors, those with less experience (which describes nearly all the tutors from Corpus II on which this HMM was built) tend to lecture the students more, and this finding is consistent with the large percentage of dialogue moves

[6] The Viterbi algorithm (Rabiner, 1989) was used to fit the best sequence of hidden states to each observation sequence.

that fell in the *Tutor Lecture* state. State 5, *Interactive Collaboration*, resembles the collaborative interaction known as Knowledge Co-Construction (KCC) (Hausmann *et al.*, 2004). Episodes of this highly interactive mode have been associated with higher learning. Recently, the study of KCC interactions in domains such as data structures for computer programming has yielded insights for the design of an intelligent peer-learning agent (Kersey *et al.*, 2009).

Although the HMM presented in this section offers descriptive insight into the tutorial strategies occurring in the corpus, one serious limitation involves a well-known property of HMMs, namely, that they can transition to a different hidden state at every time step. A model that transitions between hidden states in the middle of a pair of dependent dialogue moves (such as a question and answer pair) would violate an intuitive notion of dialogue structure; therefore, the next section presents a method by which these dependent pairs of dialogue moves, known as *adjacency pairs* (Schegloff & Sacks, 1973), are automatically discovered and joined before building an HMM on the modified input sequences.

## 6.3 Leveraging Adjacency Pairs with Bigram HMMs

The importance of adjacency pairs is well established in natural language dialogue. Adjacency pair analysis, also referred to as bigram analysis, has illuminated important phenomena in tutoring (Forbes-Riley & Litman, 2005; Forbes-Riley *et al.*, 2007). The intuition behind adjacency pairs is that certain dialogue acts naturally occur together, and by grouping these acts we capture an exchange between two dialogue participants in a single structure. This formulation is of interest primarily because when treating sequences of dialogue acts as a Markov process, with or without hidden states, the addition of adjacency pairs offers a semantically richer observation alphabet. This section presents an HMM trained on sequences of dialogue act adjacency pairs from Corpus II. As part of the evolution of the methodology, this application of HMMs to Corpus II utilized a penalized log-likelihood measure, which reduces the measure of fit as the number of parameters increases. Penalized log-likelihood measures are used to strike a balance between model complexity and model fit. To facilitate direct comparison, this section describes both a unigram and a bigram HMM fit using this penalized log-likelihood approach.

### *6.3.1 Adjacency pair identification*

To find adjacency pairs we utilize a $\chi^2$ test to assess dependence of the categorical variables $act_i$ and $act_{i+1}$ for all sequential pairs of dialogue acts that occur in the corpus. Only pairs in which the two speakers were different were considered; that is, speaker($act_i$) $\neq$ speaker($act_{i+1}$). Table 7 displays a list of all dependent adjacency pairs sorted by descending (unadjusted) statistical significance; the subscript on each dialogue act tag indicates tutor ($T$) or student ($S$).

An adjacency-pair joining algorithm was applied to join statistically significant ($p$<0.01) pairs of dialogue acts into atomic units according to a priority determined by the strength of the statistical significance. Dialogue acts that were "left out" of adjacency pair groupings were treated as atomic elements in subsequent analysis. Figure 14 shows the adjacency-pair joining algorithm and Figure 15 illustrates the application of the algorithm on a sequence of dialogue acts from the corpus.

*Table 7. Statistically significant adjacency pairs in Corpus II*

| ***act$_i$*** | ***act$_{i+1}$*** | **P(act$_{i+1}$\| act$_i$)** | **P(act$_{i+1}$\| ¬act$_i$)** | **$\chi^2$ val** | ***p*-val** |
|---|---|---|---|---|---|
| EvaluatingQuestion$_S$ | PositiveFdbk$_T$ | 0.48 | 0.07 | 654 | <0.0001 |
| Grounding$_S$ | Grounding$_T$ | 0.27 | 0.03 | 380 | <0.0001 |
| ExtraDomain$_S$ | ExtraDomain$_T$ | 0.34 | 0.03 | 378 | <0.0001 |
| EvaluatingQuestion$_T$ | PositiveFdbk$_S$ | 0.18 | 0.01 | 322 | <0.0001 |
| EvaluatingQuestion$_T$ | Statement$_S$ | 0.24 | 0.03 | 289 | <0.0001 |
| EvaluatingQuestion$_S$ | LukewarmFdbk$_T$ | 0.13 | 0.01 | 265 | <0.0001 |
| Question$_T$ | Statement$_S$ | 0.65 | 0.04 | 235 | <0.0001 |
| EvaluatingQuestion$_T$ | LukewarmFdbk$_S$ | 0.07 | 0.00 | 219 | <0.0001 |
| Question$_S$ | Statement$_T$ | 0.82 | 0.38 | 210 | <0.0001 |
| EvaluatingQuestion$_S$ | NegativeFdbk$_T$ | 0.08 | 0.01 | 207 | <0.0001 |
| ExtraDomain$_T$ | ExtraDomain$_S$ | 0.19 | 0.02 | 177 | <0.0001 |
| NegativeFdbk$_S$ | Grounding$_T$ | 0.29 | 0.03 | 172 | <0.0001 |
| EvaluatingQuestion$_T$ | NegativeFdbk$_S$ | 0.11 | 0.01 | 133 | <0.0001 |
| Statement$_S$ | Grounding$_T$ | 0.16 | 0.03 | 95 | <0.0001 |
| Statement$_S$ | PositiveFdbk$_T$ | 0.30 | 0.10 | 90 | <0.0001 |
| Statement$_T$ | Grounding$_S$ | 0.07 | 0.04 | 36 | <0.0001 |
| PositiveFdbk$_S$ | Grounding$_T$ | 0.14 | 0.04 | 34 | <0.0001 |
| LukewarmFdbk$_S$ | Grounding$_T$ | 0.22 | 0.04 | 30 | <0.0001 |
| Statement$_T$ | EvaluatingQuestion$_S$ | 0.11 | 0.07 | 29 | <0.0001 |
| Statement$_T$ | Question$_S$ | 0.07 | 0.05 | 14 | 0.0002 |
| Grounding$_T$ | ExtraDomain$_S$ | 0.07 | 0.03 | 14 | 0.002 |
| Grounding$_T$ | Grounding$_S$ | 0.10 | 0.05 | 9 | 0.0027 |
| EvaluatingQuestion$_T$ | EvaluatingQuestion$_S$ | 0.13 | 0.08 | 8 | 0.0042 |

```
Sort adjacency pair list L by descending statistical significance
For each adjacency pair ($act_1$, $act_2$) in L
    For each dialogue act sequence ($a_1, a_2, \ldots, a_n$)
    in the corpus
        Replace all pairs ($a_i$=$act_1$, $a_{i+1}$=$act_2$) with a new single symbol ($act_1act_2$)
```

*Figure 14. Adjacency-pair joining algorithm*

Original dialogue act sequence:

$G_s \rightarrow G_t \rightarrow Q_s \rightarrow S_t \rightarrow EQ_t \rightarrow EQ_s \rightarrow PF_t \rightarrow Q_s$

After adjacency pair joining:

$G_sG_t \rightarrow Q_sS_t \rightarrow EQ_t \rightarrow EQ_sPF_t \rightarrow Q_sS_t$

*Figure 15. Example of input sequences before and after adjacency-pair joining*

### *6.3.2 Model training*

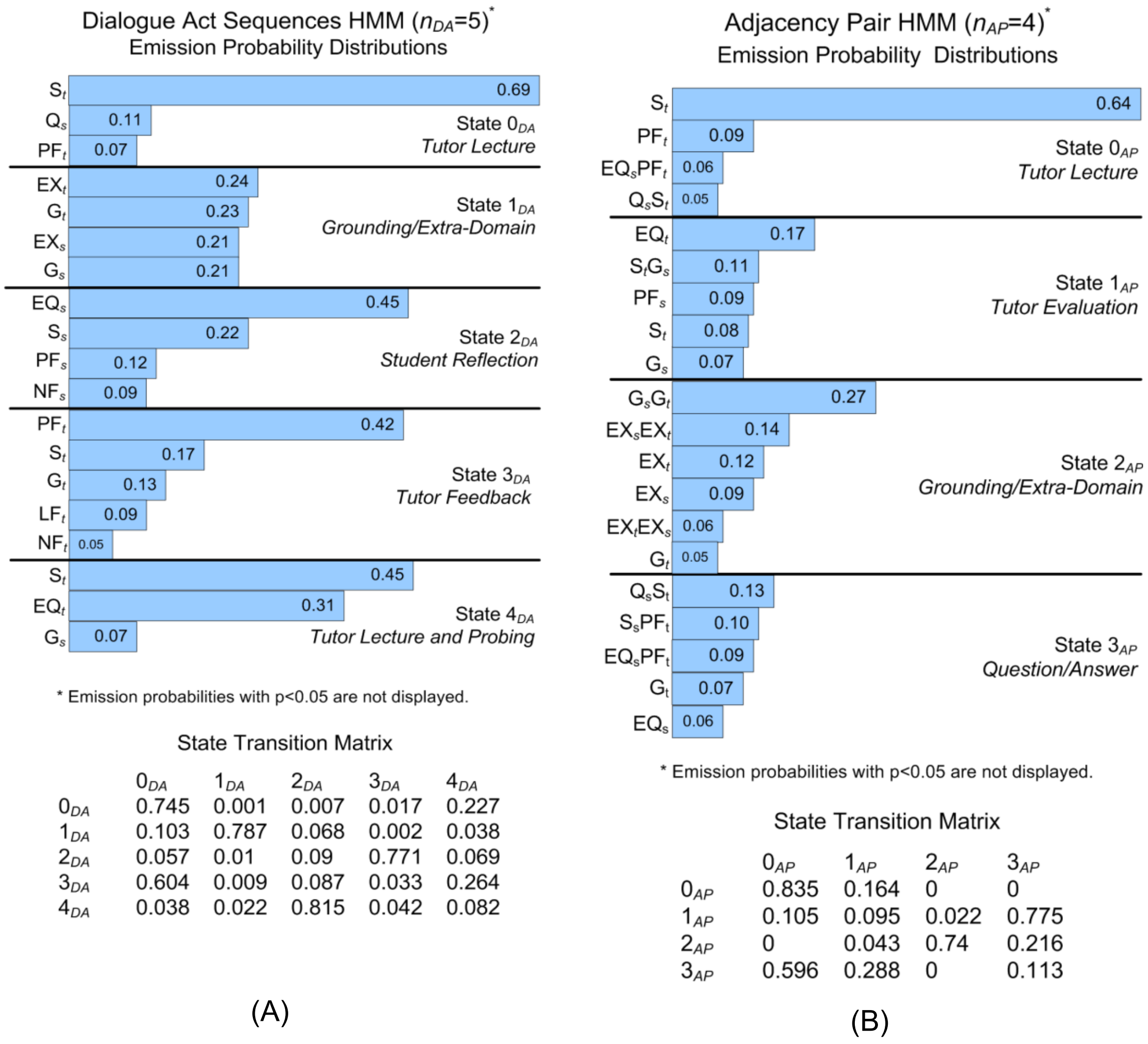
In keeping with the goal of automatically discovering dialogue structure, it was desirable to find *n,* the best number of hidden states for the HMM, during modeling. To this end, seven models were trained and ten-fold cross-validated, each featuring randomly initialized parameters, for each number of hidden states *n* from 2 to 15, inclusive. [26] The average log-likelihood fit from ten-fold cross-validation was computed across all seven models for each *n*, and this average log-likelihood $l_n$ was used to compute the Akaike Information Criterion, a maximum-penalized likelihood estimator that prefers simpler models (Scott, 2002). This modeling approach was used to train HMMs on both the dialogue act and the adjacency pair input sequences.

[26] $n$=15 was chosen as an initial maximum number of states because it comfortably exceeded the hypothesized range of 3 to 7 (informed by the tutoring literature). The Akaike Information Criterion (Scott, 2002) measure steadily worsened above $n = 5$, confirming no need to train models with $n > 15$.

The input sequences of individual dialogue acts contain 16 unique symbols because each of the 8 dialogue act tags (Table 6) was augmented with a label of the speaker, either tutor or student. The best-fit HMM for this input sequence contains $n_{DA}$=5 hidden states. The adjacency pair input sequences contain 39 unique symbols, including all dependent adjacency pairs (Table 7) along with all individual dialogue acts because each dialogue act occurs at some point outside an adjacency pair. The best-fit HMM for this input sequence contains $n_{AP}$=4 hidden states. In both cases, the best-fit number of dialogue modes implied by the hidden states is within the range of what is often considered in traditional tutorial dialogue analysis (Cade *et al.*, 2008; Graesser *et al.*, 1995).

Qualitatively evaluating the impact of grouping the dialogue acts into adjacency pairs requires a fine-grained examination of the generated HMMs to gain an insight into how each model interprets the student sessions. Figure 16(A) displays the emission probability distributions for the dialogue act HMM. State $0_{DA}$, *Tutor Lecture*,[27] is strongly dominated by tutor statements with some student questions and positive tutor feedback. State $1_{DA}$ constitutes *Grounding/Extra-Domain*, a conversational state consisting of acknowledgments, backchannels, and discussions that do not relate to the computer science task. State $2_{DA,}$ *Student Reflection,* generates student evaluation questions, statements, and positive and negative feedback. State $3_{DA}$ is comprised of tutor utterances, with positive feedback occurring most commonly, followed by statements, grounding, lukewarm feedback, and negative feedback. This state is interpreted as a *Tutor Feedback* mode. Finally, State $4_{DA}$, *Tutor Lecture/Probing*, is characterized by tutor statements and evaluative questions with some student grounding statements.

[27] For clarity, the states of each HMM have been named according to an intuitive interpretation of the emission probability distribution.



Dialogue Act Sequences HMM ($n_{DA}$=5)*
Emission Probability Distributions

* Emission probabilities with p<0.05 are not displayed.

State Transition Matrix

|  | $0_{DA}$ | $1_{DA}$ | $2_{DA}$ | $3_{DA}$ | $4_{DA}$ |
|---|---|---|---|---|---|
| $0_{DA}$ | 0.745 | 0.001 | 0.007 | 0.017 | 0.227 |
| $1_{DA}$ | 0.103 | 0.787 | 0.068 | 0.002 | 0.038 |
| $2_{DA}$ | 0.057 | 0.01 | 0.09 | 0.771 | 0.069 |
| $3_{DA}$ | 0.604 | 0.009 | 0.087 | 0.033 | 0.264 |
| $4_{DA}$ | 0.038 | 0.022 | 0.815 | 0.042 | 0.082 |

(A)

Adjacency Pair HMM ($n_{AP}$=4)*
Emission Probability Distributions

* Emission probabilities with p<0.05 are not displayed.

State Transition Matrix

|  | $0_{AP}$ | $1_{AP}$ | $2_{AP}$ | $3_{AP}$ |
|---|---|---|---|---|
| $0_{AP}$ | 0.835 | 0.164 | 0 | 0 |
| $1_{AP}$ | 0.105 | 0.095 | 0.022 | 0.775 |
| $2_{AP}$ | 0 | 0.043 | 0.74 | 0.216 |
| $3_{AP}$ | 0.596 | 0.288 | 0 | 0.113 |

(B)

*Figure 16. Dialogue act (unigram) and adjacency pair (bigram) HMMs*

The state transition diagram in Figure 16(A) illustrates that *Tutor Lecture* ($0_{DA}$) and *Grounding/Extra-Domain* ($1_{DA}$) are stable states whose probability of self-transition is high, at 0.75 and 0.79, respectively. Perhaps not surprisingly, *Student Reflection* ($2_{DA}$) is most likely to transition to *Tutor Feedback* ($3_{DA}$) with probability 0.77. *Tutor Feedback* ($3_{DA}$)

transitions to *Tutor Lecture* ($0_{DA}$) with probability 0.60, *Tutor Lecture/Probing* ($4_{DA}$) with probability 0.26, and *Student Reflection* ($2_{DA}$) with probability 0.09. Finally, *Tutor Lecture/Probing* ($4_{DA}$) very often transitions to *Student Reflection* ($2_{DA}$) with probability 0.82.

Figure 16(B) displays the emission probability distributions for the HMM that was trained on the input sequences of adjacency pairs. State $0_{AP}$, *Tutor Lecture*, consists of tutorial statements, positive feedback, and dialogue turns initiated by student questions. In this state, student evaluation questions occur in adjacency pairs with positive tutor feedback, and other student questions are answered by tutorial statements. State $1_{AP}$, *Tutor Evaluation*, generates primarily tutor evaluation questions, along with the adjacency pair of tutorial statements followed by student acknowledgements. State $2_{AP}$ generates conversational grounding and extra-domain talk; this *Grounding/Extra-Domain* state is dominated by the adjacency pair of student grounding followed by tutor grounding. State $3_{AP}$ is comprised of several adjacency pairs: student questions followed by tutor answers, student statements with positive tutor feedback, and student evaluation questions followed by positive feedback. This *Question/Answer* state also generates some tutor grounding and student evaluation questions outside of adjacency pairs.

To illustrate how the above models fit the data, Figure 17 depicts the progression of dialogue modes that generate an excerpt from the corpus.

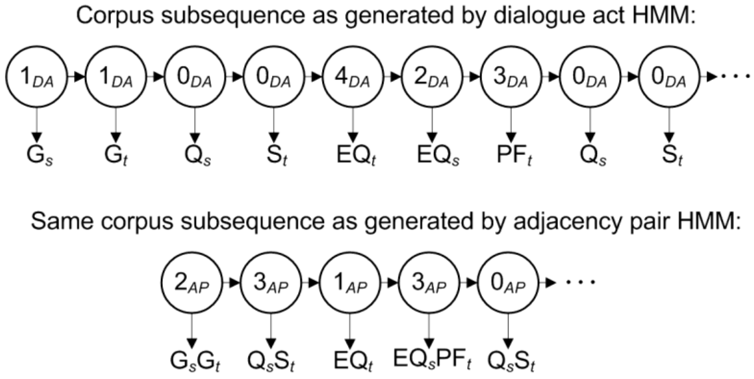


*Figure 17. Dialogue act sequences as generated by unigram and bigram HMMs*

In both models, the most commonly occurring dialogue mode is *Tutor Lecture*, which generates 45% of observations in the dialogue act model and around 60% in the adjacency pair model. Approximately 15% of the dialogue act HMM observations are fit to each of states *Student Reflection*, *Tutor Feedback*, and *Tutor Lecture/Probing*. This model spends the least time, around 8%, in *Grounding/Extra Domain.* The adjacency pair model fits approximately 15% of its observations to each of *Tutor Evaluation* and *Question/Answer*, with around 8% in *Grounding/Extra-Domain*.

### 6.3.3 Qualitative comparison of unigram and bigram HMM

While the two models presented in this section were derived from the same corpus, it is important to exercise caution when making direct structural comparisons. The models contain neither the same number of hidden states nor the same emission symbol alphabet. It is meaningful to note that the adjacency pair model with $n_{AP}$=4 achieved an average log-likelihood fit on the training data that was 5.8% better than the same measure achieved by the dialogue act model with $n_{DA}$=5, despite the adjacency pair input sequences containing greater than twice the number of unique symbols.[28]

The qualitative comparison begins by examining the modes that are highly similar in the two models. State $2_{AP}$ generates grounding and extra-domain statements, as does State $1_{DA}$. These two states both constitute a *Grounding/Extra-Domain* dialogue mode. One artifact of the tutoring study design is that all sessions begin in this state due to a compulsory greeting that signaled the start of each session. More precisely, the initial state probability distribution for each HMM assigns probability 1 to this state and probability 0 to all other states.

Another dialogue mode that is structurally similar in the two models is *Tutor Lecture*, in which the majority of utterances are tutor statements. This mode is captured in State 0 in both models, with State $0_{AP}$ implying more detail than State $0_{DA}$ because it is certain in the former that some of the tutor statements and positive feedback occurred in response to student questions. While student questions are present in State $0_{DA}$, no such precise ordering of the acts can be inferred.

Other states do not have one-to-one correspondence between the two models. State $2_{DA}$, *Student Reflection,* generates only student utterances and the self-transition probability for the state is very low; the dialogue usually visits State $2_{DA}$ for one turn and then transitions immediately to another state. Although this aspect of the model reflects the fact that students rarely keep the floor for more than one utterance at a time in the corpus, such quick dialogue mode transitions are inconsistent with an intuitive understanding of tutorial dialogue modes as meta-structures that usually encompass more than one dialogue turn. This phenomenon is perhaps more accurately captured in the adjacency pair model. For example, the dominant

[28] This comparison is meaningful because the models depicted here provided the best fit among all sizes of models trained for the same input scenario.

dialogue act of State $2_{DA}$ is a student evaluation question ($EQ_s$). In contrast, these dialogue acts are generated as part of an adjacency pair by State $3_{AP}$; this model joins the student questions with subsequent positive feedback from the tutor rather than generating the question and then transitioning to a new dialogue mode.

### *6.3.4 Discussion*

One promising result of this early work emerges from the fact that by applying hidden Markov modeling to sequences of adjacency pairs, meaningful dialogue modes have emerged that are empirically justified. The number of these dialogue modes is consistent with what researchers have traditionally used as a set of hypothesized tutorial dialogue modes. Moreover, the composition of the dialogue modes reflects some recognizable aspects of tutoring sessions: tutors teach through the *Tutor Lecture* mode and give feedback on student knowledge in a *Tutor Evaluation* mode. Students ask questions and state their own perception of their knowledge in a *Question/Answer* mode. Both parties engage in "housekeeping" talk containing such things as greetings and acknowledgements, and sometimes, even in a controlled environment, extra-domain conversation occurs between the conversants in the *Grounding/Extra-Domain* mode.

Although the tutorial modes discovered may not map perfectly to sets of handcrafted tutorial dialogue modes from the literature, it is rare for such a perfect mapping to exist even between those sets of handcrafted modes. In addition, the HMM framework allows for succinct probabilistic description of the phenomena at work during the tutoring session: through the state transition matrix, we can see the back-and-forth flow of the dialogue among its modes.

Automatically learning dialogue structure is an important step toward creating more robust tutorial dialogue management systems. This section has presented two hidden Markov models in which the hidden states are interpreted as *dialogue modes* for task-oriented tutorial dialogue. These models were learned in an unsupervised fashion from sequences of manually labeled dialogue acts. The next section discusses work to identify associations between the structure of learned HMMs and the student learning outcomes.

## 6.4 Correlations Between Hidden Dialogue State and Student Learning

The preliminary investigations into learning HMMs from corpora of tutorial dialogue and examining their descriptive power qualitatively were conducted on Corpus II from the second pilot study, as described in Sections 6.2 and 6.3. The qualitative examination suggests that HMMs can discover tutoring modes, or hidden dialogue states, in an unsupervised fashion (that is, without hand labeling of the tutoring modes), and that these modes bear a resemblance to tutoring modes from the literature. One important aspect of validating these models is to identify statistically significant correlations between the tutoring modes learned by the models and the outcome of student learning as measured by learning gain from pre-test to post-test, a line of analysis that directly investigates Hypothesis 2.1.

The HMMs in this section were learned from Corpus III, which was produced by the main observational tutoring study. Notably, this study differs from the two pilot studies because 1) the instrument to measure learning was improved based on piloting and further refined based on input from three experts in teaching introductory computer science, and 2) in addition to dialogue act annotation, the task actions in Corpus III were manually annotated (Section 4.2), providing a rich basis for learning models that capture the interplay between dialogue and task.

### *6.4.1 Learning separate HMMs by tutor*

The main tutoring study featured two paid tutors who had achieved the highest average student learning gains in the two pilot studies. Tutor A was a male computer science student in his final semester of undergraduate studies. Tutor B was a female third-year computer science graduate student. Exploratory analysis of the corpus suggested that the tutors took different approaches with respect to initiative, but achieved similar learning gains (Section 5.3). Because of the observed differences between tutoring strategies, a separate HMM was derived from the tutoring sessions with each tutor. The ten-fold cross-validation methodology was used, as described in the previous section, utilizing a penalized log-likelihood estimator to measure model fit.

The best-fit HMM for Tutor A's dialogues features eight hidden states. Figure 18 depicts a subset of the transition probability diagram with nodes representing hidden states (tutoring modes). Inside each node is a histogram of its emission probability distribution. For simplicity, only five of the eight states are displayed in this diagram; each state that was omitted mapped to less than 5% of the observed data sequences and was not significant in the correlational analysis. Each tutoring mode has been interpreted and named based on its structure. For example, State 4 is dominated by correct task actions; therefore, it is named *Correct Student Work.* State 6 is comprised of student acknowledgements, pairs of tutor statements, some correct task actions, and assessing questions by both tutor and student; this state is labeled *Student Acting on Tutor Help.* The best-fit model for Tutor B's dialogues features ten hidden states. A portion of this model, consisting of all states that mapped to more than 5% of observations, is displayed in Figure 19.

Some tutoring modes with similar structures were identified by both models. Both models feature a *Correct Student Work* mode characterized by the student's successful completion of a subtask. This state maps to 38% of observations with Tutor A and 29% of observations with Tutor B. In both cases the *Correct Student Work* mode occurs more frequently than any other mode.

The next three most frequently occurring modes each maps onto 10-15% of the observations. For Tutor A, one such mode is *Tutor Explanations with Feedback*, while for Tutor B a corresponding mode is *Tutor Explanations with Assessing Questions*. In both cases, the mode involves tutors explaining concepts or task elements. A key difference is that with Tutor A, the explanation mode includes frequent negative elaborated feedback or positive content-free feedback, while for Tutor B the explanation mode features questions in which the tutor aims to gauge the student's knowledge. A similar pattern emerges with each tutor's next most frequent mode: for Tutor A, this mode is *Student Work with Tutor Positive Feedback*; for Tutor B, the mode is *Student Work with Tutor Assessing Questions*. These corresponding modes illuminate a tendency for Tutor A to provide feedback in situations where Tutor B chooses to ask the student a question. For Tutor A, the only mode that featured assessing questions was *Student Acting on Tutor Help*, which as we will discuss, was positively correlated with student learning.

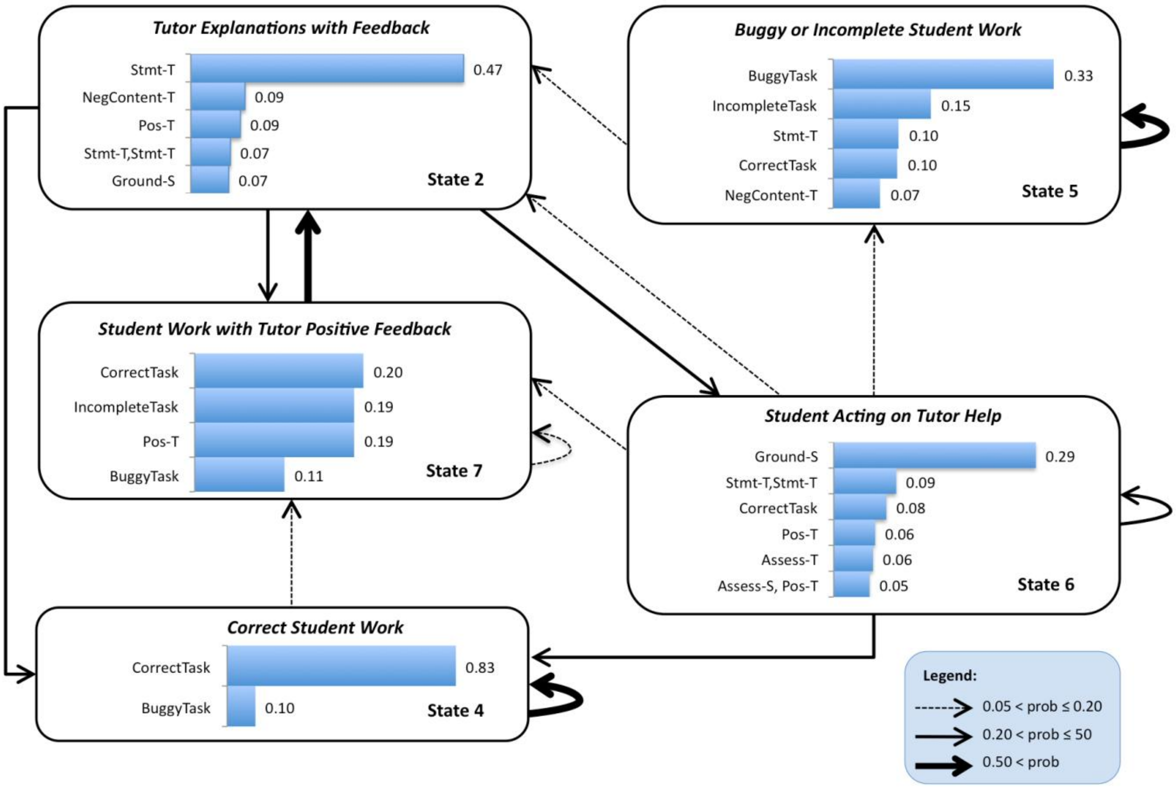


*Figure 18. Portion of bigram HMM for Tutor A, Corpus III*

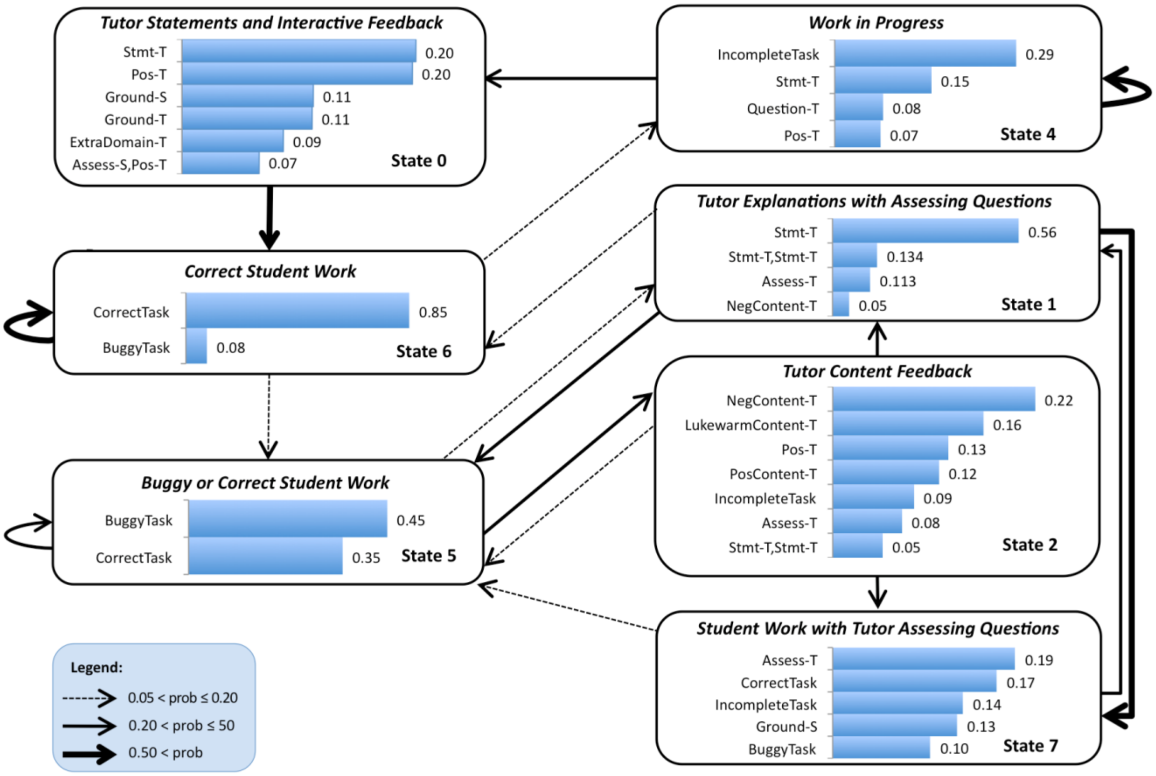
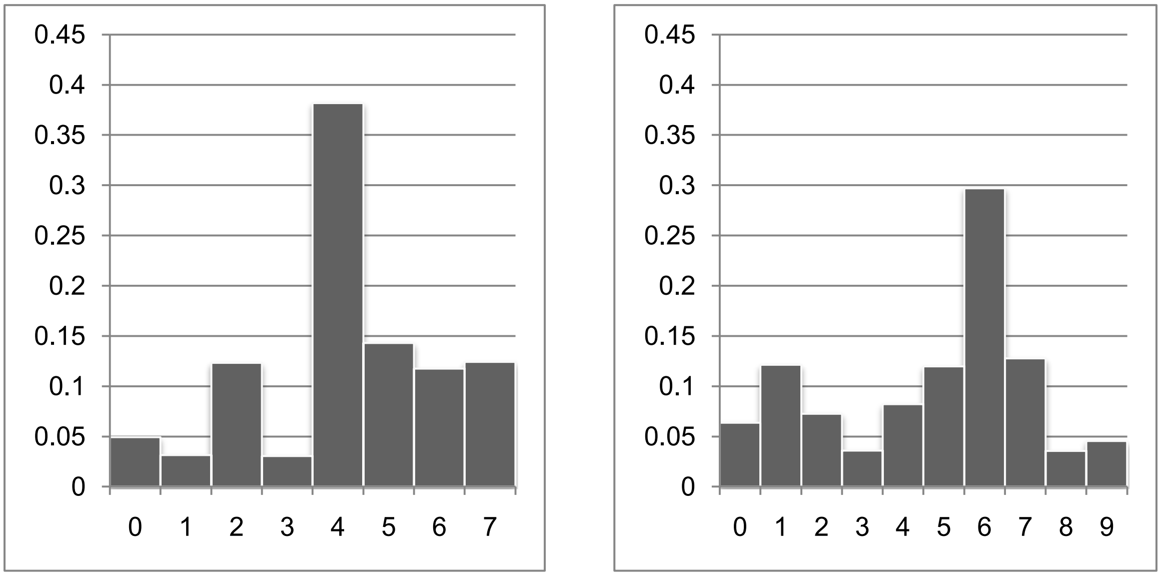


*Figure 19. Portion of bigram HMM for Tutor B, Corpus III*

### 6.4.2 Correlations between hidden dialogue states and student learning

With the learned models in hand, the next goal was to identify statistical relationships between student learning and the automatically extracted tutoring modes. The models presented above were used to fit each sequence of observed dialogue acts and task actions onto the set of hidden states (*i.e.*, tutoring modes) using maximum likelihood. The transformed sequences were used to calculate the frequency distribution of the modes that occurred in each tutoring session (*e.g.*, State 0 = 32%, State 1 = 15%. . . State 8 = 3%). The average relative frequencies of each hidden state across all tutoring sessions with each tutor are depicted in Figure 20.



(A) (B)

*Figure 20. Relative frequency of hidden states across corpus for Tutor A and Tutor B*

For each HMM, correlations were generated between the learning gain of each student session and the relative frequency vector of tutoring modes for that session to determine whether significant relationships existed between student learning and the proportion of discrete events (dialogue and problem solving) that were accounted for by each tutoring mode. For Tutor A, the *Student Acting on Tutor Help* mode was positively correlated with learning (*r=0.51;p<0.0001*). For Tutor B, the *Tutor Elaborated Feedback* mode was positively correlated with learning (*r=0.55; p=0.01*) and the *Work in Progress* mode was negatively correlated with learning (*r=-0.57; p=0.0077).*

This analysis has identified significant correlation between student learning gains and the automatically extracted tutoring modes modeled in the HMMs as hidden states. While students who worked with either tutor on average achieved significant learning, each group of students displayed a substantial range of learning gains. The correlation analysis leveraged this data spread to gain insight into which aspects of the tutorial interaction were related to higher or lower learning gains.

### *6.4.3 Discussion*

For Tutor A, the relative frequency of the *Student Acting on Tutor Help mode* was positively correlated with student learning. This mode was characterized primarily by student acknowledgments and also featured tutor explanations, correct student work, positive tutor feedback, and assessing questions from both tutor and student. The composition of this tutoring mode suggests that these observed events possess a synergy that, in context, contributed to student learning. In a learning scenario with novices, it is plausible that only a small subset of tutor explanations were grasped by the students and put to use in the learning task. The *Student Acting on Tutor Help* mode may correspond to those instances, in contrast to the *Correct Student Work* mode in which students may have been applying prior knowledge.

For Tutor B, the *Tutor Elaborated Feedback* mode was positively correlated with student learning. This mode was relatively infrequent, mapping to only 7% of tutoring events. However, providing direct feedback represents a departure from this tutor's more frequent approach of asking assessing questions of the student. Given the nature of the learning task and the corresponding structure of the learning instrument, students may have identified errors in their work and grasped actionable new knowledge most readily through this tutor's direct feedback.

For Tutor B, the *Work in Progress* mode was negatively correlated with learning. This finding is consistent with observations that in this tutoring study, students did not easily seem to operationalize new knowledge that came through tutor hints, but rather, often needed explicit constructive feedback. The *Work in Progress* mode features no direct tutor content feedback. Tutor questions and explanations (which are at a more abstract level than the student's solution) in the face of incomplete student work may not have been an effective tutoring approach in this study.

The work described in this section confirms Hypothesis 2.1, that the hidden dialogue structure extracted by HMMs is correlated with tutoring outcomes. This work takes a step toward fully automatic extraction of tutorial strategies from corpora, a contribution that has direct application in human tutoring research. The approach also can be used in tutorial dialogue system development by producing a data-driven library of system strategies, and as the next two chapters discuss, by contributing to the creation of data-driven tutorial dialogue management models.

# CHAPTER 7

# Dialogue Act Classification in Task-Oriented Tutorial Dialogue

All dialogue systems, including tutorial dialogue systems, must address the two central challenges of a) interpreting user utterances and b) selecting system dialogue moves. This chapter focuses on user utterance interpretation in terms of dialogue acts (Austin, 1962). Dialogue acts are abstractions that provide a valuable intermediate representation that can be used for dialogue management. The models presented in this chapter were developed from Corpus III data generated by the third, and main, tutoring study. The corpus consists of human textual dialogue utterances and a separate, parallel stream of user-generated task actions. To classify student utterances with respect to dialogue acts in this complex task-oriented domain, classifiers are constructed that utilize lexical and syntactic features along with structural features including task/subtask labels, dialogue act history, speaker, and hidden dialogue state in a vector-based representation. This chapter explores whether the addition of HMM and task/subtask features improves the predictive performance of the dialogue act classifiers. The results speak to Hypothesis 2.2.

Because this chapter focuses on student utterances only, some dialogue act labels from the symmetric tutor/student dialogue act classification scheme in Table 4 (Section 4. 1) were renamed to better reflect dialogue acts from a student's perspective (no re-annotation was required; there is a one-to-one correspondence between these labels and the symmetric scheme presented previously). Table 8 displays the student dialogue acts and their relative frequencies across the corpus, along with the inter-annotator agreement statistic for that particular student utterance.

*Table 8. Student dialogue acts, frequencies, and Kappas in Corpus III*

| Student Dialogue Act | Relative Frequency | Human κ |
|---|---|---|
| ACKNOWLEDGMENT (ACK) | .17 | *.90* |
| REQUEST FOR FEEDBACK (RF) | .20 | *.91* |
| EXTRA-DOMAIN (EX) | .08 | *.79* |
| GREETING (GR) | .04 | *.92* |
| UNCERTAIN FEEDBACK WITH ELABORATION (UE) | .01 | *.53* |
| UNCERTAIN FEEDBACK (U) | .02 | *.49* |
| NEGATIVE FEEDBACK WITH ELABORATION (NE) | .01 | *.61* |
| NEGATIVE FEEDBACK (N) | .05 | *.76* |
| POSITIVE FEEDBACK WITH ELABORATION (PE) | .02 | *.43* |
| POSITIVE FEEDBACK (P) | .09 | *.81* |
| QUESTION (Q) | .09 | *.85* |
| STATEMENT (S) | .16 | *.82* |
| THANKS (T) | .05 | *1* |

## 7.1 Features

To address the classification task, the models make use of features of each utterance that include the words and pairs of words, parts of speech, and syntactic structure. These features are encoded within a vector-based representation along with structural features that include dialogue act labels, task/subtask labels, and set of hidden dialogue state prediction features as described below.[29]

[29] The lexical and syntactic features were extracted collaboratively with Eunyoung Ha.

Lexical and syntactic features were automatically extracted from the utterances using the Stanford Parser default tokenizer and part of speech (POS) tagger (De Marneffe *et al.*, 2006). The parser created both phrase structure trees and typed dependencies for individual sentences. From the phrase structure trees, we extracted the top-most syntactic node and its first two children. In the case where an utterance consisted of more than one sentence, only the phrase structure tree of the first sentence was considered. Typed dependencies between pairs of words were extracted from each sentence. Individual word tokens in the utterances were further processed with the Porter Stemmer (Porter, 1980) in the NLTK package (Loper & Bird, 2004). The POS features were extracted in a similar way. Unigram and bigram word and POS tags were included for feature selection in the classifiers.

Structural features include the annotated dialogue acts (Section 4.1), the annotated task/subtask labels (4.2), and attributes that represent the *hidden dialogue state* (Section 6.2). To derive these hidden dialogue state features, an HMM was trained utilizing the methodology described in Section 6.3 and it was used to generate predictions in the form of a probability distribution over possible user utterances at each step in the dialogue. This set of stochastic features was subsequently passed to the classifier as part of the input vector, as depicted in Figure 21.

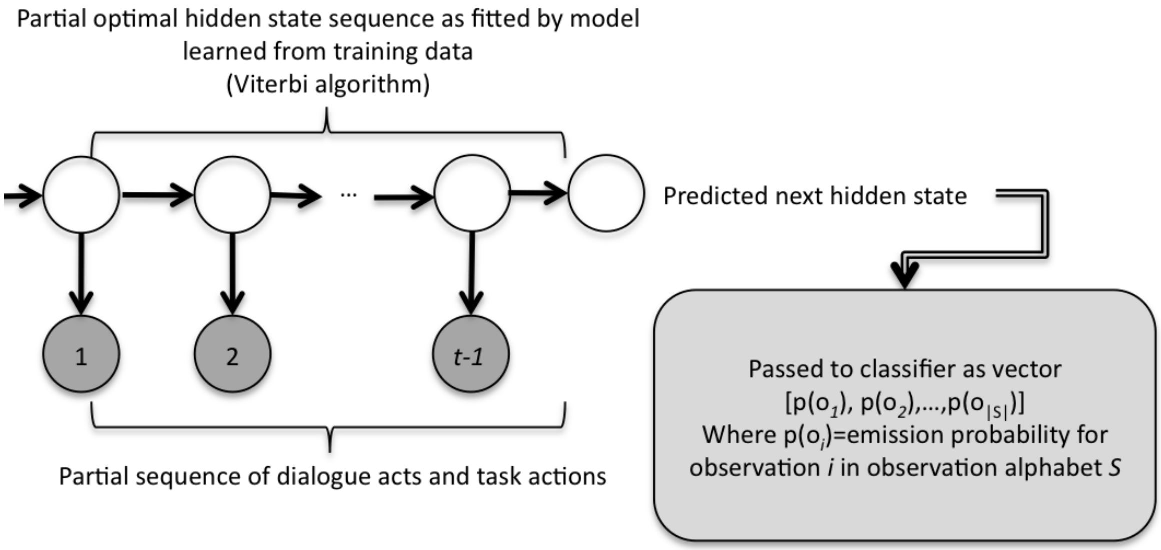

Partial optimal hidden state sequence as fitted by model learned from training data (Viterbi algorithm)
...
Predicted next hidden state
1
2
t-1
Partial sequence of dialogue acts and task actions
Passed to classifier as vector
[p(o_1), p(o_2),...,p(o_|S|)]
Where p(o_i)=emission probability for observation i in observation alphabet S

*Figure 21. Generation of hidden dialogue state features*

## 7.2 Input Vectors

The features were combined into a shared vector-based representation for training the classifier. As depicted in Table 9, the components of the feature vector include binary existence vectors for lexical and syntactic features for the current (target) utterance as well as for three utterances of left context (this left context may include both tutor and student utterances, which are distinguished by a separate indicator for the speaker). The task/subtask and correctness history features encode the separate stream of task events. There is no one-to-one correspondence between these history features and the left-hand dialogue context, because several task events could have occurred between a pair of dialogue events (or vice versa). This distinction is indicated in the table by the representation of dialogue time steps as [*t, t-1,…*] and task history steps as [*task(t), task(t-1),…*]. In total, the feature vectors included 11,432 attributes that were made available for feature selection.

## 7.3 Dialogue Act Classification Experiments

This section describes the vector-based models for classification of user dialogue acts using maximum likelihood logistic regression. In addition to investigating the accuracy of the overall model, binary dialogue act classifiers investigate the utility of feature types for discriminating between particular dialogue acts of interest.

The classifiers are based on logistic regression, which finds a discriminant for each pair of dialogue acts by assigning weights in a maximum likelihood fashion.[30] The logistic regression models were learned using the Weka machine learning toolkit (Hall *et al.*, 2009). For feature selection, attribute subset evaluation was used in conjunction with a best-first approach that greedily searched the space of possible features using a hill climbing approach with backtracking, also within the Weka toolkit. The prediction accuracy of the classifiers was determined through ten-fold cross-validation on the corpus, and the results below are presented in terms of average prediction accuracy (number of correct classifications divided by total number of classifications) as well as by the Kappa statistic, which adjusts for expected agreement by chance.

[30] In general, the model that maximizes likelihood is precisely the model that maximizes entropy under the same constraints (Berger *et al.*, 1996).

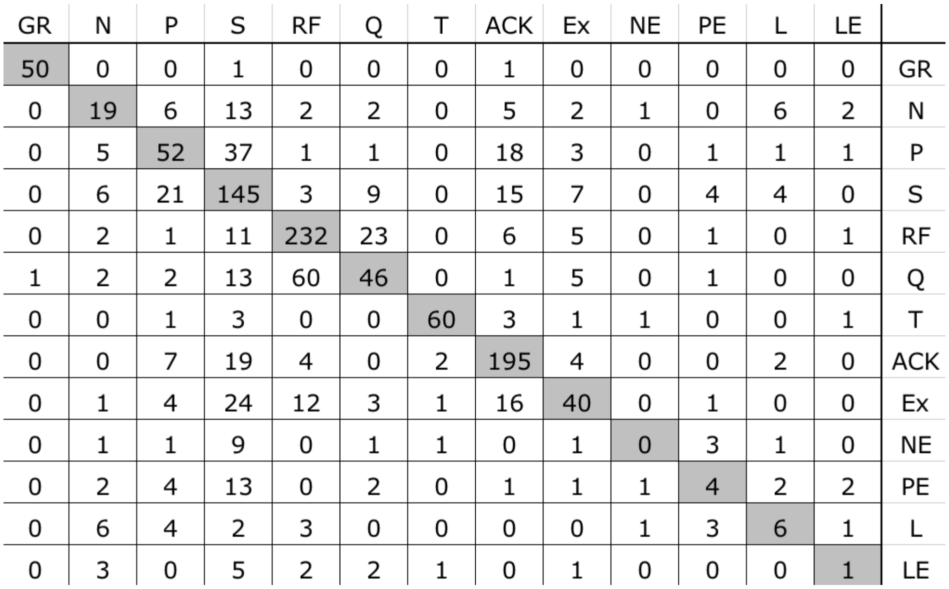
*Table 9. Feature vectors for dialogue act classification*

| **Feature vector *f*** | **Description** |
|---|---|
| $[w_{t,1}, \ldots w_{t,\|w\|},$<br>$p_{t,1}, \ldots, p_{t,\|p\|},$<br>$d_{t,1}, \ldots, d_{t,\|d\|},$<br>$s_{t,1}, \ldots, s_{t,\|s\|}]$ | Binary existence vector for word unigrams & bigrams, POS unigrams & bigrams, dependency types, and syntactic nodes for current target utterance *t* |
| $[w_{t-k,1}, \ldots w_{t-k,\|w\|},$<br>$p_{t-k,1}, \ldots, p_{t-k,\|p\|},$<br>$d_{t-k,1}, \ldots, d_{t-k,\|d\|},$<br>$s_{t-k,1}, \ldots, s_{t-k,\|s\|}]$<br>where$k=1,\ldots,3$ | Binary existence vector for word unigrams & bigrams, POS unigrams & bigrams, dependency types, and syntactic nodes for three utterances of left context |
| $[p(o_1), \ldots, p(o_{\|S\|})]$ | Probability distribution for emission symbols in predicted next hidden state as generated by HMM |
| $[da_{t-1}, da_{t-2}, da_{t-3}]$ | Dialogue act left context |
| $[sp_{t-1}, sp_{t-2}, sp_{t-3}]$ | Speaker label left context |
| $[tk_{task(t-1)}, tk_{task(t-2)}, tk_{task(t-3)}]$ | Three steps of subtask history (each level of hierarchy represented as a separate feature) |
| *pt* | Indicator for whether the target utterance was immediately preceded by a task event |

### *7.3.1 Overall classification*

The overall dialogue act classification model was trained to classify each utterance with respect to the thirteen dialogue acts (Table 8). For this task, the feature selection algorithm selected 63 attributes including some syntax, dependency, POS, and word attributes as well as two steps of dialogue act history, speaker history, and one task/subtask feature. No hidden dialogue state features or task correctness attributes were selected. The overall average classification accuracy across ten folds was 62.8% (*stdev*=3. 26%). This accuracy constitutes a 3.7-fold improvement over baseline chance of 17% (the relative frequency of the most frequently occurring dialogue act, ACK). An alternate nontrivial baseline is a bigram model on true dialogue acts (including speaker tags); this model's accuracy was 36.8%.

In addition to the classifier with all features available as described above, the experiments include classifiers that used only the lexical and syntactic features of each utterance. This approach is of interest in part because it avoids the error propagation that can happen when a model relies on a series of its own previous classifications as features. The classifier that had access to only the set of lexical and syntactic features selected 85 of these attributes and achieved an average prediction accuracy of 60.2% (*stdev*=2.44%) and *κ*=. 53 (*stdev*=0.03), slightly worse than the 62.8% achieved with the model that had access to all features (one-tailed two sample *t*-test *p*=0.027). The overall average Kappa for the full classifier was *κ*=. 57 (*stdev*=0.04). The confusion matrix for this model is depicted in Figure 22, which depicts agreements along the diagonal and disagreements elsewhere. In this figure, the row indicates the true tag and the column indicates the automatically applied tag.



| GR | N | P | S | RF | Q | T | ACK | Ex | NE | PE | L | LE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 50 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | GR |
| 0 | 19 | 6 | 13 | 2 | 2 | 0 | 5 | 2 | 1 | 0 | 6 | 2 | N |
| 0 | 5 | 52 | 37 | 1 | 1 | 0 | 18 | 3 | 0 | 1 | 1 | 1 | P |
| 0 | 6 | 21 | 145 | 3 | 9 | 0 | 15 | 7 | 0 | 4 | 4 | 0 | S |
| 0 | 2 | 1 | 11 | 232 | 23 | 0 | 6 | 5 | 0 | 1 | 0 | 1 | RF |
| 1 | 2 | 2 | 13 | 60 | 46 | 0 | 1 | 5 | 0 | 1 | 0 | 0 | Q |
| 0 | 0 | 1 | 3 | 0 | 0 | 60 | 3 | 1 | 1 | 0 | 0 | 1 | T |
| 0 | 0 | 7 | 19 | 4 | 0 | 2 | 195 | 4 | 0 | 0 | 2 | 0 | ACK |
| 0 | 1 | 4 | 24 | 12 | 3 | 1 | 16 | 40 | 0 | 1 | 0 | 0 | Ex |
| 0 | 1 | 1 | 9 | 0 | 1 | 1 | 0 | 1 | 0 | 3 | 1 | 0 | NE |
| 0 | 2 | 4 | 13 | 0 | 2 | 0 | 1 | 1 | 1 | 4 | 2 | 2 | PE |
| 0 | 6 | 4 | 2 | 3 | 0 | 0 | 0 | 0 | 1 | 3 | 6 | 1 | L |
| 0 | 3 | 0 | 5 | 2 | 2 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | LE |

*Figure 22. Confusion matrix for student dialogue act classification (row=true tag)*

### *7.3.2 Binary dialogue act classifiers*

In tutoring, some student dialogue acts are particularly important to identify because of their implications for the tutor's response or for the student model. For example, a student's REQUEST FOR FEEDBACK requires the tutor to assess the condition of the task artifact, rather than to query the in-domain factual knowledge base. UNCERTAIN FEEDBACK is another dialogue act of high importance because identifying it allows the tutor to respond in an affectively advantageous way (Forbes-Riley & Litman, 2009). Although hidden dialogue state features and task/subtask features generally were not useful for the overall dialogue classification task, it is of interest to explore whether these features are useful for differentiating particular dialogue acts of interest.

To explore which features are useful for classifying particular dialogue acts, we constructed binary dialogue act classifiers, one for each dialogue act, by preprocessing the dialogue act labels from the set of thirteen down to TRUE or FALSE depending on whether the label of the utterance matched the target dialogue act for that specialized classifier. Table 10 displays the features that were selected for each binary classifier, along with the percent accuracy and Kappa for each model. Note that for some dialogue acts the chance baseline is very high, and therefore even a model with high prediction accuracy achieves a low Kappa.

As shown in the table below, for several dialogue act models, the feature selection algorithm retained subtask and HMM features. However, in an experiment to quantify the utility of these features, the performance of the binary all-features models was compared to the performance of the overall all-features model on that dialogue act. The only dialogue acts whose specialized models outperformed the overall model ($p$<0.05, one-tailed $t$-test) were GREETING and EXTRA-DOMAIN. These results demonstrate that hidden dialogue state and task/subtask features did not improve classification accuracy for dialogue acts of high pedagogical interest.

*Table 10. Feature selection for binary dialogue act classifiers*

| DA | | Features Selected | % Correct | Model κ |
|---|---|---|---|---|
| ACK | 51 | Lexical/syntax, HMM, DA history (preceding=S), speaker history (preceding=Tutor) | .933 | *.75* |
| RF | 42 | Lexical/syntax, DA history, preceded by subtask | .905 | *.72* |
| EX | 57 | Dependency, pos, word, HMM, DA history (preceding=EX), subtask | .939 | *.45* |
| GR | 11 | Syntax, pos, word, DA (previous=EMPTY), speaker, subtask | .998 | *.97* |
| UE | 21 | Dependency, pos, word, subtask | .991 | *.33* |
| U | 63 | Syntax, dependency, pos, word, HMM, subtask | .979 | *.21* |
| NE | 44 | Dependency, pos, word, HMM, DA history (2 ago=UNCERTAIN), subtask | .987 | *0* |
| N | 83 | Lexical/syntax, DA history, subtask | .966 | *.76* |
| PE | 90 | Dependency, pos, word, HMM, subtask | .976 | *.10* |
| P | 110 | Dependency, pos, word, HMM, DA history (previous=REQUEST FEEDBACK) | .945 | *.58* |
| Q | 43 | Syntax, dep, pos, word, HMM, subtask | .940 | *.60* |
| S | 92 | Syntax, pos, word, HMM, DA history (previous=EMPTY or Q) | .901 | *.57* |
| T | 29 | Syntax, pos, word, DA history (previous=POSITIVE) (3 ago=POSITIVE) | .992 | *.92* |

## 7.4 Discussion

This chapter has presented a maximum likelihood classifier that assigns dialogue act labels to user utterances from a corpus of human-human tutorial dialogue given a set of lexical, syntactic, and structural features. Overall, this classifier achieved 62.8% accuracy in ten-fold cross-validation on the corpus.[31] However, during feature selection, the overall model did not select any hidden dialogue state features, and selected only one task/subtask feature out of more than 50 possible. Therefore, the results do not support Hypothesis 2.2, which stated that the structural features of hidden dialogue state and task/subtask would improve dialogue act classification for user utterances.

The performance (369% over chance baseline) achieved with the overall model is on par with other automatic dialogue act tagging models, both sequential and vector-based, in task-oriented domains that do not feature complex, user-driven parallel tasks. In a catalogue ordering domain with an integrated task and dialogue model, Bangalore *et al.*(2009) report 75% classification accuracy for user utterances using a maximum entropy classifier, a 275% improvement over baseline. Poesio & Mikheev (1998) report 54% classification accuracy by utilizing conversational game structure and speaker changes in the Maptask corpus, an improvement of 170% over baseline. Recent work on Maptask reports a classification accuracy of 65.7% using lexical and syntactic features alone (Sridhar *et al.*, 2009). This classifier is analogous to the lexical/syntactic feature model of this work, which achieved 60.2% accuracy.

The results of these models demonstrate that, consistent with the findings in other task-oriented domains, lexical/syntactic features are highly useful for classifying student dialogue moves in this complex task-oriented domain. The utility of these features is so great

[31] Individual words, or unigrams, were not included in an early round of classifiers, and the performance was significantly lower, at 45. 3% overall.

that hidden dialogue state and task/subtask features were not useful for improving performance of the models in terms of classification accuracy. In contrast, the next chapter investigates the use of these features within a hierarchical HMM framework for predicting human tutor moves within a corpus, where the HMM structure significantly improves performance of the models.

# CHAPTER 8

# Leveraging Hidden Dialogue State to Select Tutorial Moves

The previous chapter dealt with learning data-driven models for dialogue act classification of student utterances within task-oriented tutorial dialogue. This chapter turns to the equally important task of tutorial move selection. Historically, tutorial dialogue policies have been based either on system designers' pedagogical knowledge or on observational studies of human tutors followed by manual analysis. (Please see Section 2.2 for a detailed historical overview.) This chapter presents a data-driven approach that extracts a tutorial dialogue management policy directly from a corpus. The degree to which this tutorial dialogue policy represents the actions of the human tutors is evidenced by its accuracy on the task of predicting the tutors' dialogue moves. The results provide support for Hypothesis 2.3, which states that a hierarchical HMM that explicitly models task/subtask structure will predict tutorial moves within the corpus more accurately than a flat HMM that does not model the task structure.

For student utterances, the surface lexical and syntactic features are given at the time of classification (motivating the use of the vector-based approach to handle a large number of features). In contrast, in a real-time tutorial dialogue applications the surface features of a planned tutorial utterance are not yet available, but will be realized based on the systems' choice of dialogue act. Therefore, for the task of predicting tutor moves, rather than utilizing feature vectors as input to a classifier, a sequential representation is used in which the only input for training the models are sequences of dialogue acts and task events. In addition to flat HMMs, hierarchical hidden Markov models are constructed that explicitly model task/subtask structure. The models presented in this chapter were trained using the main corpus, Corpus III.

## 8.1 Introduction to Hierarchical Hidden Markov Models

Hierarchical hidden Markov models (HHMMs) allow for explicit representation of multilevel stochastic structure (Fine *et al.*, 1998). HHMMs include two types of hidden states: *internal nodes*, which do not produce observation symbols, and *production nodes*, which do produce observations. An internal node includes a set of sub-states that correspond to its potential children, $S=\{s_1, s_2, \ldots, s_N\}$, each of which is itself the root of an HHMM. The initial probability distribution $\Pi=[\pi_i]$ for each internal node governs the probability that the model will make a vertical transition to substate$s_i$ from this internal node; that is, that this internal node will produce substate$s_i$ as its leftmost child. Horizontal transitions are governed by a transition probability distribution similar to that described above for flat HMMs. Production nodes are defined by their observation symbol alphabet and an emission probability distribution over the symbols; HHMMs do not require a global observation symbol alphabet. HHMMs of arbitrary topology can be trained using a generalized version of the Baum-Welch algorithm (Fine *et al.*, 1998; Rabiner, 1989). The generative topology of an HHMM in context of the tutorial dialogue application at hand is illustrated in Figure 23.

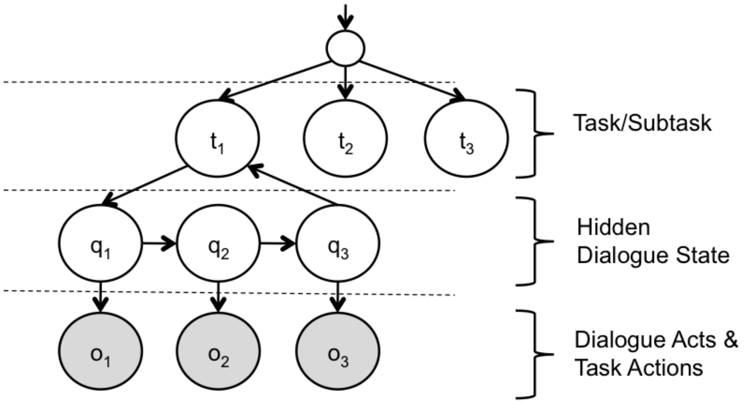


*Figure 23. Generative topology of hierarchical HMM*

## 8.2 Learned Hierarchical HMM

The HHMMs in this chapter feature a pre-specified model topology based on known task/subtask structure. A Bayesian view of a portion of the best-fit HHMM is depicted in Figure 24. Again, only the principal distribution components are shown at the sub-task level. This model was trained using five-fold cross-validation instead of ten-fold cross-validation, to address the absence of symbols from the training set that are present in the testing set, a problem that arose from splitting the data hierarchically.

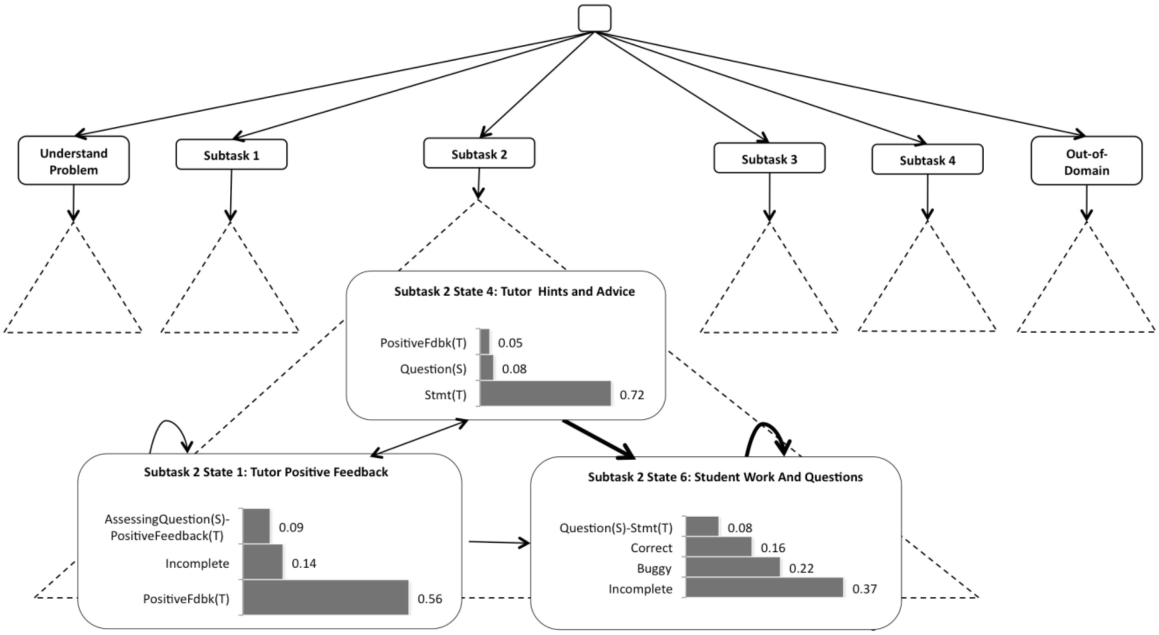


*Figure 24. Bayesian view of learned hierarchical HMM*

## 8.3 Comparison of MM, HMM, and HHMM Prediction Accuracy

Markov Models (MMs), HMMs, and HHMMs were trained on Corpus III and their prediction accuracy of tutorial dialogue acts was calculated by providing the model with partial sequences from the test set and querying for the next tutorial move. The chance baseline prediction accuracy for this task is 41.1%, corresponding to the most frequent tutorial dialogue act (STATEMENT). As depicted in Figure 25, a first-order MM performed worse than baseline ($p$<0.001)[32], at 27% average prediction accuracy ($\sigma_{MM}$=6%). HMMs performed better than baseline ($p$<0.0001), with an average accuracy of 48% ($\sigma_{HMM}$=3%).

[32] All $p$-values in this section were produced by two-sample one-tailed $t$-tests with unequal sample variances.

HHMMs averaged 57% accuracy, significantly higher than baseline ($p$=0.002), but weakly significantly higher than HMMs ($p$=0.04) and with high variation ($\sigma_{HHMM}$=23%).[33]

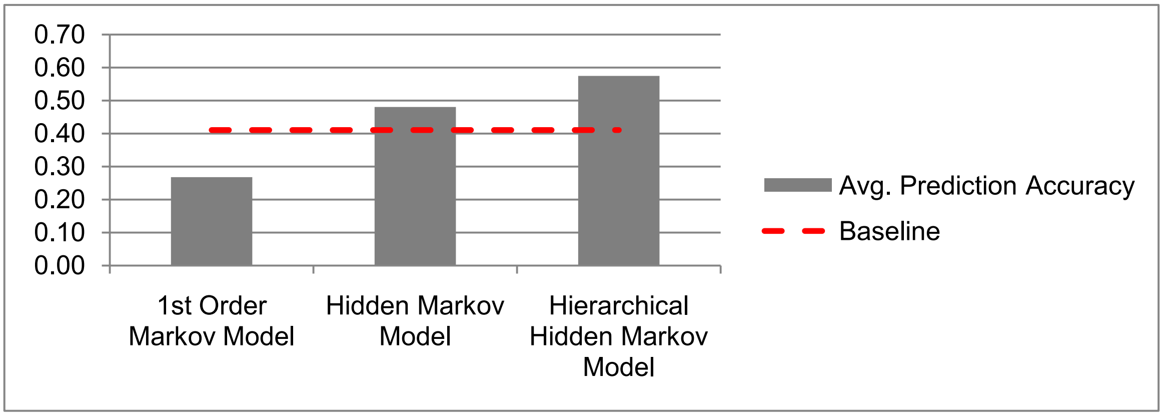


*Figure 25. Avg. prediction accuracy across folds of MM, HMM, and HHMM compared to the most-frequent class baseline*[34]

To further explore the performance of the HHMMs, Figure 26 displays their prediction accuracy on each of six labeled subtasks. These subtasks correspond to the top level of the hierarchical task/subtask annotation scheme. The UNDERSTAND THE PROBLEM subtask corresponds to the initial phase of most tutoring sessions, in which the student and tutor agree to some extent on a problem-solving plan. Subtasks 1, 2, and 3 account for the implementation and debugging of three distinct modules within the learning task, and Subtask 4 involves testing and assessing the student's finalized program. 100% of students reached Subtask 1, 94% of students reached Subtask 2, 81% of students reached Subtask 3,

[33] Because the two tutors in this study utilized different strategies (Sections 4.3 and 6.3), separate models were also built by tutor in a separate experiment. However, these models performed on par with (no statistical difference from) the aggregate models. This phenomenon is likely due to the greater amount of training data available to the aggregate models.

[34] See Section 8.4 for further discussion.

and 54% of students reached Subtask 4. The EXTRA-DOMAIN subtask involves side conversations whose topics are outside of the domain.

The HHMM performed as well as or better ($p$<0.01) than baseline on the first three in-domain subtasks. The performance on Subtask 4 was not distinguishable from baseline ($p$=0.06). The model did not outperform baseline ($p$=0.40) for the UNDERSTAND THE PROBLEM subtask, and qualitative inspection of the corpus reveals that the dialogue during this phase of tutoring exhibits limited regularities between students; additionally, only 54% of students engaged in this subtask; the remaining students began directly working on Subtask 1.

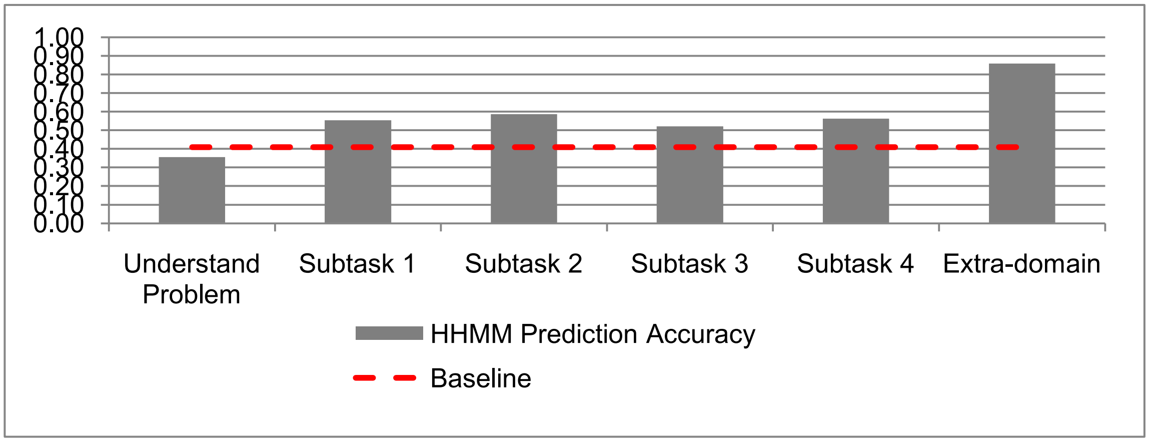


*Figure 26. Average HHMM prediction accuracy across folds by subtask*

## 8.4 Discussion

The results support Hypothesis 2.3 that HMMs, because of their capacity for explicitly representing dialogue structure at an abstract level, perform better than MMs for predicting tutor moves. The results also suggest that explicitly modeling hierarchical task structure can further improve prediction accuracy of the model. The below-baseline performance of the bigram model illustrates that, unlike conversational dialogue or task-oriented dialogue that does not feature a parallel user-driven task, in this complex task-oriented domain an immediately preceding event is not highly predictive of the next move. This finding highlights the possibility that the first-order Markov assumption may exclude important dependencies within the data, a limitation that is further discussed in Chapter 9.

Considering the performance of the HHMM on individual subtasks reveals interesting properties of the dialogues. First, the HHMM is unable to outperform baseline on the UNDERSTAND THE PROBLEM subtask, probably due to a high level of variation between individuals during this portion of the dialogues. On all four in-domain subtasks, the HHMM achieved a 30% to 50% increase over baseline. For extra-domain dialogues, which involve side conversations that are not task-related, the HHMM achieved 86% prediction accuracy on tutor moves, which constitutes a 115% improvement over baseline. This high accuracy may be due in part to the fact that out-of-domain asides were almost exclusively initiated by the student, and tutors rarely engaged in such exchanges beyond providing a single response. This regularity likely facilitated prediction of the tutor's dialogue moves during out-of-domain talk.

Only one recent project reports extensively on predicting system actions from a corpus of human-human dialogue. Bangalore *et al.*'s (2008) flat task/dialogue model in a catalogue-ordering domain achieved a prediction accuracy of 55% for system dialogue acts, a 175% improvement over baseline. When explicitly modeling the hierarchical task/subtask dialogue structure, they report a prediction accuracy of 35.6% for system moves, approximately 75% above baseline (Bangalore & Stent, 2009). These findings were obtained by utilizing a variety of lexical and syntactic features from preceding utterances along with manually annotated dialogue acts and task/subtask labels. In comparison, the HHMM in this chapter achieved an average 42% improvement over baseline using only manually annotated dialogue acts and task/subtask labels without any lexical or syntactic features.

The best model performed better than baseline by a significant margin. The absolute prediction accuracy achieved by the HHMM was 57% across the corpus, which at first blush may appear too low to be of practical use. However, the choice of tutorial move involves some measure of subjectivity, and in many contexts there may be no uniquely appropriate dialogue act. Work in other domains has dealt with this uncertainty by maintaining multiple hypotheses (Wright Hastie *et al.*, 2002) and by mapping to clustered sets of moves rather than maintaining policies for each possible system selection (Young *et al.*, 2009). Such approaches may prove useful in the complex task-oriented domain of computer science tutoring as well, and may help to more fully realize the potential of a machine-learned dialogue management model.

# CHAPTER 9

# Conclusion

Creating intelligent systems that bring the benefits of one-on-one human tutoring to a broad population of learners is a grand challenge for the field of computing. Tutorial dialogue systems hold great promise for closing the effectiveness gap that has been observed between human tutors, as models of successful tutoring, and intelligent tutoring systems. A particularly important direction involves utilizing data-driven approaches for defining the behavior of computer-based tutorial dialogue systems based on corpora of effective human tutoring. These data-driven approaches may facilitate rapid computer-driven dialogue system development, give rise to flexible dialogue management policies, support interpretation of user input, and ultimately result in a more effective computer-based learning experience for students than any current generation tutorial dialogue system has achieved. With these goals in mind, this dissertation addresses two phases of data-driven investigation. The first phase involves collecting, annotating, and exploring corpora. The second phase involves learning and evaluating computational models of hidden dialogue states, student dialogue act classification, and tutor move prediction.

## 9.1 Hypotheses Revisited

The research presented in this dissertation has produced evidence that speaks to several exploratory hypotheses.

- **Hypothesis 1.1.** Because human tutors adapt their behavior based on student characteristics including skill level, self-efficacy, and gender, *the distributions of* *dialogue acts within human-to-human tutoring sessions are dependent on these student characteristics.*

  - The results of dialogue profile analysis (Section 5.1) indicate that human tutors do adapt their dialogue profiles depending on learner characteristics, even when those characteristics are hidden from the tutors (Boyer, Vouk *et al.*, 2007). This finding suggests specific ways in which tutorial dialogue systems might adapt behavior based on learner characteristics, such as providing more acknowledgements to students with high self-efficacy, and anticipating more requests for feedback from female students.

- **Hypothesis 1.2.** Because some tutoring approaches are more effective than others, given a tutoring context, *the frequency of some tutor moves is positively correlated with student learning and motivational outcomes while other moves are negatively correlated with these outcomes*.
  - Bigrams of incorrect student problem-solving actions, and the subsequent tutor moves, were considered (Section 5.2). Their relative frequency across the corpus was correlated with learning and motivational outcomes (Boyer, Phillips *et al.*, 2008a; Boyer, Phillips *et al.*, 2008b). Explicit tutorial encouragement following incorrect student action was found to correlate positively with motivational outcomes but negatively with learning. Positive cognitive feedback, as a corrective tactic, was found to correlate with improved motivational outcomes, but not to correlate significantly with learning. These results are consistent with findings from other tutoring domains. They highlight that caution is necessary when implementing explicit motivational techniques within tutorial dialogue systems.

- **Hypothesis 1.3.** Because autonomy is an important aspect of the learning process that may impact cognitive and motivational outcomes differently, *the level of autonomy given to students during tutoring is correlated with learning and motivational outcomes*.
    - The analysis for tutor and student initiative revealed no evidence of a statistically significant link between student initiative and learning (Section 5.3). However, students who were allowed more initiative did have significantly higher self-efficacy gain from pre-test to post-test (Boyer, Phillips, Wallis *et al.*, 2009a; Boyer, Phillips, Wallis *et al.*, 2009b). These results suggest that allowing students more initiative may improve the motivational outcome of self-efficacy gain.

In addition to the exploratory hypotheses listed above, several hypotheses regarding modeling the structure of tutorial dialogue with hidden Markov models were also addressed.

- **Hypothesis 2.1.** Hidden Markov models (HMMs) are able to discover tutoring modes, or *hidden dialogue states*, that *i) qualitatively correspond to tutoring modes from the literature,* and *ii) whose frequencies of occurrence correlate with student learning*.
    - Qualitative analysis (Sections 6.1 and 6.2) suggests that the automatically extracted hidden states correspond to tutoring modes from the literature. The frequency of occurrence of some automatically extracted hidden dialogue states was found to correlate significantly with student learning (Section 6.3). These findings provide evidence that an HMM can automatically extract pedagogically relevant tutorial dialogue structure in the form of a stochastic layer formed from hidden states.

- **Hypothesis 2.2.** The hidden dialogue state and task/subtask structure are predictive of student dialogue acts.
    - An observable Markov baseline classifier, $\boldsymbol{B_1}$, was constructed using dialogue act sequences only. Model $\boldsymbol{M_1}$ was constructed using lexical and structural features, and $\boldsymbol{M_1}$**'** added to $\boldsymbol{M_1}$'s available features and attributes from the hidden dialogue state and the task/subtask structure. Both $\boldsymbol{M_1}$and $\boldsymbol{M_1}$**'** achieved higher accuracy than $\boldsymbol{B_1}$ for the overall task of classifying a user utterance with respect to the full set of 13 dialogue acts. However, contrary to the hypothesis, $\boldsymbol{M_1}$**'** did not utilize any hidden dialogue state features and only selected one out of more than 50 task/subtask features. The results demonstrate that lexical and syntactic cues are strong indicators of student dialogue acts, a finding that is consistent with user dialogue act classification from other task-oriented domains.
- **Hypothesis 2.3.** The hidden dialogue state and task/subtask structure are predictive of tutor dialogue acts.
    - A baseline first-order Markov model, $\boldsymbol{B_2}$, of sequences of dialogue acts and task events was compared with a flat HMM, $\boldsymbol{M_2}$, and further with a hierarchical HMM, $\boldsymbol{M_2}$**'**, structured according to the task/subtask hierarchy. Both $\boldsymbol{M_2}$ and $\boldsymbol{M_2}$**'** predicted human tutorial moves within the corpus more accurately than $\boldsymbol{B_{2.}}$ Furthermore, consistent with the hypothesis, hierarchical HMM $\boldsymbol{M_2}$**'** was more accurate than $\boldsymbol{M_{2.}}$ This finding suggests that for the complex task-oriented domain of tutoring introductory computer programming, models of human tutorial dialogue policy can be made more accurate by leveraging knowledge of both the task/subtask structure and the inferred hidden dialogue state.

## 9.2 Summary

Exploration of three tutorial dialogue corpora in the domain of introductory computer programming yielded new insights into the structure of the dialogue that occurs in this complex task-oriented domain. As expected, human tutors appear to adapt to learner characteristics such as incoming knowledge level, self-efficacy, and gender. Tutors undertake a variety of cognitive and motivational remediation. Sometimes these cognitive and motivational goals are at odds with each other, but it may be possible to positively impact both types of goals by selecting appropriate feedback. Finally, compared to a very proactive tutoring approach, allowing more autonomy may better support students' motivation.

A hidden Markov modeling framework was selected for development of computational models of the corpora because HMMs explicitly represent a stochastic layer of hidden structures. These hidden states were found in qualitative analysis to correspond to tutoring modes from the literature. Furthermore, some tutoring modes were found to correlate with student learning, indicating that HMMs can discover, in an unsupervised fashion, meaningful hidden dialogue structure.

Based on these encouraging results, HMMs were also utilized to produce prediction features within a larger set of attributes for vector-based maximum likelihood classification of student dialogue acts. The results indicate that HMM features as well as task/subtask features improved dialogue act classification for three student feedback acts. Additionally, overall the classifiers achieved performance on par with state-of-the-art dialogue act classification accuracy in less complex domains.

The flat HMM approach was extended to a hierarchical HMM that explicitly represented task/subtask structure within the computer programming exercise to predict tutorial moves within the corpus, a first step toward fully data-driven tutorial policy extraction. Both HMMs and hierarchical HMMs outperformed chance and a baseline Markov model. Hierarchical HMMs performed with highest accuracy overall for predicting tutor dialogue acts.

## 9.3 Limitations

This dissertation research was conducted based on corpora of naturalistic human one-on-one tutorial dialogue in the domain of introductory computer programming. The extent to which the exploratory findings and machine-learned models generalize to other domains in natural language dialogue, even to tutoring in other task-oriented scientific domains, has not been established and is an important direction of future work. Another limitation to generalizing the results was the low number of women and underrepresented groups participating in the studies. Low participation of underrepresented groups was not unexpected given national trends in CS enrollments (Zweben, 2008). These students are particularly important to include in investigations of the impact of intelligent tutoring systems, and active steps should be taken to include them in future studies.

Other limitations involve the design of the observational tutoring studies. First, while the tutors all had some level of experience with tutoring computer science, none had received formal training in pedagogical strategies. Another significant issue is that a control group was not included in any of the studies because of resource limitations and because holding subjects out for a control group would have decreased the sample size of students who were tutored, reducing the size of the corpora. Further, the tutoring sessions were one-time events rather than repeated interventions that took place over the academic term; therefore it is not clear what effect size these tutors would have achieved compared to classroom instruction.

Finally, some limitations involve the models chosen. As mentioned previously, the Markov model and hidden Markov modeling approaches were motivated by aspects of natural language dialogue that have been long noted in the literature. These properties include the strong local dependence of adjacent dialogue acts, which has led to the common practice in natural language dialogue research of making a first-order Markov assumption over input sequences to simplify the modeling process. However, as findings in Section 8.4 highlight, this local dependency may not be as strong in the task-oriented domain of introductory computer science tutoring as in other domains such as conversational speech. This issue raises questions about the most suitable baseline model for comparison in future work, and suggests that the HMMs presented here, because of their first-order assumption, might be improved by taking into account a longer window of dialogue history. However, creating more complex models with longer-range dependencies will require larger data sets and increased computation time for model training, a tradeoff that must be explored to determine the optimal approach.

## 9.4 Future Work

Perhaps the most important area for future work that is highlighted by this dissertation involves unsupervised dialogue modeling, in which manual annotation is completely eliminated from the data processing pipeline. Unsupervised dialogue modeling has only just begun to be explored for conversational and task-oriented dialogue, but the challenges are many. The complexities of a rich task-oriented domain further complicate the unsupervised dialogue modeling endeavor, yet the creation of successful unsupervised task-oriented dialogue models will constitute a critical step toward overcoming the very high development cost and barriers to effectiveness that are associated with the current generation of dialogue systems.

Learning models from corpora of expert human tutoring will be a key step toward creating highly effective data-driven tutorial dialogue policies. It is hoped that the modeling approaches presented in this dissertation will generalize successfully to corpora of expert human tutoring and to other domains.

While the accuracy of the dialogue act classifiers and predictors reported here are on par with those reported by other researchers in domains that do not involve a separate user-driven task, it is likely that including student characteristics such as incoming knowledge level, self-efficacy, and gender could improve the models substantially. For example, exploratory work reported here indicated that dialogue profiles of students with high self-efficacy differed substantially from those of students with lower self-efficacy. Therefore, including knowledge of the dialogue profile within a dialogue act classifier or tutorial move predictor may improve its performance.

## 9.5 Concluding Remarks

This dissertation was motivated by the author's desire to improve the state of computer science education through research into individual learner adaptation. Tutorial dialogue systems hold great promise for realizing that dream of bringing individualized instruction to every learner. While the Intelligent Tutoring Systems field is only just beginning to truly understand the impact of natural language dialogue, affect, collaboration, and other important phenomena on students' learning, the Natural Language Dialogue systems research community has begun to explore complex task-oriented domains as a focus for its work. The time is ripe for pedagogical goals to drive innovation in dialogue systems research, and this dissertation represents a first step toward this end.

# GLOSSARY

***acoustic*** – the properties of an utterance relating to its sound

***adjacency pair*** – two dialogue acts that co-occur because one establishes an expectation for the other to follow (*e.g.*, QUESTION-ANSWER, STATEMENT-ACKNOWLEDGEMENT)

***affect*** – emotion

***annotation*** – the process by which labels are applied, either manually or automatically, to raw data

***bigram*** – two adjacent observations in a sequence

***cognitive*** – relating to the information processing aspects of acquiring knowledge, skills, or understanding

***corpora*** – plural of *corpus*

***corpus*** – a collection of written or spoken material

***dialogue act*** – a communicative purpose, or action, that underlies a dialogue move (*e.g.*, provide positive feedback, agree, give a command)

***dialogue move*** **–** a turn taken within dialogue; also referred to as an *utterance*

***dialogue policy*** **–** a mapping from a set of states to a set of actions; this mapping defines the dialogue moves that a system takes

***domain*** – an area of application; in this dissertation, refers to the tutoring of introductory Java programming

***inter-rater agreement*** – also known as *inter-rater reliability;* the extent to which two or more human annotators agree in their application of an annotation scheme to a corpus

***Kappa statistic*** – a measure of inter-rater agreement that adjusts for how likely the annotators would be to agree by chance

***lexical*** – relating to words or vocabulary

***log-likelihood score*** – a measure of how likely a set of observations would be under a given model

***policy*** – see *dialogue policy*

***prosodic*** – relating to vocal stress and intonation

***self-efficacy*** – one's own belief in his or her capability to produce given levels of attainment on a particular task. This term differs from *confidence*, which is more general and may also refer to a person's certainty at failing instead of succeeding

***syntactic*** – relating to the arrangement of words and phrases within an utterance

***tagging* –** see *annotation*

***utterance* –** see *dialogue move*

# REFERENCES

Ai, H., Tetreault, J. R., & Litman, D. J. (2007). Comparing user simulation models for dialog strategy learning. *Proceedings of the North American Chapter of the Association for Computational Linguistics Human Technology Conference, Companion Volume,* Rochester, New York. 1-4.

Aleven, V., McLaren, B., Roll, I., & Koedinger, K. (2004). Toward tutoring help seeking: Applying cognitive modeling to meta-cognitive skills. *Proceedings of the 7th International Conference on Intelligent Tutoring Systems,* 227-239.

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

Aleven, V., Popescu, O., & Koedinger, K. R. (2001). Towards tutorial dialog to support self-explanation: Adding natural language understanding to a cognitive tutor. *International Conference on Artificial Intelligence in Education,* 246-255.

Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., & Stent, A. (2001). An architecture for a generic dialogue shell. *Natural Language Engineering, 6*(3&4), 213-228.

Allen, J., Ferguson, G., & Stent, A. (2001). An architecture for more realistic conversational systems. *Proceedings of the 6th International Conference on Intelligent User Interfaces,* 1-8.

Artstein, R., & Poesio, M. (2008). Inter-coder agreement for computational linguistics. *Computational Linguistics, 34*(4), 555-596.

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

Bandura, A. (1997). *Self-efficacy: The exercise of control.* Worth Publishers.

Bandura, A. (2006). Guide for constructing self-efficacy scales. In F. Pajares, & T. Urdan (Eds.), *Self-efficacy beliefs of adolescents* (pp. 307-337). Greenwich, Connecticut: Information Age Publishing.

Bangalore, S., Di Fabbrizio, G., & Stent, A. (2008). Learning the structure of task-driven human-human dialogs. *IEEE Transactions on Audio, Speech, and Language Processing, 16*(7), 1249-1259.

Bangalore, S., & Stent, A. J. (2009). Incremental parsing models for dialog task structure. *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics,* 94-102.

Barker, L. J., & Garvin-Doxas, K. (2004). Making visible the behaviors that influence learning environment: A qualitative exploration of computer science classrooms. *Computer Science Education, 14*(2), 119-145.

Ben-Ari, M. (1998). Constructivism in computer science education. *Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education,* 257-261.

Berger, A. L., Pietra, V. J. D., & Pietra, S. A. D. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics, 22*(1), 71.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.

Bloom, B. S. (1956). *Taxonomy of educational objectives: The classification of educational goals*. New York: David McKay.

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.

Boyer, K. E., Dwight, A. A., Fondren, R. T., Vouk, M. A., & Lester, J. C. (2008). A development environment for distributed synchronous collaborative programming. *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education,* Madrid, Spain. 158-162.

Boyer, K. E., Dwight, R. S., Miller, C. S., Raubenheimer, C. D., Stallmann, M. F., & Vouk, M. A. (2007). A case for smaller class size with integrated lab for introductory computer science. *The 38th SIGCSE Technical Symposium on Computer Science Education,* Covington, Kentucky. 341-345.

Boyer, K. E., Ha, E. Y., Phillips, R., Wallis, M. D., Vouk, M. A., & Lester, J. C. (In press). Dialogue act modeling in a complex task-oriented domain. To appear in *Proceedings of the 11th Annual SIGDIAL Meeting on Discourse and Dialogue,* Tokyo, Japan.

Boyer, K. E., Ha, E. Y., Wallis, M. D., Phillips, R., Vouk, M. A., & Lester, J. C. (2009). Discovering tutorial dialogue strategies with hidden Markov models. *Proceedings of the 14th International Conference on Artificial Intelligence in Education,* Brighton, U. K. 141-148.

Boyer, K. E., Lahti, W., Phillips, R., Wallis, M., Vouk, M. A., & Lester, J. C. (2010). Principles of asking effective questions to improve student problem solving. *41st SIGCSE Technical Symposium on Computer Science Education,* Milwaukee, Wisconsin. 460-464.

Boyer, K. E., Phillips, R., Ha, E. Y., Wallis, M. D., Vouk, M. A., & Lester, J. C. (2009). Modeling dialogue structure with adjacency pair analysis and hidden Markov models. *The North American Association for Computational Linguistics Human Language Technologies Conference (NAACL-HLT) Short Papers,* 49-52.

Boyer, K. E., Phillips, R., Ha, E. Y., Wallis, M. D., Vouk, M. A., & Lester, J. C. (2010a). Leveraging hidden dialogue state to select tutorial moves. *Proceedings of the NAACL Workshop on Innovative use of NLP for Building Educational Applications,* Los Angeles, California. 66-73.

Boyer, K. E., Phillips, R., Ha, E. Y., Wallis, M. D., Vouk, M. A., & Lester, J. C. (2010b). A preliminary investigation of hierarchical hidden Markov models for tutorial planning.

*Proceedings of the 3rd International Conference on Educational Data Mining,* Pittsburgh, Pennsylvania. 285-286.

Boyer, K. E., Phillips, R., Ingram, A., Ha, E. Y., Wallis, M. D., Vouk, M. A., & Lester, J. C. (2010). Characterizing the effectiveness of tutorial dialogue with hidden Markov models. *Proceedings of the 10th International Conference on Intelligent Tutoring Systems,* Pittsburgh, Pennsylvania. 55-64.

Boyer, K. E., Phillips, R., Wallis, M. D., Vouk, M. A., & Lester, J. C. (2008a). Balancing cognitive and motivational scaffolding in tutorial dialogue. *Proceedings of the 9th International Conference on Intelligent Tutoring Systems,* Montreal, Canada. 239-249.

Boyer, K. E., Phillips, R., Wallis, M. D., Vouk, M. A., & Lester, J. C. (2008b). Learner characteristics and feedback in tutorial dialogue. *Proceedings of the Third ACL Workshop on Innovative use of NLP for Building Educational Applications,* 53-61.

Boyer, K. E., Phillips, R., Wallis, M. D., Vouk, M. A., & Lester, J. C. (2009a). The impact of instructor initiative on student learning through assisted problem solving. *Proceedings of the 40th SIGCSE Technical Symposium on Computer Science Education,* Chattanooga, Tennessee. 14-18.

Boyer, K. E., Phillips, R., Wallis, M. D., Vouk, M. A., & Lester, J. C. (2009b). Investigating the role of student motivation in computer science education through one-on-one tutoring. *Computer Science Education, 19*(2), 111-135.

Boyer, K. E., Vouk, M. A., & Lester, J. C. (2007). The influence of learner characteristics on task-oriented tutorial dialogue. *Proceedings of the 13th International Conference on Artificial Intelligence in Education,* Marina del Rey, California. 365-372.

Cade, W., Copeland, J., Person, N., & D'Mello, S. (2008). Dialog modes in expert tutoring. *Proceedings of the 9th International Conference on Intelligent Tutoring Systems,* Montreal, Canada. 470-479.

Callaway, C. B., Dzikovska, M., Farrow, E., Marques-Pita, M., Matheson, C., & Moore, J. D. (2007). The Beetle and BeeDiff tutoring systems. *Proceedings of the SLaTE Workshop on Speech and Language Technology in Education.*

Cameron, J., & Pierce, W. D. (1994). Reinforcement, reward, and intrinsic motivation: A meta-analysis. *Review of Educational Research, 64*(3), 363.

Carletta, J. (1996). Assessing agreement on classification tasks: The Kappa statistic. *Computational Linguistics, 22*(2), 249-254.

Chi, M., Jordan, P., VanLehn, K., & Hall, M. (2008). Reinforcement learning-based feature selection for developing pedagogically effective tutorial dialogue tactics. *Proceedings of the 1st International Conference on Educational Data Mining,* Montreal, Canada. 258-265.

Chi, M., Jordan, P., VanLehn, K., & Litman, D. (2009). To elicit or to tell: Does it matter? *Proceedings of the 14th International Conference on Artificial Intelligence in Education,* 197-204.

Chi, M., VanLehn, K., & Litman, D. (2010). Do micro-level tutorial decisions matter: Applying reinforcement learning to induce pedagogical tutorial tactics. *Proceedings of the 10th International Conference on Intelligent Tutoring Systems*, 224-234.

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

Chi, M. T. H., Roy, M., & Hausmann, R. G. M. (2008). Observing tutorial dialogues collaboratively: Insights about human tutoring effectiveness from vicarious learning. *Cognitive Science, 32*(2), 301-341.

Chi, M. T. H., Siler, S. A., Jeong, H., Yamauchi, T., & Hausmann, R. G. (2001). Learning from human tutoring. *Cognitive Science, 25*(4), 471-533.

Chotimongkol, A. (2008). *Learning the structure of task-oriented conversations from the corpus of in-domain dialogs.* (Unpublished Ph. D. Dissertation). Carnegie Mellon University School of Computer Science.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement, 20*(1), 37-46.

Cohen, P. A., Kulik, J. A., & Kulik, C. L. C. (1982). Educational outcomes of tutoring: A meta-analysis of findings. *American Educational Research Journal, 19*(2), 237-248.

Core, M., & Allen, J. (1997). Coding dialogs with the DAMSL annotation scheme. *AAAI Fall Symposium on Communicative Action in Humans and Machines,* 28–35.

Davis, M. H. (1983). Measuring individual differences in empathy: Evidence for a multidimensional approach. *Journal of Personality and Social Psychology, 44*(1), 113-126.

De Marneffe, M. C., MacCartney, B., & Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006),* Genoa, Italy.

De Veaux, R. D., Velleman, P. F., & Bock, D. E. (2005). *Stats: Data and models.* Addison-Wesley Longman.

Deci, E. L., Koestner, R., & Ryan, R. M. (2001). Extrinsic rewards and intrinsic motivation in education: Reconsidered once again. *Review of Educational Research, 71*(1), 1-27.

Dickinson, L. (1995). Autonomy and motivation: A literature review. *System, 23*(2), 165-174.

Dzikovska, M. O., Callaway, C. B., Farrow, E., Marques-Pita, M., Matheson, C., & Moore, J. D. (2006). Adaptive tutorial dialogue systems using deep NLP techniques. *The Annual Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies (NAACL-HLT),* 5-10.

Elliot, A. J., & McGregor, H. A. (2001). A 2 x 2 achievement goal framework. *Journal of Personality and Social Psychology, 80*(3), 501-519.

Evens, M., & Michael, J. (2006). *One-on-one tutoring by humans and computers*. Mahwah, New Jersey: Lawrence Erlbaum Associates.

Fine, S., Singer, Y., & Tishby, N. (1998). The hierarchical hidden Markov model: Analysis and applications. *Machine Learning, 32*(1), 41-62.

Forbes-Riley, K., & Litman, D. (2005). Using bigrams to identify relationships between student certainness states and tutor responses in a spoken dialogue corpus. *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue,* 87-96.

Forbes-Riley, K., Litman, D., Huettner, A., & Ward, A. (2005). Dialogue-learning correlations in spoken dialogue tutoring. *Proceedings of the 12th International Conference on Artificial Intelligence in Education,* Amsterdam. 225-232.

Forbes-Riley, K., & Litman, D. J. (2005). Using bigrams to identify relationships between student certainness states and tutor responses in a spoken dialogue corpus. *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue,* 87-96.

Forbes-Riley, K., Rotaru, M., Litman, D. J., & Tetreault, J. (2007). Exploring affect-context dependencies for adaptive system development. *Proceedings of NAACL HLT (Short Papers),* 41-44.

Forbes-Riley, K., & Litman, D. (2009). Adapting to student uncertainty improves tutoring dialogues. *Proceedings of the 14th International Conference on Artificial Intelligence and Education,* 33-40.

Fossati, D., Eugenio, B. D., Brown, C., & Ohlsson, S. (2008). Learning linked lists: Experiments with the iList system. *Proceedings of the 9th International Conference on Intelligent Tutoring Systems* 80-89.

Fossati, D., Di Eugenio, B., Brown, C., Ohlsson, S., Cosejo, D., & Chen, L. (2009). Supporting computer science curriculum: Exploring and learning linked lists with iList. *IEEE Transactions on Learning Technologies, 2*(2), 107-120.

Fossati, D., Di Eugenio, B., Ohlsson, S., Brown, C., & Chen, L. (2010). Generating proactive feedback to help students stay on track. *Proceedings of the 10th International Conference on Intelligent Tutoring Systems,* 315-317.

Fox, B. A. (1993). *The human tutorial dialogue project*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Frampton, M., & Lemon, O. (2009). Recent research advances in reinforcement learning in spoken dialogue systems. *The Knowledge Engineering Review, 24*(4), 375-408.

Glass, M., Kim, J. H., Evens, M. W., Michael, J. A., & Rovick, A. A. (1999). Novice vs. expert tutors: A comparison of style. *The 10th Midwest Artificial Intelligence and Cognitive Science Conference,* 43-49.

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

Graesser, A. C., Lu, S., Jackson, G. T., Mitchell, H. H., Ventura, M., Olney, A., & Louwerse, M. M. (2004). AutoTutor: A tutor with dialogue in natural language. *Behavior Research Methods Instruments and Computers, 36*(2), 180-192.

Graesser, A. C., & Person, N. K. (1994). Question asking during tutoring. *American Educational Research Journal, 31*(1), 104.

Graesser, A. C., Person, N. K., & Magliano, J. P. (1995). Collaborative dialogue patterns in naturalistic one-to-one tutoring. *Applied Cognitive Psychology, 9*(6), 495-522.

Graesser, A. C., Rus, V., & Cai, Z. (2008). Question classification schemes. *Proceedings of the 1st Workshop on Question Generation,* Arlington, Virginia.

Graesser, A. C., Wiemer-Hastings, K., Wiemer-Hastings, P., & Kreuz, R. (1999). AutoTutor: A simulation of a human tutor. *Cognitive Systems Research, 1*(1), 35-51.

Guzdial, M., & Tew, A. E. (2006). Imagineering inauthentic legitimate peripheral participation: An instructional design approach for motivating computing education. *Proceedings of the Second International Computing Education Research Workshop (ICER),* Canterbury, United Kingdom. 51-58.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. (2009). The WEKA data mining software: An update. *SIGKDD Explorations, 11*(1), 10-18.

Hardy, H., Biermann, A., Inouye, R. B., McKenzie, A., Strzalkowski, T., Ursu, C., Webb, N., & Wu, M. (2006). The Amitiés system: Data-driven techniques for automated dialogue. *Speech Communication, 48*(3-4), 354-373.

Hausmann, R. G., Chi, M. T. H., & Roy, M. (2004). Learning from collaborative problem solving: An analysis of three hypothesized mechanisms. *Conference of the Cognitive Science Society,* 547-552.

Heeman, P. A. (2007). Combining reinforcement learning with information-state update rules. *Proceedings of NAACL HLT,* 268-275.

Henderson, J., Lemon, O., & Georgila, K. (2008). Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics, 34*(4), 487-511.

Ho, C. W., Raha, S., Gehringer, E., & Williams, L. (2004). Sangam: A distributed pair programming plug-in for eclipse. *Proceedings of the OOPSLA Workshop on Eclipse Technology Exchange,* 73-77.

Holt, P., Dubs, S., Jones, M., & Greer, J. (1994). The state of student modelling. *Student Modelling: The Key to Individualized Knowledge-Based Instruction,* 3-35.

Jackson, G. T., & Graesser, A. C. (2007). Content matters: An investigation of feedback categories within an ITS. *Proceedings of the 13th International Conference on Artificial Intelligence in Education,* 127-134.

Johnson, W. L., & Soloway, E. (1985). PROUST: An automatic debugger for PASCAL programs. *BYTE, 10*(4), 179-190.

Jordan, P., Makatchev, M., Pappuswamy, U., VanLehn, K., & Albacete, P. (2006). A natural language tutorial dialogue system for physics. *Proceedings of the Florida Artificial Intelligence Research Society (FLAIRS) Conference,* 521-526.

Jurafsky, D., & Martin, J. H. (2008). *Speech and language processing*. Upper Saddle River, New Jersey: Pearson.

Keller, J. M. (1983). Motivational design of instruction. In C. M. Reigeluth (Ed.), *Instructional-design theories and models* (pp. 386-434) Hillsdale: Erlbaum.

Kelly, D., & Weibelzahl, S. (2006). Raising confidence levels using motivational contingency design techniques. *Proceedings of the 8th International Conference on Artificial Intelligence in Education,* 535-544.

Kersey, C., Di Eugenio, B., Jordan, P., & Katz, S. (2009). KSC-PaL: A peer learning agent that encourages students to take the initiative. *Proceedings of the NAACL HLT Workshop on Innovative use of NLP for Building Educational Applications,* Boulder, Colorado. 55-63.

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*(1), 30-43.

Kumar, R., Rosé, C., Aleven, V., Iglesias, A., & Robinson, A. (2006). Evaluating the effectiveness of tutorial dialogue instruction in an exploratory learning context. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems,* 666-674.

Kumar, R., Rosé, C. P., Yi-Chia, W., Joshi, M., & Robinson, A. (2007). Tutorial dialogue as adaptive collaborative learning support. *Proceedings of the International Conference on Artificial Intelligence in Education,* 383-393.

Landis, J. R., & Koch, G. (1977). The measurement of observer agreement for categorical data. *Biometrics, 33*(1), 159-174.

Lane, H. C. (2004). *Natural language tutoring and the novice programmer.* (Unpublished Ph. D. Dissertation). University of Pittsburgh Department of Computer Science.

Lane, H. C., & VanLehn, K. (2004). A dialogue-based tutoring system for beginning programming. *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS),* 449–454.

Lane, H. C., & VanLehn, K. (2005). Teaching the tacit knowledge of programming to novices with natural language tutoring. *Computer Science Education, 15*(3), 183-201.

Layman, L., Williams, L., & Slaten, K. (2007). Note to self: Make assignments meaningful. *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education,* Covington, Kentucky. 459-463.

Lepper, M. R., Woolverton, M., Mumme, D. L., & Gurtner, J. L. (1993). Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors. In S. P. Lajoie, & S. J. Derry (Eds.), *Computers as cognitive tools* (pp. 75-105). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Levin, E., Pieraccini, R., & Eckert, W. (2000). A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing, 8*(1), 11-23.

Lister, R., Berglund, A., Box, I., Cope, C., Pears, A., Avram, C., Bower, M., Carbone, A., Davey, B., & de Raadt, M. (2007). Differing ways that computing academics understand teaching. *Proceedings of the Ninth Australasian Conference on Computing Education,* 97-106.

Litman, D., & Forbes-Riley, K. (2006). Correlations between dialogue acts and learning in spoken tutoring dialogues. *Natural Language Engineering, 12*(2), 161-176.

Litman, D., Moore, J., Dzikovska, M., & Farrow, E. (2009). Using natural language processing to analyze tutorial dialogue corpora across domains and modalities. *Proceedings of the 14th International Conference on Artificial Intelligence in Education,* 149-156.

Litman, D. J., Rosé, C. P., Forbes-Riley, K., VanLehn, K., Bhembe, D., & Silliman, S. (2006). Spoken versus typed human and computer dialogue tutoring. *International Journal of Artificial Intelligence in Education, 16*(2), 145-170.

Litman, D., & Silliman, S. (2004). ITSPOKE: An intelligent tutoring spoken dialogue system. *Proceedings of the North American Association for Computational Linguistics and Human Language Technologies Conference (NAACL HLT),* 5-8.

Loper, E., & Bird, S. (2004). NLTK: The natural language toolkit. *Proceedings of the ACL Demonstration Session,* Barcelona, Spain. 214-217.

Machanick, P. (2007). A social construction approach to computer science education. *Computer Science Education, 17*(1), 1-20.

Marineau, J., Wiemer-Hastings, P., Harter, D., Olde, B., Chipman, P., Karnavat, A., Pomeroy, V., Rajan, S., & Graesser, A. (2000). Classification of speech acts in tutorial dialog. *Proceedings of the ITS 2000 Workshop on Modelling Human Teaching Tactics and Strategies,* 65-71.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., & Balik, S. (2003). Improving the CS1 experience with pair programming. *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education,* Reno, Nevada. 359-362.

Ohlsson, S. (1994). Constraint-based student modeling. In J. E. Greer, & G. I. McCalla (Eds.), *Student modelling: The key to individualized knowledge-based instruction* (pp. 167-189). Berlin: Springer-Verlag.

Ohlsson, S., Di Eugenio, B., Chow, B., Fossati, D., Lu, X., & Kershaw, T. C. (2007). Beyond the code-and-count analysis of tutoring dialogues. *Proceedings of the 13th International Conference on Artificial Intelligence in Education,* 349-356.

Porayska-Pomsta, K., & Pain, H. (2004). Providing cognitive and affective scaffolding through teaching strategies: Applying linguistic politeness to the educational context. *Proceedings of the 7th International Conference on Intelligent Tutoring Systems,* Alagoas, Brazil. 77-86.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program, 14*(3), 130-137.

Purandare, A., & Litman, D. (2008). Content-learning correlations in spoken tutoring dialogs at word, turn and discourse levels. *Proceedings of the 21st International FLAIRS Conference,* 195-200.

Purver, M., Kording, K. P., Griffiths, T. L., & Tenenbaum, J. B. (2006). Unsupervised topic modelling for multi-party spoken discourse. *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL,* Sydney, Australia. 17-24.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77*(2), 257-286.

Rebolledo-Mendez, G., du Boulay, B., & Luckin, R. (2006). Motivating the learner: An empirical evaluation. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems,* Jhongli, Taiwan. 545-554.

Rosé, C., Bhembe, D., Siler, S., Srivastava, R., & VanLehn, K. (2003). The role of why questions in effective human tutoring. *Proceedings of the International Conference on Artificial Intelligence in Education,* 55-62.

Rosé, C. P., Aleven, V., & Torrey, C. (2004). CycleTalk: Supporting reflection in design scenarios with negotiation dialogue. *CHI Workshop on Designing for Reflective Practitioners: Sharing and Assessing Progress by Diverse Communities.*

Rosé, C. P., Moore, J. D., VanLehn, K., & Allbritton, D. (2001). A comparative evaluation of Socratic versus didactic tutoring. *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society,* 897–902.

Rosé, C. P., Torrey, C., Aleven, V., Robinson, A., Wu, C., & Forbus, K. (2004). CycleTalk: Toward a dialogue agent that guides design with an articulate simulator. *Proceedings of the 7th International Conference on Intelligent Tutoring Systems,* 401-411.

Schegloff, E., & Sacks, H. (1973). Opening up closings. *Semiotica, 7*(4), 289-327.

Scott, S. L. (2002). Bayesian methods for hidden Markov models: Recursive computing in the 21st century. *Journal of the American Statistical Association, 97*(457), 337-352.

Shute, V. J. (2007). *Focus on formative feedback*. Princeton, NJ: ETS.

Singh, S., Litman, D., Kearns, M., & Walker, M. (2002). Optimizing dialogue management with reinforcement learning:  Experiments with the NJFun system. *Journal of Artificial Intelligence Research, 16*, 105-133.

Slaten, K. M., Droujkova, M., Berenson, S., Williams, L., & Layman, L. (2005). Undergraduate student perceptions of pair programming and agile software methodologies:  Verifying a model of social interaction. *Proceedings of AGILE,* Denver, Colorado. 323-330.

Soh, L. K., Samal, A., & Nugent, G. (2007). An integrated framework for improved computer science education: Strategies, implementations, and results. *Computer Science Education, 17*(1), 59-83.

Sridhar, V. K. R., Bangalore, S., & Narayanan, S. (2009). Combining lexical, syntactic and prosodic cues for improved online dialog act tagging. *Computer Speech & Language, 23*(4), 407-422.

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Van Ess-Dykema, C., & Meteer, M. (2000). Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics, 26*(3), 339-373.

Tan, J., & Biswas, G. (2006). The role of feedback in preparation for future learning: A case study in learning by teaching environments. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems,* Jhongli, Taiwan. 370-381.

Tetreault, J. R., & Litman, D. J. (2008). A reinforcement learning approach to evaluating state representations in spoken dialogue systems. *Speech Communication, 50*(8-9), 683-696.

Toney, D., Moore, J., & Lemon, O. (2006). Evolving optimal inspectable strategies for spoken dialogue systems. *Proceedings of the NAACL Human Language Technology Conference, Companion Volume,* 173-176.

VanLehn, K. (2008). The interaction plateau. Keynote talk presented at the *Proceedings of the 9th International Conference on Intelligent Tutoring Systems,* Montreal, Canada.

VanLehn, K., Graesser, A. C., Jackson, G. T., Jordan, P., Olney, A., & Rosé, C. P. (2007). When are tutorial dialogues more effective than reading? *Cognitive Science, 31*(1), 3-62.

VanLehn, K., Jordan, P. W., Rosé, C. P., Bhembe, D., Bottner, M., Gaydos, A., Makatchev, M., Pappuswamy, U., Ringenberg, M., & Roque, A. (2002). The architecture of Why2-atlas: A coach for qualitative physics essay writing. *Proceedings of the International Conference on Intelligent Tutoring Systems,* 158-167.

Walker, M., & Whittaker, S. (1990). Mixed initiative in dialogue: An investigation into discourse segmentation. *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics,* Pittsburgh, Pennsylvania. 70-78.

Wang, N., Johnson, W. L., Rizzo, P., Shaw, E., & Mayer, R. E. (2005). Experimental evaluation of polite interaction tactics for pedagogical agents. *Proceedings of the 10th International Conference on Intelligent User Interfaces,* San Diego, California. 12-19.

Ward, A., & Litman, D. (2006). Cohesion and learning in a tutorial spoken dialog system. *Proceedings of FLAIRS,* Melbourne Beach, FL. 533-538.

Wenger, E. (1987). *Artificial intelligence and tutoring systems*. Los Altos, California: Morgan Kaufmann.

Wiedenbeck, S. (2005). Factors affecting the success of non-majors in learning to program. *Proceedings of the First International Computing Education Research Workshop (ICER),* Seattle, Washington. 13-24.

Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In support of pair programming in the introductory computer science course. *Computer Science Education, 12*(3), 197-212.

Wolfe, J. (2004). Why the rhetoric of CS programming assignments matters. *Computer Science Education, 14*(2), 147-163.

Wright Hastie, H., Poesio, M., & Isard, S. (2002). Automatically predicting dialogue structure using prosodic features. *Speech Communication, 36*(1-2), 63-79.

Young, S. (2000). Probabilistic methods in spoken-dialogue systems. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences, 358(1769)*, 1389-1402.

Young, S., Gasic, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., & Yu, K. (2009). The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language, 24*(2), 150-174.

Zhou, Y., & Evens, M. W. (1999). A practical student model in an intelligent tutoring system. *The 11th IEEE International Conference on Educational Tools with Artificial Intelligence,* 13-18.

Zinn, C., Moore, J. D., & Core, M. G. (2002). A 3-tier planning architecture for managing tutorial dialogue. *Proceedings of the 6th International Conference on Intelligent Tutoring Systems,* 574-584.

Zweben, S. (2008). Undergraduate enrollment in computer science trends higher; doctoral production continues at peak levels. *Computing Degree and Enrollment Trends from the 2007-2008 Taulbee Survey.*

# APPENDICES

# APPENDIX A: Select materials for Study I

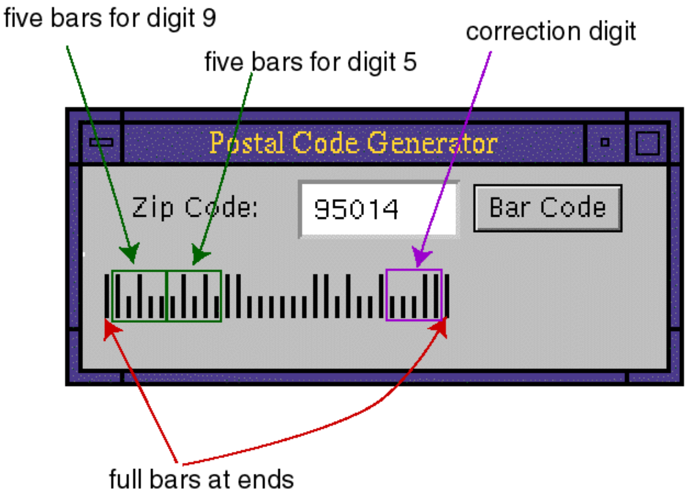
*Table 11. Programming exercise for pilot studies I and II*[35]

## Postal Bar Codes

The Problem:

For faster sorting of letters, the United States Postal Service encourages companies that send large volumes of mail to use a bar code denoting the ZIP code. Using the skeleton GUI program provided for you, you will complete this lab with code to actually generate the bar code for a given zip code.

More About Bar Codes:
In postal bar codes, there is a full-height frame bar on each end (and these are drawn automatically by the program provided for you; you don't have to write code to draw these). Each of the five encoded digits is represented by five bars. The five encoded digits are followed by a correction digit.



Zip Code: 95014

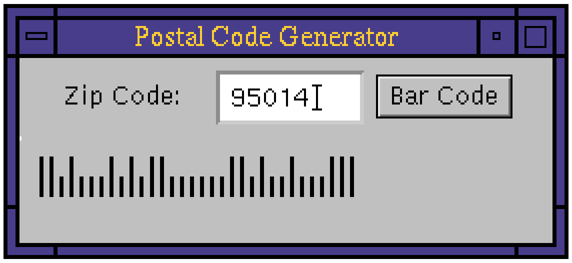
[35] Adapted directly from NC State University CSC 116 laboratory manual, Spring 2006

The Correction Digit

The correction digit is computed as follows: Add up all digits, and choose the correct digit to make the sum a multiple of 10.For example, the ZIP code 95014 has sum of digits 19, so the correction digit is 1 to make the sum equal to 20.

What's Already Written?

You can see what parts of this program are already written by running the file Main. java. When you do, you should see output like the image below, with a blank zip code slot. You can enter a zip code, and you should see that no bar code is generated (except the first and last full bars which are required for all bar codes).



What's Your Task?

Your job is to take this five-digit zip code and use it to generate a bar code. The PostalFrame class is the one which handles this task. The three methods which you must complete are:

extractDigits()<br>calculateAndDrawCDigit()<br>drawZIPCode()

For extractDigits(), you will need to add a private variable to the class which stores the zip code as separate digits.

Some Helpful Information

- If you can't remember how to do something with the software, please refer to the reference sheet on your desk.

- This lab involves a package named postal. This package contains classes Bar, FullBar, PostalBarCode, and SmallBar. The reason these classes are grouped into a package, is that the classes of the postal package logically belong together to accomplish a task. Whenever you need to use things from one package outside of that package, you just import the package. This has already been done for you in Main and PostalFrame – you will see the statement import postal. * at the top. In addition to code already provided, you will need to call methods in the PostalBarCode class from your PostalFrame class to draw full and small bars.

- Each digit of the ZIP code and the correction digit are encoded according to the following table (each digit has five bars -- a zero is a half bar and a one is a full bar). This scheme represents all combinations of two full and three half bars.

| Digit | |
|---|---|
| 0 | 1 1 0 00 |
| 1 | 0 00 1 1 |
| 2 | 0 0 1 0 1 |
| 3 | 0 0 1 1 0 |
| 4 | 0 1 0 0 1 |
| 5 | 0 1 0 1 0 |
| 6 | 0 1 1 0 0 |
| 7 | 1 0 00 1 |
| 8 | 1 0 0 1 0 |
| 9 | 1 0 1 0 0 |

*Table 12. Student pre-survey for pilot study I*

Please rate how certain you are that you can complete a one-hour laboratory assignment in the situations described below.

*Rate your degree of confidence by recording a number from 0 to 100 using the scale given below:*

| 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|
| Cannot do at all | | | | | Moderately certain can do | | | Highly certain can do | | |

| | **Confidence (0-100)** |
|---|---|
| Complete a simple lab on my own | _______ |
| Complete a challenging lab on my own | _______ |
| Complete a challenging lab if I am paired with a classmate | _______ |
| Complete a challenging lab if I work with an educational robot designed to act as my programming partner | _______ |

*Please rate the degree to which you agree or disagree with the following statements using the scale given below:*

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Strongly Disagree | Mostly Disagree | Mostly Agree | Strongly Agree |

1. I usually enjoy CSC 116 labs.
2. I usually find CSC 116 labs easy.
3. I already understand the material I will need for today’s lab.
4. I am often frustrated by CSC 116 labs.

*Table 13. Student post-survey for pilot study I*

Please rate how certain you are that you can complete a one-hour laboratory assignment in the situations described below.

*Rate your degree of confidence by recording a number from 0 to 100 using the scale given below:*

| 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|
| Cannot do at all | | | | | Moderately certain can do | | | | | Highly certain can do |

Please rate how certain you are that you could complete a future one-hour laboratory assignment in the situations described below.

| | Confidence (0-100) |
|---|---|
| Complete a simple lab using the system | ______ |
| Complete a challenging lab using the system | ______ |

*Please rate the degree to which you agree or disagree with the following statements using the scale given below:*

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Strongly Disagree | Mostly Disagree | Mostly Agree | Strongly Agree |

5. I enjoyed today's lab.
6. Using the system did not save me any time in completing the lab assignment compared to working on my own.
7. Today's lab assignment was frustrating.
8. If given the chance, I would use the system again in a lab.
9. Today's lab problem was challenging for me.
10. Using the system helped me understand the material better than if I had worked on my own.
11. The system is difficult to use.
12. If given the chance, I would use the system on my own time for programming projects.

*Please choose one answer for each question.*

13. My programming partner was:
    a. far less skilled than me
    b. a little less skilled than me
    c. about the same skill level as me
    d. a little more skilled than me
    e. far more skilled than me

14. My programming partner:
    a. asked too many questions
    b. asked just the right amount of questions
    c. should have asked more questions

15. When I asked a question, my programming partner:
    a. usually responded with a helpful answer
    b. usually responded with an answer that was not helpful
    c. usually did not respond to my question
    d. I did not ask any questions

16. I believe my programming partner was a(n):
    a. educational software robot
    b. professor
    c. graduate student
    d. CSC 116 classmate
    e. other: ________________

17. I imagined my programmer partner's race was:
    a. Caucasian
    b. African American
    c. Native American
    d. Hispanic
    e. Asian
    f. Other: ________________
    g. I did not imagine my partner's race

18. I imagined my programmer partner's gender was:
    a. Male
    b. Female
    c. I did not imagine my partner's gender

19. I imagined my programming partner's age was:

a. 18-20
b. 21-22
c. 23-25
d. 26-30
e. 31-40
f. 41-50
g. over 50
h. I did not imagine my partner's age

20. What I liked least about the system was: ________________________________.
21. What I liked best about the system was: _________________________________.
22. Additional Comments:

*Table 14. Student pre-test for pilot study I*

Circle the best answer for each question.

1. Consider the following line of java code:

   ```
   int z;
   ```

   Is z a primitive type, or an object?

   a) Primitive type
   b) Object
   c) Neither

2. Consider the following line of java code:

   ```
   String s;
   ```

   Is s a primitive type, or an object?

   a) Primitive type
   b) Object
   c) Neither

3. Which of these is true of an array in java?
   a) It's exactly the same thing as a String
   b) It's an object which stores more than one value and can be indexed to access the values
   c) It's an object whose size you never have to declare
   d) It's a primitive type which can only hold three or less integers
   e) Both a and c
   f) Both b and d
   g) None of these

4. Which of the following statements correctly converts int x to a String?
   a) String s = x. toString();
   b) String s = Integer. parseInt(x);
   c) String s = x + “”;

d) Both a and c  
e) Both b and c  
f) None of these

5. Which of the following statements correctly accesses the nth character in String s?  
a) s [n]  
b) s. charAt(n)  
c) s. length  
d) both a and b  
e) both a and c  
f) None of these

6. Consider a String s = “90210”. Which of the following statements returns the ‘1’ in this String?  
a) s. charAt(1);  
b) s. charAt(2);  
c) s. charAt(3);  
d) s. charAt(4);  
e) s. length();  
f) None of these

7. Consider the following code:

```
for (int  k = 0; _______; k++)
system. out. println(array[k]);
```

How can you fill in the blank to correctly traverse an array of 11 values?  
a) k<11  
b) k<=11  
c) k> 10  
d) k< 10  
e) k<= 10  
f) both b and c  
g) both a and e  
h) None of these

8. Write java code to declare and instantiate an array named bools of 30 booleans.  
a) arraybools = new array of Booleans;  
b) int [] bools = new int[30];

c) Boolean [] bools = new Boolean[30];
d) Any of these will work
e) None of these

9. Imagine you have a five-digit number, such as 32,451. Which of the following finds the remainder when this number is divided by 10?

a) 32451 & 10
b) 32451 / 10
c) 32451 % 10
d) 32451 – 10
e) None of these

10.Imagine you have a three-digit number such as 798. What is the result of performing the java statement 798 / 100?

a) 79. 8
b) 79
c) 7
d) 7. 98

*Table 15. Student post-test for pilot study I*

Circle the best answer for each question.

1. True or False: int in java is a primitive type

2. True or False: String in java is a primitive type

3. When might we use arrays in java?
   a) To hold a bunch of characters, but never numbers.
   b) To hold a set of values of any type so we can index into the array and retrieve them.
   c) When we do not know how large a set of values we need to hold.
   d) When we need to store five or less integers
   e) Both a and c
   f) Both b and d
   g) None of these

4. You have an int x = 31. What type should the variable y be in order to legally perform y = Integer. parseInt[x]?
   a) int
   b) Integer
   c) String
   d) Char []
   e) None of these

5. Consider an array:

   int [] x = new int [40];

   How do we access the first element in this array?

   a) x[0]
   b) x[1]
   c) x. charAt(0)

d) x. intAt(1)
e) None of these

6. Consider a String s = “54321”. What is the result of the statement s. charAt(2)?
   a) 5
   b) 4
   c) 3
   d) 2
   e) 1
   f) None of these

7. Consider the following code:

```
int [] a = new int [23];
```

   Fill in the blank in the following code to traverse array a.

```
for (int c = ___; c < 23;  c++)
     System. out. println(a[c]);
```

   a) 1
   b) 23
   c) 0
   d) false
   e) None of these

8. Write java code to declare an instantiate an array named chars of 30 chars.
   a) char [] 30 = new char;
   b) char [] chars = new chars;
   c) array chars = new array (char);
   d) char [] chars = new char[30];
   e) Any of these will work
   f) None of these

9. Imagine you have a five-digit number, such as 47,998. What is the result of the java statement 47998 % 10?

a) 4
b) 7
c) 9
d) 8
e) None of these

10. Imagine you have a three-digit number such as 364. You want to extract the first digit, the 3, and store it into an int d. How can you do this?
    a) int d = 364 % 10;
    b) int d = 364 / 10;
    c) int d = 364 – 10;
    d) int d = 364 & 10;
    e) None of these

# APPENDIX B: Materials for Main Study

*Table 16. Programming exercise for main study*

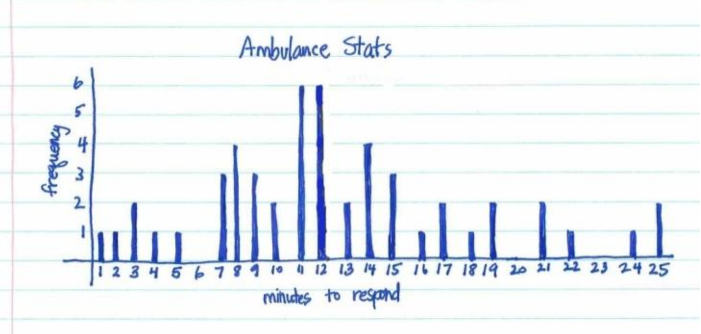
***Waimea County Ambulance Study***
**Problem Description**

To help ensure the safety of their residents, the Waimea County Emergency Response office is re-assessing their ambulance dispatch system. A study has already been conducted to gather data about the ambulance response times to 911 calls. You have been hired to analyze this data and help the emergency response office answer some questions about how quickly their ambulances are able to reach people in need. You'll be taking over for Maddie, the previous developer who was recently promoted.

Maddie already completed the class called Ambulance. java, which is a driver for the whole program (it contains the main method). She also completed AmbulanceGUI. java, which is used for displaying the ambulance response times graphically. You just need to complete a few methods in the AmbulanceData class to finish this project!

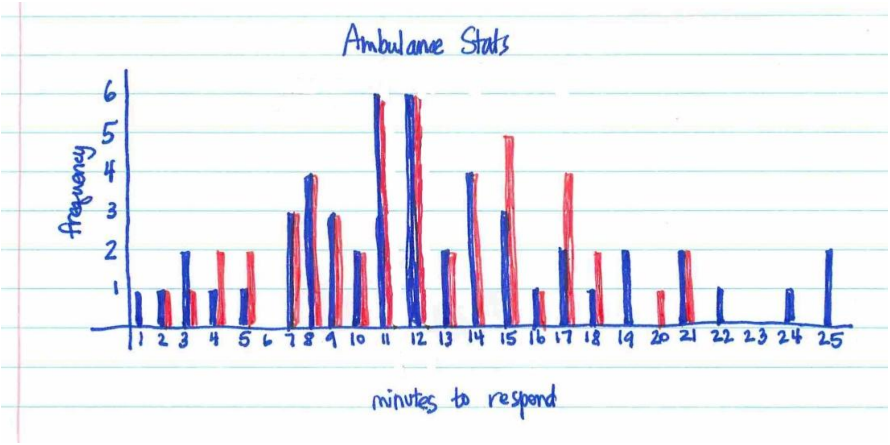
1. In the AmbulanceData class, you must complete the method plotTimes() so that all the ambulance response times in the parameter array (arrayToPlot) are displayed on a graph. Maddie already created the method outline with some comments, so you'll just need to read her comments and fill in the method.

   Maddie had an intern draw a graph by hand for the response times. This way, you know what the output of your program is supposed to look like. The x-axis is how many minutes an ambulance took to respond, the y-axis is a count of how many of the response times in the data set took that long. For instance, there were three ambulance responses that took 7 minutes.



2. The department is considering replacing its aging fleet with new ambulances. Because of the county's tight budget, these would be slightly slower ambulances than the current fleet but the county could afford more ambulances overall. The staff believe the effects of this change would be:
   - On all response times below 5 minutes, the new fleet would take 1 minute longer to respond.
   - On all response times above 18 minutes, the new fleet would take 4 fewer minutes to respond.
   - Other response times would remain the same.

   Complete the method newFleetProjections() which creates a new array of hypothetical response times given the above effects of the new fleet. You will need to create a new array because you must not overwrite the true response times in the original array.



3. There is more analysis work than Maddie originally thought, so one of your colleagues, Shannon, is writing a set of methods that perform the statistical analysis so your group can give a detailed report to the Waimea County authorities. Shannon's code needs to be able to pass an array of unsorted times to a sortArray method, and get back an array of sorted times. Write a method called sortArray in the AmbulanceData class. The sortArray method

should take an array of doubles as a parameter, and return a sorted ascending version of the parameter array *without overwriting the contents* of the original array.

The next page has some details of how your sort method should work.

*Table 17. Student pre-survey for main study*

Please rate how certain you are that you can do each of the things described below by writing the appropriate number.

*Rate your degree of confidence by recording a number from 0 to 100 using the scale given below:*

| | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cannot do<br>at all | | | | | Moderately<br>can do | | | | | Highly certain<br>Can do | | |

| | **Confidence<br>(0-100)** |
|---|---|
| Learn Computer Science. | ________ |
| Learn CSC 116 course material. | ________ |
| Complete a *simple* programming exercise on my own. | ________ |
| Complete a *challenging* programming exercise on my own. | ________ |
| Complete a *challenging* programming exercise if I am in a lab where a TA is available to help me. | ________ |
| Explain for-loops to others well. | ________ |
| Explain arrays to others well. | ________ |
| Explain method calls to others well. | ________ |
| Use for-loops in a programming exercise correctly and effectively. | ________ |
| Use arrays in a programming exercise correctly and effectively. | ________ |
| Make method calls correctly and effectively. | ________ |

*Please rate the degree to which you agree or disagree with the following statements:*

| | Not At All | | Moderate | | Very Much |
|---|---|---|---|---|---|
| I usually enjoy CSC 116 course material. | ○ | ○ | ○ | ○ | ○ |
| I usually find CSC 116 exercises challenging. | ○ | ○ | ○ | ○ | ○ |
| I understand for-loops. | ○ | ○ | ○ | ○ | ○ |
| I understand arrays. | ○ | ○ | ○ | ○ | ○ |
| I am experienced using the eclipse development environment. | ○ | ○ | ○ | ○ | ○ |
| I am experienced using the eclipse development environment. | ○ | ○ | ○ | ○ | ○ |

*Table 18. Pre/post-test for main study*

*Complete each of the following problems to the best of your ability. Even if you do not know how to completely answer the question, fill in as much as you know.*

1. Write a chunk of Java code to accomplish each of these tasks:

    a. Declare an array of integer type and give it an initial size of 100.

    b. Test the $i^{th}$ element of the array you declared in part a of this question and print “true” if the element is equal to 5 and “false” otherwise. Assume that i has already been declared and initialized.

    c. Set the $i^{th}$ element of the array you declared in part a of this question to be 5. Again, assume that i has already been declared and initialized.

2. Write a piece of Java code that prints “Cowabunga!” exactly 73 times. System. out. println can be used to print the string.

3. In a Java program, an array named firstArray of type int has been created and initialized. Write a line of Java code to create an array named secondArray that is the same size and same type as firstArray. The contents of secondArray do not need to be initialized to be the same as the contents of firstArray.

4. An array a has already been declared as an array of integers in Java.
    a. Assuming i and j are integers in the range of a, write a piece of Java code to swap the values located at a[i] and a[j].

    b. Write a piece of Java code to print the elements of a in the order they appear in the array.

    c. Write a piece of Java code to print the elements of a in the reverse order they appear in the array.

*Table 19. Student post-survey for main study*

Please rate how certain you are that you can do each of the things described below by writing the appropriate number.

*Rate your degree of confidence by recording a number from 0 to 100 using the scale given below:*

| | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cannot do<br>at all | | | | | Moderately<br>can do | | | | | Highly certain<br>Can do | | |

| | **Confidence (0-100)** |
|---|---|
| Learn Computer Science. | ________ |
| Learn CSC 116 course material. | ________ |
| Complete a *simple* programming exercise on my own. | ________ |
| Complete a *challenging* programming exercise on my own. | ________ |
| Complete a *challenging* programming exercise if I am in a lab where a TA is available to help me. | ________ |
| Explain for-loops to others well. | ________ |
| Explain arrays to others well. | ________ |
| Explain method calls to others well. | ________ |
| Use for-loops in a programming exercise correctly and effectively. | ________ |
| Use arrays in a programming exercise correctly and effectively. | ________ |
| Make method calls in a programming exercise correctly and effectively. | ________ |
| | |

*Please rate the degree to which you agree or disagree with the following statements:*

Not at all Moderate Very much

| | | | | | |
|---|---|---|---|---|---|
| I enjoyed today's programming exercise. | ○ | ○ | ○ | ○ | ○ |
| Today's programming exercise was frustrating. | ○ | ○ | ○ | ○ | ○ |
| I understand for-loops. | ○ | ○ | ○ | ○ | ○ |
| I understand arrays. | ○ | ○ | ○ | ○ | ○ |

| | | | | | |
|---|---|---|---|---|---|
| Given the chance, I would use this software again for a programming exercise. | ○ | ○ | ○ | ○ | ○ |
| Given the chance, I would work with this tutor again for a programming exercise. | ○ | ○ | ○ | ○ | ○ |
| This software would be just as helpful if there were no tutor there to help me. | ○ | ○ | ○ | ○ | ○ |
| The software was difficult to use. | ○ | ○ | ○ | ○ | ○ |

*Please choose one answer for each question.*

1. It seems like the TA was ______ knowledgeable about programming than me:

| | | |
|---|---|---|
| a. far less | c. about equally | e. far more |
| b. a little less | d. a little more | |

2.The tutor:

| | |
|---|---|
| a. asked too many questions | c. should have asked more questions |
| b. asked just the right amount of questions | |

3. When I asked a question, the tutor:

| | |
|---|---|
| a. usually responded with a helpful answer | c. usually did not respond to my question |
| b. usually responded with an unhelpful answer | d. I did not ask any questions |

4. Select a rating for each of the following:

| | Not at all | | Moderately | | Very Much |
|---|---|---|---|---|---|
| How helpful was the tutor? | ○ | ○ | ○ | ○ | ○ |
| How friendly was the tutor? | ○ | ○ | ○ | ○ | ○ |
| How empathetic was the tutor? | ○ | ○ | ○ | ○ | ○ |
| How knowledgeable was the tutor? | ○ | ○ | ○ | ○ | ○ |
| How genuine was the tutor? | ○ | ○ | ○ | ○ | ○ |
| How kind was the tutor? | ○ | ○ | ○ | ○ | ○ |

5. What I liked least about working with the tutor was:

6. What I liked best about working with the tutor was:

7. Additional comments: