Susan Obermeyer now works in product line management in the Digital Switching Division of Northern Telecom. She earned her bachelor's degree in electrical engineering from McMaster University, Hamilton, Ontario; and her master's degree in computer science from the University of Waterloo, Ontario.

THE EFFECTS OF FREQUENCY AND LENGTH OF COMMANDS AND TRAINING TRANSFER ON TEXT EDITING PERFORMANCE

VIRGINIA A. L. GUNTHER, ALEXIS GROSOFOSKY,
DANIEL J. BURNS, DAVID G. PAYNE

The nature of the command terms used by computer systems is an important issue affecting human computer interaction. As such, this area has received considerable attention recently (e.g., Ehrenreich, 1982, 1985; Grudin & Barnard, 1984; Landauer, Galotti, & Hartwell, 1983; Landauer & Galotti, 1984; Ledgard, Whiteside, Singer, & Seymour, 1980; Scapin, 1981, 1982). The issues addressed by these researchers are particularly important in light of the larger number of computer users who have little or no knowledge of the underlying logic of either the hardware or the software that they are using. As a group, these users often believe that the system should be "fast and easy to use". One area in which this expectation is especially widespread is that of wordprocessing packages. Although wordprocessing packages generally share the same basic functional purposes, the specific command terms that each uses for a particular function are generally not the same or even similar. There are even instances where different versions of the same wordprocessing package have different commands and/or functions than previous versions. Therefore, not only is the issue of the ease of learning and use of commands important, but also the issue of how these commands might be changed (either across different levels of user expertise or different versions of the software) so that there is maximum transfer of training from old commands to new commands.

Previous research in this area has been concerned with transfer of training between different wordprocessing packages (e.g., Karat, Boyes, Weisgerber, & Schafer, 1986). Since there are inherent differences between wordprocessing packages that include more than just command name changes it was felt that examining transfer of training within the same wordprocessing environment would isolate the issue of transfer of training for command terms. Therefore, the present study was concerned with addressing transfer of training issues related to command length changes within the same wordprocessing package.

Method

Subjects, Design, and Procedure. Forty students enrollled in an introductory psychology course participated in four sessions spread over two consecutive days. In each session subjects were given hardcopy versions of eight paragraphs each containing eight errors noted by standard copyediting marks. Subjects were instructed to use the text editor to correct the errors in each paragraph, store that paragraph, and then retrieve the next paragraph. Different paragraphs were used for each session but the paragraphs were the same across subjects for each session. The length of the commands used to accomplish the text editing was manipulated within-subjects. On Day 1 subjects used either whole word comands or single letter commands and on Day 2 they were switched to the other command type (i.e., words on Day 1 with letters on Day 2 or vice versa). In all cases the single letter was the first letter of the whole word. Half the subjects used words that were high frequency and the other half used low frequency commands. Subjects were not informed that they would be switching command lengths. Therefore, the design was a 2 (Length of Command: words vs. letters) x 2 (Frequency: high vs. low) x 4 (Session: 1-4) mixed-factor design.

Results

A variety of performance measures related to speed and accuracy were obtained from each session. In addition, data regarding subjects' experience with computers, word processors, and typing was also collected. Due to space limitations we will only discuss the variables of the total time to complete each session and total errors for a session. Not surprisingly, for each of the four between subjects conditions the total time required to complete each session decreased across the four sessions. Furthermore, this task completion time was greater for low frequency commands than for high frequency commands. Subjects in the letter-to-word condition showed an increase in task completion time on the third session, regardless of the relative frequency of

the whole word command. These results clearly indicate a negative transfer of training from letters to words with respect to the time required to complete a word processing task.

The error rates were higher in the letter-to-word conditions than in the word-to-letter conditions. Also, high frequency commands showed fewer errors than low frequency commands. When errors were classified according to the commands' overall functional purpose (e.g., deleting, adding, changing) we found that for high frequency commands errors were more likely to be due to the use of another command that was within the appropriate commands' functional purpose (e.g., deleting a word rather than a letter). The error classifications showed that for the letter-to-word condition most mistakes were made outside the functional grouping during sessions one and two, while the opposite was true for the word-to-letter conditions during those sessions.

Conclusions

These results suggest that relative word frequency and command type (letter vs. word) are relevant and important issues in assessing transfer of training within software packages. We would also argue that related findings in basic cognitive and learning psychology can be used to assess performance when design changes are contemplated for software packages. Finally, further research is needed to assess transfer of training issues with other aspects of the software interface such as functionality within the scope of computer aided design packages.

References

Ehrenreich, S. L. (1985). Computer abbreviations: Evidence and synthesis. Human Factors, 27, 143-155.

Grudin, J. and Barnard, P. (1984). The cognitive demands of learning and representing command names for text editing. Human Factors, 26, 407-422.

Landauer, T. K., Galotti, L. M. and Hartwell, S. (1983). Natural command names and initial learning: A study of text editing terms. Communications of the ACM, 26, 495-503.

Landauer, T. K. and Galotti, L. M. (1984). What makes a difference when? Comments on Grudin and Barnard. Human Factors, 26, 423-429.

Ledard, H. , Whiteside, J., Singer, A., and Seymour,W. (1980). The natural language of interactive systems. Communications of the ACM, 23, 556-563.

Scapin, D.L. (1981). Computer commands in restricted natural language: Some aspects of memory of experience. Human Factors, 23, 365-375.

Scapin, D.L. (1982). Generation effect, structuring and computer commands. Behavior and Information Technology, 1, 401-410.

About the Authors

Virginia A. L. Gunther is a graduate student in both the Psychology and Systems Science Departments at the State University of New York at Binghamton. She earned her M.A. degree from Towson State University in Experimental Psychology. Prior to attending SUNY Binghamton she taught psychology at the University of Maryland (European Division) and Hartford Community College. She has also been involved with research at the Kennedy Institute, Johns Hopkins Medical School. Virginia's interests include basic and applied research in the areas of human memory and attentional processes, artificial intelligence (connectionist), and human computer interface design.

Alexis Grosofsky is a graduate student in the Psychology Department at the State University of New York at Binghamton. She received her M.A. degree from SUNY Binghamton in 1985. Alexis' masters work was in the area of ecological perception. Currently Alexis is finishing her dissertation in the area of human memory. Alexis' research interests are in the areas of human memory and cognition, ecological perception, and human factors.

Daniel J. Burns received his M.A. and Ph.D. degrees in Experimental Psychology from the State University of New York at Binghamton. Dan's primary research interests are in the area of human memory.

Dan is currently an Assistant Professor in the Psychology Department at Creighton University in Omaha, Nebraska.

David G. Payne received his Ph.D. in Cognitive Psychology from Purdue University. David's current research interests include the design of person-machine interfaces, eyewitness memory, memory improvement, attentional processes and mental workload. David recently spent a summer as a NASA/ASEE faculty fellow at the NASA Langley Research Center where he conducted research concerning manual and voice control of remote video cameras. He is currently an Assistant Professor and Associate Chairman in the Psychology Department of the State University of New York at Binghamton.

A NATURAL LANGUAGE SHELL

MANTON M. MATTHEWS

This paper describes a natural language interface, that models users and maintains a knowledge base of how users are using the system. The system is developed under Unix using a combination of Prolog and C code. The natural language processing part is being implemented in UNSW Prolog with extensions to facilitate efficient processing of dictionaries and frame hierarchies.

Natural language has been viewed as overkill in many of the applications of today and rightly so. Direct manipulation techniques are much more natural and efficient in most domains than typing in "natural language." Rich however points out that natural language is a powerful communication mechanism and that in sufficiently complex domains the cost of processing natural language is justified [1]. Note that the cost of natural language is two-fold: the user must generate the natural language and the system must then "understand" it. The user must formulate his request and then currently in most cases type it into the system. However, in a few years speech processing systems will become more effective and then spoken natural language will be a viable input mechanism. Spoken natural language will be much more effective than relying totally on typed input, or totally on mouse or other pointer input. The most effective interfaces in the future will probably allow combinations of spoken natural language and pointing devices.

Our system does not attempt to incorporate all of these features into the current version, but lays a foundation that will accommodate these features. It currently uses a language that is a mixture of command language, natural language (typed) and mouse input. In this both of

> mv paper /fac/matthews
> move paper to my home directory

cause the system to move the specified file "paper" to the user's home directory. The system also allows the use of the natural constructs, to simplify communication. For instance assuming that the file "paper" had recently been referenced, perhaps by editing or viewing it, then the commands could be simplified to "move it to my home directory". The mouse input is primarily used for selecting commands and for dereferencing words or phrases, such as the words "it" or "here" in "move it here".

The system maintains a model of how the software on the system can and is being used. This knowledge base is incrementally created by the system observing commands, and trying to model what is going on. If it has a sequence of commands that it is not capable of handling it makes a note in a log. The system developer uses this log to extend the knowledge base with the assistance of the system. Extensions that commonly occur are additions to the dictionary, additions of new commands, additions of options to commands, etc.

In addition to the general knowledge base there is a model of the individual user that represents how the user has been using the system. This model is used in several ways, but the most important are (1) to provide a measure of sophistication with various pieces of software on the system and (2) as a memory extender (or extended C-shell history). The measure of sophistication is used in tailoring responses to the individual users as well as in processing queries. The "extended history" aspect allows the system to respond to

> "What was the font that I used to print the ACM paper in November?"

Note that the system could ask the user for help in disambiguating this query if it were necessary. Using this model of the system can understand and respond to user queries about the software.

References

(1) Elaine Rich, "Natural Language Understanding: How natural can it be?," Proc. of Second Conference on Artificial Intelligence Applications, pp. 372-377 (1985).

About the Author

Manton M. Matthews is Director of Graduate Studies for the Computer Science Department of the University of South Carolina. He earned his doctorate in mathematics from the University of South Carolina and spent a postdoctoral year at the University of North Carolina at Chapel Hill working in functional programming. Since 1981 he has been a member of the faculty at South Carolina. He has published papers on user modeling, knowledge-based systems, user interfaces, functional programming and graph theory.
