Transfer of Learning: Beyond Common Elements

Linda Tetzlaff

IBM, T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, N.Y. 10598

Abstract

An experiment on transfer of learning using text editors revealed significant differences in performance, based on the learning experience of the subjects. The set of commands of a text editor was divided into four subsets. Different groups of subjects learned these subsets in different orders. Depending on the order of learning, subjects formed different concepts of the editor as manifest by their choice of commands, their errors, and their model of the editor, elicited by a sorting task. Pragmatic production model approaches to transfer would need significant enhancement to accommodate this result.

Résumé

Une expérience sur le transfert de connaissance avec les éditeurs de textes a révélé des différences importantes entre les accomplissements des sujets, en fonction de leurs cours d'apprentissage. L'ensemble des commandes d'un éditeur de textes a été divisé en quatre sous-ensembles. Les différents groupes de sujets ont appris ces sous-ensembles présentés en ordres différents. Selon l'ordre d'apprentissage, les sujets ont formé des concepts différents de l'éditeur. Les concepts formés par les sujets étaient évidents en leurs choix de commandes, leurs erreurs, et leurs modèles de l'éditeur élicités par une tâche de classification. Pour accommoder ce résultat, il faut augmenter considérablement les modèles simplifiés, employant des règles de production, qui traitent du transfert d'apprentissage.

Keywords: transfer, cognitive skills, human factors, text editing, production models

Introduction

Transfer of learning is a problem vexing both users and manufacturers of computer software. Users discover that the skills and expectations that they have developed in the use of one application turn out to be inapplicable or substantially dissimilar to those appropriate to a second application on the same system. Manufacturers find that families of applications, intended to provide a common interface across different environments, nonetheless result in significant difficulties as users move from one instantiation of the system to the next [6]. Both kinds of transfer effects result in user frustration and employer retraining costs.

Transfer of learning can be described using production modelling, introduced to the analysis of human-computer interaction by Card, Moran and Newell [1]. A production model describes actions at the interface in terms of a set of if-then rules, in which a user recognizes some condition and then performs an appropriate action. Polson and Kieras [7] go a step further, quantifying this activity by counting the rules necessary to accomplish a task. The number of rules executed predict learning and performance times. In cases in which the user can take advantage of previous experience, learning time is predicted from the number of *new* productions to be learned: Productions shared by the old and the new task are assumed to transfer at no cost. This notion that common elements are the substance of transferred learning has its psychological origins in work by Thorndyke [9].

Polson and Kieras assume that performance time is a linear function of the number of rules. While they obtain a striking fit between the modelled interactions and subject performance data, many unresolved, non-trivial issues remain. Their results may reflect ceiling effects in an experimental environment which approximates idealized learning: Simple tasks were run within a single session, minimizing memory decay as a relevant factor. Neither the model nor the experimental design addressed issues of choice or judgement. Furthermore, production rule generation, so critical to an understanding of the relevant cognitive processes, and the linchpin of the quantitative modelling, remains an ill-defined process, burdened with unspecified assumptions.

The work of Gick and Holyoak on analogical reasoning [4] points to some fundamental limitations of a common elements approach to transfer. An implicit assumption underlying the Polson and Kieras work is that the user will in fact discern the common elements, and then transfer them without cost. What Gick and Holyoak demonstrate is that it is remarkably difficult for subjects to perceive common elements in the first place. Their work suggests that spontaneous inference of common elements is poor and that the ability to form inferences is but little assisted by advance knowledge of appropriate abstractions. Gentner [3] finds that identification of commonality is

based primarily on surface characteristics rather than on underlying functional similarities.

The primary focus of this experiment has been the concern that there are important aspects of learning and transfer of learning that are not well described by pragmatic production modelling. It is, of course, the case that retrospectively and descriptively *any* human interaction with a computer can be detailed as a set of production rules. However, for production modelling to have practical, predictive power, a number of simplifying assumptions must be made in advance of rule generation. The Polson and Kieras approach, for example, does not independently model the user's structural representation of the interface; it is implicitly assumed that the representation is captured in the production rules. Neither does it model the performance or transfer of the selection process. Card et al. also assume that experienced users select optimal paths at no cost (p. 267).

The pilot experiment described here attempts to explore the problem of transfer of learning in a context in which subjects must choose among methods to accomplish a task. An editor is developed (the composite editor) containing two ways of performing each modification (insert, delete, etc.). The composite editor is split into two pairs of editors. A 'structured' pair groups the commands into a line editor version and a full screen version. A 'disjoint' pair creates two functionally complete editors, each containing a mix of both line and full screen style commands. Subjects learn either the structured pair or the disjoint pair in each of two sessions. In the third session both groups work with the composite editor. Learning commands in a disjoint structure seems a reasonable, if somewhat extreme, approximation of the piecemeal way in which users acquire knowledge of complex systems. Learning commands coherently, in their intended context, explores an alternative instructional strategy which may also have implications for design.

In each treatment, subjects ultimately learn precisely the same command set. In the final session they must also develop selection rules to mediate amongst modification methods. Production models, which typically assume that performance is independent of instruction technique, order of command presentation, or the logical relationships among commands should predict equivalent performance in the final session. However, if presentation order and perceived editor structure has caused the subjects to develop different models of the editor, one might expect differences in performance.

This experiment is very similar to the order manipulations performed by Polson and Kieras, and not appreciably more complex, at least as compared to real world systems. Should significant differences in performance occur, extensions would have to be made for their model to be accommodate the results.

Subjects

Six temporary employees of the IBM Research Laboratory, with little or no previous editor experience, participated in the experiment. There were five women and one man, with a median age of 20 years. Subjects were given a free lunch for each day of participation. Subjects were divided into two groups of three, on the basis of the results of a typing test and the spatial memory test from the Cognitive Factor-Referenced Test Battery [2]. The groups were matched as equally as possible on both scores, following the evidence provided by Gomez et al. [5] that scores on these tests correlate significantly with success in learning to use a text editor[1].

One group (structured) was given an orderly progression of editors - a line editor followed by a full screen editor. The second (disjoint) received an arbitrarily composed, but functionally complete sequence.

Editor task

Materials

Five editors were constructed using the macro facilities of IBM's System Product Editor (XEDIT), and the Restructured Extended Executor language (REXX). The functions supported in each editor were: scroll up/down, locate string, move lines, delete lines/character, insert line/character, append and replace. All commands could be abbreviated using one or two characters, and the functionality of matched command pairs was identical. (For example, both li 'text' on the command line, and i in the prefix area, followed by text on the generated line, resulted in line insertion.) Commands within a pair differed with respect to their place of invocation, the setup required for successful execution and the command name. Each editor constituted a functionally complete subset of the facilities available in the composite editor (Table 1).

A manual was created for each editor, and a performance aid, showing command syntax and function, was created for use during the sessions.

The experiment was run using an IBM AT to monitor keystrokes and simulate a dedicated IBM 3278 terminal. The AT was connected to a VM timesharing system. Average system response time for editing tasks was less than .1 second. The keyboard environment was restricted to valid inputs by the tailoring facility of E78, an IBM internal use 3278 emulator for the IBM PC.

Seventeen single page, double spaced text documents were created, using general interest, non-fiction materials. The order of presentation of fifteen files was randomized across the experiment, with one additional file used by all subjects for practice and another for a verbal protocol.

[1] However, no relationship was actually found between spatial memory scores and any experimental variable in this experiment.

Single page documents were employed to eliminate variance introduced by page turning. Text appeared on the page in the same line format as it appeared on the screen in order to obviate the need to perform spatial translation to locate text on the screen. The documents averaged 217 words, 27 lines (8 words/line), and spanned two screens of display, enabling observation of possible variations in navigational technique.

A program was written in order to generate randomly positioned aberrations in the text, resulting in 14 modifications per document. Two text modifications were designed to elicit correction by each command type. The resulting mutilated document was marked up in red pencil for correction by the subject during testing. Each subject saw fifteen text documents, randomly ordered, five in each session. An additional document was used for initial training and for a final verbal protocol.

A driver was written to control and monitor the actual test session. The driver took the subject through a minimum of two, and up to 10 trials for each of five different documents. Each trial consisted of presentation of a mutilated document for correction. On filing, the driver compared the resulting file with its correct original. If an error was made, or if the time to completion was not within 15% of the previous trial, the same document was presented again. If no error was made, and if the time was within 15% of the previous trial (the presumptive asymptote of the learning curve), the subject advanced to the next document.

Commands were captured using CMON, and keystrokes using MMON, both IBM internal tools.

	LEDIT	FEDIT	AEDIT	BEDIT	MEDIT
file	CL	CL	CL	CL	CL
cursor up/down	--	ALL	ALL	ALL	ALL
cursor left/right	CL	ALL	ALL	ALL	ALL
scroll up	CL	CL	CL	CL	CL
scroll down	CL	CL	CL	CL	CL
locate string	CL	CL*	CL	CL*	CL/CL*
append char	CL	TA	TA	CL	CL/TA
replace char	CL	TA	TA	CL	CL/TA
delete char	CL	TA	CL	TA	CL/TA
insert char	CL	TA	TA	CL	CL/TA
delete line	CL	PA	PA	CL	CL/PA
insert line	CL	PA	CL	PA	CL/PA
move lines	CL	PA	PA	CL	CL/PA

Table 1. Distribution of commands across the editors, indicating place of invocaton

Key: CL - command line, TA - text area, PA - prefix area, ALL - anywhere on the screen. There were two commands to loacate a string. One, indicated by an asterisk, was optimized for text area modifications; the other, for command line modifications. Both were available in the mixed editor.

Procedure

The experiment consisted of three sessions, run on different days within a single week. Subjects were run individually by a single experimenter.

At the first session there was a general introduction to the experimental task, as well as to the keyboard, display and task materials.

On each day, the subject reviewed the manual describing the editor to be used for that session for as much time as necessary. With the assistance of the experimenter, the subject performed a series of standardized edits on a practice file, using each of the editor commands in turn. When practice on individual commands was complete, the subject practiced using all the commands, exactly as in the experimental task.

At this time, the subject progressed to the composite experimental task proper, under control of the automated driver. A performance aid, listing the available commands, was present throughout the session, and the experimenter only rarely intervened (e.g. to reset the screen following an asynchronous and irrelevant system message).

Following the completion of the last session, the subject edited one trial of a single new document, during which the subject was encouraged to comment on the reasons for his/her command selections.

Results

A number of measures were taken in order to evaluate the effect of editor structure on subjects' performance, and to capture a qualitative picture of what they had learned. Informally collected, mean practice times were 48, 34 and 21 minutes for the three sessions, respectively, indicating general task learning. On average, the structured group took 34 3.5 minute trials to complete the third session, while the disjoint group took 53 4.6 minute trials. Think time[2] was 2.0 minutes and 2.7 minutes respectively. (See Table 2 for a complete summary of the means and the analysis of variance.) Despite nearly comparable starting points, the structured group clearly shows a more favorable performance history.

Group	Total Time	Think Time	Trials
Structured			
LEDIT	284.5	201.7	12.1
FEDIT	226.9	129.2	22.9
MEDIT	210.4	118.0	34.2
Disjoint			
AEDIT	316.0	198.5	12.8
BEDIT	355.0	232.7	34.0
MEDIT	274.8	162.6	53.5
(p <)	.01	.02	.11

Table 2. Analysis of variance

Time per trial reported in seconds; P gives probability of difference between MEDIT sessions.

Command protocols as well as summary statistics on command usage in the MEDIT session were examined. In general, the structured group adopted a monolithic command strategy, utilizing the full screen command set presented in their second editor. The disjoint group used an approach which mixed not only commands from the line and full screen orientations, as one would expect, but also commands from both the initial and most recent sessions (Table 3).

	Line	Full Screen	First Editor	Second Editor
Structured	43	721	43	721
Disjoint	617	537	505	649

Table 3. Mixed editor session - command summary

Commands are broken down by source, as coming from either the line or full screen cluster, and as coming from the first or second editor presented. Commands share over all three sessions are not considered.

Command protocol analysis revealed that for both groups, navigation commands were appropriately paired with modification commands. Thus, subjects used line editor locate commands to perform line editor modifications and full screen locates for text area modifications. Choice errors, in which a subject started to issue a navigation command, and then backspaced over it to issue another, were observed for the disjoint group only. The few command line commands issued by the structured group were directed at line targets (e.g. delete line); all character oriented modifications were done in full screen mode.

Card Sorting task

Materials

The name of each command was written on a 3 x 5 index card. As many cards as the subject wanted were made for each command, and each command pile was placed on the table in alphabetical order at the start of the sorting task. A sample set containing ordinary objects was also created in order to demonstrate the task.

Procedure

At the end of the experiment, subjects were asked to sort the cards with the command names on them. This task represented an effort to characterize the subject's 'model' of the editor. The subject was asked to arrange the commands in as many piles as seemed appropriate, feeling free to use commands in multiple piles. In addition, the subject was asked to provide some kind of label for each pile.

[2] Think time was defined as the difference between elapsed time and typing time. Typing time was taken as the sum of character and cursor keystroking times; character keystroking time was computed assuming five characters per word and the subject's typing speed; consecutive cursor keystrokes were assumed to have been produced by the typamatic feature, which was measured at .1 second per stroke.

Results

The card sorting data were analyzed with multidimensional scaling and hierarchical clustering techniques. In addition, the labels assigned to the individual piles were evaluated.

Hierarchical clustering indicated that subjects in the structured group primarily viewed commands belonging to either the line editor or the full screen editor, while the disjoint group first thought of commands as serving navigational or modification functions (Figure 1).

```
Structured  up / down
            cursor keys
            command line commands
                locates
                character oriented commands
                line oriented commands
            full screen commands
                line oriented commands
                character oriented commands

Disjoint    navigation commands (up, down and locates)
            cursor keys
            character oriented commands
                command line commands
                full screen commands
            line oriented commands
                command line commands
                full screen commands
```

Figure 1. Hierarchical clustering of commands by group

MDS analysis produced similar results. Distance measures for the MDS analysis were computed by assigning a value of one to each command pair grouped by the subject. Command pair values were summed across subjects, with high values representing a high degree of perceived similarity. The analysis indicated that the subjects' view of the commands could be reasonably described as distributing across three dimensions (Table 4).

The structured group emphasized procedural separation of commands by their usage in line or full screen mode. Secondarily, they divided them according to function, as navigation or modification commands. Finally, they focussed on the command object, either character or line. While the both groups divided along the same dimensions, the dimensions differed markedly in importance. Most of the variance for the structured group was accounted for by the procedural dimension, while the bulk of the variance for the disjoint group was accounted for by command function - the procedural dimension adding only 18% to the variance accounted for. At the same time, both groups rank functional class above object type.

DIMENSION			Structured RSQ	Structured var	Disjoint RSQ	Disjoint var
(Procedure)	Line	/ Screen	.430	43.0	.885	17.6
(Function)	Modify	/ Navigate	.675	24.5	.437	43.7
(Object)	Character	/ Line	.825	15.0	.709	27.2

Table 4. MDS analysis

Variance accounted for is computed from the difference between dimensional r-square (RSQ) values.

A manual analysis was performed on the labels assigned to the sorted piles. They were categorized as relating to syntax (e.g. commands that take a numeric parameter), to function (e.g. commands used for deleting) and to ease of use (e.g. difficult commands). While both groups had a similar number of groups devoted to function, the disjoint group seemed considerably more preoccupied with the details of syntax (Table 4).

	Structured Number	Structured %	Disjoint Number	Disjoint %
Syntax	5	16	28	57
Function	22	69	20	41
Difficulty	5	16	1	2
Total	32		49	

Table 5. Label analysis

General Discussion

In this experiment, the structured group learned first about functions available from the command line, and subsequently about commands available in the text area. The disjoint group encountered both kinds of commands in both sessions. Thus, the perception of 'structure' in this experiment was enabled by temporally contiguous presentation of commands, similarly placed on the screen, and hence having correlated physical attributes. The evidence suggests that the structured group developed a coherent picture of distinct sets of commands that could be productively used together, and that the crisp organization at this high level obviated the need for continual decision at the command level. The poorer performance of the disjoint group, their experience of choice errors, and the heterogeneity of their working command set suggest that they were less successful at discerning the underlying structure of the system, and less able to develop efficient strategies.

Transfer of learning is difficult to model for many reasons, including the uncertainty that components previously 'learned' will be present in memory and hence available for transfer, the dependence of transfer on the conditions of learning, and the inability of the designer to control acquisition of concepts, strategies and knowledge structures above the level of mechanical procedures.

While it is likely that the necessity for new learning was reduced by presence of common elements, the substantial quantitative and qualitative differences in performance between these two groups suggest that there is more that must be taken into account. The Polson-Kieras model assumes that the constituent elements are independent of one another: once learned, if appropriate to a new condition, an element will transfer. The results of this experiment suggest, however, that there is more to be learned than the set of component productions. As it stands, the information contained in the component productions is insufficient to describe the base strategies available for problem solving in a new situation. This, too, must surely be considered a component of transferred learning.

Well structured interfaces, in which distinctive and functionally complete features or subsystems can be perceptually discerned, would seem to offer natural categories in which temporal association, physical features, motor productions and functional scripts can be correlated [8]. These should share the benefits with other natural categories of faster response times, speed of learning, facilitated transfer. They should permit operational selection at the categorical level, which limits the range of applicable decisions and facilitates procedural optimization.

There are two ways to think about the implications of this experiment for future system design. One view is that like structural features must be either temporally, physically or logically proximate to be perceived as coherent wholes. Another view is that whatever the structure of the system, it must be presented in a way that calls attention to related features, in order for users to capitalize on their similarities. This should maximize transfer of learning and facilitate development of efficient performance strategies.

References

1. Card, S. K., Moran, T. P., and Newell, A. *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, Hillsdale, N.J., 1983.

2. Ekstrom, R. S., French, J. W., and Harman, H. H. *Manual for kit of factor-referenced cognitive tests.* Educational Testing Service, Princeton, N.J., 1976.

3. Gentner, D. Analogical subprocesses, Rutgers University, Analogica '85.December 1985. Unpublished conference report.

4. Gick, M. L. and Holyoak, K. J. Schema induction and analogical transfer. *Cognitive Psychology*, 15:1 - 38, 1983.

5. Gomez, L. M., Egan, D. E., Wheeler, E. A., Sharma, D. K., and Gruchacz, A. M. How interface design determines who has difficulty learning to use a text editor. *Proceedings CHI'83 Human Factors in Computing Systems. (Boston, December 12-15)*, 1983.

6. Karat, J., Boyes, L., Weisberger, S., and Schafer, C. Transfer between word processing systems. *Proceedings CHI'86 Human Factors in Computing Systems. (Boston, April 13-17)*, 1986.

7. Polson, P. G. and Kieras, D. E. A quantitative model of the learning and performance of text editing knowledge. *Proceedings: CHI'85 Human Factors in Computing Systems. (San Francisco, April 14-18)*, 1985.

8. Rosch, E. *Cognition and Categorization*, chapter Principles of Categorization. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1978.

9. Thorndyke, E. L. and Woodworth, R. S. The influence of improvement in one mental function upon the the efficiency of other functions.. *Psychological Review*, 8:157-203, 1901.