Learning from Instruction: A Comprehension-Based Approach

Stephanie M. Doane
Young Woo Sohn
David Adams
Department of Psychology
University of Illinois
Champaign, IL 61820
sdoane@s.psych.uiuc.edu

Danielle S. McNamara
Institute of Cognitive Science
University of Colorado
Boulder, CO 80309
mcnamara@clipr.colorado.edu

Abstract

A comprehension-based approach to learning assumes that incoming information and background knowledge are integrated to form a mental representation which is subsequently used to incorporate new knowledge. We demonstrate that this approach can indicate when people will learn from instructions. Specifically, we show that a computational model based on the construction-integration theory of comprehension (Kintsch, 1988) can explain and predict how individual users will comprehend help prompts that guide their generation of successful complex commands within an operating system.

Theoretical Background

Learning from problem solving episodes has previously been examined by different traditions. Case-based planning (e.g., Hammond, 1989) assumes that we acquire knowledge by storing cases in memory which are, in general terms, the specific plans for different problems. Search-based models like SOAR learn by chunking the results of the search process (Rosenbloom et al., 1991). The present research builds upon a third emerging theory of learning, a comprehension based approach, which uses an association between problem descriptions (in this case incoming instructions to produce a command) and background knowledge to first activate relevant knowledge to construct a coherent situation model. This mental representation is then used to incorporate new knowledge (e.g., Schmalhofer & Tschaitschian, 1993).

We hypothesize that a computational model based on the construction-integration theory of comprehension (Kintsch, 1988) can explain and predict how individual users will comprehend and learn from help instructions as they attempt to generate complex computer commands. We are analyzing learning in the context of a theory of comprehension that has been used to explain story comprehension (Kintsch, 1988), algebra story problem comprehension (Kintsch, 1988), the solution of simple computing tasks (Mannes & Kintsch, 1991), and the Tower of Hanoi task (Schmalhofer & Tschaitschian, 1993). Thus, we are performing our research in the context of a general architecture of cognition.

The main goal of this study is to determine if this comprehension-based framework can be extended to account for learning from technical instructions. Specifically, we evaluate whether the comprehension strategies of UNICOM (Doane, Kintsch, & Polson, 1989), a construction-integration model containing knowledge of UNIX commands, adequately account for the type of instructions users find helpful to their command production performance. We have detailed empirical data on learning to produce complex, sequence-dependent commands in the UNIX operating system, a complex problem solving task. In previous empirical studies (Doane, McNamara, Kintsch, Polson, & Clawson, 1992), we asked users of varying experience with the UNIX operating system to produce complex UNIX commands, and then provided help prompts when the commands they produced were erroneous. The help prompts were designed to assist subjects with both knowledge and processes that our previous research has suggested are lacking in less expert users (Doane, Pellegrino, & Klatzky, 1990). The results showed significant differences in learning from instructions (prompts) as a function of UNIX background knowledge.

In the present work, we extended our model to include comprehension-based learning mechanisms in order to model the individuals in the prompting study. In our modeling, each subject's performance was analyzed to identify their initial knowledge base, which represents the knowledge they displayed without prompting. Using this knowledge base, we then "give" the model the same prompts that the subject was given when it executes an unsuccessful action plan, and then run the model again so the incoming prompt instructions can activate knowledge to attempt to solve the problem again. In so doing, we extend this theory to understanding how users learn from instructions to plan complex actions, and we can provide a detailed analysis of the match between modeled and actual performance.

UNICOM Construction/Integration Model

The construction-integration model is a cognitive architecture that is based on a general theory of discourse comprehension and that represents knowledge as associations (Kintsch, 1988). In the context of the present research, using UNIX refers to the comprehension of brief instructions to produce legal UNIX commands and associating this instruction with knowledge about UNIX in order to develop an action plan. The focus of our analysis is not so much on understanding the text per se, but on the way these instructions activate the UNIX knowledge relevant to the performance of the specified task.

The model activates knowledge in parallel through activation and selection of contextually relevant knowledge. It does not plan ahead; rather, the model reacts to the current state of the world to guide knowledge activation and selection in a step-by-step fashion. The instructional text and the current state of the operating system serve as cues for activation of the relevant knowledge and for organizing this knowledge to produce an action sequence. The symbolic/connectionist architecture of this model is similar to that of ECHO (Thagard, 1989) and ACME (Holyoak & Thagard, 1989): It uses symbolic rules to interrelate knowledge to develop a situation model, and then spreads activation throughout this representation using constraint

satisfaction. This results in focused activation of knowledge relevant to the current goal.

Classes of knowledge. The UNICOM model requires that three classes of knowledge be available to simulate command production. The first class, *world knowledge*, represents the state of the world at the current moment. Examples of world knowledge include knowledge of the current task, what files exist on the current directory, what directory you are in, and what system is in use (UNIX).

The next class of knowledge, *general knowledge*, refers to facts about UNIX. To facilitate discussion of the UNICOM knowledge base, we will start with the command "nroff -ms ATT2>ATT1" as an example. This command formats the contents of the file ATT2 using the utility nroff and the -ms macro package, and then stores the results in the file ATT1. There are four types of general knowledge required to produce a composite command. The model must know: (a) command syntax, (b) I/O redirection syntax, (c) conceptual facts about the redirection of input and output, and (d) conceptual facts about command redirection.

An example of command syntax knowledge is knowing the nroff command and the -ms flag. An example of I/O redirection syntax is knowing the ">" redirection symbol. Examples of conceptual facts about the redirection of input and output include the conceptual knowledge that redirection of input and output can occur between commands. This is separate from the syntax specific knowledge of I/O redirection symbols. (Some users appear to know that redirection can occur, and not know the specific syntax.) Finally, an example of a conceptual fact about command redirection would be the knowledge that the output of nroff can be redirected to a file.

Finally, the third class of knowledge, called *plan elements*, are the "executable" forms of knowledge about UNIX. Plan elements describe actions that can be taken in the world, and they specify conditions under which actions can be taken. Thus, users have condition-action rules that they can consider and execute if conditions are correct. In the model, plan elements have three parts. The first is the name of the plan element. The second component contains the preconditions that must be present either in the world or in the general knowledge for the plan element to fire. For example, if a file to be formatted does not exist in the world knowledge, then a plan element that requires that file exist will not be able to fire. Finally, there is an outcome component of plan elements. These contain facts that will be added to the world if the plan element fires. For example, once a file is formatted, the world knowledge will change to reflect the fact that the formatted contents of the file exist in the world. World knowledge will also change when incoming prompt instructions are introduced.

Model execution. Many plan elements may be selected in sequence to form an entire action plan (described below). The model operates in a cyclical fashion. The model fires the most activated plan element whose preconditions are satisfied in the world. When it fires, the outcome of the selected plan element is added to the world. The selection of the next plan element is determined by the modified contents of the world. For plans to be selected for execution, they must be contextually relevant to the current task (as dictated by the world knowledge) and the facts that allow them to fire (preconditions) must exist in the knowledge base.

Research Goals of Prompting Study

The goal of the empirical prompting study was to determine more precisely what users at different levels of expertise know about UNIX, what information is lacking when users produce erroneous commands, and what information (i.e., prompt contents) helps the users. The experiment used a prompting paradigm to assess the knowledge and processes of users at various levels of expertise. We assumed that users have different amounts of the required four types of knowledge and that displaying the prompts that help with each type of knowledge will influence subsequent user performance, if they lack this knowledge.

In the full study, twenty-two computer science and electrical engineering majors received 21 composite command production tasks. All subjects had received prior instruction about redirecting standard input and output in their coursework and had experience using redirection symbols to complete coursework or other tasks using UNIX. Novices had less than 1.25 years of experience with UNIX; intermediates had between 1.25 and 3.0 years experience with UNIX; and experts had greater than three years experience with UNIX. We can only summarize the procedure here - see Doane et al. (1992) for details.

All production tasks were performed on a computer. The stimuli were task statements, a fixed directory of file names and a series of help prompts, displayed when appropriate on the screen, as well as three "error cards" presented by the experimenter. Subjects typed their command or series of commands on the keyboard to accomplish a given task and then used the mouse to "click" on a display button to obtain an evaluation of their answer. The task instructions described actions that could best be accomplished by combining two or three commands through the use of redirection (i.e., composite problems; see Table 1). Accompanying the task statement was a fixed directory listing of all file names that were used in the experiment.

For incorrect responses, there was a series of help prompts designed to address specific types of deficits in the subjects' UNIX knowledge. These prompts were displayed on the screen one at a time in a fixed order regardless of the type of error that the subject had made. The system simply parsed an answer to determine if it contained the required syntax in the requisite order and if it did not, then the system would display the next prompt in the sequence.

There are seven different areas in which the help prompts could assist the subjects, and these are best described by referring to the example in Table 1. Prompt 1 parses the task statement into relevant command concepts. Prompt 2 identifies actual command syntax (command syntax knowledge). Prompt 3 explains concepts of redirection in an abstract fashion, independent of syntax (conceptual I/O redirection knowledge). Prompt 4 identifies the actual I/O redirection symbols (I/O redirection syntax knowledge) required for the problem. Prompts 5 and 7 remind the user of the complete set of items that have already been identified in previous prompts. Prompt 6 is the first prompt that

determines the order of commands and symbols for the user (providing command redirection knowledge and help with tracking intermediate results). This information is repeated in Prompt 8. Finally, Prompt 9 gives the user the correct production.

Modeling prompting performance. To develop individual knowledge bases for simulation, we scored each subject's answers to determine what knowledge they displayed prior to instruction on that knowledge. For example, if a subject used the command "nroff" before we provided any information about the command, then their initial knowledge base would receive credit for the command syntax knowledge for nroff and the plan to produce nroff by itself. If the subject tried to use nroff in the context of a redirection symbol (e.g., "nroff -ms filename | lpr") then they would be given credit for the command redirection knowledge for nroff that its output could be redirected as input to another command.

The subject's knowledge was entered into an initial knowledge base, and the model was given the initial problem statement in the "world" knowledge. The system then went through a series of construction-integration cycles to produce an action plan. If the model's plan was not correct (regardless of the subject's actual performance), it was given the text description of the first prompt, and the process began again. If the next attempt was again unsuccessful, then the text description for prompt 2 was added to the world knowledge, and so on.

The text descriptions of the prompts influence the activation of knowledge in the system. If they overlap sufficiently with existing knowledge and are relevant to the goal, then they will be able to provide preconditions to the plan element that fires. If this takes place, the knowledge given in the prompt that is used in a fired plan (correct or not) is permanently added to the knowledge base. Only the four most highly activated prompt propositions can remain in the world knowledge during any one cycle. Thus, if an incoming prompt was not highly associated with the existing knowledge or the goal, it was dropped from the knowledge base. Below we discuss the empirical and

Table 1. Example of task description and prompts for the problem nroff -ms ATT2>ATT1

Task Description

Format the text in ATT2 using the -ms macro package and store the formatted version in ATT1

Prompts

Prompt 1. You will need to use the following command
One that will format the contents of a file using the -ms macro package

Prompt 2 You will need to use this command
nroff -ms will format the contents of a file using the -ms macro package

Prompt 3 You will need to use a special symbol that redirects command output to a file

Prompt 4 You will need to use the arrow symbol " ≥ " that redirects output from a command to a file

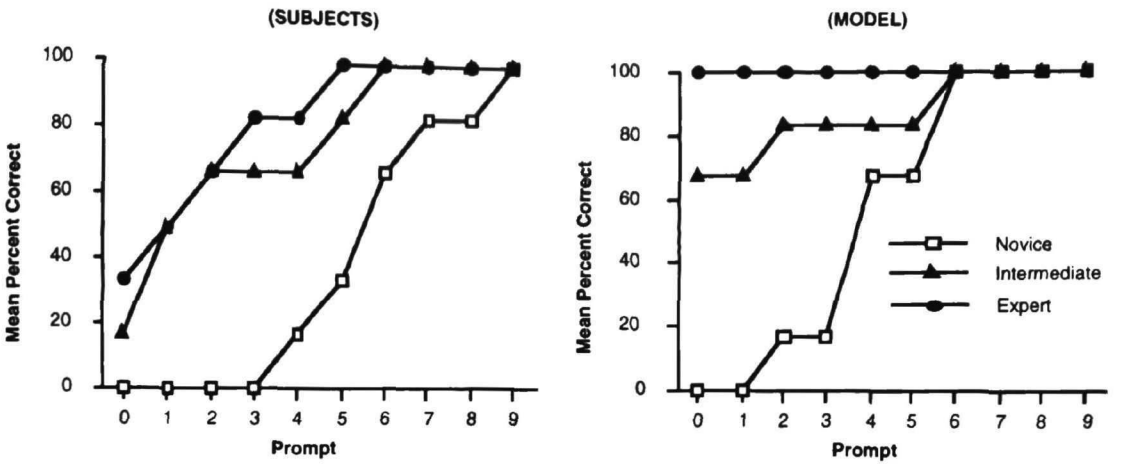
Prompt 5 You will need to use the arrow symbol " ≥ " and the command nroff -ms

Prompt 6 You'll need to use an nroff -ms on ATT2 (which will output the formatted contents of ATT2), and you'll need to redirect this output as input to ATT1

Prompt 7 You will need to use exactly the following command elements (though not necessarily in this order): ≥, nroff -ms

Prompt 8 You'll need to use the command nroff -ms followed by the arrow symbol " ≥ "

Prompt 9 The correct production is
nroff -ms ATT2>ATT1
Please enter this production now



Figures 1(a) and 1(b). Mean percent correct productions for novice, intermediate, and expert subjects, and for the simulation model of these subjects using UNICOM.

modeling results for the 6 subjects modeled (2 novices, 2 experts, 2 intermediates).

Results and Discussion

Scoring correct command productions. For the empirical work, productions were scored as correct by the computer if they matched the syntax of the idealized command (spaces were not counted). Thus, a subject had to produce the command that required the least number of keystrokes (i.e., subjects could not substitute "sort file1>temp; head temp>file2" for the command "sort file1 | head>file2"). Productions produced by each model were scored using the same rules of correctness. The problems simulated in this paper are those requiring the greatest percentage (60-100%) of new knowledge for solution, as detailed in Doane et al. (1992). This subset of problems is discussed in this paper.

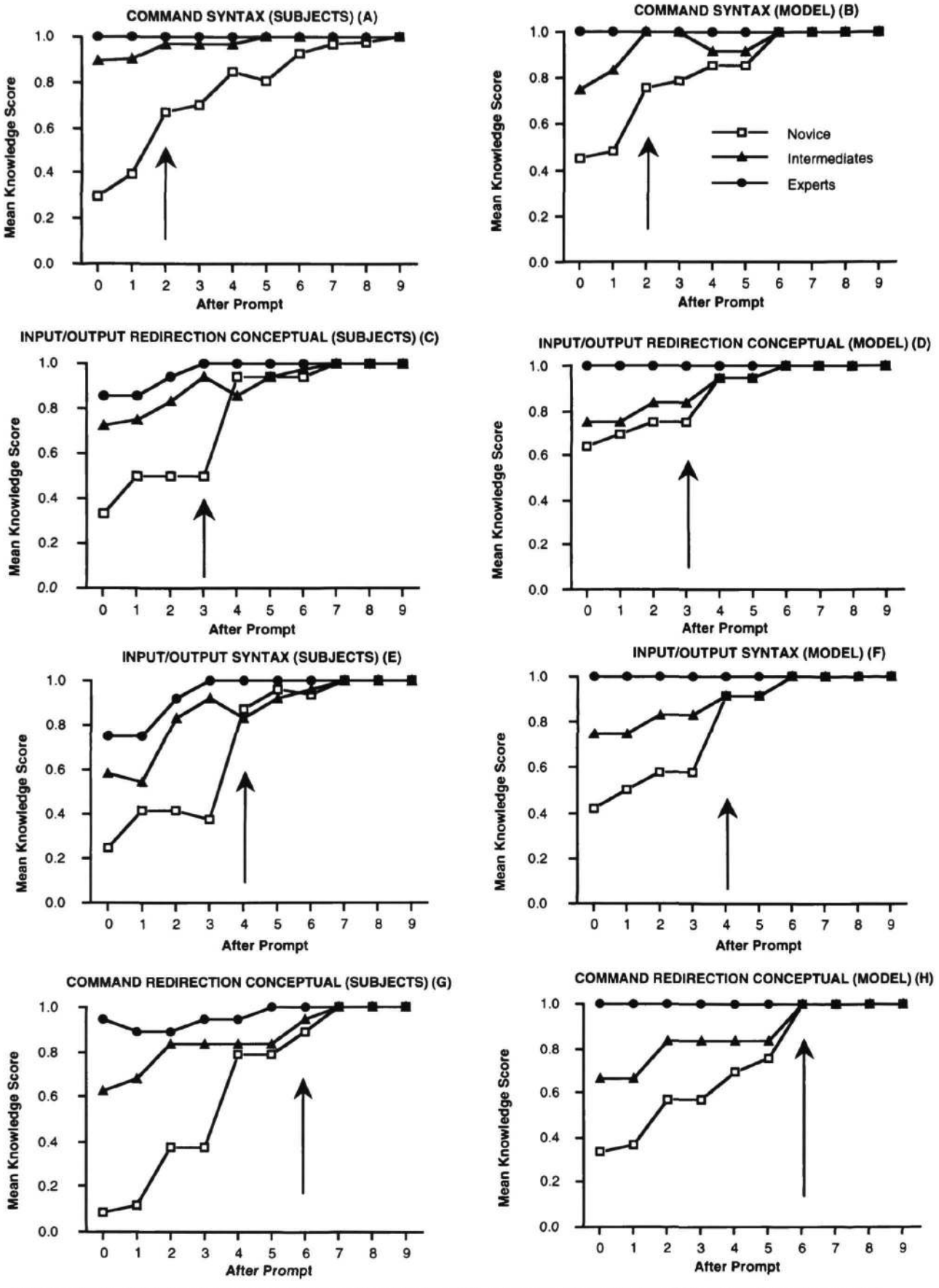
Correct productions as a function of prompt. Figures 1(a) and 1(b) show the cumulative percentage of correct composite productions for the three subject groups and modeled subject groups as a function of prompt. The data are cumulative; subjects (and modeled subjects) who correctly produced a problem at Prompt 4 were included as correct data points at Prompts 5-9 as well. Thus, at Prompt 9, all of the subjects in each expertise group were at 100% correct performance. Looking at the empirical results in Figure 1(a), experts have the highest correct percentage overall, followed by the less expert groups. Prompts have differential influences on correcting performance for the three expertise groups. For example, the change in percent correct performance from Prompt 3 to Prompt 4 is zero for intermediates and experts, suggesting that Prompt 4, which gives I/O syntax information (see Table 1) provided little or no new information to them. Conversely, the same change between Prompts 3 and 4 for the novices is large, suggesting that this prompt does provide them with significant new information. Experts and intermediates require fewer prompts in order to obtain perfect performance. Novices, in contrast, only obtain perfect performance once they are exposed to the final prompt which gives the exact command, Prompt 9. Looking at the modeling results in Figure 1(b), we obtain the basic expertise effect[1]. For the model, as for the subjects, Prompt 4 helps the novices but not the intermediate group.

Scoring of knowledge. Each of the problems given to subjects and to the Learning-UNICOM model required a certain amount of the four types of component knowledge discussed earlier in the section describing the UNICOM model. Answers for the present tasks were scored for the percentage of each type of knowledge displayed by a subject and by a model of the subject at each prompt level.

Knowledge analyses. Figures 2 (a-h) show the mean knowledge scores for the three expertise groups after prompts 0-9 for both the subjects and the modeled subjects. The arrow markers specify which prompt first provided information relevant to the knowledge type displayed in the graph. For example, in Figure 2(a), Prompt 2 is the first prompt that describes all of the command syntax knowledge required to complete the task (see Table 1 for an example of all prompt types described in this section), and the arrow indicates knowledge displayed after presentation of Prompt 2. The change in the knowledge score for command syntax between Prompts 1 and 2 indicates the effect of Prompt 2. The component knowledge shown is higher than the percent correct scores shown in Figure 1(a). This is because an attempt can show high, but not perfect component knowledge, and component knowledge must be perfect for an attempt to be entered as correct in Figures 1(a) and 1(b).

The difference between percent correct performance and the amount of knowledge displayed in an attempt can be examined by comparing the knowledge scores shown in Figures 2 (a-b) with the percent correct performance shown

[1]We chose to model two representative experts, and scored across their performance, they displayed all of the requisite knowledge without prompting. This led to 100% correct performance by the model (see Figure 1(b)).



Figures 2(a) - 2(h). Mean command syntax knowledge, I/O conceptual knowledge, I/O syntax knowledge and command redirtection knowledge for expert, intermediate, and novice subjects, and modeled subjects.

Knowledge Type	Knowledge Scores		Changes in Knowledge Scores	
	Novice	Intermediate	Novice	Intermediate
Command Syntax	.99	.74	.82	.44
I/O Redirection	.98	.80	.41	.18
I/O Syntax	.97	.81	.80	.07
Command Red.	.94	.97	.48	.83

Table 2. R^2 values for model fit with subject knowledge scores and changes in knowledge scores.

in Figures 1(a) and 1(b). Figure 2(a) suggests that for novice and intermediate groups, presentation of Prompt 2 improves command syntax knowledge, but only intermediates show improvement in percent correct performance (see Figure 1(a)). The lack of change in percent correct performance for the novice groups (see Figure 1(a)) suggests that for them, the prompt is not sufficient to guarantee that subsequent attempts will show perfect command syntax knowledge. Figure 2(b) shows a similar pattern of improvement for modeled novices and intermediates in response to the command syntax prompt. The novice model shows the greatest improvement in command syntax knowledge at Prompt 2. The novice model also shows an increase in correct performance (see Figure 1(b) at Prompt 2), which differs from the subject performance. The remaining figures (Figures 2(c-h)) can be examined in a similar fashion, where the comparisons show that the model does a good job of predicting what type of information is important to improve the amount of knowledge displayed in an attempt for novices, and slightly less so for intermediates.

To quantify the fit between the model and the subject data, correlations were performed on the knowledge scores as a function of prompt and on the change in knowledge scores as a function of prompt. The change scores provide a more stringent test of the fit between the model and the subject data because it pinpoints the changes between prompts rather than the general increase in knowledge. Table 2 shows the resulting R^2 values. Descriptively speaking, Table 2 suggests that the model does a good job of predicting the pattern of improvement in percent correct performance, showing the best fit with the novice data. The change scores indicate that the model does a good job of predicting the syntax-based knowledge for novices, and command redirection knowledge for the intermediates The fit between expert's performance and the model's predictions was not calculated due to ceiling performance.

The analyses suggest that there is a good match between what the model learns from instructions, and what the actual subjects learn. The differences in what knowledge is relevant as a function of expertise is consistent for the actual novice and intermediate subjects, and their models.

General Discussion

We have shown that the construction-integration model can be extended to account for learning from technical instructions. Using this comprehension-based approach, we are able to predict what prompt instructions users will apply and learn as a function of their background knowledge. This work has implications for computer-aided instruction and intelligent tutoring. If we can specify what instructions will be effective based on the activation resulting from the overlap between incoming instructions and background knowledge, then we can design more effective instructional systems.

Acknowledgments

The authors thank Gary Bradshaw, Walter Kintsch, Suzanne Mannes, and Gregory Murphy for their insightful comments on this research and on earlier versions of this manuscript.

References

Doane, S. M., Kintsch, W. & Polson, P. (1989). Action Planning: Producing UNIX commands. *Proceedings of the 11th Annual Conference of the Cognitive Science Society.* (pp. 458-465). Erlbaum.

Doane, S. M., McNamara, D. S., Kintsch, W., Polson, P. G., Clawson, D. (1992). Prompt comprehension in UNIX command production. *Memory and Cognition, 20(4),* 327-343.

Doane, S. M., Pellegrino, J. W., & Klatzky, R. L. (1990). Expertise in a computer operating system: Conceptualization and performance. *Human-Computer Interaction, 5,* 267-304.

Hammond, k. (1989). *Case-based planning.* London: Academic Press.

Holyoak, K. J., & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science, 13,* 295-355.

Kintsch, W. (1988). The use of knowledge in discourse processing: A construction-integration model. *Psychological Review, 95,* 163-182.

Mannes, S. M. & Kintsch, W. (1991). Routine computing tasks; Planning as understanding. *Cognitive Science, 15(3),* 305-342.

Polson, P. G. & Lewis, C. H. (1990). Theory-based design for easily learned interfaces. *Human-Computer Interaction, 5,* 191-220.

Rosenbloom, P. S., Laird, J. E., Newell, A., & McCarl, R. (1991). A preliminary analysis of the SOAR architecture as a basis for general intelligence. *Artificial Intelligence, 47,* 289-325.

Schmalhofer, F., & Tschaitschian, B. (1993). The acquisition of a procedure schema from text and experiences. *Proceedings of the 15th Annual Conference of the Cognitive Science Society.* (pp. 883-888). Erlbaum.

Thagard, P. (1989). Explanatory coherence. *Brain and Behavioral Sciences, 12,* 435-467.